TESE DE DOUTORADO

SELEÇÃO DE PARÂMETROS DE MÁQUINAS DE VETORES DE SUPORTE USANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM META-HEURÍSTICAS

Carlos Eduardo da Silva Santos

Brasília, março de 2019

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TESE DE DOUTORADO

SELEÇÃO DE PARÂMETROS DE MÁQUINAS DE VETORES DE SUPORTE USANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM META-HEURÍSTICAS

Carlos Eduardo da Silva Santos

Tese de Doutorado submetida ao Departamento de Engenharia Mecânica como requisito parcial para obtenção do grau de Doutor em Sistemas Mecatrônicos

Banca Examinadora

Prof. Dr. Carlos H. Llanos Quintero, ENM/FT/UnB _____
Orientador

Prof. Dr. Alexandre Ricardo Soares Romariz, ENE/UnB _____
Examinador externo

Prof. Dr. Flávio de Barros Vidal, CIC/UnB _____
Examinador interno

Prof. Dr. Gilberto Reynoso Meza, PPGEPS/PUCPR _____
Examinador externo

Prof. Dr. Leandro dos Santos Coelho, PPGEPS/PUCPR e PPGEE/UFPR _____
Co-orientador

FICHA CATALOGRÁFICA

SANTOS, CARLOS EDUARDO
SELEÇÃO DE PARÂMETROS DE MÁQUINAS DE VETORESDE SUPORTE USANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM META-HEURÍSTICAS [Distrito Federal] 2019.
xvi, 136 p., 210 x 297 mm (ENM/FT/UnB, Doutor, Engenharia Mecânica, 2019).
Tese de Doutorado - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Mecânica

1. Meta-heurísticas
2. Máquinas de Vetores de Suporte
3. Aprendizagem de Máquinas
4. Otimização Multiobjetivo.
I. ENM/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SANTOS, C.E. (2019). *SELEÇÃO DE PARÂMETROS DE MÁQUINAS DE VETORESDE SUPORTE USANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM META-HEURÍSTICAS*. Tese de Doutorado, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 136 p.

CESSÃO DE DIREITOS

AUTOR: Carlos Eduardo da Silva Santos
TÍTULO: SELEÇÃO DE PARÂMETROS DE MÁQUINAS DE VETORESDE SUPORTE USANDO OTIMIZAÇÃO MULTIOBJETIVO BASEADA EM META-HEURÍSTICAS.
GRAU: Doutor em Sistemas Mecatrônicos ANO: 2019

Carlos Eduardo da Silva Santos
Depto. de Engenharia Mecânica (ENM) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Agradecimentos

Agradeço aos meus pais, por todo o aporte, compreensão, ensinamentos ao longo da vida sem os quais não teria, de forma alguma, realizado meus sonhos.
Ao meu orientador, Prof. Carlos Llanos, pela valiosa amizade, confiança, ensinamentos, companheirismo, compreensão e liberdade de pesquisa oferecida ao longo destes anos.
Ao meu co-orientador, Prof. Leandro dos Santos Coelho, pela valiosa contribuição, paciência e valorosa orientação durante todo o projeto.
Aos demais professores que contribuíram com seus ensinamentos sobre os temas das pesquisas aqui desenvolvidas.
Aos meus colegas e amigos de trabalho e pesquisa que contribuíram imensamente no desenvolvimento desta pesquisa, em especial a Renato Sampaio, Daniel Muñoz, Eduardo Mesquita, Fabián Barrera, Guillermo Alvarez, Oscar Anacona, Sérgio Cruz, Sérgio Pertuz, João Carlos, Javier Alexis, Reurisson, Everaldo e a tantos outros que participaram do LEIA - Laboratório de Sistemas Embarcados e Aplicações de Circuitos Integrados da Universidade de Brasília.
Ao meu irmão Tercio Filho, ao Prof. Sergio Francisco pelo apoio, disponibilizando computadores da UFG Catalão para realização dos experimentos do projeto final.
Aos demais amigos evolutivos intra e extrafísicos pelo companheirismo ao longo das vidas.

Carlos Eduardo da Silva Santos

RESUMO

Máquinas de Vetores de Suporte (*Support Vector Machines* - SVMs) representam uma técnica de Aprendizagem de Máquina (*Machine Learning* - ML), que modelam classificadores e regressores, amplamente utilizados principalmente devido às suas propriedades matemáticas que incluem boa capacidade de generalização e robustez. O modelo de treinamento das SVMs busca minimizar o risco empírico simultaneamente à capacidade de generalização. Entretanto, para obter modelos com boa precisão e baixa complexidade, é necessário definir o *kernel* e os seus parâmetros, assim como os parâmetros do modelo de treinamento. Os parâmetros do *kernel* e do modelo de treinamento juntos são denominados de hiperparâmetros do problema de seleção de parâmetros das SVMs. Porém minimizar a complexidade e maximizar a capacidade de generalização das SVM/SVRs são critérios contraditórios e, por isso, neste trabalho, o problema de seleção de parâmetros é modelado com um problema de otimização multiobjetivo (MOOP). Para resolver este problema, foi desenvolvida uma meta-heurística multiobjetivo chamada de *Adaptive Parameter with Mutant Tournament Multi-Objective Differential Evolution* (APMT-MODE), *Multi-Objective Particle Swarm Optimization* (MOPSO), objetivando resolver o problema de seleção de parâmetros. Os algoritmos desenvolvidos foram testados com um conjunto de treinamento para classificadores e regressores (obtidos no repositório *University of California*, Irvine), combinados com os *kernels* gaussiano, cauchy, polinomial e arco cosseno. Para validar as meta-heurísticas desenvolvidas, foi realizado o teste estatístico de Friedman e os testes *post hoc*, os quais mostraram que o algoritmo APMT-MODE é superior ao clássico algoritmo *Non Sorting Genetic Algorithm - II*. Além desses estudos, foram realizados estudos de comparação da complexidade computacional entre modelos de SVMs com diferentes *kernels*, nos quais os *kernels* gaussiano e polinomial configurados pelo APMT-MODE obtiveram melhor desempenho. Como aplicação em situações reais, o APMT-MODE foi empregado para obtenção de modelos para predição da penetração e largura de cordões de solda, que são utilizados como parâmetros para o controle de processo de soldas. A partir das análises, concluiu que os modelos gerados pelo APMT-MODE são mais eficientes que os encontrados para redes neurais do tipo perceptron. Finalmente, o *Grid Search* foi empregado para caracterizar os espaço de busca e a complexidade do problema de seleção de parâmetros modelado como um MOOP.

Palavras-chave: Meta-heurísticas. Máquinas de vetores de suporte. Aprendizagem de máquinas. Otimização multiobjetivo. Problema de seleção de parâmetros.

ABSTRACT

Support Vector Machines (SVMs) represent a Machine Learning technique (ML), which model classifiers and regressors, widely used mainly because of their mathematical properties that include good generalizability and robustness. The SVM training model seeks to minimize the empirical risk simultaneously to the generalization capacity. However, to obtain models with good precision and low complexity, it is necessary to define the kernel and its parameters, as well as the parameters of the training model. The parameters of the kernel and the training model together are called hyperparameters of the parameter selection problem of the SVMs. However, minimizing the complexity and maximizing the generalization capacity of the SVM/SVRs are conflicting criteria and, therefore, in this work, the problem of parameter selection is modeled with a multiobjective optimization (MOOP) problem. In order to solve this problem, a multiobjective meta-heuristic called Adaptive Parameter with Mutant Tournament Multi-Objective Differential Evolution (APMT-MODE) was developed, aiming at solving the problem of parameter selection. The algorithms developed were tested with a training set for classifiers and regressors (obtained from the University of California, Irvine repository), combined with the Gaussian, Cauchy, polynomial, and cosine arc textures. In order to validate the developed heuristics, the Friedman statistical test, and the post-hoc tests were performed, which showed that the APMT-MODE algorithm is superior to the classic Non-Sorting Genetic Algorithm-II algorithm. In addition to these studies, computational complexity was compared between models of SVMs with different kernels, in which the Gaussian and polynomial kernels configured by APMT-MODE obtained better performance than others considered algorithms. As an application in real situations, the APMT-MODE was used to obtain models for penetration prediction and width of weld beads, which are used as parameters for the process control of welds. From the analysis, it was concluded that the models generated by APMT-MODE are more efficient than those found for neural networks of the perceptron type. Finally, Grid Search was used to characterize the search space and complexity of the parameter selection problem modeled as a MOOP.

Keywords: Meta-heuristics. Support vectors machines. Machine learning. Multi-objective optimization. Parameters selection problem.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE SÍMBOLOS

Símbolos Latinos

K	Matriz ou função *kernel*
N	Cardinalidade do conjunto de treinamento
C	Parâmetro de regularização
$\mathbf{w}$	Vetor diretor do hiperplano classificador
$\mathbf{x}$	Vetor caracterí stica do dado de treinamento
b	*bias*

Símbolos Gregos

α_i	i-é sima variável Lagrangeana
ξ_i	i-é sima variável de folga do problema de otimização
Φ	Função de transformação
γ	Parâmetro do *kernel* gaussiano
β	Coeficiente do *kernel* polinomial
κ	Termo independente do *kernel* polinomial
π	Constante irracional pi
δ	Parâmetro do *kernel* sigmoide
ρ	Parâmetro do *kernel* sigmoide
ϵ	Largura do tubo ϵ-*insensitive*

Grupos Adimensionais

e	Número de Euler

Sobrescritos

α^*	Variável de decisão ótima
$\hat{f}$	Preditor, função aproximadora f
$\bar{y}$	Média da variável y

Siglas

AG	Algoritmo Genético
AP-MODE	*Adaptive Parameter Multi-Objective Differential Evolution*
APMT-MODE	*Adaptive Parameter with Mutant Tournament* - MODE
AR	Atrativo Repulsivo
AUC	*Area Under Curve*
CLO	*Criticism of Lexicographic Ordering*
DE	*Differential Evolution*
EA	*Evolutionary Algorithm*
FP	Falso positivo
FPGA	*Field Programmable Gates Arrays*
FSM	*Finite State Machines*
GS	*Grid Search*
GUIDE	*Grafical User Interface Development Environment*
H-C	Hermite-Chebyshev
H-C	Hermite-Gaussian
HDL	*Hardware Description Language*
KKT	Karush-Kuhn-Tucker
LEIA	*Laboratory of Embedded Systems and Integrated Circuits Applications*
MAC	*Multiply-Accumulate*
MODE	*Multi-Objective Differential Evolution*
MOOP	*Multi-objective Optimization Problem*
MOPSO	*Multi-Objective Particle Swarm Optimization*
MOUD	*Muti-objective uniform design*
MSE	*Mean Squared Error*
NIOTS	*Nature Inspired Optimization Tools for SVM*
NP-Completo	Não Polinomial Completo
NSGA-II	*Non Sorting Genetic Algorithm II*
OAO	*One Against One*
OBL	*Oposition Based Learning*
PF	*Pareto Front*
PD	Problema de Decisão
PO	Pareto Ótimo
PSM	Problema de Seleção de Modelos
PSO	*Particle Swarm Optimization*
PSP	Problema de Seleção de Parâmetros
RBF	*Radial Basis Function*

Siglas

RNA	Redes Neurais Artificiais
ROC	*Receiver Operating Characteristic*
SA	*Simulated Annealing*
SLHD	*Symmetric Latin Hypercube Design*
SMO	*Sequential Minimal Optimization*
SRM	*Structural Risk Minimization*
SV	*Support Vectors*
SVM	*Support Vectors Machine*
SVR	*Support Vectors Regressor*
T	Turing
TIC	Tecnologia da Informação e Comunicação
UCI	*University of California, Irvine*
VHDL	*Very High Description Language*
VC	Vapnik Chervonenkis

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DA PROPOSTA

Diante da crescente gama de informações disponíveis, selecionar uma informação específica, a partir de conjuntos de dados enormes, representa um grande desafio nos dias atuais, especialmente na Ciência da Computação e em diversas áreas da Engenharia. Nesse contexto, dados podem ser bancos de imagens, de sinais, ou qualquer medida que necessite de classificação e/ou predição. Nesse cenário, o reconhecimento de padrões apresenta-se como um processo de interesse para solucionar problemas de classificação, o qual envolve inferir um conjunto de regras definidas, ou seja, um algoritmo (THEODORIDIS et al., 2010).

Algoritmo é uma sequência finita de instruções corretamente definida que devem ser executadas em um intervalo de tempo e esforço finito (SIPSER, 2006). Uma visão mais formal sobre o tema liga o conceito de algoritmo a uma *Máquina de Turing*, associando um problema de decisão (PD) a um algoritmo (A) , sendo que, nesse caso, (A) resolve PD se, e somente se, pode ser computado em uma Máquina de Turing (T), atingindo um estado de parada para toda instância $I \in PD$ (SIPSER, 2006).

Uma restrição adicional usada para a definição de um algoritmo é que ele deve garantir a solução ótima para *PD* (SIPSER, 2006). Assim, quando um algoritmo não garante a solução ótima para um problema (garantindo apenas uma solução de boa qualidade), é dito ser uma *heurística* (XING; GAO, 2014). No contexto prático da solução de problemas, a maioria dos problemas somente podem ser resolvidos por algoritmos com complexidade computacional não polinomial (SIPSER, 2006). Nesse caso, na prática, são usadas soluções heurísticas para os mesmos. Por exemplo, para algumas tarefas, não existem algoritmos que, diante de determinada entrada, possam transformá-la em uma saída adequada, como, por exemplo, no caso do reconhecimento de textos manuscritos. Se, por um lado, cada pessoa possui uma caligrafia própria, por outro, um indivíduo alfabetizado, mesmo diante de variadas caligrafias, é capaz de ler e melhorar sua eficiência leitora com a experiência. Nesse exemplo, está intrínseco o conceito de aprendizagem. Para o reconhecimento de manuscritos, não existe um conjunto de regras explícitas, o que torna o algoritmo algo complexo de ser definido. Diante do exposto, tem-se uma introdução à definição de aprendizagem de máquina (ALPAYDIN, 2014).

No contexto geral, *aprendizagem de máquina* é o processo de aprender um conjunto de regras (associado a um algoritmo) a partir de exemplos (conjunto de treinamento). Ou seja, criar uma função capaz de generalizar, diante de entradas novas, que apresentam variações, como o caso da caligrafia de cada pessoa (KOTSIANTIS, 2007). Por exemplo, o reconhecimento de textos manuscritos consiste em um problema de classificação de símbolos (letras), ou mesmo, um conjunto delas (palavras) dentro de um conjunto que possui um padrão, mas com variações significativas.

Por outro lado, a aproximação de funções (regressão) consiste em um problema semelhante à classificação, diferente desta pelo fato de as saídas dos ditos *regressores* serem dados contínuos.

Nesse contexto, o reconhecimento de padrões é uma parte da ciência da computação cujo objetivo é a classificação de objetos em determinado número de categorias ou classes, em que os objetos podem ser imagens, sinais elétricos ou qualquer medida que necessita ser classificada (THEODORIDIS; KOUTROUMBAS, 2009).

As técnicas de reconhecimento de padrões são aplicadas em diversas atividades, tornando-as autônomas ou um suporte para tomada de decisões. Entre várias aplicações, citam-se: (a) reconhecimento de movimento dos dedos da mão (ANAM; AL-JUMAILY, 2014), (b) sequenciamento genético (ZIEN et al., 2000; GUYON et al., 2002), (c) reconhecimento de voz (GANAPATHIRAJU; HAMAKER; PICONE, 2004), (d) reconhecimento de textos manuscritos (AYYAZ; JAVED; MAHMOOD, 2016), (e) predição de falências de empresas (NAGARAJ; SRIDHAR, 2015; LU et al., 2015), e (f) detecção de falhas em rolamentos (JACK; NANDI, 2002), entre outras possíveis.

Diante do exposto, fica evidente a importância de se tornar um processo de reconhecimento de padrão manual e sujeito a falhas humanas em um processo autônomo confiável, seja ele associado à produção industrial, à análise clínica, científica ou comercial.

O problema de aprendizagem de máquinas consiste em vários subproblemas, geralmente trabalhados por meio de heurísticas, a saber: (a) identificação dos dados necessários; (b) pré-processamento dos dados; (c) definição do conjunto de treinamento; (d) seleção do algoritmo de treinamento; (e) seleção de parâmetros; e (f) avaliação com o conjunto de testes.

A qualidade da solução, de cada um desses subproblemas, influencia o resultado final da máquina gerada, nos quesitos capacidade de generalização e complexidade. Diante disso, tem-se como objetivo gerar uma *máquina* com máxima capacidade de generalização e com complexidade mínima, tornando a aprendizagem de máquina num conjunto de problemas de otimização (KOTSIANTIS, 2007).

Neste trabalho, foi escolhido o subproblema (d) com a técnica de *Support Vector Machines* (SVM), que utiliza os conceitos de *Structural Risk Minimization* (SRM) para maximizar a capacidade de generalização. Entretanto a qualidade dos resultados dessa técnica é fortemente dependente do subproblema (e) mencionado no parágrafo anterior (SMOLA; SCHÖLKOPF, 2004).

Nos últimos anos, a SVM é uma ferramenta que mostra resultados satisfatórios, principalmente quando comparada a Redes Neurais Artificiais (RNA), que se baseiam, diferentemente dessas, por possuir um processo de treinamento analítico com solução única, baseada em um modelo matemático (não sendo uma caixa-preta). Entretanto sua eficiência está intimamente ligada à definição dos seus parâmetros na fase de modelagem, que afeta complexidade e capacidade de generalização das SVMs (NIU et al., 2016). Portanto, definir os parâmetros do problema de treinamento é um problema complexo muitas vezes tratado por técnicas meta-heurísticas (COELLO; VELDHUIZEN, 2007).

Meta-heurística é a combinação do prefixos gregos *meta*, que significa nível superior ou avançado, e *heurístico*, que significa encontrar (YANG, 2013). As meta-heurísticas constituem um conjunto de metodologias que harmonizam uma interação entre processos de aperfeiçoamento local (refinamento da solução) e estratégias de alto nível com o propósito de criar um processo no qual seja capaz escapar-se de mínimos/máximos locais e, assim, fazer uma extensa exploração por meio do espaço das soluções (XING; GAO, 2014). Além disso, as meta-heurísticas não requerem conhecimento específico sobre o problema ou uma função objetiva contínua/diferenciável. Técnicas recentes bioinspiradas, como aquelas baseadas em algoritmos genéticos e otimização de enxames de partículas, produziram resultados satisfatórios em diferentes áreas da engenharia e da ciência da computação (COELLO et al., 2013; NOZ et al., 2010).

É importante ressaltar que o uso de meta-heurísticas é adequado mesmo em situações dinâmicas em que o objetivo ou as restrições estão mudando ao longo do tempo, seja por causas exógenas ou autoinduzidas, e os ajustes de parâmetros e medições de condicionamento físico são perturbados. O mesmo acontece quando o espaço de busca é multidimensional, multimodal ou fractal e não pode ser tratado por métodos tradicionais, especialmente aqueles que usam uma previsão global da análise de superfície local (BACK; FOGEL; MICHALEWICZ, 1997).

Muitos trabalhos encontram soluções para o problema de seleção de parâmetros, visando otimizar um único critério (geralmente a capacidade de generalização do modelo). Entretanto minimizar os critérios, tais como o erro e a complexidade dos modelos de SVM/SVRs (*Support Vector Regressors*-SVR) , simultaneamente, é um problema contraditório. Portanto, neste trabalho, o problema de seleção de parâmetros é modelado como um *problema de otimização multiobjetivo* em que a complexidade é medida pela: (1) *quantidade de vetores de suporte* (SV) e (2) a *capacidade de generalização pela métrica erro*.

Geralmente, nos métodos multiobjetivos, utilizam-se estratégias como *c-distance*, hipervolume, *ranking*, ou a ponderação das funções objetivo para gerar a fronteira de Pareto. A estratégia adotada, neste trabalho, para definir a melhor solução, é o *Criticism of Lexicographic Ordering* (CLO), que define qual função objetivo será considerada para decidir qual o melhor indivíduo entre os não dominantes (COELLO; VELDHUIZEN, 2007).

As meta-heurísticas geralmente são aplicadas em problemas que não têm um algoritmo ou heurística específico(a) que produza uma solução satisfatória; ou mesmo, quando não é possível implementar esta metodologia ótima. Estas técnicas têm como principal objetivo problemas combinatórios e NP-Completos (SUN; LAI; WU, 2016).

Neste trabalho, é proposta uma estratégia fundada nas meta-heurísticas *Multi-Objective Particle Swarm Optimization* (MOPSO) e *multi-Objective Differential Evolution* (MODE), a fim de obter os parâmetros do modelo de uma SVM, considerando o equilíbrio entre a complexidade mínima e a máxima capacidade de generalização (COELLO; VELDHUIZEN, 2007; ABRAHAM; JAIN, 2005; ZHAO et al., 2011; RIQUELME; LÜCKEN; BARAN, 2015).

1.2 DEFINIÇÃO DO PROBLEMA E MOTIVAÇÕES

As Máquinas de Vetores de Suporte têm obtido uma atenção considerável devido à sua eficiência e fundamentação teórica diante de outros algoritmos de aprendizagem em diferentes aplicações. Entretanto, para obter uma SVM de alto desempenho, deve-se escolher adequadamente o parâmetro de regularização e os parâmetros da função *kernel* apropriados a cada tipo de dados a serem classificados (MIRANDA et al., 2012). Assim, a função *kernel* é responsável por mapear os dados a um espaço de dimensão superior ao original, de forma a tornar os dados linearmente separáveis.

O *parâmetro de regularização* C e os parâmetros da função *kernel*, em conjunto, são denominados *hiperparâmetros* de uma SVM, sendo que o problema de ajustar esses parâmetros de modo a obter a melhor SVM (segundo determinados critérios) é conhecido como *Problema de Seleção de Modelo* (PSM) da SVM ou *Problema Seleção de Parâmetros* (PSP) da SVM (NARZISI, 2007).

O problema de seleção de modelo da SVM foi inicialmente abordado (segundo a literatura) por uma busca exaustiva, em que um parâmetro é fixado, e o outro variado, gerando, assim, uma grade de soluções, sendo a melhor delas escolhida como a solução ótima; Vale ressaltar que essa técnica é conhecida como *Grid Search* (GS) (NARZISI, 2007; FRIEDRICHS; IGEL, 2005a). Entretanto a inviabilidade computacional deste método é evidente. Nessa direção, um método outrora empregado é o *método do gradiente*. Contudo esse método também não é eficiente, tendo em conta que ele exige que as funções sejam diferenciáveis, condição que não é garantida para o problema de seleção de hiperparâmetros. Adicionalmente, o problema de seleção de modelo da SVM é multimodal, o que sugere que um algoritmo de gradiente pode rapidamente convergir para um mínimo/máximo local (NARZISI, 2007).

Assim, resolver o problema de seleção de modelo da SVM requer encontrar um compromisso aceitável entre reduzir a complexidade do modelo e, ao mesmo tempo, obter um modelo com alto nível de precisão. Algumas vezes, tem-se um modelo com uma ótima generalização, mas muito complexo em termos de tempo e espaço (NARZISI, 2007). Diante do exposto, o problema é modelado como um *Multi-objective Optimization Problem* (MOOP) por possuir, por definição, pelo menos dois critérios contraditórios a serem otimizados, especificamente: a complexidade e a capacidade de generalização.

O *parâmetro de regularização*, que é uma penalização para os erros cometidos na fase de treinamento da máquina, faz o balanço entre o superajuste e o sobreajuste da SVM (GOMES et al., 2012; SMOLA; VAPNIK, 1997). Uma SVM superajustada não possui boa capacidade de generalização para novos dados desconhecidos. Por outro lado, o subajuste gera modelos de baixa complexidade, mas que também não possuem boa capacidade de generalização. Adicionalmente, a função *kernel* também possui parâmetros a serem definidos que influenciam no desempenho da máquina, dependendo da função adotada, sendo que as funções *kernel* mais conhecidas são: (a) a polinomial, (b) a sigmoide e (c) a RBF Gaussiana (*Gaussian Radial Basis Function*) (LORENA; CARVALHO, 2003).

Um tema tratado também neste trabalho é a implementação em *hardware* das SMV/SVRs, mediante o desenvolvimento de arquiteturas do tipo *ad-hoc*, que podem ser descritas mediante Linguagens de Descrição de *Hardware* (*Hardware Description Language* - HDL), objetivando aplicações que exigem que as SVMs/SVRs sejam utilizadas em tempo real, em que o compromisso entre complexidade e precisão é fundamental.

Neste trabalho, as implementações em *hardware* serão direcionadas para sua implementação em *Field Programmable Gates Arrays* (FPGAs). Para resolver o problema, aplicam-se, então, os métodos de otimização, de modo a se obter o *conjunto de Pareto* das soluções do problema, permitindo, assim, ao projetista a possibilidade de optar pela solução que melhor atenda às limitações e às necessidades da aplicação em questão.

Entretanto a complexidade dos modelos SVM/SVRs está intimamente relacionado não apenas à quantidade de SV, mas também ao *kernel* empregado, pois a lei de formação da função pode envolver operações aritméticas, trigonométricas e outras funções transcendentais, desde que a função final atenda ao teorema de Merce. A complexidade das operações citadas não pode ser ignorada quando se comparam modelos de SVM/SVRs com *kernels* distintos, assim foi realizado um estudo, para avaliar a complexidade computacional desdes modelos, empregando como métrica o tempo necessário para executá-los em um dispositivo ARM (Raspberry Pi 3).

Para validar o desempenho das meta-heurísticas desenvolvidas (APMT-MODE, AP-MODE e MOPSO), foram realizados testes estatísticos de Friedman e os testes *post hoc* de Nemenyi, Shaffer e Bergmann, todos recomendados para testes comparativo do tipo $N \times N$ recomendados por Derrac *et al.* em (DERRAC et al., 2011). Em outros trabalhos, desenvolvidos no *Laboratory of Embedded Systems and Integrated Circuits Applications* (LEIA), em que é necessária a comparação de algoritmos meta-heurísticos para resolver problemas, como, por exemplo, o controle preditivo, a alocação de tarefas em multiprocessadores, a autolocalização de robôs autônomos e outros; a metodologia de análise estatística tem apresentado bons resultados.

1.3 HIPÓTESES DA PROPOSTA

Trabalhos que tratam o problema de seleção de parâmetros de *kernels* adotam, em sua maioria, uma abordagem mono-objetivo (ZHANG et al., 2017; THARWAT; MOEMEN; HASSANIEN, 2017; YE; LOU; SUN, 2017; SUBASI; KEVRIC; CANBAZ, 2017; FARIS et al., 2018b). Uma abordagem mono-objetivo facilmente pode levar a soluções subótimas tendo em conta que a superfície de busca da solução é multidimensional e multimodal.

Assim, uma **primeira hipótese** adotada neste trabalho é que uma abordagem multiobjetivo é apropriada para este tipo de problema, se for previamente sintonizada nas suas estratégias de busca, assim como nos parâmetros que as meta-heurísticas costumam utilizar. Nesse sentido, alguns poucos trabalhos têm reportado na literatura um esforço para utilizar técnicas multiobjetivos para o problema de seleção de parâmetros das SVMs.

Por exemplo, no trabalho de Igel (IGEL, 2005), é proposto um algoritmo evolutivo autoadaptativo, que atualiza indivíduos baseado em uma distribuição normal $N(0, 1)$, que não se altera durante o processo evolutivo. O algoritmo é o primeiro autoadaptativo, reportado na literatura. O mesmo usa um ordenamento não dominante, chamado de NSES (IGEL, 2005). O NSES foi desenvolvido para emprego do *kernel* gaussiano para classificadores e destaca o estudo das possíveis funções objetivo do MOOP. Igel (2005) propõem outro trabalho, o qual modela o problema de seleção de parâmetros com três funções objetivos e emprega o mesmo algoritmo de Igel (2005) em *benchmarks*. Nesse caso, as funções objetivo propostas ao *Non-dominated Sorting Genetic Algorithm II* (NSGA-II), para classificar a presença de pedestres em imagens de carros autônomos.

Em Chatelain et al. (2007), o NSGA-II é empregado para resolver o problema de seleção de parâmetros das SVMs, em que as funções objetivos consideradas são os Falsos Positivos (FP) e Falsos Negativos (FN) , ou seja, não avalia complexidade da SVM. No trabalho de Chatelain et al. (2007), as variáveis de decisão do problema é o γ do *kernel* Gaussiano e dois parâmetros de regularização, pois trata cada classe independentemente.

O algoritmo *Muti-objective uniform design* (MOUD), proposto por Li, Liu e Gong (2011), emprega a técnica *Uniform Design*, sendo esta semelhante ao *Symmetric Latin Hypercube Design* (SLHD) . O MOUD não considera complexidade das SVM, mas a sensibilidade e a especificidade como critérios multiobjetivos para obter os hiperparâmetros ótimos das SVMs, assim como o *kernel* utilizado é o gaussiano e o polinomial.

Miranda et al. (2012) desenvolveram um algoritmo híbrido em que uma técnica *Meta-Learning* (ML), baseada na características dos problemas, possa gerar um enxame de soluções de boa qualidade como enxame inicial para o MOPSO. Esse MOPSO emprega o *kernel* Gaussiano para gerar classificadores e os critérios do MOOP são $\#SV$ e a precisão. Uma restrição deste MOPSO é que o espaço de busca foi discretizado limitando-o a 399 diferentes configurações de C e γ os hiperparâmetros adotados e consequentemente comprometendo a qualidade dos modelos. Miranda et al. (2014) propõem um outro trabalho com modificações no processo de obtenção do ML e no MOPSO .

O NSGA-II também é empregado no trabalho de Narzisi (2008), em que os critérios adotados são $\#SV$ e a precisão. Narzisi G. realiza testes com o *kernel* gaussiano e sigmoide, para tratar dados não linearmente separáveis.

Os algoritmos multiobjetivos citados nos parágrafos anteriores (*vide* Tabela 4.1) utilizam, em sua maioria, o NSGA-II, com *kernel* Gaussiano para desenvolver classificadores, ou seja, uma subcategoria do problema. Deve-se observar que, para diferentes *kernels*, se tem PSP com características distintas.

Nesse contexto, nenhum dos estudos encontrados na literatura abordam O PSP das SVRs, o que evidência a necessidade de um estudo mais criterioso sobre o assunto, empregando diferentes *kernels* e algoritmos que resolvam o problema de seleção de parâmetros, como um MOOP. Dessa maneira, tem-se, assim, a **segunda hipótese** do trabalho.

Adicionalmente, nenhum trabalho anterior considerou a complexidade dos *kernels* em termos de operações matemáticas. Este quesito tem um impacto direto no desempenho na execução da SVM/SVR em tempo de execução. Especialmente, quando se requeira a implementação em *hardware* do classificador/regressor a complexidade, em termos de quantidade e complexidade de operações matemáticas (do tipo soma-substração, multiplicação/divisão e operações trigonométricas/exponenciais), é determinante no custo da arquitetura, em termos de recursos de *hardware* (por exemplo no FPGA) e consumo de energia.

Assim, a **terceira hipótese** abordada aqui é a possibilidade de se tratar do problema de complexidade dos *kernels*, quando o objetivo é seu(s) mapeamento(s) em *hardware*, pois, com parâmetros adequados *kernels* clássicos, como o Gaussiano, o polinomial e o Cauchy podem gerar modelos de SVM/SVRs com boa precisão e que requerem poucos recursos de *hardware*.

A **quarta hipótese** refere-se à natureza do PSP, dado que, para cada *kernel* e cada conjunto de treinamento, tem-se um problema distinto e, portanto, as meta-heurísticas desenvolvidas, no passado, em outros trabalhos tratam o PSP mono-objetivo, geralmente, com um único *kernel*. Entretanto, com o emprego de técnicas de adaptação de parâmetros como nas meta-heurísticas APMT-MODE e AP-MODE, tem-se a possibilidade de encontrar soluções de boa qualidade para vários *kernels* combinados com diversos conjuntos de treinamento.

1.4 OBJETIVOS

Os objetivos do trabalho são divididos em *objetivo geral*, que apresenta o foco principal deste estudo, e *objetivos específicos*, que são resultados a serem alcançados no processo de atingir o objetivo geral.

1.4.1 Objetivo geral

O objetivo geral é desenvolver meta-heurísticas com a finalidade de sintonizar os hiperparâmetros das SVMs, ou seja, encontrar soluções para o problema de seleção de parâmetros primando otimizar a complexidade e capacidade de generalização simultaneamente, gerando um conjunto de Pareto que permitirá ao projetista sem conhecimento aprofundado do problema escolher soluções de boa qualidade para aplicação em questão.

1.4.2 Objetivos específicos

Os objetivos específicos do trabalho são:

(a) Desenvolver uma ferramenta que tenha como entrada os dados de treinamento e saída o conjunto de Pareto dos hiperparâmetros da SVM, assim como seus respectivos modelos, sendo esta capaz de resolver diferentes problemas da engenharia.

(b) Definir algoritmos meta-heurísticos adequados e técnicas associadas para determinação do conjunto de Pareto, no problema de seleção de hiperparâmetros.

(c) Definir quais métricas para avaliar os algoritmos desenvolvidos são mais eficientes.

(d) Avaliar os custos das operações aritméticas de *kernels* considerados clássicos.

(e) Aplicar as meta-heurísticas para resolver o PSP de SVM/SVRs em *benchmarks* sugeridos pela literatura.

(f) Empregar a ferramenta desenvolvida para resolver um problema prático da engenharia.

(g) Definir as perspectivas da aplicabilidade das técnicas SVM/SVRs para sistemas embarcados os quais têm limitações nos recursos computacionais, no desempenho e no consumo de potência.

1.5 ASPECTOS METODOLÓGICOS DO TRABALHO

Para alcançar os objetivos citados na seção anterior, adotaram-se os seguintes procedimentos metodológicos:

(a) Levantamento bibliográfico a respeito das SVMs, identificando suas características, aplicações e problemas associados ao tema.

(b) Análise de diferentes estratégias de modelagem do PSP como um MOOP.

(c) Levantamento bibliográfico e análise a respeito de técnicas meta-heurísticas empregadas de resolução do MOOP.

(d) Implementação dos algoritmos meta-heurísticos e estratégias complementares a estes, tais como métodos de inicialização, diferentes tipos de distribuição de números aleatórios, técnicas de adição de diversidade e técnicas adaptativas.

(e) Validação das técnicas por meio de testes realizados com problemas gerados pelas combinação dados de treinamento *benchmarks* e diferentes *kernels*, sendo estes resultados validados com testes de hipótese e significância estatística.

(f) Desenvolvimento de um sistema que simplifique a execução dos testes e obtenção dos dados para testes estatísticos, assim como viabilizar a análise dos modelos de SVM/SVRs.

(g) Para alcançar o objetivo de gerar modelos de SVM/SVRs com baixa complexidade e boa capacidade de generalização, foram abordados trabalhos sobre *kernels*, tendo em vista a influência que estes têm sobre a qualidade dos modelos de SVM/SVRs.

(h) Definir metodologia de análise de diferentes tipos de *kernels*, tendo em vista a diversidade das operações matemáticas que podem ser empregadas para definir um *kernel*.

(i) Implementação em *hardware* dos modelos de SVM/SVRs, obtidos pela ferramenta desenvolvida, para análise dos recursos computacionais, do desempenho e do consumo de potência.

1.6 CONTRIBUIÇÕES DAS PROPOSTAS

O estudo tem como principal contribuição a definição de técnicas e modelos que possibilitem escolher os parâmetros de uma SVM de modo a definir um compromisso ótimo entre o erro empírico e o erro generalização, tendo em vista que os parâmetros não podem ser definidos *a priori*, pois cada conjunto de dados determina um problema de natureza diferente.

Como principais contribuições deste trabalho, citam-se: (a) um novo algoritmo denominado *Adaptive Parameter with Mutant Tournament Multi-objective Differential Evolution* - APMT-MODE capaz de encontrar boas soluções para o problema de seleção de parâmetros SVM/SVRs para quatro *kernels* distintos; (b) a aplicação da estratégia *Criticism of Lexicographic Ordering* - CLO, para transformar o problema de seleção de parâmetros em MOOP, definindo qual seria a melhor solução pertencente ao conjunto de soluções não dominantes; (c) a aplicação da estratégia CLO para definir a melhor partícula individual e global MOPSO; e (d) a validação dos algoritmos desenvolvidos pelo Teste de Friedman e *post hoc*, tendo como problema a combinação de *benchmarks* e *kernels*, de modelos SVM/SVRs.

Os programas desenvolvidos neste estudo são úteis no estudo das SVMs para aplicações específicas, como, por exemplo, controle preditivo não linear, modelo de controle de corrente no processo de solda, predição climática, como modelos para implementação em *hardware* ou como referência para o desenvolvimento de outros algoritmos envolvidos e novas teorias. Nesse sentido, são testadas métricas de otimização alternativas, pois estas desempenham papel importante no processo de otimização e influenciam fortemente no processo de busca meta-heurística.

O sistema denominado NIOTS (*Nature Inspired Optimization Tool for SVMs*) é composto pelos ambientes: (a) *Optimization*; (b) *Ensembles*; (c) *Prediction*; e (d) *Statistics*. O ambiente *Optimization*, por sua vez, consiste na implementação dos algoritmos de seleção de parâmetros APMT-MODE, *Adaptive Parameter - Multi-objective Differential Evolution* - AP-MODE, MOPSO e GS, juntamente com a possibilidade de escolha do *kernel*, a ser aplicado para criação do modelo, estratégias de adição de diversidade, quantidade de repetição do experimento e demais parâmetros inerentes do algoritmo em questão.

No ambiente *Ensembles*, é possível criar *ensembles* com dois, três ou quatro modelos oriundos dos conjuntos soluções obtidas pelos algoritmos de otimização. Já, no ambiente *Prediction*, é possível gerar gráficos com os modelos gerados pelo NIOTS. E, finalmente, no ambiente *Statistics*, é possível fazer análise de convergência baseada nas métricas de avaliação dos conjuntos não dominantes obtidos em cada iteração, assim como comparações estatísticas baseado no ANOVA (caso os dados sejam normais) ou teste não paramétrico de Wilcoxon.

No ambiente *Optimization*, do NIOTS, destaca-se o algoritmo desenvolvido APMT-MODE, pois sua característica adaptativa combinada com a estratégia de mutação torneio mostrou-se mais eficiente que o clássico NSGA-II e os demais algoritmos implementados.

1.7 ESTRUTURA DO TEXTO

O texto está divido em mais oito capítulos. No Capítulo 2, são definidos conceitos a respeito de Máquina de Vetores de Suporte, partindo do caso em que os dados são linearmente separáveis e, em seguida, para dados não linearmente separáveis. A definição e as condições para que os *kernels* possam ser usados em SVMs não lineares são discutidas no Capítulo 2.

No Capítulo 3, é descrito o PSP modelado como um MOOP, o algoritmo meta-heurístico *Particle Swarm Optimization* - PSO e *Differential Evolution* - DE e as adaptações que eles sofreram para tratar o problema de seleção de parâmetros como MOOP.

O estado da arte sobre os algoritmos MODE e MOPSO e algoritmos para resolução do problema de seleção de parâmetros é apresentado no Capítulo 4. O estado da arte fomenta a discussão de como esses problemas têm sido tratados nos últimos anos.

O Capítulo 5 descreve os algoritmos desenvolvidos e as adaptações realizadas para tratar o PSP como um MOOP, assim como descreve as funções objetivo e as métricas por elas empregadas.

O Capítulo 6 descreve cada ambiente do sistema NIOTS (sistema desenvolvido), suas características e funcionalidades.

No Capítulo 7, *benchmarks* de classificação e regressão foram aplicados aos algoritmos desenvolvidos, implementados no *Nature Inspired Optimization Tools for SVM* - NIOTS, e comparados com o algoritmo *Non Sorting Genetic Algorithm* - NSGA-II, utilizando testes estatísticos, com objetivo de verificar a eficiência dos algoritmos MOPSO, APMT-MODE e AP-MODE.

Finalmente, no Capítulo 8, são apresentadas as conclusões obtidas a partir dos resultados encontrados e dos testes realizados no Capítulo 7.

2 MÁQUINAS DE VETORES DE SUPORTE

As SVMs possuem um modelo matemático para obtenção dos pesos que consiste em um *Problema de Otimização Quadrático*. Como resultado é obtida uma função classificadora (ou aproximadora) com pesos esparsos. Neste capítulo, são apresentados os conceitos necessários para determinar os pesos. Adicionalmente, são discutidas as demais definições associadas aos problemas de aprendizagem de máquina, como, por exemplo, capacidade de generalização e risco empírico.

2.1 INTRODUÇÃO

O problema de classificação é inerente à natureza humana, tendo em conta que, desde o nosso primeiro contato com o mundo, diferenciamos nossos pais de outras pessoas, o que gostamos de comer ou não, o ambiente em que nos encontramos, entre outros. Infelizmente, esse processo não é tão simples quanto parece, considerando que os objetos possuem muitas características para serem analisadas, e estas podem possuir semelhanças entre si, dificultando a classificação.

Atualmente, com o grande avanço da Tecnologia da Informação e Comunicação - TIC, a classificação de dados tornou-se extremamente importante, sendo que as SVMs representam uma metodologia que fornece bons resultados a problemas reais. Entre as várias aplicações, pode-se citar a predição de falência de empresas (SHIN; LEE; KIM, 2005; LU et al., 2015), o reconhecimento de voz (GANAPATHIRAJU et al., 2004), o reconhecimento de manuscritos (HSU; LIN, 2002; HOFMANN; SCHöLKOPF; SMOLA, 2008), a identificação de sequências de proteínas (ZIEN et al., 2000), a análise de genes para identificação de câncer (GUYON et al., 2002), a detecção de falhas (JACK; NANDI, 2002), entre outras (JAFARI et al., 2017; AHMAD et al., 2014; MASTINU et al., 2017).

As SVMs são obtidas a partir de um subconjunto finito de dados conhecidos e de seus respectivos rótulos (conjunto de treinamento), que, por meio de uma técnica de indução, buscam classificar dados que não fazem parte do conjunto de treinamento. Portanto, a SVM é uma função induzida a partir de um conjunto de treinamento, capaz de separar os dados em classes, baseada na informação fornecida pelo conjunto de treinamento (LIMA, 2004; BURGES, 1998).

O processo de aprendizagem ocorre dependendo da forma como o algoritmo de indução atua sobre o conjunto de treinamento. Este processo é dividido em três paradigmas de aprendizagem: (a) *supervisionada*, (b) *não supervisionada* e (c) *por reforço*. Na aprendizagem supervisionada, o processo ocorre com auxílio de um professor externo, que possui conhecimento sobre os dados de treinamento e seus respectivos rótulos e atua avaliando o erro cometido e corrigindo-os, aprimorando a função classificadora em cada etapa. No processo não supervisionado, os dados não são

rotulados e, portando, não existe um conhecimento inicial das classes, nesse caso, os dados são classificados por afinidade. O aprendizado por reforço é uma técnica que permite que o modelo seja aproximado do ideal, por meio da aprendizagem baseada em punição e recompensa, sendo realizada pela interação entre a técnica de treinamento e o seu ambiente. Nesse contexto, a função das SVMs é induzida pelo paradigma supervisionado.

A eficiência de uma função classificadora é medida tanto pela complexidade quanto pela capacidade de generalização. A *capacidade de generalização* é a habilidade da função em classificar corretamente dados que não pertencem ao conjunto de treinamento, enquanto que a complexidade se refere à quantidade de elementos que são necessários para compor a função, como, por exemplo, os pontos do conjunto de treinamento das SVMs, os centros das redes RBFs, os neurônios na camada escondida nas RNAs. Neste trabalho, entende-se por *complexidade* a quantidade de pontos do conjunto de treinamento necessários para compor a função classificadora ou aproximadora.

Entretanto minimizar a complexidade e maximizar a capacidade de generalização são objetivos divergentes ou contraditórios e, portanto, o que se busca nestas máquinas é o compromisso entre esses dois quesitos, tendo em vista a aplicação prática que elas terão.

Conforme Smola et al. (1999), entre as principais características de uma SVM, citam-se as seguintes:

- Grande capacidade de generalização.
- Robustez para dados de grande dimensões, como, por exemplo, imagens.
- O problema de otimização que modela o treinamento é quadrático e convexo, o que garante uma solução ótima única para o problema de treinamento.

O conjunto de treinamento é composto por $X = \{(\mathbf{x}_i, y_i)\}$, $i = 1, 2, ..., N$ tal que $\mathbf{x}_i \in \mathbb{R}^n$ e $y_i \in \{-1, 1\}$, em que $\mathbf{x}_i$ é o vetor das características, y_i os rótulos e N a cardinalidade de X.

A função $f(\mathbf{x}_i, \alpha)$ é uma máquina treinada a partir do conjunto X, sendo α o parâmetro de ajuste da função obtido no processo de treinamento e $f(\mathbf{x}_i, \alpha)$ o rótulo atribuído ao vetor $\mathbf{x}_i$ pelo modelo. Para vetores que não pertencem ao conjunto X, o erro cometido pela máquina é calculado por 2.1

$$R(\alpha) = \frac{1}{2} \int c(f(\mathbf{x}, \alpha), y) dP(\mathbf{x}, y), \tag{2.1}$$

em que c é uma métrica para a diferença entre $f(\mathbf{x}, \alpha)$ e y, $R(\alpha)$ é denominado *risco esperado* ou simplesmente *risco*. Entretanto não é possível calcular o risco esperado, pois a distribuição de probabilidade $P(\mathbf{x}, y)$ é desconhecida, assim o que se pode calcular é o *risco empírico* definido pela Equação 2.2

$$R_{emp}(\alpha) = \frac{1}{2N} \sum_{i=1}^{N} c(f(\mathbf{x}_i, \alpha), y_i). \tag{2.2}$$

O risco empírico $R_{emp}(\alpha)$ é determinado sobre o conjunto de treinamento por meio da técnica de *cross-validation* ou por meio de um conjunto de vetores característico com rótulos conhecidos, mas que não pertencem ao conjunto de treinamento.

No treinamento de uma máquina, é importante conhecer o risco esperado, mas, como a distribuição de probabilidade é desconhecida, define-se um limite superior para o risco com probabilidade $\eta - 1$ tal que $0 < \eta < 1$. Este limite superior é estimado pela Equação 2.3

$$R(\alpha) \leq R_{emp} + \sqrt{\left(\frac{h(log(2N/h) + 1) - log(\eta/4)}{N}\right)}. \tag{2.3}$$

Implícito à Equação 2.3 está o conceito de dimensão de Vapnik Chervonenkis - VC definido como: dado um conjunto X de vetores de características com cardinalidade N e $y_i \in \{-1, 1\}$, então existe 2^N formas de atribuir rótulos ao conjunto X. A classe de funções $\{f(\mathbf{x}, \alpha_j)\}$ é dita ter dimensão VC igual a N se esta classe é capaz de classificar todas as 2^N possibilidades de rotular X (CORTES; VAPNIK, 1995).

A variável h na Equação 2.3 é a dimensão de VC, sendo relacionada com a complexidade da SVM e, consequentemente, com sua capacidade de classificação. A parcela que soma ao risco empírico na Equação 2.3 é denominada de *termo de capacidade*.

A minimização do limite superior do risco esperado é conhecido como *minimização do risco estrutural*. Entretanto minimizar o risco estrutural envolve objetivos contraditórios (risco empírico e termo de capacidade), quanto menor o risco empírico, maior o termo de capacidade e vice-versa. A Figura 2.1 ilustra a relação existente entre o limite do risco esperado, o termo de capacidade e o risco empírico (LORENA; CARVALHO, 2007).

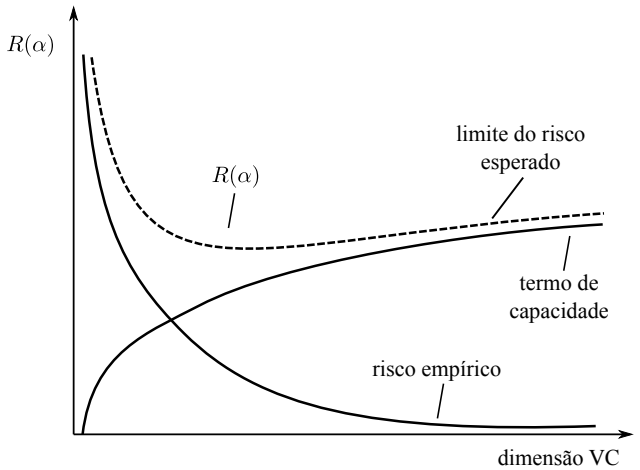


Figura 2.1 – Princípio de minimização do risco estrutural, adaptado de Lorena e Carvalho (2007).

Um modelo com risco empírico mínimo para um dado conjunto de treinamento não garante que este tenha boa capacidade de generalização, esse fenômeno é conhecido como *sobreajuste* ou, em inglês, *overfitting*. Um modelo sobreajustado perde a capacidade de generalização, pois este assimila, durante o processo de treinamento, ruídos e informações indesejadas oriundas de

X. O contrário é chamado de *subajuste* ou, em inglês, *underfitting*, ou seja, o risco empírico não atende às restrições do problema no conjunto de treinamento.

Especificamente, as SVMs separam os vetores de características por hiperplanos e o Teorema 2.1.1 apresenta a condição necessária e suficiente para que uma classe de hiperplanos seja capaz de separar um conjunto finito de pontos (BURGES, 1998).

Os conceitos da Figura 2.1 são descritos pelo Teorema 2.1.1 de acordo com (BURGES, 1998):

Teorema 2.1.1 *Considere um conjunto de m pontos em $\mathbb{R}^n$ e defina qualquer um dos N pontos como origem. Então, os m pontos podem ser separados por hiperplanos se, e somente se, a posição dos pontos restantes são linearmente independentes* [1].

A partir do Teorema 2.1.1, pode ser definido o seguinte corolário (BURGES, 1998):

Corolário 2.1.2 *A dimensão VC de um conjunto de hiperplanos orientados em $\mathbb{R}^n$ é $n+1$, assim pode-se sempre escolher $n+1$ pontos e, então, escolher um dos pontos como origem, tal que a posição dos n pontos restantes são linearmente independentes, mas nunca pode se escolher $n+2$ pontos* [1].

A dificuldade em se obter uma função $f(\mathbf{x}, \alpha)$ que minimize o risco estrutural está no fato de que minimizar o risco empírico e o termo de capacidade são contraditórios. As SVMs buscam minimizar simultaneamente o risco empírico e o termo de capacidade por meio da escolha de um hiperplano de margens largas, ou seja, define um plano separador com distância máxima dos vetores características mais próximos no conjunto de treinamento (LIMA, 2004).

O modelo empregado no treinamento da SVM para determinar tal hiperplano consiste em um problema de programação quadrática convexo e, portanto, possui solução ótima única, podendo ser calculada por um método exato, diferente das redes neurais do tipo *Perceptron*, que determina a função separadora por métodos heurísticos, não tendo como garantia o ótimo global (LUTS et al., 2010; CORTES; VAPNIK, 1995).

No caso de os dados serem linearmente separáveis em seu espaço de origem, as SVMs são denominadas de margens rígidas por não haver a necessidade de permitir erros nos dados de treinamento. Dados de aplicações reais geralmente são não linearmente separáveis, logo, nessa situação, empregam-se SVMs de margens suaves e o *kernel trick* para gerar o modelo adequado. Na Seção 2.2, serão tratadas as duas situações iniciando pelas SVMs lineares por questões didáticas.

2.2 MÁQUINAS DE VETORES DE SUPORTE - CLASSIFICADORES

As SVMs têm como objetivo definir um hiperplano que separe os dados de treinamento, garantindo uma boa generalização. Assim é introduzido o conceito de *margens*, que são hiperplanos

[1] A demonstração pode ser encontrada no apêndice de Burges (1998).

paralelos ao hiperplano classificador, sendo estes definidos por pontos, com propriedades inerentes ao conjunto de treinamento denominados de *vetores de suporte*. Para que a generalização seja a melhor possível, as margens devem ser definidas de tal forma que a distância entre elas seja máxima com o mínimo de risco empírico (LIMA, 2004).

Por questões didáticas, o modelo matemático das SVMs foi dividido em duas categorias: (a) linearmente separáveis e (b) linearmente não separáveis. As SVMs linearmente separáveis dividem-se em: (a) SVMs de margens rígidas e (b) SVMs de margens suaves. Entretanto, em aplicações reais, as SVMs linearmente não separáveis são transformadas em margens suaves com o uso de *funções kernel*, descritas na Subseção 2.2.3.

2.2.1 SVMs - Linearmente Separáveis

Um conjunto de treinamento X é denominado linearmente separável se existe um hiperplano que separe suas classes sem qualquer erro. A SVM, nesse caso, define o hiperplano classificador com margens mais distantes possíveis, minimizando o erro de generalização, para este conjunto de treinamento.

Na Figura 2.2, as retas tracejadas H_1 e H_2 são as margens da reta classificadora F, e os pontos que satisfazem as equações das margens são denominados SV. O objetivo é identificar quais são os vetores de suporte que definam margens de largura máxima que separem as classes, sem qualquer tipo de erro, nessas condições, as margens são denominadas de *margens rígidas* (CORTES; VAPNIK, 1995).

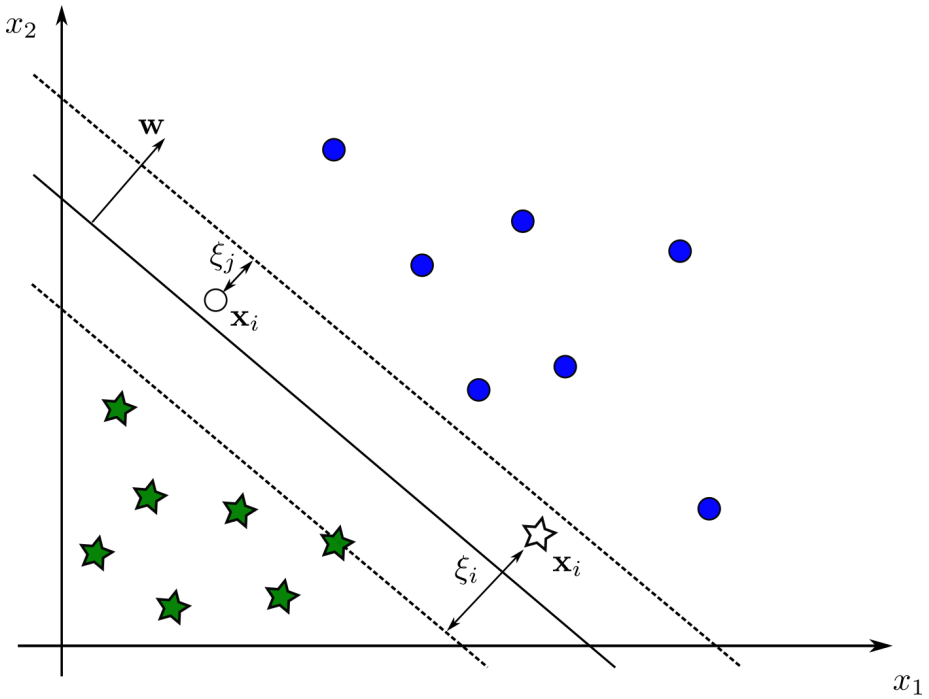


Figura 2.2 – SVM - Margens rígidas.

Os pontos que estão sobre as margens satisfazem a Equação 2.4. Esta equação representa a restrição do problema de treinamento e garante que não haja erro empírico na classificação.

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0. \tag{2.4}$$

No processo de treinamento, busca-se maximizar a distância entre as margens H_1 e H_2 dada pela Equação 2.5.

$$d(H_1, H_2) = \frac{2}{\|\mathbf{w}\|}. \tag{2.5}$$

As margens máximas são calculadas minimizando $\|\mathbf{w}\|$ na Equação 2.5, sujeito à restrição da Equação 2.4, gerando, assim, o problema de otimização primal 2.6 nas variáveis $\mathbf{w}$ e b.

$$\begin{aligned} &\textbf{Min} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &s.a. \\ &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0 \qquad i = 1, ..., N, \end{aligned} \tag{2.6}$$

em que N é a cardinalidade do conjunto de treinamento.

O Problema de Otimização 2.6 possui dois tipos de variáveis distintas e N restrições, que tornam o problema complexo de ser resolvido, para simplificar o modelo de treinamento, empregam-se os multiplicadores de Lagrange e o teorema de Karush-Kuhn-Tucker - KKT para transformar o problema primal no seu correspondente dual (GRIVA; NASH; SOFER, 2008). Segundo o teorema da dualidade fraca, os dois problemas têm a mesma solução ótima e, nesse caso, o problema dual possui conceitos que permitem a sua generalização para conjuntos com dados não linearmente separáveis. O Problema de Otimização 2.7 é conhecido como dual de Wolf, nas variáveis α_i.

$$\begin{aligned} &\textbf{Max} \quad L_d(\alpha_i) = \textstyle\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ &\text{s.a} \\ &\textstyle\sum_{i=1}^{N} \alpha_i y_i = 0 \\ &\alpha_i \geqslant 0 \qquad\qquad i = 1, ..., N, \end{aligned} \tag{2.7}$$

em que α_i são os multiplicadores de Lagrange, ou seja, as variáveis do problema de otimização, $\mathbf{x}$ e y são os vetores característicos e os rótulos de X respectivamente.

No problema definido por 2.7, a função L_d é uma função quadrática e a única restrição linear, ambas convexas, sendo que esta característica é condição necessária e suficiente (segundo o teorema de KKT) para solução ótima única. Essa propriedade garante que o classificador obtido por este método seja o ótimo global, dado um conjunto de treinamento e os parâmetros do modelo (GRIVA; NASH; SOFER, 2009).

Ao resolver o problema 2.7, têm-se os valores ótimos de α_i^*. Uma vez conhecidos os α^*,

calcula-se o $\mathbf{w}^*$ a partir da Equação 2.8.

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i. \tag{2.8}$$

O b^* é calculado implicitamente isolando-o na equação $\alpha_i^*[y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1] = 0$; sendo que as margens são equidistantes do hiperplano classificador e o termo b^* determina uma translação em relação à origem, toma-se sua média, assim tem-se:

$$b^* = \frac{1}{\#SV} \sum_{i=1}^{\#SV} \left(\frac{1}{y_i} - \mathbf{w}^* \cdot \mathbf{x}_i \right). \tag{2.9}$$

Usando os resultado das equações 2.9 e 2.8, obtém-se a função classificadora dependente dos α^*, cujos índices pertencem ao conjunto $SV = \{\alpha^* : \alpha^* \neq 0\}$, dada pela Equação 2.10

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{\#SV} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \right). \tag{2.10}$$

A função 2.10 é capaz de classificar corretamente todos os dados do conjunto de treinamento com a melhor generalização possível, mas, no caso de os problemas não serem linearmente separáveis, pode-se relaxar as restrições do Problema de Otimização 2.6. Essas SVMs são denominadas de margens suaves, as quais são descritas na Subseção 2.2.2.

2.2.2 SVMs de Margens Suaves

Em situações em que os dados são não linearmente separáveis, mas que podem ser divididos em duas classes, por um hiperplano, adota-se um classificador que aceita tais erros. Estes erros são controlados por um parâmetro denominado *parâmetro de regularização*, sendo esses classificadores conhecidos como *SVMs de margens suaves*.

No Problema 2.6, adiciona-se a variável de folga ξ_i, tornando o modelo tolerável a erros empíricos, assim como a dados entre as margens, possibilitando, desse modo, que as SVMs classifiquem dados não linearmente separáveis. Na Figura 2.2, a estrela sem preenchimento é um exemplo de erro empírico penalizada por ξ_i, enquanto que o círculo sem preenchimento representa um caso de dado classificado corretamente entre as margens penalizado por ξ_j. O problema de otimização primal para SVMs de margens suaves é dado pela Equação 2.11 (CORTES; VAPNIK, 1995)

$$\begin{aligned} &\text{Min} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C \textstyle\sum_{i=1}^{N} \xi_i \\ &s.a. \\ &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) + \xi_i - 1 \geqslant 0 \\ &\xi_i \geqslant 0. \end{aligned} \tag{2.11}$$

Na Equação 2.11, C é o parâmetro de regularização, que pondera o erro empírico cometido pela máquina.

O parâmetro de regularização (C) é um parâmetro a ser determinado pelo projetista da máquina, sendo que a escolha correta do valor desse parâmetro determina o compromisso que uma SVM possui entre a sua complexidade e a capacidade de generalização. Caso o valor de C tenda ao infinito, a SVM obtida se aproxima do modelo de margens rígidas, valorizando dados muito específicos ou mesmo ruídos, comuns em dados reais. Por outro lado, se o valor de C tende a zero, a SVM permitirá tantos erros que não será capaz de classificar os dados, tendo baixa capacidade de generalização.

O problema de otimização dual das SVMs com margens suaves pode ser obtido de maneira similar às SVMs de margens rígidas. Os cálculos detalhados para obter o problema dual 2.12 encontram-se no Apêndice A.

$$\begin{array}{ll} \text{Max} & L_d(\alpha_i, \xi_i) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{s.a} & \\ \sum_{i=1}^{N} \alpha_i y_i = 0 & \\ \xi_i \geqslant 0 & i = 1, ..., N \\ 0 \leqslant \alpha_i \leqslant C & i = 1, ..., N. \end{array} \tag{2.12}$$

A função objetivo assim como as restrições do problema da Equação 2.12 são contínuas, diferenciáveis e convexas. Nesse caso, as condições de KKT são necessárias e suficientes para existência de ponto ótimo único, *vide* Apêndice A (GRIVA; NASH; SOFER, 2009). Essas características do problema, juntamente com as condições de KKT, garantem que a SVM, obtida a partir do conjunto de treinamento (e definido um parâmetro de regularização), seja ótima global. Nesse contexto, fica em aberto a escolha do parâmetro de regularização C.

Determinados os valores ótimos de α_i^*, as variáveis $\mathbf{w}^*$ e $\mathbf{b}^*$ são definidas pelas equações 2.8 e 2.9, respectivamente. As variáveis ξ_i não têm influência na equação ótima do hiperplano, entretanto é um indicativo do erro empírico da máquina. Considerando a Equação A.37 em que os $\alpha_i \neq 0$, tem-se:

$$\xi_i = \max\left\{0, 1 - y_i(\mathbf{x}_i \cdot \mathbf{w} + b)\right\}. \tag{2.13}$$

A condição na Equação 2.13 faz-se necessária, pois, no caso de valores classificados corretamente, a variável ξ_i pode assumir valores negativos.

Uma vez determinados os α^*, a função classificadora é dada pela função sinal 2.14:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{\#SV} \alpha_i^* y_i (\mathbf{x}_i \cdot \mathbf{x}) + b^*\right), \tag{2.14}$$

em que $SV = \{\alpha^* : \alpha^* \neq 0\}$.

As SVMs de margens suaves são capazes de resolver alguns casos muito específicos de pro-

blemas não lineares, entretanto problemas reais possuem forte não linearidade o que torna essas máquinas ineficientes. Na Seção 2.2.3, são combinados os conceitos de SVMs de margens suaves ao do *kernel trick*, o que torna as SVMs uma ferramenta notável na classificação de dados não linearmente separáveis.

2.2.3 Máquinas de Vetores de Suporte Não Lineares

Na classificação de dados reais, geralmente, as classes não são linearmente separáveis, de tal modo que as SVMs de margens suaves não produzem resultados satisfatórios. A Figura 2.3 é um exemplo de tal situação. Nesse caso, aplica-se uma transformação não linear $\Phi(\mathbf{x})$, que leva os dados do espaço de entrada $\mathcal{L}$ para um espaço das características $\mathcal{H}$ de dimensão superior, conforme ilustrado na Figura 2.3 (CORTES; VAPNIK, 1995; THEODORIDIS; KOUTROUMBAS, 2009).

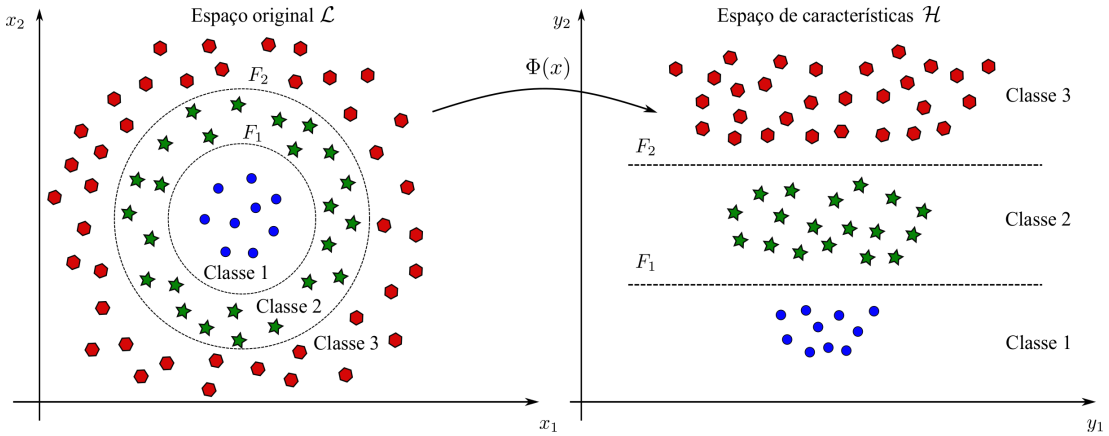


Figura 2.3 – Dados no espaço de origem.

O Teorema de Cover determina quais são as condições para que um conjunto de dados, com qualquer combinação de rótulos, possa ser separável em $\mathcal{H}$, por determinada probabilidade. As condições são:

(a) A transformação $\Phi(\mathbf{x})$ seja não linear.

(b) A dimensão do espaço de característica deve ser suficientemente alta.

A probabilidade dos dados $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ serem linearmente separáveis no espaço de características é (THEODORIDIS; KOUTROUMBAS, 2009):

$$P(N,d) = \begin{cases} \left(\frac{1}{2}\right)^{N-1} \sum_{m=0}^{d} \binom{N-1}{m} & se\ N > d+1 \\ \\ 1 & se\ N \leqslant d+1. \end{cases} \tag{2.15}$$

A Equação 2.15 define a probabilidade de separar N dados, seja qual for sua distribuição, em uma dimensão d. Percebe-se também que, se a dimensão d for maior que N, os dados sempre

serão separáveis. Entretanto considerar a dimensão de um espaço de características na mesma ordem da cardinalidade do conjunto solução não é viável computacionalmente, pois acrescentaria uma enorme quantidade de dados redundantes a este conjunto. O objetivo, então, é escolher a dimensão mínima que torna os dados linearmente separáveis (THEODORIDIS; KOUTROUMBAS, 2009).

No gráfico da Figura 2.4, a variável r é a razão entre a cardinalidade do conjunto de dados e a dimensão do espaço das características. No gráfico, tem-se a relação entre r e a probabilidade de os dados serem linearmente separáveis. Observa-se que a cardinalidade do conjunto deve ser, no máximo, duas vezes $d + 1$, com o objetivo de obter uma probabilidade (de cinquenta por cento) de um hiperplano separar os dados.

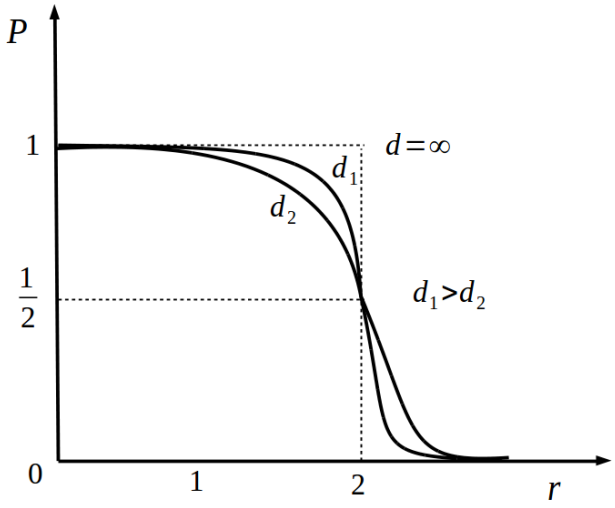


Figura 2.4 – Probabilidade N dados ser separável em uma dimensão d.

Fundamentada nas condições do Teorema de Cover e nas propriedades do espaço de Hilbert, tem-se a teoria necessária para definir um método que leve os dados do espaço de Entrada para um espaço das Características com dimensão adequada (THEODORIDIS; KOUTROUMBAS, 2009).

O espaço $\mathcal{H}$ deve ser um espaço de Hilbert, ou seja, um espaço vetorial no qual é definido o produto interno, tendo em conta que o modelo da SVM com margens suaves depende apenas do produto interno $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ no espaço $\mathcal{H}$, conforme o modelo 2.16.

$$\begin{array}{ll} \text{Max} & L_d(\alpha_i) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) \\ \text{s.a} & \\ \sum_{i=1}^{N} \alpha_i y_i = 0 & \\ \xi_i \geqslant 0 & i = 1, ..., N \\ 0 \leqslant \alpha_i \leqslant C & i = 1, ..., N. \end{array} \tag{2.16}$$

Desse modo, todas as características das SVMs de margens suaves são mantidas para as SVMs não lineares. Entretanto determinar a lei de formação da função $\Phi(\mathbf{x})$ e, posteriormente, calcular o

produto interno no espaço das características não é viável. Nesse contexto, denomina-se produto interno a operação definida por:

Definição 2.2.1 *Seja $\mathcal{V}$ um espaço vetorial real. Suponha que cada par de vetores $\mathbf{u}$ e $\mathbf{v} \in \mathcal{V}$ esteja associado a um número real, denotado por $\langle \mathbf{u}, \mathbf{v} \rangle$. Esta função é chamada* produto interno *em $\mathcal{V}$ se satisfaz os seguintes axiomas:*

$$\begin{aligned} \langle a\mathbf{u}_1 + b\mathbf{u}_2, \mathbf{v} \rangle &= a\langle \mathbf{u}_1, \mathbf{v} \rangle + b\langle \mathbf{u}_2, \mathbf{v} \rangle & (2.17) \\ \langle \mathbf{u}, \mathbf{v} \rangle &= \langle \mathbf{v}, \mathbf{u} \rangle & (2.18) \\ \langle \mathbf{u}, \mathbf{u} \rangle &\geqslant 0; \langle \mathbf{u}, \mathbf{u} \rangle = 0 \; sse \; \mathbf{u} = \mathbf{0}. & (2.19) \end{aligned}$$

(HILL, 2006)

O Axioma 2.17 refere-se à linearidade do produto interno em relação ao primeiro (ou segundo) fator do produto. O Axioma 2.18 garante a sua simetria e o Axioma 2.19 assegura que o produto seja definido positivo.

Na próxima subseção, será definido o *Kernel Trick*, que consiste em uma técnica em que Φ é definido implicitamente, entregando ao modelo o resultado do produto interno no espaço $\mathcal{H}$.

2.2.4 Truque do Núcleo (*Kernel Trick*)

O algoritmo de treinamento de uma SVM depende somente do produto interno em $\mathcal{H}$, ou seja, nas funções da forma $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Portanto, pode-se definir uma função K tal que $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, denominada função núcleo (*kernel*), definindo $\Phi(\mathbf{x})$ implicitamente. Umas das vantagens desta substituição está na simplicidade do cálculo no algoritmo de treinamento. Para obter o modelo com a função núcleo, substitui-se $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ por $K(\mathbf{x}_i, \mathbf{x}_j)$ no modelo 2.16 (THEODORIDIS; KOUTROUMBAS, 2009; LORENA; De Carvalho, 2008) Assim, tem-se:

$$\begin{array}{l} \text{Max} \quad L_d(\alpha_i) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.a} \\ \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \xi_i \geqslant 0 \qquad\qquad i = 1, ..., N \\ 0 \leqslant \alpha_i \leqslant C \qquad\quad i = 1, ..., N. \end{array} \tag{2.20}$$

A função classificadora, após a substituição da função núcleo, toma a forma da Equação 2.21,

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{SV} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b \right), \tag{2.21}$$

sendo SV o conjunto dos índices dos vetores de suporte.

Uma função K representa um produto interno entre os elementos do conjunto de treinamento se, somente se, ela satisfaz as condições do Teorema de Mercer.

Teorema 2.2.1 (Mercer) *Tem-se* $\mathbf{x} \in \mathbb{R}^l$ *e um mapeamento* Φ

$$\mathbf{x} \to \Phi(\mathbf{x}) \in \mathcal{H}$$

em que $\mathcal{H}$ *é um espaço de Hilbert. Nesse espaço, o produto interno tem a seguinte representação equivalente a*

$$\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}))\rangle = K(\mathbf{x}, \mathbf{z}), \tag{2.22}$$

em que $\langle \cdot, \cdot \rangle$ *denota a operação produto interno em* $\mathcal{H}$*. Então,* $K(\mathbf{x}, \mathbf{z})$ *é uma função contínua simétrica que satisfaz as seguintes condições:*

$$\int_C \int_C K(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) dx dz \geqslant 0, \tag{2.23}$$

para qualquer $g(\mathbf{x})$*,* $\mathbf{x} \in C \subset \mathbb{R}^l$ *tal que*

$$\int_C g(\mathbf{x})^2 dx < +\infty, \tag{2.24}$$

em que C *é um subconjunto compacto (finito) de* $\mathbb{R}^l$*. O contrário também é verdadeiro.*

Para um dado *kernel* K, a função de mapeamento Φ induzida pelo Teorema de Mercer depende fundamentalmente do domínio $\mathcal{X}$. A matriz $K = k_{ij}$ tal que $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, também conhecida como *matriz de Gram*, possui autovalores e autovetores que desempenham papel fundamental na estimava do erro na teoria da aprendizagem de máquina.

Dado um domínio $\mathcal{X}$ e um *kernel* K fixo sobre $\mathcal{X} \times \mathcal{X}$, existem, de fato, infinitas funções Φ associadas a K; e, finalmente, a matriz K deve ser semidefinida positiva para atender às condições do Teorema de Mercer (MINH; NIYOGI; YAO, 2006).

O modelo, representado na Equação 2.20, será convexo se, somente se, a matriz K satisfaz o Teorema de Mercer, ou seja, a matriz K deve ser semidefinida positiva e simétrica. Um *kernel* que não satisfaz essas condições produz um modelo com mínimos locais, tornando o processo de treinamento extremante complexo.

A Proposição 2.2.2 diz respeito às propriedades das funções *kernel* que satisfazem o Teorema de Mercer. Nesse contexto, pode-se adotar métricas diferentes da distância Euclidiana, associadas ao produto interno.

Proposição 2.2.2 *Considere,* k_1 *e* k_2 *kernels definidos positivos, arbitrários sobre* $\mathcal{X} \times \mathcal{X}$*, em que* $\mathcal{X}$ *é um conjunto não vazio:*

(a) O conjunto de kernels definido positivo é um cone convexo fechado, sendo,

   (a) se $\alpha_1, \alpha_2 \geqslant 0$*, então* $\alpha_1 k_1 + \alpha_2 k_2$ *é definido positivo;*

   (b) se $k(x_i, x_j) := \lim_{n\to\infty} k_n(\mathbf{x_i}, \mathbf{x_j})$ *existe para todos* x_i*,* x_j *então* k *é definido positivo.*

(b) O produto $(k_1 \cdot k_2)(\mathbf{x}_i, \mathbf{x}_j) = k_1(\mathbf{x}_i, \mathbf{x}_j) \cdot k_2(\mathbf{x}_i, \mathbf{x}_j)$

(c) Assumindo que, para $i = 1, 2, ...$, k_i é um kernel definido positivo sobre $\mathcal{X}_i \times \mathcal{X}_i$, em que $\mathcal{X}_i$ é um conjunto não vazio. Então, o produto tensorial $k_1 \otimes k_2$ e a soma direta $k_1 \oplus k_2$ são kernels definidos positivos sobre $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$.

(THEODORIDIS; KOUTROUMBAS, 2009; BURGES, 1998)

Na Tabela 2.1, estão as funções núcleo mais utilizadas (BURGES, 1998).

Tabela 2.1 – Tipos de *kernel*.

kernel	**Função** $K(\mathbf{x}_i, \mathbf{x}_j)$	**Parâmetros**
Polinomial	$(\beta(\mathbf{x}_i, \mathbf{x}_j) + \kappa)^d$	β, κ, d
Gaussiano - RBF	$e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\gamma}}$	γ
Sigmoidal	$\tanh(\delta(\mathbf{x}_i \cdot \mathbf{x}_j) + \rho)$	δ, ρ

A função sigmoidal não é semidefinida positiva para valores específicos de δ e κ.

Uma função de base radial tem a forma geral

$$f(\|\mathbf{x} - \mathbf{c}_i\|),$$

sendo $\|\mathbf{x} - \mathbf{c}_i\|$ a uma métrica de distância entre os centros $\mathbf{c}_i$ e o dado $\mathbf{x}$. Por isso, a função f é chamada de função de base radial (*Radial Basis Function* - RBF). As funções de base radial podem assumir várias formas, por exemplo, aquelas definidas em 2.25 e 2.26.

$$f(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{\gamma}\right) \tag{2.25}$$

$$f(\mathbf{x}) = \frac{\sigma^2}{\sigma^2 + \|\mathbf{x} - \mathbf{c}_i\|^2} \tag{2.26}$$

A função Gaussiana (Equação 2.25) é largamente utilizada na literatura (THEODORIDIS; KOUTROUMBAS, 2009; GANAPATHIRAJU et al., 2004; GASPAR; CARBONELL; OLIVEIRA, 2012; MIRANDA et al., 2014; SOARES; BRAZDIL; KUBA, 2004). Uma função contínua qualquer $g(\mathbf{x})$ pode ser aproximada por uma combinação linear de funções RBF para valores de k grande o suficiente, conforme a Equação 2.27

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{k} w_i \exp\left(-\frac{\|(\mathbf{x} - \mathbf{c}_i)\|^2}{\gamma}\right). \tag{2.27}$$

Nas SVMs, os $\mathbf{c}_i$ são os vetores de suporte e k a cardinalidade do conjuntos de vetores de suporte. Na situação em que a quantidade de vetores de suporte é igual ao número de dados de treinamento, tem-se uma generalização pouco eficiente, pois a SVM está considerando importante informações que podem ser *outliers* ou ruídos, implicando maior esforço computacional e baixa capacidade de generalização. Para valores de $k \ll N$ e pequenos erros no conjunto de

treinamento, tem-se uma máquina com bom desempenho computacional e uma boa capacidade de generalização (THEODORIDIS; KOUTROUMBAS, 2009).

A capacidade de uma função *kernel* linearizar dados pode ser medido pela dimensão VC; para as funções RBF Gaussiano, essa capacidade é caracterizada pelo teorema 2.2.3.

Teorema 2.2.3 *Dada uma classe de kernels que satisfazem o Teorema de Mercer para o qual* $K(\mathbf{x}_1, \mathbf{x}_2) \to 0$ *quando* $\|\mathbf{x}_1 - \mathbf{x}_2\| \to \infty$ *e para o qual* $K(\mathbf{x}_1, \mathbf{x}_2)$ *é* $O(1)$*, e assuma que os dados podem ser escolhidos arbitrariamente de* $\mathbb{R}^d$*. Então, a família de classificadores de SVMs com kernels que satisfazem essas condições, e, para o qual a penalidade para o erro pode assumir qualquer valor, tem dimensão* VC *infinita (BURGES, 1998).*

Demonstração: A matriz do *kernel* $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ é uma matriz de Gram em $\mathcal{H}$. Claramente, pode-se escolher os dados de treinamento tal que os elementos da matriz fora da diagonal principal $K_{i \neq j}$ podem ser feitos arbitrariamente pequenos e os elementos da diagonal por suposição são $O(1)$. Assim, a matriz K de posto completo e, portanto, o conjunto de vetores, cujo produtos em $\mathcal{H}$ formam K, são linearmente independentes e consequentemente os vetores em $\mathcal{H}$ também. Pelo teorema da dimensão VC, os pontos são linearmente separáveis por SVMs com penalidade para erro suficientemente grande. Sendo verdadeiro para qualquer números finitos de pontos, a dimensão VC desse classificador é infinita (BURGES, 1998).

As funções núcleo RBF atendem às condições do Teorema 2.2.3, e, desse modo, podem levar dados do espaço de entrada a um espaço de características de dimensão infinita. Portanto, de acordo com o teorema de Cover e o Teorema 2.1.1, as funções *kernels* RBF oferecem condições necessárias para determinação do hiperplano no espaço das características de qualquer conjunto dados X seja qual for a disposição de seus rótulos.

A teoria discutida até o momento sobre SVMs diz respeito a classificadores, mas essa teoria com as devidas adaptações pode ser estendida para produzir aproximadores de funções (regressão). Essas adaptações para gerar aproximadores de funções serão discutidas na Seção 2.3.

2.3 MÁQUINAS DE VETORES DE SUPORTE REGRESSÃO - SVR

Os *regressores* (ou aproximadores de funções) são modelos matemáticos utilizados para determinar o comportamento de um sistema do tipo *caixa preta*, simplificar modelos matemáticos complexos, eliminar informações indesejadas (ruídos e *outliers*); e podem ser empregados como preditores. As *Support Vector Regressors - SVR* são modelos matemáticos baseados na mesma teoria empregada nas SVMs como classificadores.

Os SVRs são obtidas a partir de um conjunto de treinamento $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N), \} \subset X \times \mathbb{R}$, em que $X = \mathbb{R}^d$ denota o espaço do padrão de entrada. Os SVRs são definidos de

maneira semelhante às SVMs classificadoras, bastando para isso adequar a função de penalização dos erros cometidos no conjunto de treinamento.

Como no caso dos classificadores, inicialmente, será tratado o caso linear, sendo a base para os demais conceitos desenvolvidos. A função $f : X \to \mathbb{R}$ é definida por:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b,$$

em que $\mathbf{w} \in X$, $b \in \mathbb{R}$.

As margens de um aproximador são dadas por:

$$\begin{array}{ll} H_1 : & \mathbf{w} \cdot \mathbf{x} + b + \epsilon = 0 \\ H_2 : & \mathbf{w} \cdot \mathbf{x} + b - \epsilon = 0, \end{array} \tag{2.28}$$

em que ϵ é o erro empírico permitido para cada dado do conjunto de treinamento. Nesse caso, a distância entre as margens é dada pela Equação 2.29 e deve ser maximizada.

$$d(H_1, H_2) = \frac{2\epsilon}{\|\mathbf{w}\|}. \tag{2.29}$$

A Equação 2.29 deve ser maximizada, para que as margens sejam o mais largas quanto possível e, assim, garantir uma boa generalização. O problema de otimização primal para o caso linear é dado pelo modelo 2.30:

$$\begin{array}{ll} \text{Min} & \frac{1}{2}\|\mathbf{w}\|^2 \\ s.a. & \\ y_i - (\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant \epsilon & i = 1, ..., N \\ (\mathbf{w} \cdot \mathbf{x}_i + b) - y_i \geqslant \epsilon & i = 1, ..., N, \end{array} \tag{2.30}$$

em que n é a cardinalidade do conjunto de treinamento.

O problema de aproximação de funções, modelado como em 2.30, pode conter dados que não são linearmente separáveis devido, por exemplo, a erros adicionados aos dados de treinamento, os quais não devem ser considerados. Nesse caso, permitem-se erros no conjunto de treinamento que são penalizados por uma constante positiva. O modelo desse problema é conhecido como margens suaves, descrito pelo modelo 2.31:

$$\begin{array}{ll} \text{Min} & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ s.a. & \\ y_i - (\mathbf{w} \cdot \mathbf{x}_i + b) \geqslant \epsilon + \xi_i & i = 1, ..., N \\ (\mathbf{w} \cdot \mathbf{x}_i + b) - y_i \geqslant \epsilon + \xi_i^* & i = 1, ..., N \\ \xi_i, \quad \xi_i^* \geqslant 0. & \end{array} \tag{2.31}$$

O parâmetro de regularização C penaliza somente os dados que estão fora da região definida

pelo parâmetro ϵ. Essa região é definida pela função de perda (*loss function*). Existem vários tipos de funções de perda, sendo as mais comuns a ϵ-insensitive, a Laplaciana, a Gaussiana, a perda robusta de Huber, a polinomial e polinomial por partes (SMOLA; SCHÖLKOPF, 2004; SMOLA et al., 1996). Neste trabalho, foi adotada a ϵ-*insensitive*, definida por Vapnik.

A função aproximadora é definida dentro do *insensitive tube*, conforme ilustrado na Figura 2.5. Os dados são penalizados conforme a função de perda ϵ-*insensitive*. A Figura 2.5 representa o gráfico da Equação 2.32.

$$|\xi|_\epsilon = \begin{cases} 0, & \text{se } |\xi| \leq \epsilon \\ |\xi| - \epsilon, & \text{caso contrário .} \end{cases} \tag{2.32}$$

A função de perda ϵ-insensitive é atrativa, pois, ao contrário das funções perda quadrática e Huber, em que todos os pontos do conjunto de treinamento são vetores de suporte, a solução daquela função perda é esparsa. Devido a essa característica, as funções aproximadoras possuem maior eficiência computacional e uma boa aproximação das funções ideais (BRERETON; LLOYD, 2010).

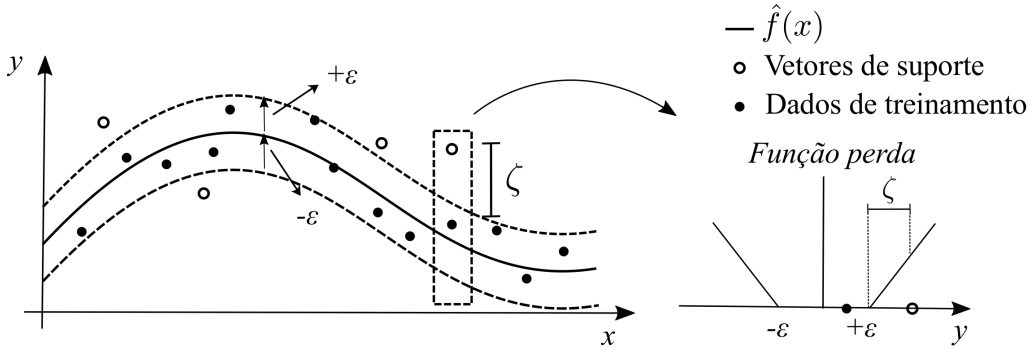


Figura 2.5 – Representação da função ϵ-insensitive, função aproximadora $\hat{f}(x)$. Adaptado de (SMOLA; SCHÖLKOPF, 2004).

Na Figura 2.5, a função $f(x)$ é a função ideal e a função $\hat{f}(x)$ é a função aproximadora.

A largura da região insensível ao erro, definido pelo valor de ϵ, está diretamente relacionada à função aproximadora resultante. Quanto maior o seu valor, mais folga existe para definição da função, gerando uma aproximação de baixa qualidade, mas com poucos vetores de suporte. Enquanto que valores menores de ϵ geram uma função superajustada no conjunto de treinamento, com muitos vetores de suporte, aumentando a sua complexidade e comprometendo a capacidade de generalização da máquina.

A partir do modelo 2.31, pode-se definir o problema dual, acrescentando ao problema os multiplicadores de Lagrange, referente a cada uma das restrições do problema, e somando à

função objetivo:

$$\begin{aligned}\text{Max} \quad & L(\mathbf{w}, b, \xi_i, \xi_i^*) = \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\alpha_i\left[\epsilon + \xi_i - y_i + (\mathbf{w}\cdot\mathbf{x}_i + b)\right] \\ & - \sum_{i=1}^{N}\alpha_i^*\left[\epsilon + \xi_i^* - y_i + (\mathbf{w}\cdot\mathbf{x}_i + b)\right] - \sum_{i=1}^{N}\left(\lambda_i\xi_i + \lambda_i^*\xi_i^*\right) \\ & \alpha_i, \alpha_i^*, \lambda_i, \lambda_i^* \geqslant 0 \qquad i = 1, ..., N.\end{aligned} \tag{2.33}$$

Derivando parcialmente em relação às variáveis de primais $\mathbf{w}$, b, ξ_i e ξ_i^* e igualando cada derivada parcial a zero, para determinar o ponto de sela, têm-se as igualdades:

$$\mathbf{w} = \sum_{i=1}^{N}\mathbf{x}_i(\alpha_i - \alpha_i^*) \tag{2.34}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \tag{2.35}$$

$$\lambda_i = C - \alpha_i, \qquad i = 1, ..., N. \tag{2.36}$$

$$\lambda_i^* = C - \alpha_i^*, \qquad i = 1, ..., N. \tag{2.37}$$

Substituindo 2.34, 2.35, 2.36 e 2.37 em 2.33 e desenvolvendo a propriedade distributiva, somando os termos semelhantes e adicionando as restrições, tem-se o modelo das SVR de margens suaves.

$$\begin{aligned}\text{Max} \quad & L(\alpha, \alpha^*) = -\frac{1}{2}\sum_{i,j=1}^{N}(\mathbf{x}_i\cdot\mathbf{x}_j)(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) - \epsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{N}y_i(\alpha_i - \alpha_i^*) \\ & \text{s.a.} \\ & \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C] \qquad i = 1, ..., N.\end{aligned} \tag{2.38}$$

A solução ótima do problema 2.38 permite definir o conjunto de vetores de suporte dado por $SV = \{x_i \in X : \alpha_i - \alpha_i^* \neq 0\}$. A cardinalidade desse conjunto normalmente é menor que a metade da cardinalidade do conjunto de treinamento X. A função aproximadora é dada pela Equação 2.39

$$f(x) = \sum_{i=1}^{\#SV}(\alpha_i - \alpha_i^*)(\mathbf{x}_i\cdot\mathbf{x}) + b, \tag{2.39}$$

em que b é a *bias* definida por 2.40

$$b = \frac{1}{2}\left\{\min\left(y_i - \sum_{i=1}^{\#SV}(\alpha_i - \alpha_i^*)(\mathbf{x}_i\cdot\mathbf{x})\right) + \max\left(y_i - \sum_{i=1}^{\#SV}(\alpha_i - \alpha_i^*)(\mathbf{x}_i\cdot\mathbf{x})\right)\right\}. \tag{2.40}$$

Os conceitos utilizados para definir a função, dada pela Equação 2.39, podem ser expandidos para funções não lineares, utilizando o truque do *kernel*. Todas as condições e definições a respeito dos *kernels* (citadas na Subseção 2.2.4) também são aplicadas aos regressores.

O produto escalar no modelo definido pela Equação 2.38 é substituído por um *kernel* não linear, tomando a forma do modelo dado pela Equação 2.41:

$$\begin{aligned} \text{Max} \quad L(\alpha, \alpha^*) = & -\frac{1}{2}\sum_{i,j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) - \epsilon\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \\ & \text{s.a.} \\ & \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ & \alpha_i, \alpha_i^* \in [0, C] \qquad i = 1, ..., N. \end{aligned} \tag{2.41}$$

Assim, a função aproximadora da Equação 2.39 toma a forma da Equação 2.42

$$f(\mathbf{x}) = \sum_{i=1}^{\#SV}(\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}) + b, \tag{2.42}$$

em que b é a bias definida pela Equação 2.43

$$b = \frac{1}{2}\left\{\min\left(y_i - \sum_{i=1}^{\#SV}(\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x})\right) + \max\left(y_i - \sum_{i=1}^{\#SV}(\alpha_i - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x})\right)\right\}. \tag{2.43}$$

As máquinas de vetores de suporte, dadas as devidas modificações, podem representar classificadores ou aproximadores de funções. Neste trabalho, quando citado SVM, faz-se referência aos classificadores, enquanto os modelos aproximadores de funções será tratado como SVR.

2.4 CONCLUSÕES DO CAPÍTULO

As máquinas de vetores de suporte, sejam aquelas modeladas como classificadoras, seja aquelas modeladas como aproximadora de funções, possuem um modelo matemático que consiste em um problema de programação quadrática convexo, possuindo a vantagem de ter solução ótima única. Essa característica, juntamente com o fato de a solução ótima desses problemas ser vetores esparsos, gera máquinas pouco complexas, o que as tornam apropriadas para modelar e resolver diversos problemas em engenharia.

Entretanto, em aplicações reais, o conjunto de treinamento possui milhares de vetores característicos, gerando um modelo com a mesma quantidade de variáveis. Isto torna o treinamento, em certos casos, uma tarefa árdua. Vários algoritmos vêm sendo utilizados para resolver esse

problema, tais como o algoritmo do pontos interiores, *Sequential Minimal Optimization* - SMO, coordenadas descendentes, método dos conjuntos ativos, resolução do problema primal pelo Método de Newton e subgradiente estocástico com projeção (SHAWE-TAYLOR; SUN, 2011).

Uma notável característica das máquinas de vetores de suporte é que sua complexidade computacional é independente da dimensão do espaço $\mathcal{H}$ ao qual o *kernel* leva os dados de entrada. Assim, o problema de efetuar cálculos em altas dimensionalidades é contornado pelo *kernel trick*.

Uma vez treinada a SVM e obtidos os α^* ótimos, a função dada pela Equação 2.21 é calculada com um baixo esforço computacional, pois, geralmente, o número de vetores de suporte é muito menor que a quantidade de dados no conjunto de treinamento.

O equilíbrio entre uma boa generalização e a baixa complexidade de uma SVM está na determinação do parâmetro de regularização C e dos parâmetros da função núcleo, sendo a técnica *Grid Search* usualmente utilizada para resolver este problema. Entretanto esta possui desvantagens em relação ao esforço computacional necessário e à definição do tamanho da malha (CHEN et al., 2016; NARZISI, 2008; HUANG; DUN, 2008).

As funções *kernel* mais utilizadas são a Guassiana, a polinomial e a sigmoide (hiperbólica) (LORENA; CARVALHO, 2007). A função gaussiana possui um único parâmetro γ a ser determinado e flexibilidade para ajustar os diversos tipos de funções com menos vetores de suporte que os outros *kernels* citados. Essas características do *kernel* gaussiano fazem esse *kernel* amplamente empregado em muitas problemas reais com um bom desempenho (GANAPATHIRAJU et al., 2004), embora outras funções *kernel* possam obter resultados melhores em situações específicas. Portanto, a questão em aberto para se obter uma SVM ótima consiste em determinar a função *kernel* e os hiperparâmetros do modelo, esta escolha torna-se complexa devido aos hiperparâmetros depender na natureza dos conjuntos de dados de treinamento (Chatelain et al., 2007).

O problema de determinação dos hiperparâmetros e da função *kernel* é uma limitação das SVMs, que será tradado no Capítulo 3 através de algoritmos meta-heurísticos de otimização multiobjetivo, tendo em vista a importância da determinação desses parâmetros para a qualidade da SVMs. Os conceitos de otimização multiobjetivo são definidos e exemplificados no Capítulo 3.

3 ALGORITMOS META-HEURÍSTICOS E O PROBLEMA DE SELEÇÃO DE PARÂMETROS

O problema de seleção de parâmetros das SVMs possui pelo menos dois objetivos contraditórios (a) a complexidade do modelo e (b) a sua capacidade de generalização, assim, caracterizando um problema multiobjetivo. Um conjunto de soluções desse problema multiobjetivo oferece a possibilidade de escolha de modelos ótimos, adequados a cada aplicação, que possuem compromissos distintos entre os dois objetivos contraditórios. Entretanto esses problemas são mais complexos de serem resolvidos e, por isso, utilizam-se algoritmos meta-heurísticos para resolver esse tipo de problema. Os algoritmos meta-heurísticos são inspirados no comportamento natural de espécies ou fenômenos físicos consistindo de uma sequência simples de passos que podem alcançar soluções de boa qualidade.

Neste capítulo, são apresentados os conceitos importantes para a compreensão da abordagem multiobjetivo, a descrição dos algoritmos PSO e DE básicos e as técnicas de adição de diversidade *Oposition Based Learning* - OBL e o Atrativo Repulsivo - AR.

3.1 INTRODUÇÃO

Em problemas de aplicações práticas, são comuns situações em que se deseja otimizar vários objetivos contraditórios a serem satisfeitos simultaneamente, como, por exemplo, ao comprar um carro, o consumidor prefere um carro com menor valor e maior conforto. Desse modo, a classe de problemas são denominados multiobjetivos, multicritério, multiperformance ou problema de otimização vetorial (COELLO; VELDHUIZEN, 2007). Esses problemas possuem funções objetivas e contraditórias, pois, caso contrário, essas poderiam ser transformadas em uma única função sem perda de informação.

A Definição 3.1.1 apresenta a forma geral do problema de otimização multiobjetivo.

Definição 3.1.1 *Um* Multi-Objective Optimization Problem - *MOOP é definido como*

$$\begin{aligned} &\textit{Min (ou Max)} \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})) \\ &\textit{s.a} \\ &g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., m \\ &h_j(\mathbf{x}) = 0 \quad j = 1, ..., p \\ &\mathbf{x} \in \Omega \end{aligned} \tag{3.1}$$

Uma solução do MOOP minimiza (ou maximiza) as componentes do vetor $\mathbf{F}(\mathbf{x})$*, em que* $\mathbf{x}$ *é uma variável de decisão n-dimensional* $\mathbf{x} = (x_1, x_2, ..., x_n)$ *de algum conjunto universo* Ω *(COELLO; VELDHUIZEN, 2007).*

Conforme a Definição 3.1.1, um MOOP consiste de k funções objetivos, $m + p$ restrições e n variáveis de decisão. As funções objetivos podem ser lineares ou não lineares e de natureza contínua ou discreta. O cálculo da função $\mathbf{F} : \Omega \rightarrow \Delta$ é o mapeamento das variáveis de decisão do seu espaço de origem para o espaço das funções objetivo. A Figura 3.1 mostra o mapeamento de $\Omega \in \mathbb{R}^2$ para $\Delta \in \mathbb{R}^2$.

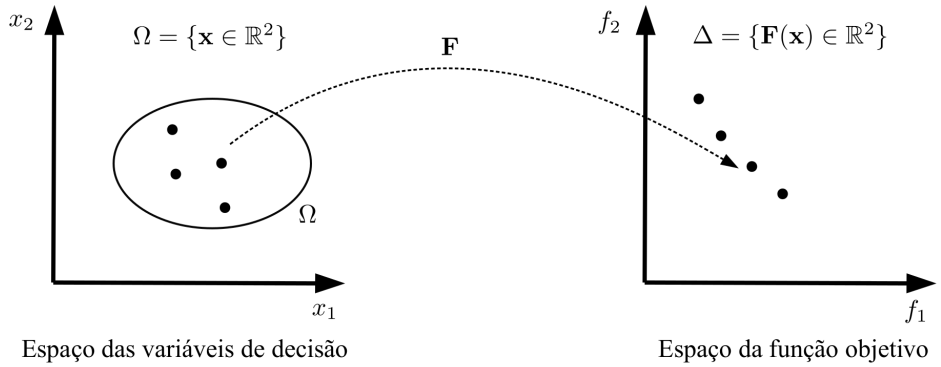


Figura 3.1 – Mapeamento das variáveis de decisão no espaço da função objetivo. Adaptado de (ABRAHAM; JAIN, 2005).

Resolver um problema multiobjetivo consiste em determinar um conjunto de soluções $\mathbf{x}^* = (x_1^*, x_2^*, ..., x_n^*)$, que satisfaçam as restrições do problema e produzam valores ótimos para todas as f_i componentes da função objetivo $\mathbf{F}(\mathbf{x}^*)$ (ABRAHAM; JAIN, 2005; COELLO; VELDHUIZEN, 2007). Deste ponto em diante, todas as funções f_i são de minimização, a não ser que seja explícito o contrário.

No processo de encontrar soluções que satisfaçam as condições de otimalidade, definir qual ou quais soluções são melhores é trivial. Entretanto, no processo de otimização multiobjetivo, uma pode ser melhor ou igual que outra em uma ou mais funções f_i, e pior nas demais, gerando um dilema de qual solução é melhor.

Diante do exposto, para definir a melhor solução, admitindo que todas as funções objetivo são igualmente importantes, a definição de *dominância* e *não dominância* entre soluções responde a este dilema (COELLO; VELDHUIZEN, 2007). No contexto de otimização multiobjetivo, uma solução é dita dominar outra (e melhor que outra) se as condições da Definição 3.1.2 forem atendidas.

Definição 3.1.2 (Dominância) *Considere um problema de otimização multiobjetivo em que todas as funções objetivo são de minimização. Um vetor, candidato à solução,* $\mathbf{u} = (u_1, u_2, ..., u_n)$ *é dito dominar outro vetor* $\mathbf{v} = (v_1, v_2, ..., v_n)$ *(denotado por* $\mathbf{u} \preceq \mathbf{v}$*) se, somente se,* $\mathbf{u}$ *é parcialmente menor que* $\mathbf{v}$*, isto é,* $\forall i \in \{1, ..., n\}, u_i \leq v_i \wedge \exists i \in \{1, ..., n\} : u_i < v_i$.

Caso um vetor $\mathbf{u}$ não domine $\mathbf{v}$, são definidos como *não dominantes*, sendo iguais em qualidade. O conjunto de Pareto Ótimo - PO é definido como o conjunto de soluções não dominantes dado pela Definição 3.1.3.

Definição 3.1.3 (Pareto ótimo) *Para um dado problema de otimização multiobjetivo,* $\mathbf{F}(\mathbf{x})$*, o conjunto de Pareto ótimo,* P^**, é definido como:*

$$P^* := \{\mathbf{x} \in \Omega | \neg \exists \mathbf{x}' \in \Omega \ \ \mathbf{F}(\mathbf{x}') \preceq \mathbf{F}(\mathbf{x})\}.$$

O conjunto de vetores não dominados são plotados no espaço das funções objetivas, em que são coletivamente conhecidos como *Fronteira de Pareto* - PF, de acordo com a definição 3.1.4.

Definição 3.1.4 (Fronteira de Pareto) *Para um dado MOOP,* $\mathbf{F}(\mathbf{x})$*, e o Conjunto de Pareto Ótimo,* P^**, a* Fronteira de Pareto PF^* *é definida como*

$$PF^* := \{\mathbf{u} = \mathbf{F}(\mathbf{x}) | \mathbf{x} \in P^*\}. \tag{3.2}$$

Na Figura 3.2, tem-se a representação do espaço das funções objetivo, em que os *Objetivos 1* e *2* são de maximizar. Os pontos são os candidatos às soluções do problema, no canto inferior esquerdo, existe um conjunto de soluções em que todas são dominadas pelas demais, ou seja, são as piores soluções, as soluções no alto e à direita são melhores soluções em cinza, pois dominam as demais. As soluções marcadas por pontos pretos são soluções que pertencem à Fronteira de Pareto (FP), estas que pertencem à FP dominam todas as demais do espaço de busca e são não dominantes entre si.

Nesses problemas, o objetivo é encontrar o conjunto de soluções mais próximas possível da Fronteira de Pareto (curva pontilhada-tracejada) e melhor distribuído sobre ela.

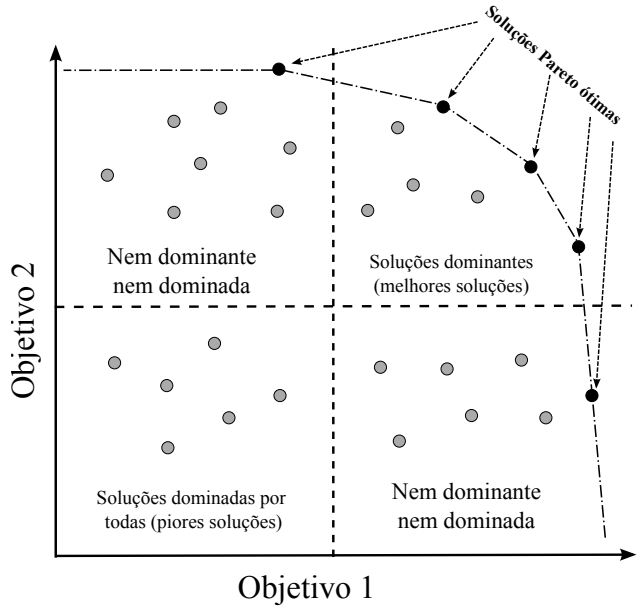


Figura 3.2 – Dados no espaço de origem. Adaptado de (ABRAHAM; JAIN, 2005).

O vetor $\mathbf{F}(\mathbf{x})$ das soluções e a Fronteira de Pareto são definidos no espaço das funções objetivos em que as coordenadas de um ponto A são $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x}))$. Na Figura 3.2, o espaço é o $\mathbb{R}^2$ da funções objetivos 1 e 2.

Os problemas de otimização que envolvem apenas uma função objetivo, nas últimas décadas, vêm sendo exaustivamente estudados e, consequentemente, é proposta uma pletora de algoritmos para resolvê-los, sendo, em sua maioria, meta-heurísticas, por buscarem solução de problemas NP-Completo (COELLO; VELDHUIZEN, 2007; ALI; SIARRY; PANT, 2012). Geralmente, os problemas multiobjetivo também são NP-Completo, e, portanto, é natural empregar essas técnicas para encontrar soluções de boa qualidade.

3.1.1 Problema de Seleção de Parâmetros da SVM

As Máquinas de Vetores de Suporte, embora possuam características matemáticas desejáveis, tais como modelo de treinamento convexo, vetores de pesos ótimos esparsos, minimização do risco empírico e termo de capacidade simultaneamente, deixam, assim, algumas decisões que devem ser tomadas pelo projetista, a saber: (a) parâmetro de regularização; (b) função *kernel*; (c) parâmetros do *kernel*; e (d) ϵ-*insensitive* no caso das SVR.

O parâmetro de regularização C, os parâmetros do *kernel* γ e do ϵ-*insensitive*, no caso da SVR, em conjuntos são chamados de hiperparâmetros e a determinação deles *Problema de Seleção de Parâmetros* - PSP ou *Problema de Seleção de Modelos* - PSM.

A capacidade de generalização e a complexidade de uma SVM/SVR estão intimamente ligadas à definição de seus hiperparâmetros. As máquinas de vetores de suporte com o valor de C tendendo ao infinito minimiza o erro sobre o conjunto de treinamento, porém, perde capacidade de generalização. Enquanto que a definição do γ depende da função *kernel* em questão, no caso de SVR, o parâmetros ϵ-*insensitive* está relacionado aos ruídos e *outliers*. Existe também uma interdependência entre os parâmetros que está relacionada ao conjunto de treinamento em questão (ALMASI; KHOOBAN, 2018).

O problema de seleção de modelos das SVMs possui ao menos duas funções objetivos conflitantes, ou seja, não é possível encontrar solução para uma função sem degradar a outra, caracterizando um MOOP. Neste problema, é considerada como funções objetivo a complexidade que é medida pela quantidade de vetores de suporte do modelo, e o erro cometido pelo modelo sobre o conjunto de validação (erro empírico). O problema de seleção de modelos das SVM é modelado como um problema multiobjetivo conforme descrito na Equação 3.3,

$$\min_{\omega \in \Omega} \quad \mathbf{F}(\omega) = (E_e(\omega), C_{SV}(\omega)) \tag{3.3}$$

em que $\mathbf{F}$ é o vetor das funções objetivos, $E_e(\omega)$ é a métrica que mensura a precisão de estimação do modelo no conjunto de validação, a função C_{SV} é a cardinalidade do conjunto de vetores de suporte, ω é o vetor de hiperparâmetros e Ω o espaço de busca dos hiperparâmetros.

A função $E_e(\omega)$, para dados de treinamento com dados de saídas contínuas, consiste no Erro

Médio Quadrático (*Mean Squared Error* - MSE) que é definido pela Equação (3.4)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.4}$$

em que y_i é a saída do dado x_i, $\hat{y}_i$ a saída predita pelo modelo em avaliação dada a entrada x_i e n a cardinalidade do conjunto de dados em avaliação.

Para modelos classificadores, a precisão é largamente utilizada como métrica para avaliar a desempenho destes modelos. Entretanto a precisão não é capaz de capturar diferentes fatores que caracterizam a eficiência de um classificador. Por exemplo, a precisão é especialmente enganosa na classificação de dados desbalanceados, ou seja, um conjunto de treinamento que possui oitenta porcento dos dados de uma classe, se o modelo independente da entrada sempre decidir por essa classe, ter-se-á uma precisão de oitenta porcento. Desse modo, para problemas desbalanceados e multiclasses, a métrica proposta por Carbonero-Ruz *et al.* em (CARBONERO-RUZ et al., 2017) adapta-se melhor para direcionar os algoritmos meta-heurísticos no processo de otimização. Em um problema com Q classes e N instâncias, em que N_q são as instâncias pertencentes a q-ésima classe, a precisão do classificador é dada pelas Equações 3.5 e 3.6

$$C = \sum_{q=1}^{Q}\frac{C_q}{N_q}\frac{N_q}{N}, \tag{3.5}$$

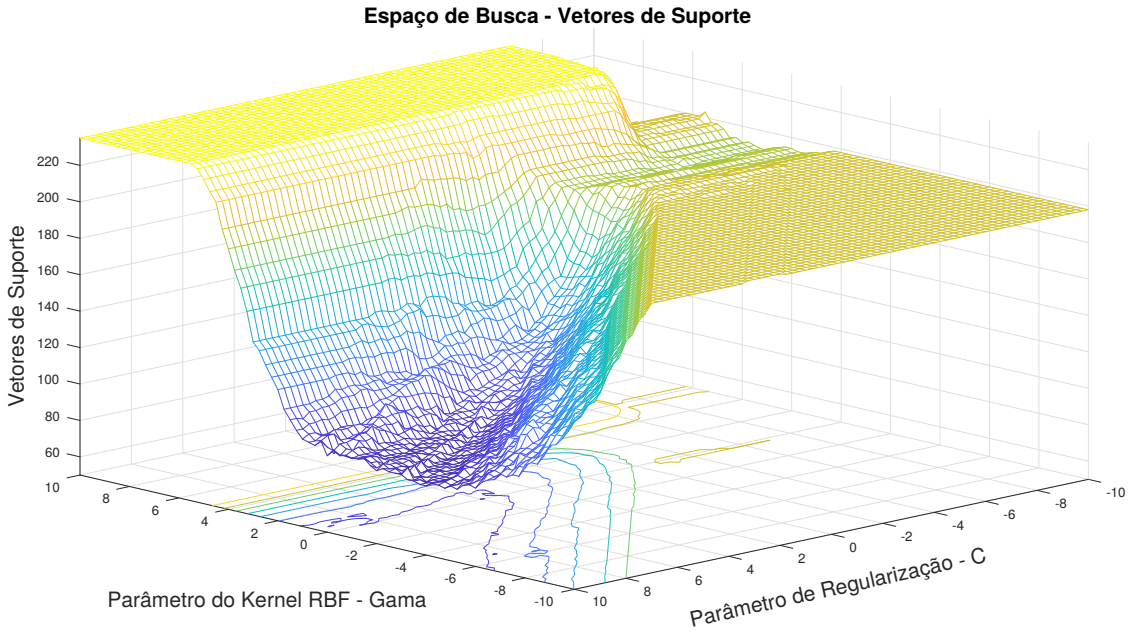
em que C_q é a quantidade de dados classificados corretamente pertencente à classe q, $\frac{C_q}{N_q}$ é a precisão na classe q e $\frac{N_q}{N}$ é a probabilidade (baseado na disponibilidade do conjunto de treinamento) que uma instância tem de pertencer à mesma classe,

$$D = \sqrt{\sum_{q=1}^{Q}\left(\frac{C_q}{N_q} - C\right)^2\frac{N_q}{N}}. \tag{3.6}$$

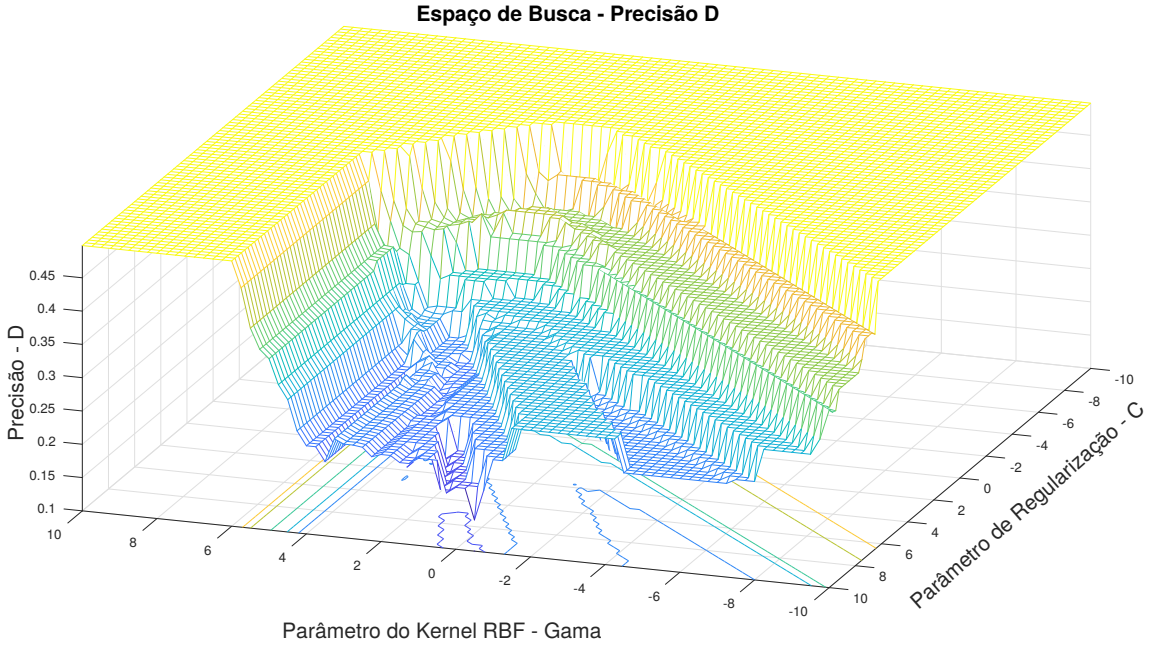
Valores de D próximos de zero corresponde à grande representatividade de C. Um modelo tem valor ótimo D quando este é exatamente igual a zero. Isto somente acontece no caso de a taxa de sucesso ser igual em todas as classes. Por outro lado, D aumenta com diferentes taxas de sucesso em cada classe. Isto se torna muito útil quando o equilíbrio de precisão entre as classes é importante (CARBONERO-RUZ et al., 2017). Assim para as SVMs, no modelo da Equação 3.3, $E_e(\omega) = D$ para um dado conjunto de validação.

Adicionalmente, o PSP é especialmente complexo, pois cada conjunto de treinamento consiste em um problema multimodal diferente. Além disso, o problema possui regiões de nível, em que os algoritmos meta-heurísticos não conseguem informação para direcionar o movimento das partículas/indivíduos. A Figura 3.3 foi obtida por meio do processo de busca exaustiva GS e mostra o comportamento das funções objetivos do *benchmark* Ecoli do repositório da *University*

of California, Irvine - UCI, modelado com o *kernel* RBF - Gaussiano (CORTES; VAPNIK, 1995; WU, 2011).



(a) Espaço de busca da função Vetores de Suporte.



(b) Espaço de busca da métrica de Precisão D.

Figura 3.3 – Espaço de busca *benchmark* Ecoli.

Na Figura 3.3, é possível observar grandes desníveis, indício de que pequenas variações nos hiperparâmetros produzem grandes alterações nos modelos resultantes, tanto no que diz respeito a vetores de suporte quando na métrica precisão. Além dessas dificuldades do PSP, é relevante considerar o esforço computacional para gerar cada modelo.

Os parâmetros possuem uma faixa de busca que pode variar consideravelmente dependendo da aplicação ou do *benchmark*. Para contornar esse problema, foi utilizada a codificação $C \in \{e^i, i \in [lb, lu] \wedge i \in \mathbb{R}\}$ e $\gamma \in \{e^i, i \in [lb, lu] \wedge i \in \mathbb{R}$. Esta codificação foi inspirada no trabalho de Li et al. (2013) o qual utiliza a codificação $C \in \{2^i, i \in [-10, 10] \wedge i \in \mathbb{Z}\}$ e $\gamma \in \{2^i, i \in [-5, 5] \wedge i \in \mathbb{Z}\}$ tornando o espaço de busca discreto.

A codificação para os parâmetros C e γ, proposta neste trabalho, possibilita que a avaliação de faixas de busca maiores sejam atingidas com quantidade reduzida de iterações e que o espaço de busca seja contínuo favorecendo a fase de refinamento dos algoritmos meta-heurísticos desenvolvidos. Para o ϵ-*insensitive* e o grau do *kernel* polinomial, a estratégia de codificação é aplicada.

Os algoritmos MOPSO, AP-MODE e APMT-MODE foram desenvolvidos especificamente para lidar com o PSP descritos nesta seção. Todos os testes foram realizados levando em consideração as métricas e estratégias definidas nesta seção.

Na Seção 3.2, apresentam-se a teoria e os fundamentos dos algoritmos PSO e DE básicos.

3.2 ALGORITMOS INSPIRADOS NA NATUREZA

Meta-heurísticas inspiradas em fenômenos naturais para resolver problemas complexos são utilizadas desde vários anos (BIRATTARI et al., 2001). Os algoritmos que simulam redes neurais, tendo como referência o cérebro humano, foram os primeiros do gênero a serem desenvolvidos. Esse algoritmo é empregado principalmente no reconhecimento de padrões (FRANçA, 2005). O Algoritmo Genético - AG, também inspirado na natureza, é amplamente empregado na resolução de problemas complexos, proposto inicialmente por Holland (HOLLAND, 1992) baseado na teoria da seleção natural proposta por Charles Darwin.

Um grupo de algoritmos inspirados na natureza baseia-se no comportamento coletivo, ao invés de criar uma sequência a partir de uma solução, em que são produzidas várias soluções simultaneamente. Neste caso, *indivíduos* exploram o espaço de busca movimentando-se, regidos por um conjunto de operações simples, adquirindo informações sobre o espaço de busca e compartilhando-as com seus vizinhos ou toda a população sem a necessidade de um comando central e, desse modo, produzindo uma forma de inteligência coletiva (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009).

O *Particle Swarm Optimizer* - PSO faz parte deste grupo de meta-heurísticas que faz uso do conhecimento coletivo, baseado no comportamento social de pássaros e peixes. Esses animais utilizam o conhecimento social e individual para proteger-se de predadores e obter comida com maior eficiência. O PSO básico é inspirado nesse padrão de comportamento para resolver problemas de otimização (KENNEDY; EBERHART, 1995).

O algoritmo *Differential Evolution* ou Evolução Diferencial é uma meta-heurística baseada na

teoria da evolução e apresenta operadores simples e que primam pelo equilíbrio entre a exploração do espaço de busca e o refinamento das soluções encontradas (STORN; PRICE, 1997a; ALI; SIARRY; PANT, 2012; XUE; SANDERSON; GRAVES, 2003).

Devido ao sucesso em obter soluções de boa qualidade para problemas de otimização mono-objetivo, os algoritmos inspirados na natureza têm sido aplicados na resolução de problemas multiobjetivos (Chatelain et al., 2007). Nas Subseções 3.2.1 e 3.2.2, são descritos os algoritmos básicos do PSO e DE.

3.2.1 *Particle Swarm Optimizer - PSO*

O PSO resolve um problema criando uma enxame de soluções candidatas, chamadas de *partículas*, e movimentando essas partículas pelo espaço de busca guiado pela memória individual e social. Esses métodos meta-heurísticos não garantem um ótimo global, mas, geralmente, encontram uma solução de boa qualidade (COELLO; VELDHUIZEN, 2007). A forma geral do problema de otimização tratado pelo PSO é definido pela Equação 3.7,

$$\begin{aligned} &\text{Min } f(\mathbf{x}) \\ &\mathbf{x} = (x_1, x_2, ..., x_n) \\ &\mathbf{x} \in [x_i^{min}, x_i^{max}] \quad i = 1, ..., n \end{aligned} \tag{3.7}$$

em que n é a dimensão do espaço de busca, $\mathbf{x}_{min}$ e $\mathbf{x}_{max}$ são os limites inferiores e superiores respectivamente de cada variável de decisão. No PSO, cada partícula é um vetor de dimensão n que representa uma possível solução do espaço de busca (ZHAO et al., 2011).

Durante o processo de busca, as partículas se deslocam de um ponto a outro do espaço das variáveis de decisão conforme a equação de velocidade, Equação 3.8.

$$v_{ij}^{(t+1)} = v_{ij}^t + c_1 U_{1j}(y_{ij}^t - x_{ij}^t) + c_2 U_{2j}(y_{sj}^t - x_{ij}^t), \tag{3.8}$$

A velocidade é um vetor que indica a direção, o sentido e o módulo do movimento da partícula em questão, sendo v_{ij}^t a velocidade da *i-ésima* partícula na *j-ésima* variável de decisão, t indica a iteração corrente, c_1 e c_2 são constantes de aceleração associadas ao conhecimento individual e social, respectivamente, U_{1j} e U_{2j} são números aleatórios distribuídos uniformemente no intervalo $[0, 1]$, y_{ij} é a melhor posição da partícula i e y_{sj} é a melhor partícula do enxame, ou seja, a partícula na posição melhor avaliada pela função objetivo $f(\mathbf{x})$ até a iteração t.

A posição da partícula é atualizada somando a velocidade v_{ij}^{t+1} à posição atual, conforme a Equação 3.9,

$$x_{ij}^{(t+1)} = x_{ij}^t + v_{ij}^{t+1}. \tag{3.9}$$

As constantes c_1 e c_2 representam o grau de confiança na memória individual e social respectivamente, obtendo melhores resultados atribuindo valores de tal forma que sua soma seja maior que quatro. Quando se abordam problemas unimodais, aconselha-se valorizar o conhecimento

social, enquanto que, para funções multimodais, deve haver um equilíbrio entre as duas constantes para não prejudicar a fase de exploração e tampouco a de refinamento (POLI, 2008; CLERC; KENNEDY, 2002).

Na resolução de problemas de otimização multimodal, é importante considerar a *sociometria do enxame*, ou seja, a forma como cada partícula compartilha sua informação com as demais (RICHARDS; VENTURA, 2003). As duas principais topologias de compartilhamento são *gbest* (estrela ou melhor global) e *ibest* (anel ou melhor local). A Figura 3.4 apresenta o esquemático das duas topologias (BLACKWELL; KENNEDY, 2018).

Na topologia *gbest*, a partícula de melhor *fitness* atrai as demais, acelerando a convergência para esta posição, na topologia *lbest*, cada partícula influencia seus k vizinhos, gerando uma convergência lenta, mas com uma exploração mais detalhada do espaço de busca, valorizando o processo de refinamento.

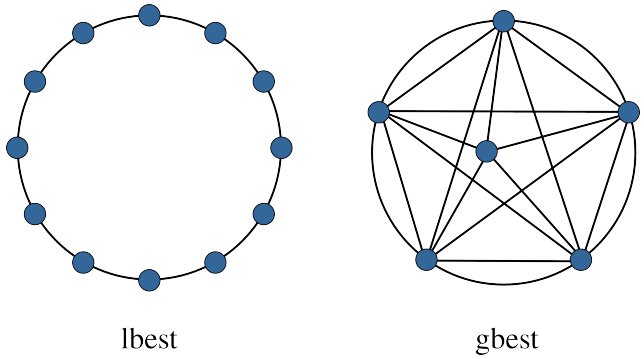


Figura 3.4 – Topologia de compartilhamento de informação do PSO.

O PSO armazena a melhor posição visitada por cada partícula ($Y_{S\times n}$), a melhor de todas as partículas $Ysj_{(1\times n)}$ e a posição atual das partículas $X_{S\times n}$, em que S é a quantidade de partículas no enxame. A cada iteração, as matrizes das partículas têm seus dados atualizados sem alterar suas dimensões e, portanto, o volume de dados armazenados não se altera no decorrer das iterações. No Algoritmo 1, tem-se o pseudocódigo do PSO básico.

Para iniciar o PSO, são necessários os parâmetros S, N, c_1, c_2, $\mathbf{x}_{max}$, Max_{ite} e $threshold$, que são respectivamente o número de partículas, de variáveis de decisão, os coeficientes de aceleração individual e social, o limite superior das variáveis de decisão (considera-se o espaço de busca simétrico definindo o limite inferior por $-\mathbf{x}_{max}$), o número máximo de iterações e o erro máximo aceitável para o caso em estudo.

Inicialmente, as S partículas são distribuídas aleatoriamente no espaço de busca e são calculados os valores funcionais $f(\mathbf{x})$ delas. O próximo passo consiste em identificar a melhor posição de cada partícula e qual a melhor posição entre todas as partículas. Em seguida, das linhas 10 a 15, é calculada a atualização da velocidade e da posição da partícula, por fim, é avaliado o critério de parada.

O PSO finaliza quando o critério de parada é atendido, que pode ser vários simultâneos ou

Algorithm 1: Pseudocódigo do PSO básico.

Input: $S, N, c_1, c_2, x_{max}, Max_{ite}, threshold$
Output: posição da melhor partícula e sua aptidão $f(x)$

Início:
inicializa o enxame
repeat
  for $k = 1 : S$ **do**
    if $f(x_k) \leq f(y_{ik})$ **then**
      $y_{ik} = x_k$
    end if
  end for
  Calcule y_s usando os S valores de aptidão $f(y_{ik})$
  for $k = 1 : S$ **do**
    for $j = 1 : N$ **do**
      $v_{kj} = v_{kj} + c_1 U_1(y_{kj} - x_{kj}) + c_2 U_2(y_{sj} - x_{kj})$
      $v_{kj} = x_{kj} + v_{kj}$
    end for
  end for
until $f(y_s) < threshold$
fim

único. Como exemplo de critérios de parada, têm-se: (a) o valor da função objetivo menor que $threshold$; (b) a quantidade predeterminada de iterações; (c) o valor funcional da melhor partícula não melhora por uma quantidade predefinida de iterações; e (d) a variação da função objetivo não se altera por uma quantidade predefinida de movimentos. O algoritmo de Evolução Diferencial é descrito na Subseção 3.2.2.

3.2.2 Evolução Diferencial - DE

Os Algoritmos Evolutivos (*Evolutionary Algorithms* - EA) são uma classe de algoritmos heurísticos que simulam a evolução das espécies descrita pela Teoria da Seleção Natural formulada por Charles Darwin, em que os indivíduos de determinado ecossistema tornam-se cada vez mais adaptados, transmitindo as suas características para sua prole a cada geração (DAS; SUGANTHAN, 2011a).

O DE faz parte da classe de EA. Este faz a adaptação dos indivíduos por meio de sucessiva aplicação dos operadores genéticos mutação, recombinação e seleção que são realizados por várias gerações. As operações mutação e recombinação são responsáveis por modificar as características dos indivíduos, enquanto o operador seleção é responsável por escolher os indivíduos mais adaptados ao meio ambiente em que vivem. Esta descrição simplificada do processo de evolucionário das espécies é simulada matematicamente para encontrar soluções de boa qualidade para problemas de otimização complexos.

O DE é uma simplificação dos operadores empregados pelo Algoritmo Genético, mas o DE mantém as principais características das fases de exploração e refinamento do AG e adicional-

mente possui a capacidade de resolver problemas de variáveis contínuas, enquanto, no AG, é necessária uma codificação especial para fazê-lo.

No DE, cada indivíduo é representado por um vetor $\mathbf{x} \in \mathbb{R}^n$, sendo n a dimensionalidade das variáveis de decisão do problema, estas podem representar grandezas físicas ou parâmetros que possuem limites inferiores e superiores, x^i_{min}, x^i_{max}, $i = 1, ..., n$ respectivamente.

O número de indivíduos $\mathbf{x}$ da população inicial é NP, sendo estes manipulados durante o processo de busca e, devido à dinâmica do movimento, NP deve ser maior ou igual a quatro. Os indivíduos da população inicial devem estar distribuídos aleatoriamente no espaço de busca. A Equação 3.10 garante a distribuição uniforme dentro do espaço de busca factível,

$$x_{j,i} = x_{j,\min} + rand_{i,j}[0,1] \cdot (x_{j,\max} - x_{j,\min}), \tag{3.10}$$

em que $rand_{i,j}$ é um número aleatório de distribuição uniforme pertencente ao intervalo $[0, 1]$.

Um algoritmo meta-heurístico robusto não deve depender da população inicial, entretanto, na busca por maior eficiência em relação à convergência, pode-se utilizar algoritmos heurísticos, tal como *Greedy*, ou alguma outra estratégia para gerar uma população inicial específica que seja favorável ao processo de busca. Vale ressaltar que esses algoritmos são conhecidos como híbridos.

No DE, a evolução dos indivíduos é direcionada pelos operadores *mutação*, *recombinação* e *seleção* que são aplicados nesta ordem a cada geração. A mutação consiste em uma variação abrupta nas característicos (valores) dos indivíduos no DE, definida pela Equação 3.11,

$$\mathbf{v}_{i,G} = \mathbf{x}_{r^i_1,G} + F(\mathbf{x}_{r^i_2,G} - \mathbf{x}_{r^i_3,G}) \quad i = 1, ..., NP, \tag{3.11}$$

em que $\mathbf{x}_{r^i_1}$, $\mathbf{x}_{r^i_2}$ e $\mathbf{x}_{r^i_3}$ são vetores escolhidos aleatoriamente da população e mutuamente distintos. O fator F é um escalar tipicamente entre $[0.4, 1]$ e G é a geração atual. A Figura 3.5 representa, em duas dimensões, a dinâmica de determinação do vetor mutante $\mathbf{v}_{i,G}$.

O operador recombinação aprimora o potencial de diversidade da população combinando o vetor $\mathbf{x}_{i,G}$ da população com o vetor mutante $\mathbf{v}_{i,G}$, conforme a Equação 3.12,

$$u_{j,i,G} = \begin{cases} v_{i,j,G} & \text{se } rand_{i,j} \leq Cr \text{ ou } j = j_{rand} \\ x_{i,j,G} & \text{caso contrário,} \end{cases} \tag{3.12}$$

em que a taxa de recombinação Cr é usualmente um escalar no intervalo $[0,3; 0,5]$ e $rand_{i,j}$ é um valor aleatório proveniente de uma distribuição uniforme no intervalo $[0; 1]$.

O operador seleção no DE é o equivalente na natureza às adversidades do ambiente em que os indivíduos são submetidos e que elimina as característica que não se ajustam ao meio. No DE básico, utiliza-se a seleção elitista, ou seja, o vetor $\mathbf{x}_{i,G}$ somente será substituído por $u_{i,G}$ caso o

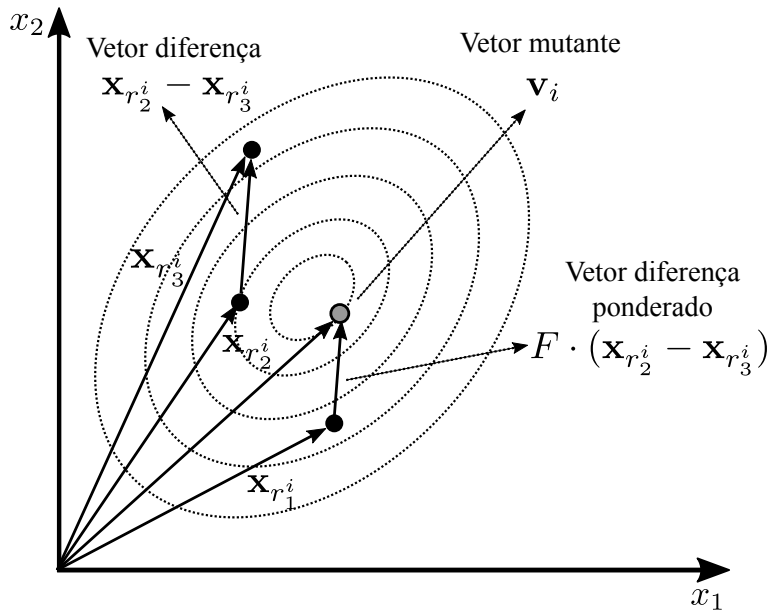


Figura 3.5 – Movimento do DE, vetor mutação. Adaptado de (STORN; PRICE, 1997a).

último seja igual ou mais adaptado, no caso de problemas de minimização, se $f(\mathbf{u}_{i,G}) \leq f(\mathbf{x}_{i,G})$ (STORN; PRICE, 1997a).

Uma iteração do DE consiste na aplicação em sequência aos indivíduos da população dos operadores mutação, recombinação e seleção. O processo é repetido até que o critério de parada seja alcançado. O Algoritmo 2 apresenta o pseudocódigo do DE.

No pseudocódigo, o laço de repetição da linha 5 controla a quantidade de gerações. O operador mutação é calculado na linha 8 sendo aplicado a cada posição de forma independente, desse modo, esta implementação favorece trabalhar com variáveis de limites e grandezas diferentes. O condicional na linha 10 implementa o operador recombinação que define quais posições do vetor $u_{j,i,G}$ assume o valor do vetor mutante.

No operador seleção, o resultado de $f(\mathbf{u}_i)$ é comparado com $f(\mathbf{x}_i)$, sendo que o indivíduo mais adaptado sobrevive avançando para a próxima geração, enquanto o menos adaptado perece.

Frenquentemente, os algoritmos meta-heurísticos ficam presos em mínimos locais e exploram apenas uma região específica do espaço de busca. Na Seção 3.3, são descritas duas técnicas de Adição de Diversidade que aumentam a capacidade de exploração dos desses algoritmos.

3.3 MÉTODOS DE ADIÇÃO DE DIVERSIDADE

Na resolução de problemas de otimização multimodais, os algoritmos de evolucionários geralmente sofrem de convergência prematura. A convergência prematura aprisiona os indivíduos em mínimos locais nas iterações iniciais do processo, restringindo a exploração do espaço de busca a determinadas regiões que, muitas vezes, são pobres. Por outro lado, os algoritmos que

Algorithm 2: Pseudocódigo do algoritmo DE.

Input: NP, Cr, F, Max_{ite}

Início:
Inicializa a população
Avalia cada indivíduo
$G = 1$
repeat
  for $i = 1 : NP$ **do**
    Escolha aleatoriamente r_1, r_2 e r_3.
    $\mathbf{v}_{i,G} = \mathbf{x}_{r_1^i,G} + F \cdot (\mathbf{x}_{r_2^i,G} - \mathbf{x}_{r_3^i,G})$
    for $j = 1 : n$ **do**
      if $(rand_{i,j} \leq Cr) \vee (j = j_{rand})$ **then**
        $u_{j,i,G} = \mathbf{v}_{i,j,G}$
      else
        $u_{j,i,G} = \mathbf{x}_{i,j,G}$
      end if
    end for
  end for
  Avalia cada indivíduo u_i
  $C = U \cup X$
  $X =$ seleção(C, NP)
  $G = G + 1$
until $G \geq Max_{ite}$
fim

possuem uma convergência muito lenta prejudica a fase de refinamento, também não satisfazendo as restrições de tempo e o esforço computacional na maioria das aplicações reais, sendo assim a convergência desses algoritmos consiste em um dilema contraditório (RIGET; VESTERSTRØM, 2002).

O problema de determinação dos hiperparâmetros de uma SVM é um problema multimodal e, dessa maneira, o PSO e o DE sofrem de convergência prematura. Para evitar a convergência na fase inicial, são utilizados métodos de adição de diversidade, tais como AR (RIGET; VESTERSTRØM, 2002) e o OBL (TIZHOOSH, 2005).

O método AR, no PSO, consiste em duas fases distintas: a *atrativa* e a *repulsiva*. Na fase atrativa, as partículas deslocam-se aglomerando na região da melhor partícula do enxame e, na fase repulsiva, as partículas movimentam-se no sentido contrário, afastando-se da melhor partícula, aumentando, assim, a diversidade do enxame. Matematicamente, esse processo é realizado multiplicando-se a velocidade de cada partícula por menos um na fase repulsiva e, na fase atrativa, o algoritmo se comporta-se conforme o PSO básico.

A diversidade é medida por um escalar d calculado pela Equação 3.13, sendo a métrica $d(S)$ independente do tamanho do enxame, da dimensionalidade do problema e dos limites do espaço de busca. Na iteração em que $d(S)$ é menor que o limite inferior d_{min}, a técnica entra na fase

repulsiva e, se $d(S)$ for maior que d_{max}, o algorítimo entra na fase atrativa

$$d(S) = \frac{1}{\#S \cdot |L|} \cdot \sum_{i=1}^{\#S} \sqrt{\sum_{i=1}^{n} (p_{ij} - \bar{p}_j)^2}, \tag{3.13}$$

em que S é o conjunto que representa o exame, $\#S$ a cardinalidade do exame, $|L|$ a maior diagonal do espaço do busca, n é a dimensionalidade do problema, p_{ij} é o valor da i-ésima partícula na j-ésima dimensão e $\bar{p}_j$ é o valor da j-ésima média do ponto $\bar{p}$ (RIGET; VESTERSTRØM, 2002).

O método de adição de diversidade AR não acrescenta grande esforço computacional aos algoritmos meta-heurísticos, sendo uma boa técnica para controlá-los no processo de exploração do espaço de busca e refinamento das soluções.

O algoritmo OBL adiciona diversidade aos algoritmos de busca meta-heurísticas e foi introduzido por Tizhoosh (TIZHOOSH, 2005). Os algoritmos de buscas meta-heurísticos, em sua maioria, inicializam o processo de busca por um conjunto de soluções aleatórias que podem estar próximas ou não da solução ótima.

Na situação em que um ponto P está distante da solução ótima, o ponto oposto $\tilde{P}$, como dado pela Definição 3.3.1, é uma tentativa melhor que um ponto aleatório segundo Rahnamayan (RAHNAMAYAN; TIZHOOSH; SALAMA, 2008).

Definição 3.3.1 *Seja $P(x_1, x_2, ..., x_n) \in \mathbb{R}^n$, sendo $x_1, x_2, ..., x_n \in \mathbb{R}$ e $x_i \in [a_i, b_i]$. O ponto oposto $\tilde{P}$ é definido por suas coordenadas $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$, sendo*

$$\tilde{x}_i = a_i + b_i - x_i \quad i = 1, ..., n. \tag{3.14}$$

No DE e no PSO, o OBL age sobre a população/exame inicial e sobre a população/exame a cada iteração, de acordo com a Definição 3.3.2.

Definição 3.3.2 *Sejam os pontos P e o seu oposto $\tilde{P}$ candidatos à solução de um espaço de busca e $\mathbf{F}(P)$ e $\mathbf{F}(\tilde{P})$ seus correspondentes no espaço das soluções, P fará parte da população/enxame se $\mathbf{F}(P)$ domina $\mathbf{F}(\tilde{P})$, caso contrário, P é substituído por $\tilde{P}$.*

Para soluções de problemas multiobjetivos, uma solução P será substituída por $\tilde{P}$ se esta domina P, ou se $\tilde{P}$ e P forem não dominadas. Este critério favorece a diversidade da população/enxame.

3.4 CONCLUSÕES DO CAPÍTULO

A filosofia de otimização multiobjetivo agrega muita informação aos modelos matemáticos quando comparados aos modelos de otimização mono-objetivo. O MOOP, por considerar mais de um critério, representa melhor problemas práticos, entretanto essa representatividade adiciona

dificuldade extra em encontrar a solução ótima, que, nesse caso, é dada pelo conjunto de Pareto (Von L u cken; BARáN; BRIZUELA, 2014).

Os algoritmos meta-heurísticos empregam equações simples para explorar o espaço de busca, independente se este é discreto ou contínuo, e as equações objetivos são diferenciáveis ou não. Devido a essas características, eles têm sido utilizados para resolver problemas complexos inclusive problemas multiobjetivos.

Especificamente, o algoritmo PSO possui, quando se trata de MOOPs, uma dificuldade em especial, pois, no PSO básico, é necessário definir a melhor partícula social e a melhor posição visitada pela partícula, que contraria a filosofia dos MOOPs, em que um conjunto de soluções são iguais em qualidade. Assim surge a questão: Qual dessas soluções escolher sem prejudicar a convergência e a qualidade do conjunto de Pareto obtido? Várias técnicas têm sido empregadas para resolver essa questão, tais como (WANG; YANG, 2010; JIANG et al., 2014; SHIH-YUAN et al., 2007; MEZA et al., 2016).

No DE, essa questão surge no operador seleção, que, no DE básico, define se uma solução é melhor que a outra. Nesse caso, uma técnica simples é adotar a solução filha quando as duas são não dominantes entre si.

Embora possuam particularidades para tratar os problemas de otimização modelados como multiobjetivo os algoritmos, PSO e DE possuem uma boa capacidade de exploração e refinamento das soluções em espaços de busca, que apresentam desafios em encontrar o conjunto de Pareto. Portanto, uma implementação desses algoritmos é descrito no Capítulo 5.

No Capítulo 4, foi realizada um levantamento bibliográfico sobre a aplicação dos algoritmos PSO e DE, assim como dos algoritmos empregados nos últimos 16 anos para resolver o problema de seleção de modelo das SVMs.

4 TRABALHOS RELACIONADOS

Os inúmeros trabalhos publicados desde o surgimento do problema de seleção de modelos até os dias atuais, *vide* Tabela 4.1, destacam a preocupação dos pesquisadores com o tema e a importância de uma solução de boa qualidade para este problema nas diversas aplicações em que as SVMs são utilizadas. Neste capítulo, é apresentada uma revisão sobre os algoritmos empregados entre os anos de 2002 e 2018, na busca por uma solução adequada para o problema de seleção de modelos.

Devido à grande popularidade dos algoritmos meta-heurísticos, existem vários artigos de revisão de literatura sobre esses algoritmos. Neste capítulo, buscou-se fazer uma revisão criteriosa destes artigos, a fim de obter informações mais concisas sobre o assunto.

4.1 INTRODUÇÃO

Desde a popularização das SVMs, o problema de seleção dos hiperparâmentros tem sido extensamente estudado. Na revisão bibliográfica, apresentada na Tabela 4.1, foram compilados 60 artigos sobre o problema de seleção de parâmetros para SVMs, sendo esse problema tratado por diversos algoritmos meta-heurísticos. Porém, poucos trabalhos lançam mão de algoritmos multiobjetivos para resolver tal problema, sendo que, nesta revisão, eles representam treze porcento do total.

Nesse cenário, devido ao potencial dos algoritmos PSO e DE em resolver problemas complexos e haver poucos trabalhos que abordam o problema de seleção do modelo de SVMs como um MOOP, foram implementadas versões multiobjetivos do PSO e DE com o objetivo de encontrar, ao invés de uma única solução, um conjunto de soluções que produzam SVMs de alta qualidade, avaliadas segundo as métricas descritas na Subseção 6.1.1.

Os algoritmos meta-heurísticos são uma representação simplificada de sistemas naturais complexos (para este caso), tais como o comportamento de animais (individualmente ou em grupos), na teoria da seleção natural, comportamento presa-predadores, leis físicas, comportamento físico de materiais e outros (JR et al., 2013). Eles apresentam capacidade de resolver problemas complexos multimodais sem exigir que estes sejam diferenciáveis ou mesmo contínuos, sendo capazes de explorar o espaço de busca e refinar as soluções, produzindo bons resultados. Nas SVMs, por exemplo, o critério *número de vetores de suporte* não é diferenciável, impossibilitando, assim, o uso de técnicas que envolvam o cálculo do gradiente.

Na Seção 4.2, evidencia-se a importância dos algoritmos meta-heurísticos para resolver problemas multiobjetivos e, na Seção 4.3, é realizado um levantamento de trabalhos publicados que desenvolveram algoritmos meta-heurísticos na tentativa de resolver o problema de seleção de

modelo das SVMs, a fim de salientar a preocupação da comunidade científica em obter os hiperparâmetros ótimos. Este levantamento mostra também as técnicas utilizadas para alcançar esses resultados.

4.2 ALGORITMOS META-HEURÍSTICOS MULTIOBJETIVOS

Em muitos MOOP, obter o conjunto de Pareto completo e exato é uma tarefa árdua e, por isso, a aproximação do conjunto de Pareto é aceita como solução dos problemas da engenharia. Os algoritmos evolucionários têm provado serem capazes de encontrar uma boa aproximação para problemas multiobjetivos, multimodais complexos.

Vários trabalhos têm sido publicados envolvendo diversos tipos de algoritmos meta-heurísticos; o que parece confirmar o grande interesse da comunidade científica nesses algoritmos e, consequentemente, sua importância. Alguns exemplos de artigos de revisão bibliográfica sobre esses algoritmos são (CORREIA, 2013; ULUNGU; TEGHEM, 1994; AGARWAL; MEHTA, 2014) e (Von L u cken; BARáN; BRIZUELA, 2014). Entre esses algoritmos, destacam-se os algoritmos PSO e o DE.

Um ano depois, R. Storn e K. V. Price publicaram o primeiro relatório técnico descrevendo o DE (STORN; PRICE, 1997a), este algoritmo adquiriu popularidade depois do *First International Conference on Evolutionary Computation*, no qual alcançou o terceiro lugar como melhor algoritmo evolucionário para resolver problemas testes de variáveis reais. Nos anos que seguiram, o DE e as suas variações sempre figuraram entre os melhores algoritmos em competições internacionais (DAS; SUGANTHAN, 2011b). Em 2007 e 2009, o *Differential Evolution* alcançou o segundo lugar e o primeiro lugar respectivamente para solução de problemas multiobjetivos. O sucesso nestes eventos, diante dos demais algoritmos, motivou o desenvolvimento de um MODE para determinação dos hiperparâmetros das SVMs.

O PSO e o DE são métodos que têm alcançado bons resultados na determinação de soluções de boa qualidade na resolução de problemas NP-Completos. O PSO consiste em um método de otimização estocástico baseado em população. Esse algoritmo tem sido aplicado com sucesso em diversos problemas, como, por exemplo, treinamento de redes neurais, otimização de funções, controle *fuzzy* e reconhecimento de padrões, controle preditivo e outros. O crescente interesse pelo PSO pode ser observado na Figura 4.1.

Os critérios para construção do histograma da Figura 4.1 são descritos em Correia (2013). Observando a curva da Figura 4.1, fica evidente o quanto este algoritmo tem sido importante na resolução de problemas complexos. Na Figura 4.1, a quantidade de artigos publicados cresceram rapidamente entre 1997 e 2009. Nos anos seguintes, o interesse pelo algoritmo manteve-se alto, alcançando, em 2012, quase 4.500 publicações (que continham o PSO no título do trabalho).

Na Seção 4.3, foi realizada uma busca bibliográfica a respeito dos algoritmos meta-herísticos empregados no problema de seleção de parâmetros das SVMs.

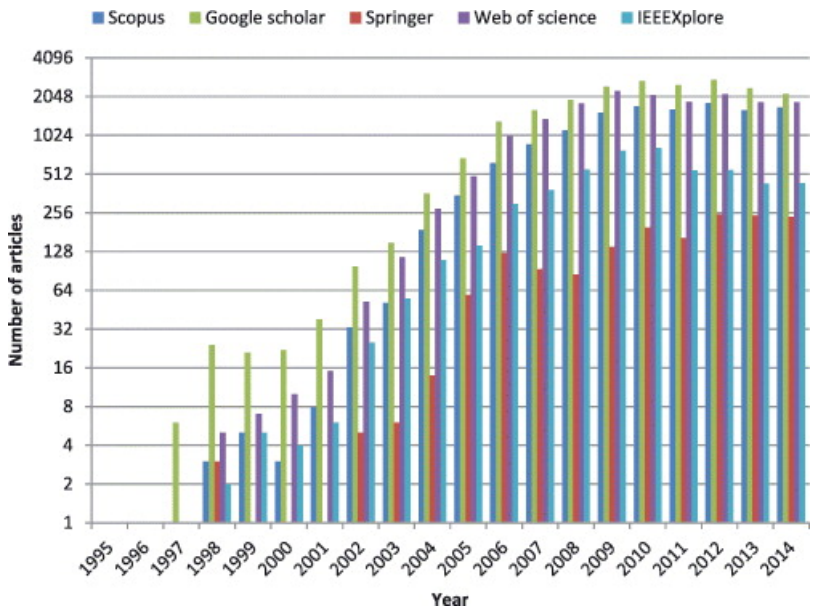


Figura 4.1 – Número de artigos publicados com "Particles Swarm Optimization" em seu títulos nas bases de dados Scopus, Google Scholar, Springer, Web of Science e IEEE Xplore (CORREIA, 2013).

4.3 PROBLEMA DE SELEÇÃO DE PARÂMETROS

No processo de modelagem, o parâmetro de regularização e os parâmetros do *kernel* devem ser definidos *a priori*, interferindo diretamente na qualidade final do classificador obtido, ou seja, na sua capacidade de generalização e complexidade, conforme definido na seção 2.1.

A definição dos hiperparâmetros é um problema complexo, pois, para cada conjunto de treinamento, tem-se um problema distinto e, geralmente, esses problemas possuem vários mínimos locais, dificultando, assim, sua determinação ótima (MIRANDA; PRUDÊNCIO, 2013). No Apêndice B, têm-se os gráficos das funções custo precisão e cardinalidade dos vetores de suporte para cada combinação de *benchmarks* e *kernels*, em que pode ser observado o comportamento destas funções e a complexidade que envole encontrar o conjunto de Pareto.

A determinação dos hiperparâmetros ótimos das SVMs tem atraído a atenção da comunidade desde sua consolidação (no final dos anos noventa) com a publicação dos trabalhos de Vapnik (CORTES; VAPNIK, 1995) e Smola (SMOLA; VAPNIK, 1997). A Tabela 4.1 apresenta vários trabalhos sobre o tema compreendidos entre os anos de 2002 a 2018, enquanto, na Figura 4.2, esses trabalhos foram quantificados por ano de publicação. A interesse pelo tema nos anos de 2016 e 2017 é maior que nos demais anos, ficando clara a relevância das SVM/SVRs e a importância de definir hiperparâmetros adequados para estes modelos.

Na Tabela 4.1, os artigos foram classificados como se segue: (a) ano de publicação; (b) quanto ao algoritmo empregado na solução do problema de seleção do modelo; (c) tipo de dados empregados para validar o algoritmo; (d) quanto à natureza multi ou mono-objetivo da formulação do problema de seleção de modelos; (e) distinção entre modelos classificadores e regressores; e (f) a quantidade de saídas do modelo SVM/SVR.

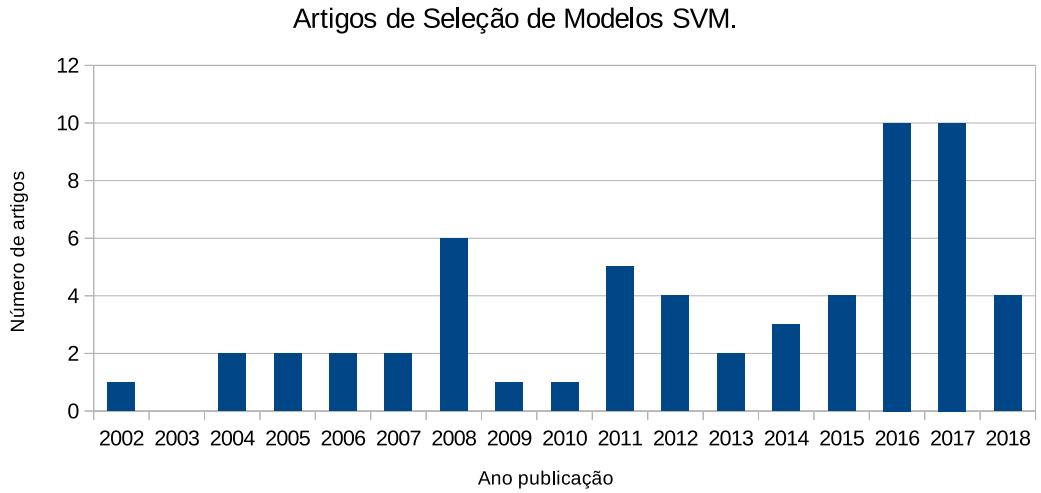


Figura 4.2 – Artigos que trata o PSP-SVM classificados por ano.

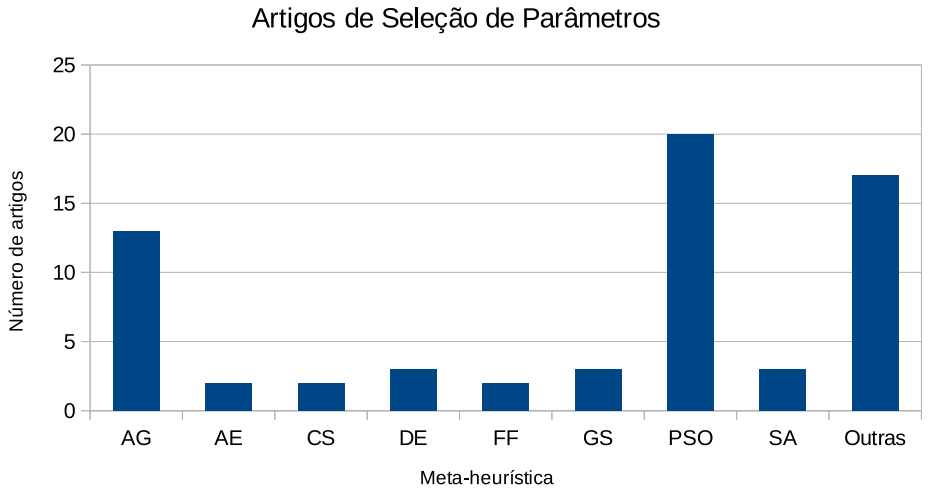


Figura 4.3 – Artigos que trata o PSP-SVM classificados por meta-heurísticas.

O algoritmo PSO, descrito na Seção 3.2.1, é o mais empregado, seja na sua forma clássica ou com modificações para evitar mínimos locais ou melhorar a convergência. Algoritmos baseados no PSO têm mostrado-se tão eficientes que, dos 59 encontrados na literatura *vide* (Tabela 4.1), 20 o envolvem. Na Figura 4.3, pode-se observar que o PSO se destaca dos demais em quantidade, sendo em maior quantidade que o grupo *outros* que consiste na soma de todos as meta-heurísticas que somaram apenas um trabalho encontrado.

O algoritmo *Simulated Annealing* - SA também merece destaque, pois, embora seja considerada uma técnica clássica, ainda hoje é utilizado para resolver problemas NP-Completo. Esse algoritmo meta-heurístico simula o aquecimento de materiais e seu resfriamento controlado com o objetivo de criar cristais perfeitos. Essa técnica é reconhecidamente robusta para diversos tipos de problemas NP-Completo, sendo originalmente desenvolvida para resolver o problema do

caixeiro viajante (KIRKPATRICK et al., 1983). O SA foi utilizado para resolver o problema de seleção de modelos em trabalhos publicados em 2008, 2012 e 2013, conforme a Tabela 4.1.

O Algoritmo Genético (AG) de Holland (1975) é uma meta-heurística inspirada na teoria da seleção natural que afirma que, em determinado ambiente, os indivíduos mais adaptados a ele têm mais chance de se reproduzir e passar seus genes adiante. O AG básico consiste nos operadores *seleção*, *crossover* e *mutação* que atuam sobre uma população de indivíduos, definindo a próxima geração. Uma característica importante do AG é sua capacidade de sair de mínimos locais, característica, em grande parte, atribuída ao operador mutação.

O AG foi adaptado para tratar problemas multiobjetivos, sendo o *Non Sorting Genetic Algorithm* (NSGA-II) a proposta mais popular (DEB et al., 2002). Por exemplo, nesta revisão, dos sete algoritmos multiobjetivos, três envolvem, de alguma forma, o NSGA-II, mostrando que não existe uma variedade de algoritmos multiobjetivos que tratam o problema de seleção de hiperparâmetros das SVMs. Na Figura 4.3, o NSGA-II foi classificado como AG, por aquele ser uma variação deste último.

Entre as técnicas encontradas, algumas não utilizam a filosofia das heurísticas na resolução do problema de seleção de parâmetros, tendo uma natureza determinística. Todavia esses métodos necessitam de um conhecimento prévio dos dados do conjunto de treinamento e, muitas vezes, são computacionalmente inviáveis. Neste cenário, três artigos (CHERKASSKY, 2004; MIRANDA; PRUDÊNCIO, 2013; JI et al., 2017a) foram encontrados corroborando com essa informação.

Ainda na Tabela 4.1, encontram-se sete trabalhos que tratam o problema de otimização como multiobjetivo, mostrando que essa importante teoria vem sendo pouco explorada, e o mais recente deles é o trabalho de Miranda et al. (2012), o que motiva o desenvolvimento de algoritmos MOOP especializados no PSP. Na Figura 4.4, os trabalhos da Tabela 4.1 foram categorizados em multiobjetivo e mono-objetivo, a porção que representa os artigos que tratam o PSP como MOOP representam pouco menos de doze porcento do total.

No trabalho de Igel (2005), é proposto um algoritmo evolutivo autoadaptativo, que atualiza os indivíduos baseado na distribuição normal $N(0, 1)$ que não se altera durante o processo evolutivo. O algoritmo é o primeiro autoadaptativo ES que usa ordenamento não dominante, chamado de NSES. O NSES foi desenvolvido para emprego do *kernel* gaussiano para classificadores e destaca o estudo das possíveis funções objetivo do MOOP. Suttorp e Igel (2006) propõe um outro trabalho, no qual modela o problema de seleção de parâmetros com três funções objetivos e emprega o mesmo algoritmo de (IGEL, 2005) em *benchmarks* e as funções objetivos propostas ao NSGA-II, para classificar a presença de pedestres em imagens de carros autônomos.

Para Chatelain et al. (2007), o NSGA-II é empregado para resolver o problema de seleção de parâmetros das SVMs, em que as funções objetivos consideradas são os Falsos Positivos - FP e Falsos Negativos - FN, ou seja, não avalia complexidade da SVM. No trabalho de Chatelain et al. (2007), as variáveis de decisão do problema são γ do *kernel* Gaussiano e dois parâmetros de regularização, pois trata cada classe independentemente.

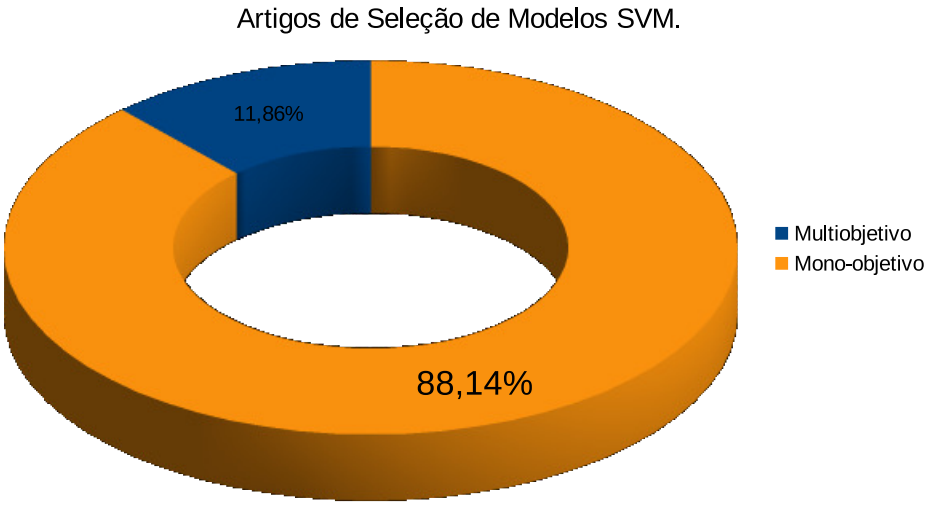


Figura 4.4 – Artigos que trata o PSP-SVM classificados por quantidade de funções objetivos.

O algoritmo *Multi-objective uniform design* (MOUD), proposto por Li, Liu e Gong (2011), emprega a técnica *Uniform Design* sendo esta semelhante ao *Symmetric Latin Hypercube Design* - SLHD. O MOUD não considera complexidade das SVM, mas a sensibilidade e a especificidade como critérios multiobjetivos para obter os hiperparâmetros ótimos das SVMs, e o *kernel* utilizado é o Gaussiano e o polinomial.

Miranda et al. (2012) desenvolveram um algoritmo híbrido em que uma técnica *Meta-Learning* - ML baseada na características dos problemas possa gerar um enxame de soluções de boa qualidade como enxame inicial para o MOPSO. Esse MOPSO emprega o *kernel* Gaussiano para gerar classificadores e os critérios do MOOP são $\#SV$ e a precisão. Uma restrição deste MOPSO é que o espaço de busca foi discretizado limitando-o a 399 diferentes configurações de C e γ, situação que compromete a qualidade dos modelos. Miranda *et al.* propõem um outro trabalho com modificações no processo de obtenção do ML e no MOPSO (MIRANDA et al., 2014).

O NSGA-II também é empregado no trabalho de Narzisi (2008), em que os critérios adotados são $\#SV$ e a precisão. Narzisi G. realiza testes com o *kernel* Gaussiano e sigmoide, para tratar dados não linearmente separáveis.

Os algoritmos multiobjetivos citados nos parágrafos anteriores vide (Tabela 4.1) utilizam, em sua maioria, o NSGA-II, com *kernel* Gaussiano para desenvolver classificadores, ou seja, uma subcategoria do problema. Deve-se observar que, para diferentes *kernels*, têm-se problemas de seleção de parâmetros com características distintas. Adicionalmente, nenhum dos estudos encontrados na literatura abordam a definição de hiperparâmetros das SVRs, o que deixa evidente a necessidade de um estudo mais criterioso sobre o assunto, empregando diferentes *kernels* e algoritmos que resolvam o problema de seleção de parâmetros como um MOOP.

A maioria dos artigos que optam pela abordagem mono-objetivo utilizam o MSE (SVR) e a precisão (SVM) como métrica para direcionar o algoritmo meta-heurístico pelo espaço de busca. Esta prática pode gerar máquinas com boa capacidade de generalização, mas, por outro lado,

muito complexas, impossibilitando a aplicação em situações em que o tempo de processamento e memória são limitados.

Tabela 4.1 – Referências de seleção de parâmetros.

Ref.	**Ano**	**Algoritmo**	**Dados**	**Tipo de Otimização**	**Máquina**	**Saídas**
(JACK; NANDI, 2002)	2002	Algoritmo Genético	Detecção de falhas em rolamentos	Mono	SVM	Binária
(SOARES; BRAZDIL; KUBA, 2004)	2004	*Meta-learning*	Projeto METAL http://www.metal-kdd.org	Mono	SVR	*Single-target*
(CHERKASSKY, 2004)	2004	Método analítico baseado nos dados de treinamento	Artificial	Mono	SVR	*Single-target*
(FRIEDRICHS; IGEL, 2005b)	2005	Estratégia de evolução adaptativa baseada na matriz de covariância – CMA-ES.	Câncer de Mama, Diabetes, Coração e Tiróide	Mono	SVM	Binária
(IGEL, 2005)	2005	Algoritmo Evolutivo	*Benchmarks* UCI	Multi-objetivo	SVM	Multi-classe
(HUANG; WANG, 2006)	2006	GA	*Benchmarks* UCI	Mono	SVM	Multi-classe
(SUTTORP; IGEL, 2006)	2006	Combinação de AE e NSGA-II	Identificação de pedestres	Multi-objetivo	SVM	Binária
(CHATELAIN et al., 2007)	2007	NSGA-II	Reconhecimento de manuscritos	Multi-objetivo	SVM	Multi-classe
(SHAOWU et al., 2007)	2007	*Differential Evolution*	Artificial	Mono	SVR	*Single-target*
(HUANG; DUN, 2008)	2008	PSO	Dados artificiais	Mono	SVM	Multi-classe
(GUO et al., 2008)	2008	PSO	*Benchmarks* UCI	Mono	LS-SVM	Binária
(NARZISI, 2008)	2008	NSGA-II	*Benchmarks* UCI, Stalog, Delve	Multi-objetivo	SVM	Binária
(LORENA; De Carvalho, 2008)	2008	Algoritmo Genético	*Benchmarks* UCI	Mono	SVM	Multi-classe

Continua na próxima página.

Tabela 4.1 – *Continuando da página anterior.*

Ref.	**Ano**	**Algoritmo**	**Dados**	**Tipo de Otimização**	**Máquina**	**Saídas**
(LIN et al., 2008b)	2008	PSO	*Benchmarks* UCI	Mono	SVM	Multi-classe
(LIN et al., 2008a)	2008	*Simulated Annealing*	*Benchmarks* UCI	Mono	SVM	Multi-classe
(ZHANG; GUO, 2009)	2009	PSO	Reconhecimento de voz	Mono	SVM	Multi-classe
(TANG et al., 2010)	2010	PSO	*Benchmarks* UCI	Mono	SVM	Multi-classe
(WU; LAW, 2011)	2011	PSO	Projeto de peças de plástico	Mono	SVR	*Single-target*
(CHEN-GLIN et al., 2011)	2011	PSO	Detecção de falhas em sensores wireless	Mono	SVM	Binária
(WU, 2011)	2011	PSO	Projeto de peças de plástico	Mono	SVR	*Single-target*
(GANG; ZHUPING, 2011)	2011	PSO	Predição de congestionamentos	Mono	SVR	*Single-target*
(LI; LIU; GONG, 2011)	2011	*Multi-objective uniform design*	*Benchmarks* UCI	Multi-objetivo	SVM	Multi-classe
(GOMES et al., 2012)	2012	Busca Tabu e PSO	*Benchmarks*	Mono	SVR	*Single-target*
(Dos Santos et al., 2012)	2012	Differential Evolution	Identificação de processos termais	Mono	SVR/I-dentificação	*Single-target*
(MIRANDA et al., 2012)	2012	MOPSO	*Benchmarks* UCI, WEKA	Multi-objetivo	SVM	Binária
(GASPAR; CARBONELL; OLIVEIRA, 2012)	2012	*Simulated Annealing*	*Benchmarks* UCI	Mono	SVM	Binária
(BONESSO, 2013)	2013	*Simulated Annealing*	Imagens	Mono	SVM	Multi-classe
(MIRANDA; PRUDÊNCIO, 2013)	2013	*Active Testing*	*Benchmarks* UCI	Mono	SVM	Multi-classe
(MIRANDA et al., 2014)	2014	MOPSO	*Benchmarks* UCI	Multi-objetivo	SVM	Multi-classe

Continua na próxima página.

Tabela 4.1 – *Continuando da página anterior.*

Ref.	Ano	Algoritmo	Dados	Tipo de Otimização	Máquina	Saídas
(YOU; GAO; KATAYAMA, 2014)	2014	*Grid Search*	Erros em processos de solda	Mono	SVM	Multi-classe
(ANAM et al., 2014)	2014	PSO	Reconhecimento de movimentos dos dedos	Mono	SVM	Multi-classe
(García Nieto et al., 2015)	2015	ABC	Predição de Cianotoxinas	Mono	SVR	*Single-target*
(SHAMSHIRBAND et al., 2016)	2015	FFA, QPSO	Predição da eficiência de motores Diesel	Mono	SVR	*Single-target*
(YAO; XUE; ZHOU, 2015)	2015	*Grid Search*	Alimentação de arames (Solda)	Mono	SVR	*Single-target*
(SHERIN, 2015)	2015	Algoritmo Genético	Classificação subaquática	Mono	SVM	Multi-classe
(García Nieto et al., 2016)	2016	PSO	Modelo preditivo do ciclo de crescimento da S. Platensis	Mono	SVR	*Single-target*
(CHEN et al., 2016)	2016	Algoritmo Genético	Diagnose de circuitos analógicos	Mono	SVM	Multi-classe
(YI et al., 2016)	2016	*Fruit Fly*	Crescimento de folha de arroz	Mono	SVR	*Single-target*
(PHAN; NGUYEN; BUI, 2016)	2016	Algoritmo Genético	*Benchmarks* UCI	Mono	SVM	Multi-classe
(MAHMOUDI; OROUJI; FALLAHMEHDIPOUR, 2016)	2016	*Shuffled Frog Leaping Algorithm*	Qualidade da água	Mono	SVR	*Single-target*
(BAMAKAN; WANG; RAVASAN, 2016)	2016	PSO	*Benchmarks* UCI	Mono	SVM	Binária

Continua na próxima página.

Tabela 4.1 – *Continuando da página anterior.*

Ref.	Ano	Algoritmo	Dados	Tipo de Otimização	Máquina	Saídas
(GHORBANI; ZARGAR; JAZAYERI-RAD, 2016)	2016	Algoritmo Genético	Precipitação de asfaltenos	Mono	SVR	*Single-target*
(MIN, 2016)	2016	Algoritmo Genético	*Benchmarks* UCI	Mono	SVM	Binária
(RASTGOUFARD; DIMITRIOS, 2016)	2016	HS, ACO, PSO e DE	Avalição de Segurança Estática (Sistema de Potência)	Mono	SVM	Multi-classe
(LI et al., 2016)	2016	Algoritmo Genético	Diagnose de falhas em transformadores	Mono	SVM	Multi-classe
(JI et al., 2017a)	2017	*Ensemble Kalman Filter*	*Benchmarks* UCI, LIBSVM	Mono	SVM	Multi-classe
(THARWAT; MOEMEN; HASSANIEN, 2017)	2017	*Whale Optimization Algorithm*	Classificação de toxidade de remédios	Mono	SVM	Multi-classe
(ZHANG et al., 2017)	2017	*Cuckoo Search*	Previsão da velocidade do vento	Mono	SVR	*Single-target*
(SUKAWATTANAVIJIT; CHEN; ZHANG, 2017)	2017	Algoritmo Genético	Imagens de satélite	Mono	SVM	Multi-classe
(THASEEN; KUMAR, 2017)	2017	Técnica baseada na variância	Detecção de invasão de redes de computadores	Mono	SVM	Multi-classe
(GUI et al., 2017)	2017	PSO, GS, GA	Falhas em estruturas de construção civil	Mono	SVM	Binária
(JI et al., 2017b)	2017	*Ensemble Kalman Filter Scheme* (EnKFS)	*Benchmarks* UCI	Mono	SVM	Multi-classe

Continua na próxima página.

Tabela 4.1 – *Continuando da página anterior.*

Ref.	Ano	Algoritmo	Dados	Tipo de Otimização	Máquina	Saídas
(NIU; DAI, 2017)	2017	EMD-GRA-MPSO-LSSVM	Predição de consumo de energia	Mono	SVR	*Single-target*
(ZHANG; WANG; ZHANG, 2017)	2017	*Cuckoo Search* (CS)	Predição de consumo de energia	Mono	SVR	*Single-target*
(ALWAN; KU-MAHAMUD, 2017)	2017	ACO	*Benchmarks* UCI	Mono	SVM	Multi-classe
(ZENG et al., 2018)	2018	*Switching Delayed* PSO	Diagnose da doença de Alzheimer	Mono	SVM	*Binária*
(FARIS et al., 2018a)	2018	*Multi Verse Optimizer* (MVO)	*Benchmarks* UCI	Mono	SVM	Multi-classe
(DAS; PADHY, 2018)	2018	*Teaching Learning Based Optimization* (TLBO)	Predição de valores de *commodities*	Mono	SVR	*Single-target*
(ALMASI; KHOOBAN, 2018)	2018	*Adaptive* PSO	*Benchmarks* UCI	Mono	SVM	Multi-classe

No contexto da pesquisa bibliográfica, trinta e seis porcento dos trabalhos pesquisados utilizaram o repositório do UCI para realizar seus experimentos, entre eles sete com abordagem multiobjetivo, o que possibilita a comparação direta com os algoritmos APMT-MODE e MOPSO desenvolvidos neste trabalho. O algoritmo NSGA-II possui código-fonte aberto, assim a alteração de suas funções objetivo para tratar o problema de seleção de modelos das SVMs foi implementada e comparada com os algoritmos APMT-MODE e MOPSO. A disponibilidade do código-fonte do NSGA-II também justifica seu emprego em três dos sete trabalhos que utilizam a metodologia multiobjetivo.

Adicionalmente, pode ser observado na Figura 4.5 que dois dos cinco trabalhos pesquisados (CHATELAIN et al., 2007; SUTTORP; IGEL, 2006), ou seja, $28,57\%$ modelam o PSP das SVMs como um MOOP, sendo o NSGA-II, empregado em ambos casos para definir os hiperparâmetros dos modelos de SVMs classificadores. Diante do exposto, conclui-se que existe uma carência no desenvolvimento de meta-heurísticas que abordem o PSP como um MOOP para aplicações

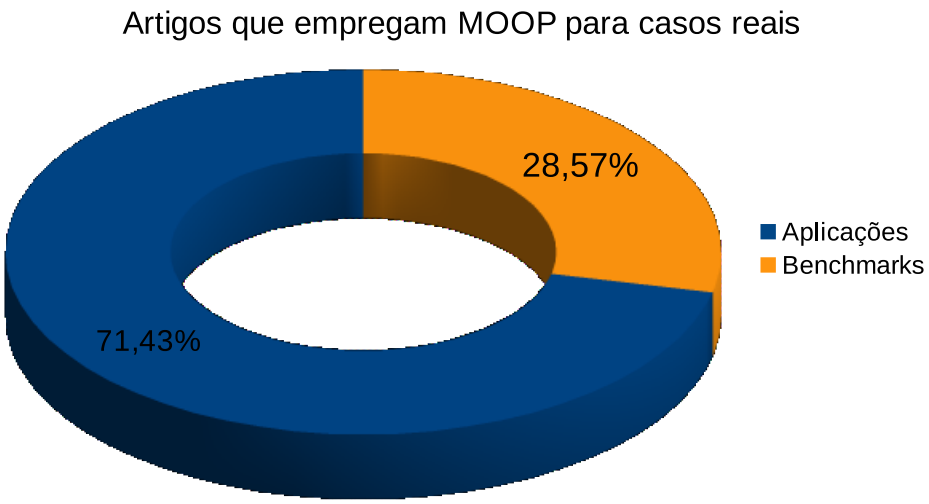


Figura 4.5 – Artigos que modelam o PSP como MOOP para aplicações reais vs. *benchmarks*.

reais em que os modelos possuem saídas são contínuas, principalmente quando estas possuem restrições de energia, recursos físicos (*hardware*) e tempo de resposta do modelo.

Metade dos trabalhos pesquisados e apresentados na Tabela 4.1 utiliza as SVM binária ou SVR *single-targe* em que não há necessidade de alteração no modelo original de Vapnik. A outra metade envolve classificação (SVMs) multiclasse em que são necessárias adaptações ou técnicas adicionais, como, por exemplo, *One-Against-One* (OAO) e *One-Against-All* (OAA). As técnicas que envolvem estudos multiclasses exigem um esforço maior na resolução do problema de seleção de modelo da SVM, pois deve-se atribuir valores aos hiperparâmetros de várias máquinas simultaneamente ou definir hiperparâmetros iguais que atendam ao modelo de forma satisfatória como um todo.

Embora existam técnicas que definem SVRs *multi-target*, não foram encontrados estudos específicos que tratam a seleção de parâmetros dessas máquinas, o que indica que pode ser uma área de pesquisa promissora. Essas máquinas possuem várias aplicações, visto que é comum em aplicações práticas saídas contínuas interdependentes, como, por exemplo, o controle de corrente, a tensão e a alimentação do arame em processos de soldagem (YAO; XUE; ZHOU, 2015).

4.4 MÉTRICAS DE AVALIAÇÃO DA FRONTEIRA DE PARETO - TRABALHOS RELACIONADOS

Um problema MOOP consiste em encontrar a solução ótima de várias funções objetivos contraditórias simultaneamente, como definido na Seção 3.1, e possui como decorrência desta definição um conjunto de soluções não dominantes que são adotadas como igualmente boas. Neste contexto, a fronteira de Pareto consiste em um conjunto de pontos (soluções) não dominantes pertencente ao Espaço Objetivo, cujos elementos devem ser avaliados em conjunto.

Segundo Okabe *et al.* (OKABE; JIN; SENDHOFF, 2003), a fronteira de Pareto é avaliada em três quesitos: (a) cardinalidade da fronteira de Pareto, (b) precisão das soluções contidas no conjunto, ou seja, proximidade da fronteira de Pareto teórica e (c) e distribuição ou espalhamento, sendo estas duas significam características semelhantes, mas, em essência, diferentes. A distribuição refere-se à relativa distância entre as soluções do conjunto não dominante, enquanto que espalhamento refere-se à faixa coberta por estas soluções (RIQUELME; LÜCKEN; BARAN, 2015).

A Tabela 4.2 apresenta as métricas utilizadas por algoritmos multiobjetivos entre os anos de 1998 e 2019, devido à diversidade de métricas utilizadas, conclui-se que o problema de avaliar a qualidade da fronteira Pareto ainda está em aberto, por não apresentar um consenso entre os trabalhos apresentados.

Tabela 4.2 – Revisão de métricas de avaliação de algoritmos multiobjetivo.

Ref.	**Ano**	**Algoritmo**	**Problemas**	**Métricas**
(ZITZLER; THIELE, 1998)	1998	VEGA, NSGA, AVOW, NPGA	*Knaspsack Problem*	Hipervolume
(DEB et al., 2002)	2002	NSGA-II	*Benchmarks*	*Generational distance*, *Spacing*
(SUTTORP; IGEL, 2006)	2006	NSGA-II	UCI, Classificação de Pedestres	*Area Under Curve*, Hipervolume
(ALBERTO; MATEO, 2011)	2009	EA - *Standart*	*Benchmarks* CEC (2007)	Hipervolume, Diferença a um conjunto referência, *Generational distance Set coverage*
(Huang et al., 2009)	2009	*objective-wise* MOSaDE	CEC2009	IGD
(WAGNER; TRAUT-MANN, 2010)	2010	SMS-EMOA	*Benchmarks* (ZDT1, ZDT2, ZDT3, ZDT4)	*Dominated HV*, *Additive* ϵ, R^2 *indicator*
(MATEO; ALBERTO, 2012)	2011	EA	CEC (2007)	Hipervolume, Diferença a um conjunto referência, *Set coverage*
(ALI; SIARRY; PANT, 2012)	2012	*Multi-Objective Differential Evolution Algorithm* (MODEA)	*Benchmarks* (ZDT1, ZDT2, ZDT3, ZDT4)	*Generational distance*, *Diversity metric*, *Spacing*
(FORTIN; PARIZEAU, 2013)	2013	NSGA-IIr	Benchmarks (ZDT1, ZDT2, ZDT3, ZDT4)	*Generational distance*, *Spacing*

Continua na próxima página.

Tabela 4.2 – *Continuando da página anterior.*

Ref.	Ano	Algoritmo	Problemas	Métricas
(LI et al., 2014)	2014	*Multiobjective evolutionary algorithm based on decomposition* (MOEA/D)	CEC2009	IGD, Hipervolume
(DEB; JAIN, 2014)	2014	NSGA-III	*Benchmarks* (DTLZ1, DTLZ3, DTLZ6)	IGD
(LIN et al., 2015)	2015	*Multi-Objective Particle Swarm Optimization* (MOPSO)	*Benchmarks* (ZDT1, ZDT2, ZDT3, ZDT4)	IGD
(MIRJALILI, 2016)	2016	*Dragonfly*	*Benchmarks* (ZDT1, ZDT2, ZDT3)	IGD
(MIRJALILI et al., 2016)	2016	*Grey Wolf Optimizer* (GWO)	*Benchmarks*	IGD, *Spacing*, *Maximum spread*
(MEZA et al., 2017)	2017	MOVPSO	*Benchmarks Binh*, Fonseca, Viennet, Viennet-2	Hipervolume, *Generational distance*
(BEJINARIU; LUCA; COSTIN, 2018)	2018	MOPSO, MOGSA, *Black Hole Algorithm*	Segmentação de imagem	Avaliação gráfica da fronteira de Pareto
(ZAPOTECAS-MARTÍNEZ; GARCÍA-NÁJERA; LÓPEZ-JAIMES, 2019)	2019	MOGWO/D	*Benchmarks* (DTLZ1, DTLZ3, DTLZ6)	Hipervolume, *Inverted Generational Distance plus*

A Figura 4.6 foi obtida a partir da Tabela 4.2, em que se pode verificar que o hipervolume é a métrica mais empregada seguida pela métrica IGD, o que sugere que as duas métricas são bem aceitas pela comunidade. Ainda, na Figura 4.6, observa-se que há uma vasta variedade de métricas, sendo a coluna *Outros* composta por métricas que aparecem uma única vez na Tabela 4.2. Entretanto, segundo a pesquisa realizada em (RIQUELME; LÜCKEN; BARAN, 2015), a métrica hipervolume é a mais utilizada em trabalhos que envolvem algoritmos evolutivos, enquanto que a métrica IGD ocupa a quarta colocação.

O hipervolume é calculado sem que seja necessário o conhecimento da fronteira de Pareto, bastando definir um ponto conhecido como *nadir* (pior solução possível) e tendo este como vér-

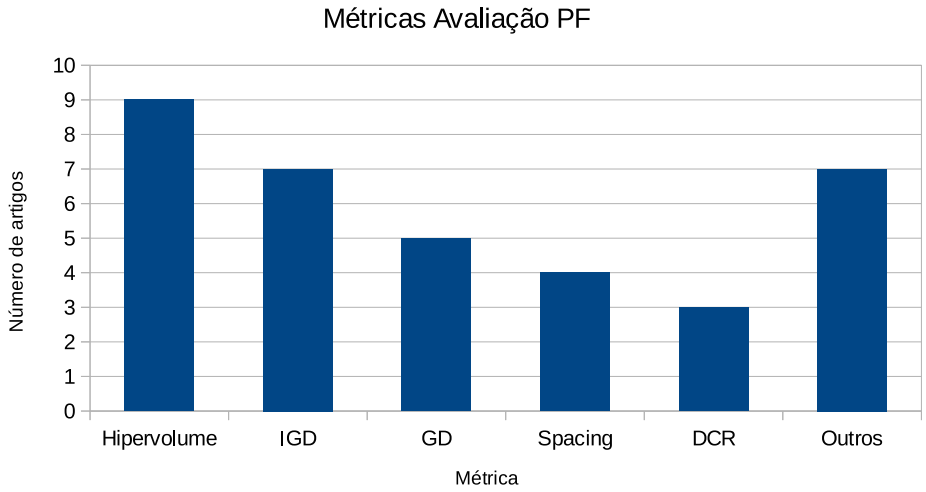


Figura 4.6 – Classificação das métricas por trabalho pesquisado.

tice calcular o hipervolume do poliedro gerado pelo conjunto candidato à fronteira de Pareto. Essa métrica é unária, ou seja, retorna um único valor escalar que representa simultaneamente a diversidade e a precisão das soluções do conjunto em avaliação. Quanto maior o hipervolume, maior a diversidade do conjunto e mais distante do ponto *nadir*, portanto, mais próximo da solução ótima.

O cálculo da métrica hipervolume, segundo Bader (BADER, 2010), na melhor das hipóteses, possui complexidade máxima de $\mathcal{O}\left((d/2)^N\right)$, em que d é a dimensionalidade do espaço objetivo e N a cardinalidade do conjunto em avaliação. Ainda segundo Bader (BADER, 2010), o cálculo do hipervolume consiste em um problema não polinomial. Diante do exposto e dos resultados obtidos durante a elaboração deste trabalho, o hipervolume não apresentou um comportamento esperado para avaliar os algoritmos implementados, pois, para todos os algoritmos, o hipervolume os descreve como tendo uma convergência com menos de 30 iterações. Entretanto, quando observada as fronteiras geradas nas iterações finais (maior que cem), os conjuntos de soluções não dominantes são significativamente melhores em valores absolutos, ou seja, o hipervolume gera um falso positivo para convergência dos algoritmos. A Figura 4.7 apresenta o comportamento descrito.

Para o cálculo da métrica IGD, é necessário que seja conhecida a fronteira de Pareto. Esta métrica é unária e, assim como o hipervolume, avalia a diversidade e a qualidade do conjunto candidato a fronteira de Pareto. Na Tabela 4.2, todos os trabalhos que empregam o IGD como avaliação utilizou como problema *benchmarks* em que as fronteiras de Pareto são conhecidas, evidenciando que esta métrica é empregada com certo sucesso apenas nas situações em que se conhece a fronteira de Pareto.

Neste trabalho, para gerar o conjunto de soluções não dominante mais próximo possível da fronteira Pareto, para cada problema, foram reunidos todos os conjuntos de solução não dominantes de todos os algoritmos a serem avaliados em um único conjunto, então, foi empregada a função *Truncate*, para determinar o conjunto de soluções não dominantes de referência, para

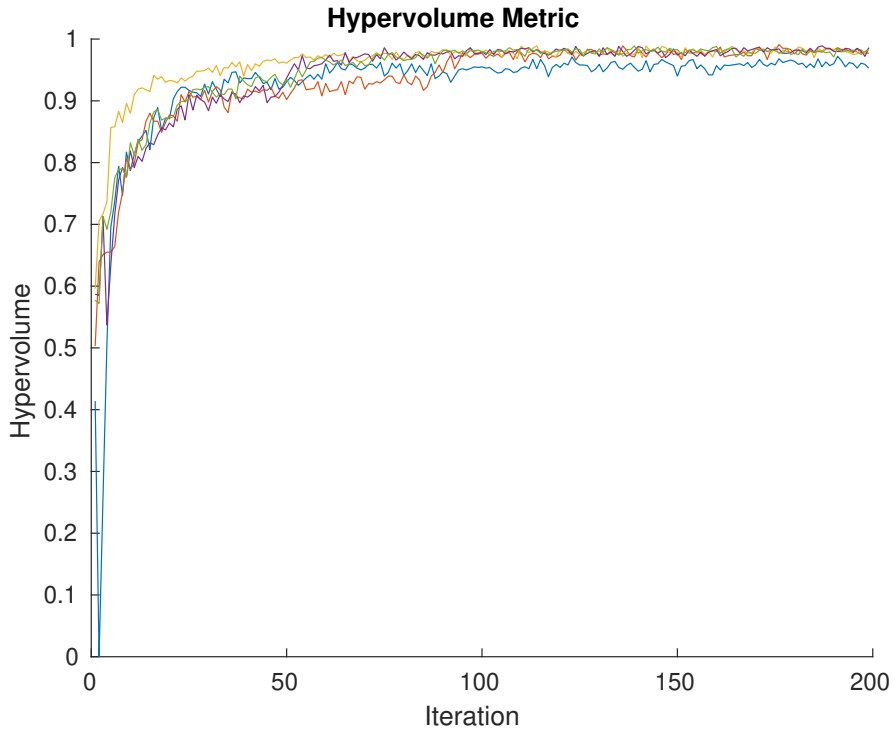


Figura 4.7 – Hipervolume obtido pelo MOPSO para o *benchmark* Ecoli.

ser utilizado pelo IGD como fronteira de Pareto. Essa estratégia permite avaliar os conjuntos de soluções encontradas por cada algoritmos em relação ao melhor conjunto encontrado entre todos.

Nos trabalhos pesquisados, Tabela 4.1, em nenhum trabalho, foi encontrada a técnica CLO, pois, segundo Coello e Veldhuizen (2007), a principal desvantagem do CLO é a aleatoriedade da função objetivo, que, no caso de MOOP, com grande número de funções objetivo, a aleatoriedade pode tender a determinada função ou grupo delas. Porém, para o PSP modelado como um MOOP, têm-se duas funções objetivo o que não caracteriza a desvantagem citada do CLO. Por outro lado, o CLO possui vantagem de tratar problemas que possuem a fronteira de Pareto convexa, formato que tem apresentado as fronteiras dos problemas tratados neste trabalho.

4.5 CONCLUSÕES DO CAPÍTULO

Os algoritmos meta-heurísticos são amplamente utilizados na resolução de problema multi-objetivos, em parte devido ao sucesso que obtiveram no tratamento de problemas NP-Completo Mono-objetivo. Para o problema de seleção de parâmetros das SVMs, dos 61 artigos encontrados sobre o assunto *vide* (Tabela 4.1), sete tratam o problema como um MOOP.

Por exemplo, para solução de problemas multimodais mono-objetivos, foi desenvolvida uma imensa variedade de técnicas de adição de diversidade com o objetivo de encontrar o equilíbrio entre a exploração do espaço de busca e o refinamento das soluções. Para problemas multiobjeti-

vos, praticamente foram implementados apenas os algoritmos básicos das meta-heurísticas, com pouca ou nenhuma adição de diversidade, ou técnicas para ajustes de parâmetros.

As métricas de avaliação da qualidade do conjunto de soluções não dominantes encontradas pelos algoritmos meta-heurísticos têm-se mostrado um desafio, seja devido à complexidade computacional ou à dependência da fronteira de Pareto conhecida. Entretanto é possível gerar um conjunto de soluções não dominantes que é usado como referência para definir quais algoritmos possuem um desempenho melhor quando utilizada a métrica IGD.

A pesquisa bibliográfica também expõe a carência de algoritmos meta-heurísticos que tratam o PSP como um MOOP, que foram desenvolvidos para geração de modelos SVM/SVRs para serem utilizados em situações reais, principalmente em relação a aplicações práticas em que os modelos são aproximadores de funções. Neste sentido, destaca-se a relevância dos algoritmos desenvolvidos neste trabalho, que possibilitam o ajuste dos hiperparâmetros das SVM/SVRs buscando o compromisso entre a complexidade e a precisão.

Minimizar os critérios complexidade e capacidade de generalização dos modelos SVM/SVRs possui grande relevância em aplicações em que há restrição de energia, disponibilidade de *hardware* e, ainda, se preza pela precisão do modelo, condições estas que motivaram o desenvolvimento das meta-heurísticas APMT-MODE, AP-MODE e MOPSO.

A maioria dos trabalhos apresentados na Tabela 4.1 utilizam o *kernel* Gaussiano para levar os dados ao espaço característica de dimensão superior em que os dados são linearmente separáveis. Entretanto, para cada *benchmark* ou aplicação, existe uma função *kernel* mais eficiente para cada caso, quando considerados os critérios complexidade e precisão simultaneamente. Nenhum dos trabalhos multiobjetivo pesquisados mencionam o uso de diversas funções *kernel*, ou mesmo, um estudo de qual é mais eficiente para o problema abordado, o que mostra que há uma lacuna a ser preenchida, o que expõe os questionamentos: Qual função *kernel* é mais apropriada para determinado conjunto de treinamento, considerando os objetivos complexidade e precisão? Quais os hiperparâmetros adequados para cada PSP, quando combinados, os parâmetros do *kernel* e o conjunto de treinamento?

Toda função, cuja lei de formação, que satisfaça as condições do Teorema de Merce, pode ser empregada como *kernel* para um modelo SVM/SVRs, ou seja, existem inúmeros *kernels* com características diversas, em cuja lei de formação podem ser empregadas operações trigonométricas, operações aritméticas, exponenciais, constantes irracionais, raízes e outras. Portanto, reduzir as funções *kernels* a operações comuns ou definir métricas para comparar a complexidade destas funções é uma tarefa que ainda não foi abordada na literatura, tendo em vista que os poucos trabalhos que consideram comparar a complexidade entre modelos de *kernels* distintos o fazem considerando a quantidade de vetores de suporte do modelo, o que computacionalmente não é uma comparação justa.

Como solução para comparação da complexidade entre os modelos de SVM/SVRs com *kernels* diferentes, neste trabalho, os modelos foram implementados em linguagem de programação C e, então, os modelos foram executados no ARM e tomados seus tempos de execução.

5 ALGORITMOS DESENVOLVIDOS: APMT-MODE E MOPSO

Neste capítulo são apresentados os algoritmos APMT-MODE, AP-MODE e MOPSO, desenvolvidos para encontrar o *Conjunto* e a *Fronteira de Pareto* para o PSP das SVM/SVRs, bem como as modificações realizadas nas estratégias utilizadas para aumentar a eficiência dos algoritmos em encontrar hiperâmetros que gerem modelos de SVM/SVRs que atendam às restrições de aplicações.

5.1 OTIMIZAÇÃO MULTIOBJETIVO POR ENXAME DE PARTÍCULAS - MOPSO

O PSO básico é um algoritmo inspirado na natureza que simula o comportamento social de pássaros e cardumes de peixes. Devido à sua simplicidade na implementação e na rápida convergência, tem atraído o interesse da comunidade científica, sendo utilizado para resolver diversos problemas aplicados à engenharia (LIN et al., 2015). Portanto, devido às características mencionadas, o PSO foi adaptado para resolver problemas multiobjetivos. As principais modificações aplicadas ao PSO básico foram: (a) estratégia de inicialização das partículas; (b) aplicação da técnica de CLO para determinação do melhor individual e global.

Essas adaptações consistem em contribuições distintas para o PSO desenvolvido denominado como *Multi-Objective Particle Swarm Optimization* - MOPSO, sendo cada uma delas descritas com detalhes no decorrer desta seção.

A Figura 5.1 é o diagrama de fluxo do MOPSO, em que o passo *Inicialização*, os parâmetros NP, w_0, w_f, c_1, c_2 e v_{max}, o tamanho da população, o fator de inércia inicial e final, os coeficientes cognitivos e social e a velocidade máxima, respectivamente, são inicializados.

Muitas implementações do MOPSO usam a distribuição uniforme para inicializar as partículas, o que não garante um espalhamento adequado delas sobre o espaço de busca. Assim, para gerar o enxame inicial, foi usado o método denominado *Symmetric Latin Hypercube Design* - SLHD (ZHAO et al., 2016). O pseudocódigo do SLHD é apresentado na Algoritmo 3.

O SLHD gera uma matriz de números aleatórios simetricamente distribuídos entre $[0, 1]$ em todas as dimensões do espaço de busca. Assim, a distribuição evita eventuais concentrações em determinadas regiões do espaço de busca, que pode ocorrer com a distribuição uniforme.

Para evidenciar a diferença entre a população gerada pela distribuição uniforme e o SLHD, foi utilizada a métrica de diversidade do método de adição de diversidade AR para avaliar as populações iniciais. Na Figura 5.2a, os pontos representam as partículas geradas pelo SLHD, enquanto que, na Figura 5.2b, os pontos representam as partículas geradas pela distribuição uniforme.

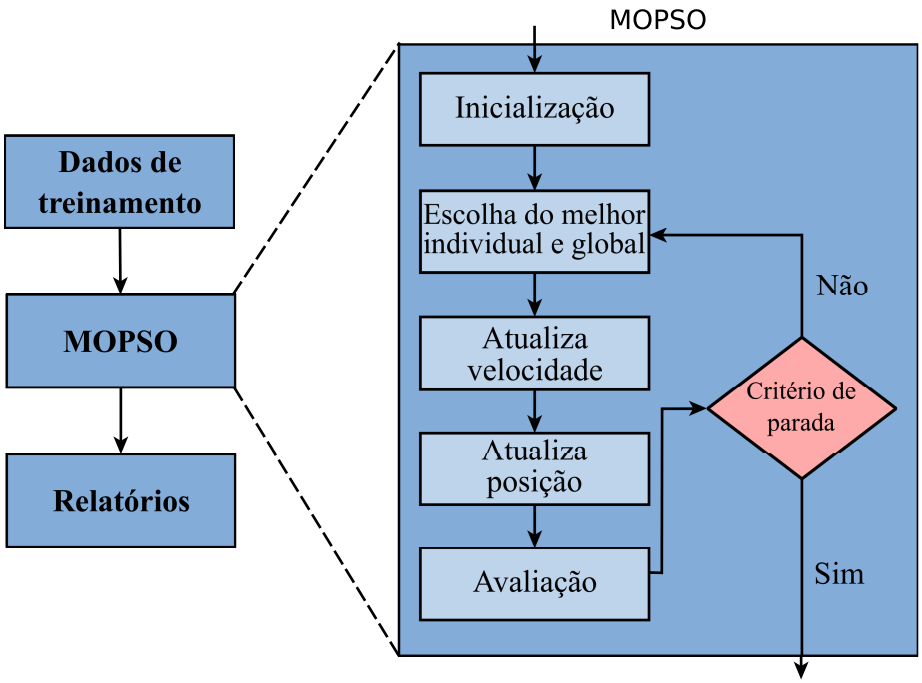


Figura 5.1 – Diagrama de fluxo do algoritmo MOPSO.

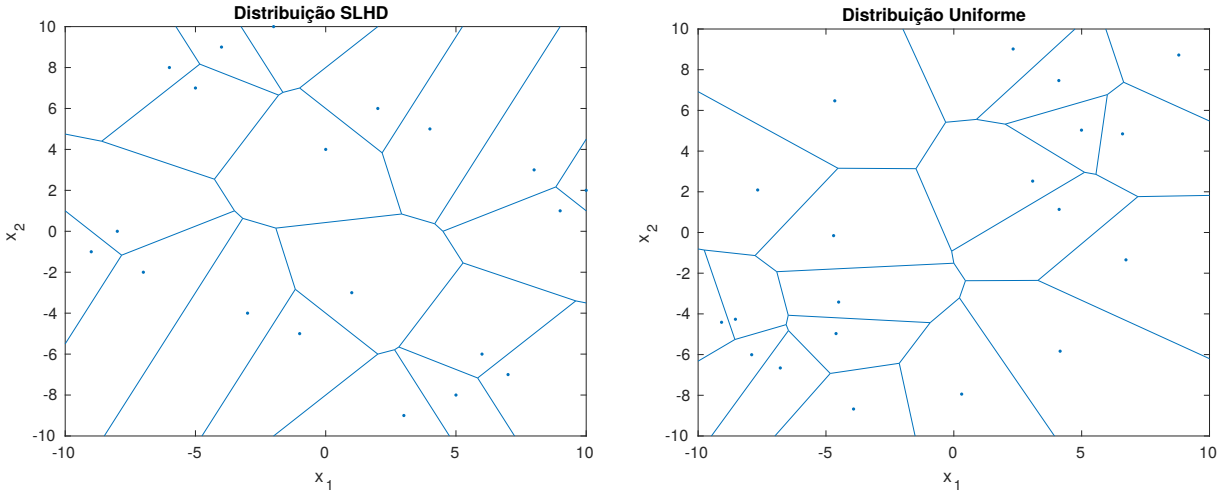
Algorithm 3: Pseudocódigo do algoritmo SLHD.

Input: NP, n

Início:
Inicializa matriz $NP \times n$
if NP for ímpar **then**
  $M((NP+1)/2, j) = (NP+1)/2$, para $j = 1, ..., n$.
end if
for $j = 1 : n$ **do**
  Escolha aleatoriamente a permutação de $1, ..., k$ e denote-a por ϕ_j
end for
for cada par (i, j), em que $i = 1, ...k$ e $j = 1, ..., n$ **do**
  Gere um número aleatório com distribuição uniforme $w_{ij} \in [0, 1]$.
  if $w_{ij} \leq 0,5$ **then**
    $M(i, j) = \phi_j(i)$
    $M(NP + 1 - i, j) = NP + 1 - \phi_j(i)$
  else
    $M(i, j) = NP + 1 - \phi_j(i)$
    $M(NP + 1 - i, j) = \phi_j(i)$
  end if
end for
for $j = 1 : n$ **do**
  $\pi_j = M(:, j)$
  Particione o intervalo $[a_j, b_j]$ em NP subintervalos de tamanhos iguais, sendo $c_j^{(i)}$ o ponto médio de $[a_j, b_j]$.
end for
for $I = 1 : NP$ **do**
  o i-ésimo ponto SLHD é dado por $(c_1^{(\pi_1(i))}, c_2^{(\pi_2(i))}, ..., c_n^{(\pi_n(i))})$
end for

Os diagramas da Figura 5.2 são denominados de diagrama de Voronoi, sendo composto por polígonos convexos onde cada um é formado pela interseção entre as mediatrizes dos pontos mais próximos de um dado ponto fixo.

Na Figura 5.2a, os polígonos são mais uniformes e com áreas semelhantes, pois a distribuição simétrica produz polígonos simétricos aos pares, produzindo, assim, uma distribuição das partículas melhor que a distribuição uniforme, apresentada na Figura 5.2b.



(a) População obtida pelo SLHD.

(b) População obtida pela distribuição uniforme.

Figura 5.2 – Comparação entre SLHD e distribuição uniforme.

A métrica diversidade, dada pela Equação 3.13, foi utilizada para avaliar a distribuição da população inicial gerada pelo SLHD e pela distribuição uniforme. Foram geradas 32 amostras independentes de populações por cada estratégia e calculada suas respectivas média, mediana e desvio-padrão (DP) da métrica diversidade para os dois casos. Estes dados são apresentados na Tabela 5.1.

Tabela 5.1 – Comparação da métrica *diversidade* entre distribuição uniforme e SLHD.

Distribuição	**Média**	**Mediana**	**DP**
SLHD	0,276	0,276	1,13E-16
Uniforme	0,208	0,208	5,64E-17

Os dados da Tabela 5.1 mostram que o SLHD gera populações com diversidade média e mediana maior que a distribuição uniforme, enquanto o desvio-padrão do SLHD é menor. Isto indica que o SLHD gera populações iniciais melhores distribuídas, na maioria das vezes, em relação à distribuição uniforme. O SLHD não acrescenta grande esforço computacional ao MOPSO, pois é utilizado apenas para gerar a população inicial.

O próximo passo consiste em definir as partículas, *melhor individual* e *melhor global*, ou seja, a melhor posição visitada por cada indivíduo e a melhor partícula já encontrada pelo enxame respectivamente, pois a velocidade de cada partícula depende da identificação destas partículas.

Entretanto, devido ao conceito de dominância, a melhor partícula pode ser qualquer uma do conjunto de soluções não dominadas.

A melhor partícula individual é obtida ordenando por *rank* de não dominância (função *Truncate*), todas as posições já visitadas pela partícula, e entre as partículas que possuem *rank* igual é escolhida aquela que possui menor função objetivo definida pela técnica CLO. A melhor partícula global é obtida a partir de um conjunto criado com todas as soluções não dominantes visitadas pelo algoritmo até a iteração atual, deste conjunto de soluções não dominantes, uma é escolhida de acordo com o critério (função objetivo) definido pelo CLO.

O passo *Atualiza velocidade* define a direção, o sentido e o módulo do vetor velocidade de cada partícula do enxame. O vetor velocidade é calculado pela Equação 3.8.

Em seguida, o passo *Atualiza posição* movimenta a partícula a partir da posição atual no sentido, na direção e no módulo do vetor velocidade v_{ij}^{t+1}, conforme Equação 3.9.

No passo *Avaliação*, cada partícula representa os hiperparâmetros de uma SVM/SVR. Os modelos são gerados pela ferramenta LibSVM (CHANG; LIN, 2011), que utiliza os hiperparâmetros juntamente com os dados de treinamento para gerar o problema de otimização e resolvê-lo, entregando como solução os modelos de SVM/SVR. O modelo gerado pela LibSVM é avaliado com os dados de validação e produz os valores das funções objetivos, que medem a capacidade de generalização do modelo e sua complexidade. O critério de parada é o número máximo de iterações definido pelo usuário.

No MOPSO, a posição do enxame na iteração $t + 1$ não é aceita diretamente, como ocorre no PSO básico, ao invés disso, a matriz que representa o enxame na nova posição é concatenada ao enxame da posição atual, formando uma matriz de ordem $2NP \times n$. Então, essa matriz é processada pela função *Truncate* que faz ranqueamento das soluções não dominadas e as NP melhores soluções consistem no posicionamento do enxame na iteração $t + 1$. Assim, de uma iteração para outra, somente as partículas com posições melhores e não dominadas são aceitas. Nesse processo, as partículas tendem sempre a melhorar seus valores funcionais, tendo uma grande capacidade de refinamento das soluções.

Na Seção 5.2, uma versão do DE foi adaptada para tratar o problema de seleção de parâmetros como um MOOP. Este algoritmo possui dois parâmetros ajustados por uma técnica adaptativa, assim como são empregadas diferentes estratégias de mutação.

5.2 PARÂMETROS ADAPTATIVOS COM TORNEIO MULTIOBJETIVO EVOLUÇÃO DIFERENCIAL - APMT-MODE

O algoritmo de *Parâmetros Adaptativos com Torneio Multiobjetivo Evolução Diferencial - APMT-MODE* foi baseado no algoritmo DE básico proposto por Storn and Price (STORN;

PRICE, 1997b) e no processo adaptativo desenvolvido originalmente por Fan e Zhang (FAN; ZHANG, 2016).

O DE básico possui algumas características que são desejadas em aplicações práticas, tais como habilidade de tratar funções objetivos não diferenciáveis, não lineares e multimodais, capacidade de paralelização para trabalhar com funções complexas, poucos parâmetros para serem ajustados, alta capacidade de convergência para soluções de boa qualidade em várias execuções independentes (STORN; PRICE, 1997b).

As modificações realizadas no DE básico objetivam contornar algumas dificuldades que este possui na solução do problema de seleção de parâmetros, tais como: (a) diferentes espaços de busca obtidos pela combinação dos conjuntos de treinamento e *kernels* disponíveis e (b) problema com dois objetivos contraditórios.

Na dificuldade (a), cada conjunto de treinamento das SVM/SVR possui características diferentes e, portanto, exige ajustes distintos do parâmetros Cr, F e, também, diferentes estratégias de mutação e recombinação, então, foram utilizadas técnicas de adaptação propostas por (FAN; ZHANG, 2016). No item (b), a função *Truncate* é utilizada para definir os conjuntos de soluções não dominadas e a técnica CLO para definir os melhores indivíduos.

O fluxograma de solução do problema de seleção de parâmetros é apresentado na Figura 5.3. O sistema representado pelo fluxograma é iniciado fornecendo os dados de treinamento para o bloco *Otimizador*, que gera o Conjunto e a Fronteira de Pareto correspondentes, assim como seus modelos. As linhas pontilhadas no bloco Otimizador evidenciam, no fluxograma, o algoritmo APMT-MODE, descrito em detalhes nos parágrafos subsequentes.

(a) *Inicialização*: os indivíduos da população inicial possuem importante papel na qualidade final das soluções, pois uma distribuição pobre dos indivíduos sobre o espaço de busca pode induzir o algoritmo para mínimo local. Várias implementações do DE usam a distribuição uniforme para gerar a população inicial, o que não garante uma espalhamento razoável sobre o espaço de busca. Assim, para gerar a população inicial, foi usado o SLHD (ZHAO et al., 2016), como descrito na Seção 5.1.

(b) *Adaptação dos parâmetros*: os parâmetros F e Cr são atualizados pela Equação 5.1 e Equação 5.2 respectivamente.

$$F_i^{G+1} = N\left(F_w^{G+1}, \sigma\right) \tag{5.1}$$

$$Cr_i^{G+1} = N\left(Cr_w^{G+1}, \sigma\right) \tag{5.2}$$

em que $\sigma = 0,1 + 0,40 \times (1 - (G/G_{max})^2)$ é a variância, F_w^{G+1} e Cr_w^{G+1} são as médias da distribuição Normal definidas pela Equação 5.3 e Equação 5.4 respectivamente,

$$F_w^{G+1} = \sum_{i=1}^{NP} w_i^G \times F_i^G \tag{5.3}$$

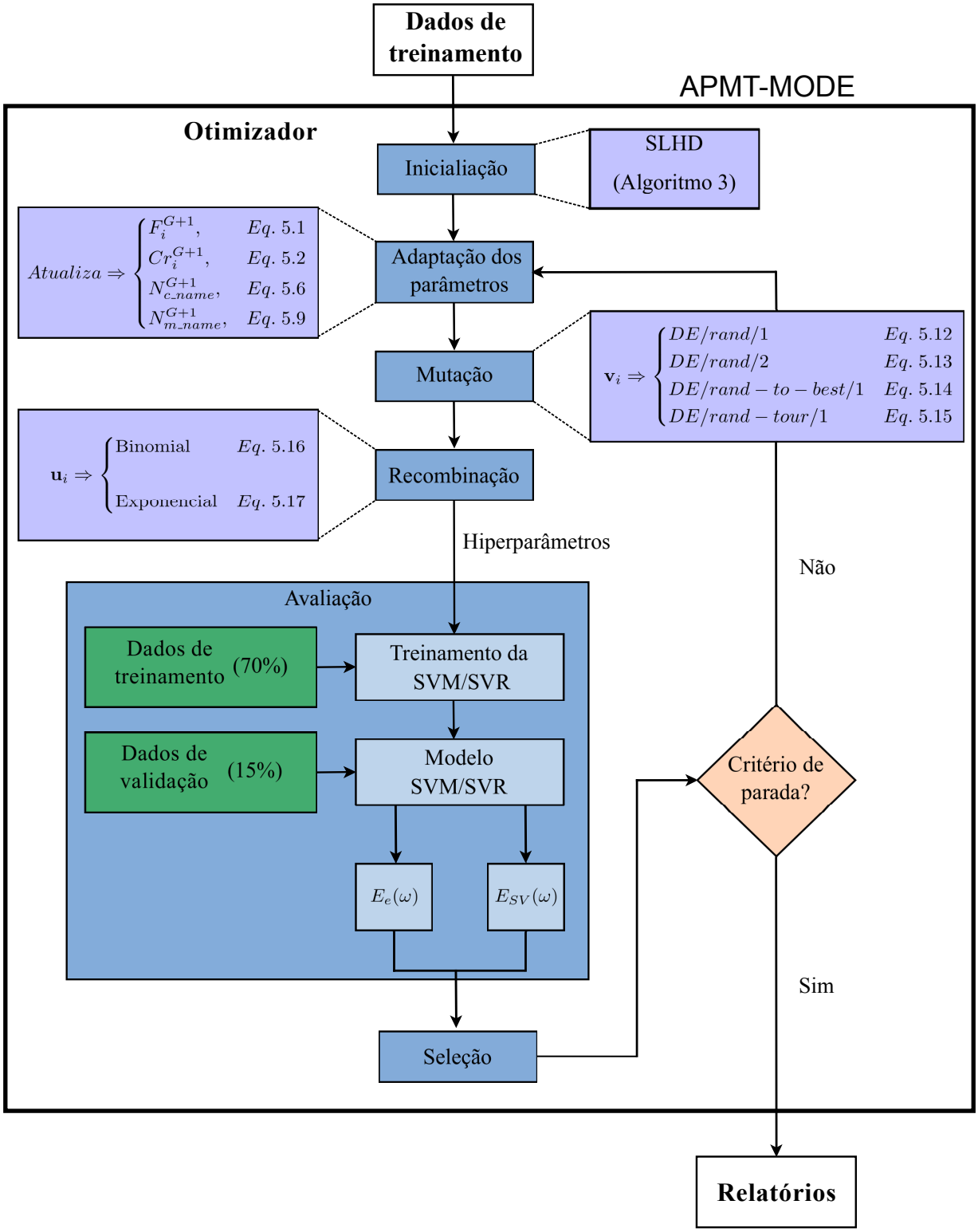


Figura 5.3 – Diagrama de fluxo do algoritmo APMT-MODE.

$$Cr_w^{G+1} = \sum_{i=1}^{NP} w_i^G \times Cr_i^G \tag{5.4}$$

em que w_i^{G+1} é calculado por Equação 5.5

$$w_i^{G+1} = \frac{\left| f_j\left(\mathbf{x}_i^G\right) - \max\left(f_j\left(\mathbf{x}^G\right)\right)\right|}{\sum_{i=1}^{NP}\left|f_j\left(\mathbf{x}_i^G\right) - \max\left(f_j\left(\mathbf{x}^G\right)\right)\right|}, \tag{5.5}$$

em que w_i^{G+1} é a média ponderada do parâmetro de controle e o peso do parâmetro de recombinação (Cr_i^{G+1}) é calculado por Equação 5.4.

O parâmetro σ controla a variância da distribuição Normal e decresce quadraticamente de $0,5$ a $0,1$, diminuindo, variedade dos vetores mutantes nas iterações finais do processo de otimização. Este comportamento aumenta a capacidade de refinamento do APMT-MODE, conforme apresentado na Figura 5.4.

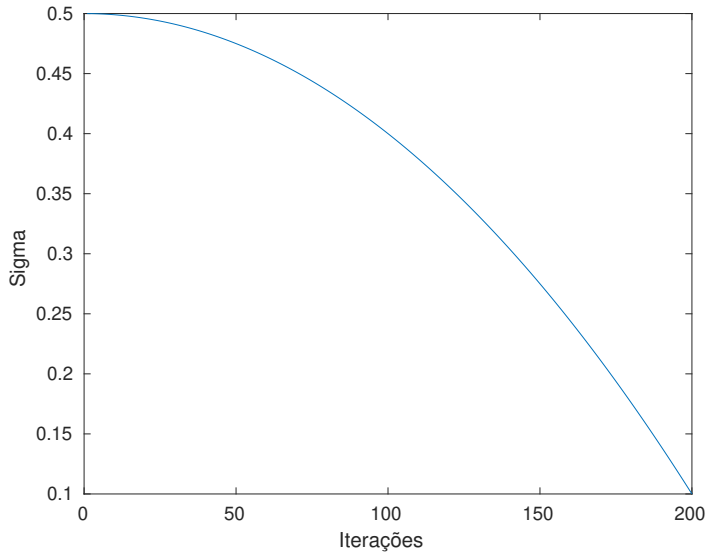


Figura 5.4 – Comportamento da variância σ.

Nesta etapa, também é definida, cardinalidade dos subgrupos de indivíduos que são afetados por cada estratégia de recombinação, calculada pela Equação 5.6.

$$N_{c\_name}^{G+1} = \begin{cases} N_{c\_name}^G + 1, \text{ se } N_{c\_name}^{G+1} > N_{c\_name}^G \\ N_{c\_name}^G - 1, \text{ se } N_{c\_name}^{G+1} < N_{c\_name}^G \\ N_{c\_name}^G, \quad \text{caso contrário} \end{cases} \tag{5.6}$$

sendo G a geração atual, c_name o nome da estratégia de recombinação e $N_{c\_name}^{G+1}$ calculado pela Equação 5.7

$$N_{c\_name}^{G+1} = round\left(PS \times \frac{S_{c\_name}^{G+1}}{S^{G+1}}\right), \tag{5.7}$$

sendo PS o número de indivíduos na população, $round$ é uma função de arredondamento,

$S^{G+1} = S^{G+1}_{cross\_bin} + S^{G+1}_{cross\_exp}$ e $S^{G+1}_{c\_name}$ é a soma das diferenças calculadas pela Equação 5.8

$$S^{G+1}_{c\_name} = \sum_{k=1}^{N^G_{c\_name}} \left| f_j\left(\mathbf{x}^G_{c\_name,k}\right) - \max\left(f_j\left(\mathbf{x}^G\right)\right)\right| \tag{5.8}$$

em que $f_j\left(\mathbf{x}^G_{c\_name,k}\right)$ é o valor funcional da partícula da k-ésima partícula do conjunto referente à estratégia de recombinação c_{name}, para a j-ésima função objetivo.

Finalmente, o *Adaptação dos Parâmetros* também designa o melhor número de indivíduos para cada estratégia de mutação, a cada cinco iterações. Esta quantidade é definida pela Equação 5.9

$$N^{G+1}_{m\_name} = \begin{cases} N^G_{m\_name} + 1, \text{ se } N^{G+1}_{m\_name} > N^G_{m\_name} \\ N^G_{m\_name} - 1, \text{ se } N^{G+1}_{m\_name} < N^G_{m\_name} \\ N^G_{m\_name}, \quad \text{caso contrário} \end{cases} \tag{5.9}$$

sendo $N^{G+1}_{m\_name}$ calculado pela Equação 5.10,

$$N^{G+1}_{m\_name} = round\left(PS \times \frac{S^{G+1}_{m\_name}}{S^{G+1}}\right) \tag{5.10}$$

em que S^{G+1} é a diferença entre o valor objetivo de cada indivíduo e o máximo valor objetivo da população, como mostrado na Equação 5.11

$$S^{G+1}_{m\_name} = \sum_{k=1}^{N^G_{m\_name}} \left| f_j\left(\mathbf{x}^G_{m\_name,k}\right) - \max\left(f_j\left(\mathbf{x}^G\right)\right)\right|, \tag{5.11}$$

em que m_name são os índices que representam a estratégia de mutação, o f_i é a mesma função empregada na estratégia de recombinação.

(c) *Mutação*: produz NP mutantes distribuídas entre as estratégias de mutação definidas pela Equação 5.9. As estratégias de mutação utilizadas no APM-MODE são apresentadas nas equações 5.12 - 5.15,

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) \tag{5.12}$$

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F_i\left(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\right) \tag{5.13}$$

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i\left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F_i\left(\mathbf{x}_{best} - \mathbf{x}_i\right) \tag{5.14}$$

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F_i\left(\mathbf{x}_{t\_best} - \mathbf{x}_i\right) \tag{5.15}$$

em que r_1, r_2, r_3, r_4 e r_5 são números aleatórios mutuamente diferentes escolhidos da população de indivíduos da geração atual, $\mathbf{x}_{best}$ é o melhor indivíduo da população atual e $\mathbf{x}_{t\_best}$ é o indivíduo com a melhor avaliação da função objetivo de um subconjunto escolhido aleatoriamente da população. Note que o parâmetro F_i é ajustado pelo passo (b) do algoritmo.

(d) *Crossover*: recombinação binomial verifica cada posição do vetor mutante e o altera, como descrito pela Equação 5.16.

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, & \text{se } rand_{i,j} \leq Cr \quad \text{ou} \quad j = j_{rand} \\ x_{ij}^{G}, & \text{caso contrário,} \end{cases} \tag{5.16}$$

em que $\mathbf{x}$ são os vetores pais, $\mathbf{u}$ são os vetores filhos, j é o *j*-ésimo posição do *i*-ésimo indivíduo, $rand_{i,j}$ é um número aleatório de distribuição uniforme que pertence aos intervalo $[0, 1]$, e j_{rand} é um inteiro escolhido aleatoriamente na faixa $[1, D_m]$ em que D_m é a dimensionalidade do indivíduo. A condição $j = j_{rand}$ assegura que, ao menos, uma posição dos filhos tenha uma informação do vetor mutante. A recombinação exponencial divide o vetor mutante e o pai em uma posição aleatória e troca suas partes para criar um novo vetor como definido pela Equação 5.17:

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^{G+1}, & \text{se } j = \langle n \rangle_{D_m}, \langle n+1 \rangle_{D_m}, ..., \langle n+L-1 \rangle_{D_m} \\ x_{ij}^{G}, & \text{caso contrário,} \end{cases} \tag{5.17}$$

em que $\langle\rangle_D$ denota o operador módulo com módulo D. O número L é um inteiro obtido do intervalo $[1, D_m]$ e n é um número inteiro.

Vale observar que as equações 5.16 e 5.17 são aplicadas sobre conjuntos disjuntos escolhidos aleatoriamente da população com NP indivíduos. Inicialmente, os subconjuntos possuem a mesma cardinalidade, e depois, a cada cinco iterações, a cardinalidade dos subconjuntos são atualizadas, neste caso, dependendo da Equação 5.6.

(e) *Avaliação*: neste passo, os indivíduos representam os hiperparâmetros necessários para treinar uma SVM/SVR. O treinamento de uma Máquina de Vetores de Suporte consiste em resolver um problema de otimização Equação 2.20 para um dado conjunto de treinamento. O modelo é gerado pela biblioteca LibSVM (CHANG; LIN, 2011), é validado com os conjunto de dados de validação. Este processo permite calcular o $E_e(\omega)$, ou seja, o MSE para as SVRs e a precisão para as SVMs, assim como a cardinalidade do conjunto de vetores de suporte ($E_{SV}(\omega)$). O passo *Avaliação* é descrito pelo diagrama de fluxo da Figura 5.3.

(f) *Seleção*: os indivíduos filhos são concatenados ao conjunto população e, então, a função *Trucante* é aplicada à matriz com dimensão duas vezes o tamanho da população. Então, os indivíduos, ordenados por *ranking* em ordem crescente, são escolhidos até a quantidade preestabelecida de indivíduos da população. Por exemplo, seja uma população de vinte indivíduos e o conjunto de indivíduos do *rank 1*, da função *truncate*, igual a quinze, então inclui cinco indivíduos do *rank 2*.

O processo iterativo repete-se após o passo (f) voltando ao passo (b) até que o número de iterações preestabelecido seja alcançado.

A notação das estratégias de mutação seguem o padrão $DE/x/y$, em que DE designa o algoritmo Evolução Diferencial, x nomeia o vetor a ser alterado pelo operador mutação e y é quantidade de vetores diferenças (STORN; PRICE, 1997b). As estratégias de mutação utilizadas são apresentadas nas equações 5.12, 5.13, 5.14 e 5.15 denominadas *DE/rand/1*, *DE/rand/2*, *DE/rand-to-best/1*, *DE/rand-tournament/1* respectivamente.

A estratégia de mutação definida na Equação 5.15 é denominada *torneio*. Esta estratégia de mutação busca o balanço entre a exploração e o refinamento. Primeiro, um subconjunto da população é criado com vinte e cinco porcento dos indivíduos escolhidos aleatoriamente. Então, o indivíduo com a melhor avaliação da função objetivo é escolhido deste subconjunto para ser o $\mathbf{x}_{t\_best}$. Esse processo garante que um indivíduo, que possui baixa qualidade, tenha chance de ser o melhor e direcionar o movimento de um subconjunto população, evitando possíveis mínimos locais.

Uma variação do APMT-MODE, denominada de *Adaptive Parameters Multi-Objective Differential Evolution* - AP-MODE, foi desenvolvida substituindo a mutação torneio, do APMT-MODE, pela estratégia definida na Equação 5.18

$$\mathbf{v}_i = \mathbf{x}_i + F_i \left(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\right) + F_i \left(\mathbf{x}_{c\_best} - \mathbf{x}_i\right). \tag{5.18}$$

A estratégia de mutação definida na Equação 5.18 gera um novo indivíduo nas proximidades da melhor solução atual explorando a sua vizinhança (GONG et al., 2017).

Nos problemas multiobjetivos, todas as funções objetivos têm o mesmo grau de importância. Porém, para escolher o melhor indivíduo no APMT-MODE, uma das funções objetivos é escolhida aleatoriamente para ser o critério de decisão. Este método é chamado *Criticism of Lexicographic Ordering* - CLO (COELLO; VELDHUIZEN, 2007).

O CLO consiste em eleger uma das funções objetivo, por sorteio, como critério para definir o melhor indivíduo entre as soluções não dominantes. O CLO é empregado a cada cinco gerações, esta frequência na escolha oferece a oportunidade de os indivíduos melhorarem em relação a determinado critério antes de realizar outro sorteio. Todas as estratégias de mutação que, de alguma forma, empregam a definição de melhor indivíduo utilizam CLO.

A principal desvantagem do CLO é a aleatoriedade da escolha da função objetivo, que, no caso de várias funções, tende a favorecer funções específicas. Entretanto, no nosso caso, o problema aqui abordado possui apenas duas funções objetivos e, nesse caso, este problema é minimizado (COELLO; VELDHUIZEN, 2007).

A principal vantagem do CLO é que ele é capaz de descrever uma Fronteira de Pareto côncava, embora isto dependa da distribuição da população e do problema em si (COELLO; VELDHUIZEN, 2007). Assim sendo, o método SLHD foi utilizado para inicializar a população do APMT-MODE, assim como o CLO, como estratégia de definição do melhor indivíduo nas estratégias de mutação e ajuste de parâmetros.

5.3 CONCLUSÕES DO CAPÍTULO

O algoritmo PSO foi modificado para tratar MOOP, agregando a função *truncate*, a técnica CLO e o SLHD para inicialização da população inicial, agregando técnicas eficientes para problemas distintos, mas que possui características semelhantes ao problema de seleção de modelo das SVMs. O desenvolvimento do APMT-MODE seguiu metodologia semelhante de agregação de técnicas distintas que ora foram utilizadas para algoritmos mono-objetivo e, neste trabalho, foi adaptado para tratar o problema como MOOP. O APMT-MODE e o MOPSO foram validados utilizando diferentes *benchmarks* classificadores e regressores, como também aplicações práticas a problemas de engenharia.

A ferramenta denominada NIOTS *vide* (Capítulo 6) busca tornar mais prática a avaliação e o desenvolvimento de técnicas e a utilização dos algoritmos APMT-MODE e MOPSO, possibilitando que projetistas possam produzir bons resultados para os problemas de seleção de parâmetros das SVMs. Algumas técnicas promissoras envolvem a análise do conjunto não dominante para definir: (a) o movimento das partículas/indivíduos; (b) as métricas para ordenar o conjunto de soluções não dominantes; (c) a adoção de técnicas adaptativas para controlar os parâmetros do APMT-MODE e suas metodologia de mutação e recombinação; (d) as metodologias de escolha dos *gbest* e *lbest* no MOPSO; (e) as distribuições alternativas de variáveis aleatórias, como Normal, Cauchy e Levy; e (f) a geração da população inicial com uso da técnica *Symmetric Latin Hypercube Design*.

Uma proposta promissora para o desenvolvimento de SVMs, com alto grau de precisão e grande capacidade de generalização, são os *comitês de máquina* (REN; ZHANG; SUGANTHAN, 2016b), que combinam duas ou mais SVMs que trabalham em conjunto, resultando em um classificador/regressor mais eficiente. Esta metodologia acrescenta um esforço considerável na obtenção dos hiperparâmetros, pois cada máquina do conjunto deve ter seus parâmetros ajustados.

No Capítulo 6, são descritas as funcionalidades do sistema desenvolvido chamado *Nature Inspired Optimization Tools for SVMs* - NIOTS.

6 *NATURE INSPIRED OPTIMIZATION TOOLS FOR SVM* - NIOTS (FERRAMENTA DESENVOLVIDA NESTE TRABALHO)

O sistema *Nature Inspired Optimization Tools for SVM* - NIOTS foi desenvolvido com o objetivo de tornar prática a utilização dos algoritmos desenvolvidos, *vide* Capítulo 3, em trabalhos futuros e nos testes de validação estatística. Neste capítulo, cada funcionalidade dos ambientes gráficos dos NIOTS são descrito em detalhes.

6.1 INTRODUÇÃO

Os algoritmos meta-heurísticos APMT-MODE e MOPSO, desenvolvidos no Capítulo 3, foram codificados em linguagem Matlab com auxílio da ferramenta GUIDE - *Grafical User Interface Development Environment*, originando o sistema denominado NIOTS. Este sistema é composto dos ambientes de (a) Otimização, (b) *Ensembles*, (c) Predição e (d) Estatística. Cada um desses ambientes são descritos nas subseções 6.1.1, 6.1.2, 6.1.3 e 6.1.4.

O sistema possibilita a utilização dos conceitos associados a SVM/SVR apresentados no Capítulo 2, os algoritmos de otimização multiobjetivo APMT-MODE, AP-MODE e MOPSO e as técnicas de Adição de Diversidade descritas no Capítulo 3. O NIOTS possibilita que usuários com interesse em obter modelos de SVM/SVRs para áreas específicas do conhecimento, sem a realização de estudos profundos dos conceitos envolvidos.

Adicionalmente, o NIOTS gera relatórios e modelos que podem ser testados e analisados estatisticamente nos ambientes Predição e Estatística.

O ambiente *Ensembles* é uma versão inicial de trabalhos futuros, em que será desenvolvido um algoritmo meta-heurístico para escolha do melhor conjunto de modelos para gerar o *ensemble* a partir de Conjuntos e Fronteiras de Pareto obtido no ambiente de Otimização. Na versão atual do NIOTS, os *ensembles* são obtidos testando todas as combinações possíveis de modelos de dois, três e quatro modelos de SVR provenientes de uma PF escolhida.

6.1.1 NIOTS: Ambiente de Otimização

No ambiente de otimização do NIOTS, é possível realizar a otimização dos hiperparâmetros de classificadores e regressores, usando os algoritmos meta-heurísticos MOPSO e APMT-MODE para resolver o problema de seleção de parâmetros formulado como um MOOP. Os algoritmos

MOPSO e APMT-MODE definem as soluções candidatas que são os parâmetros da ferramenta de treinamento LiBSVM (CHANG; LIN, 2011).

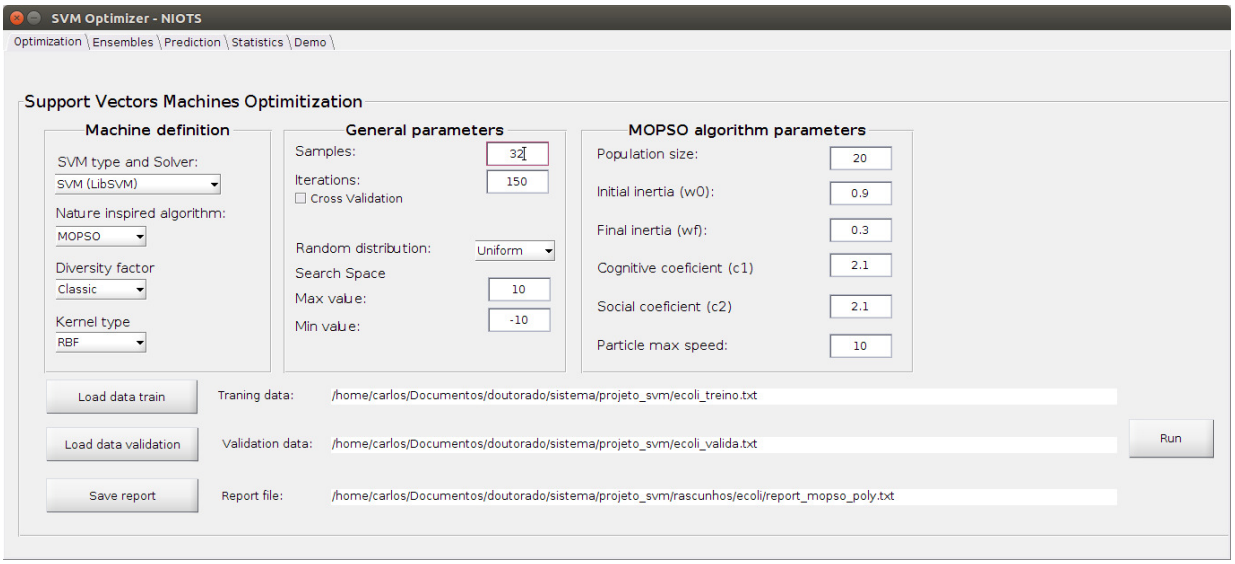


Figura 6.1 – Ambiente gráfico NIOTS - Otimização.

A interface gráfica do NIOTS, Figura 6.1, foi desenvolvida para facilitar o emprego das amplas possibilidades do sistema pelos usuários. O ambiente de otimização está dividido em três painéis: (a) *Definição da Máquina*, (b) *Parâmetros Gerais* e (c) *Parâmetros dos Algoritmos*, sendo cada opção desses painéis apresentado na Tabela 6.1.

Tabela 6.1 – Descrição das opções do ambiente Otimização.

Painel	**Menu/Campo**	**Opções/descrição**
Definição da Máquina	Tipo de Modelo	SVM/LiBSVM
		SVR/LiBSVM
		Grid Search SVR
		Grid Search SVM
	Algoritmo de Otimização	MOPSO
		APMT-MODE
	Fator de diversidade	Clássico
		AR
		OBL
	Tipo de Kernel	RBF
		Polinomial
		Acos
		Cauchy
Parâmetros Gerais	Amostras	Quantidade de Amostras independentes

Continua na próxima página.

Tabela 6.1 – *Continuando da página anterior.*

Painel	**Menu/Campo**	**Opções/descrição**
	Iterações	Número de iterações do algoritmo de otimização
	Cross Validation	Ao ser desativada esta opção, é necessário carregar um arquivo com o conjunto de validação
		Caso contrário utiliza a Validação Cruzada
	Distribuição	Uniforme
		Normal
		Cauchy
	Espaço de busca	Limite superior de C e Gama
		Limite inferior de C e Gama
Parâmetros do MOPSO	Tamanho da população	Quantidade de partículas do enxame
	Inércia inicial	Fator de inércia inicial
	Inércia final	Fator de inércia final
	Coeficiente Cognitivo	Grau de confiança na partícula
	Coeficiente Social	Grau de confiança no enxame
	Velocidade máxima da partícula	Define a velocidade máxima permitida pela partícula durante o processo de otimização
Parâmetros do APMT-MODE	Tamanho da população	
	Fator de escala	Fator inicial
	Taxa de recombinação	Taxa inicial

A ferramenta LiBSVM realiza o treinamento utilizando os hiperparâmetros fornecidos pelos algoritmos APMT-MODE e MOPSO. Esses modelos são avaliados com os dados do conjunto de validação, retornando as métricas da capacidade de generalização e complexidade do modelo.

No fluxograma da Figura 6.2, a variável ω é o vetor dos hiperparâmetros e os valores funcionais E_e e C_{SV} são as métricas da capacidade de generalização e da complexidade, respectiva-

mente. As setas preenchidas, em preto, representam o fluxo de dados entre os blocos, enquanto as setas brancas indicam as possíveis opções que o usuário pode escolher no processo de otimização, também apresentadas na Tabela 6.1.

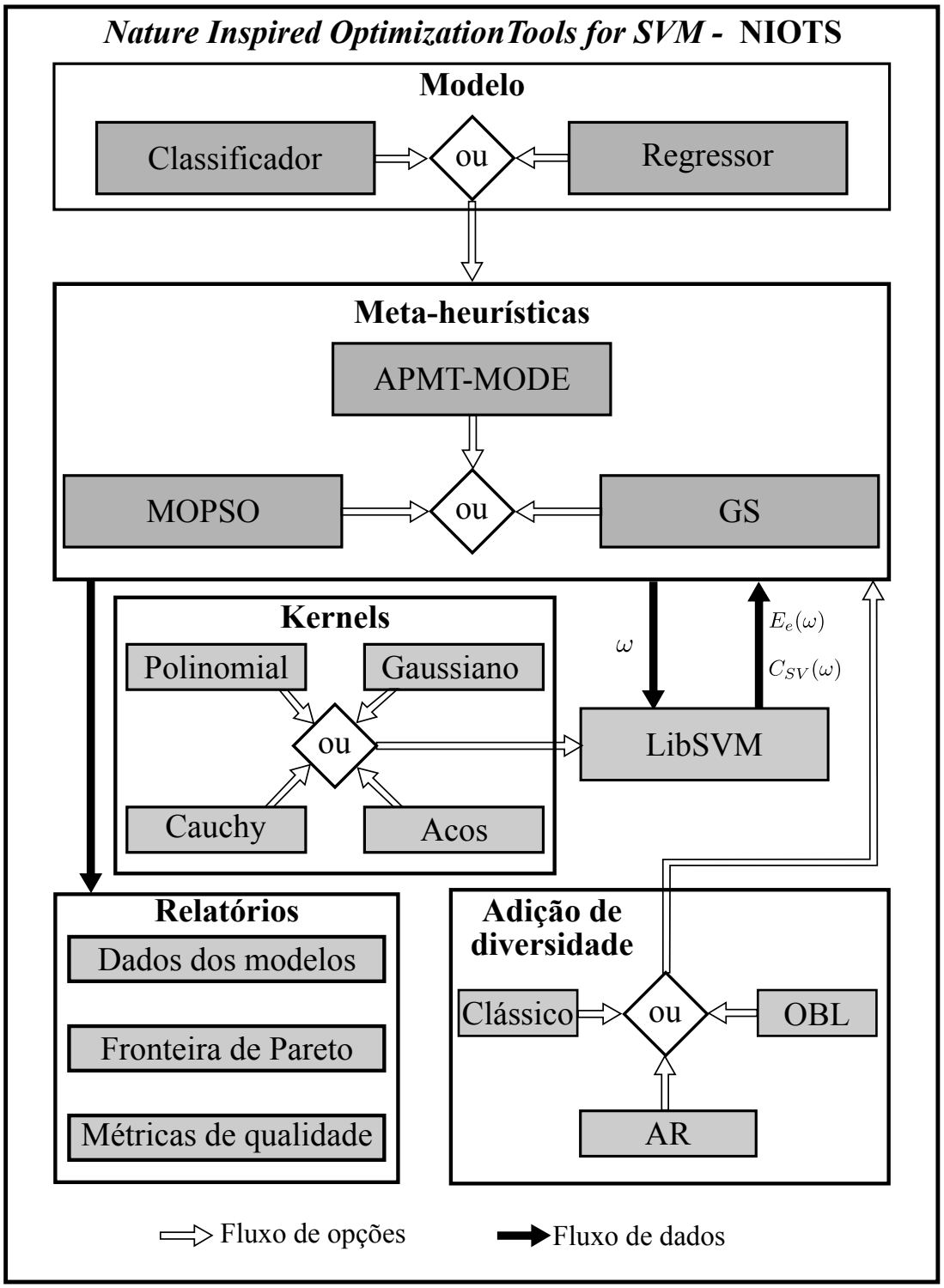


Figura 6.2 – Diagrama de fluxo de dados e opções do NIOTS.

No final do processo de otimização, são gerados três tipos de relatórios representados na

Figura 6.2 pelo bloco *Relatórios*. Para cada hiperparâmetro do conjunto de Pareto, é gerado um arquivo relatório, denominado *Dados do Modelo*, que possui todos os dados necessários para implementação do modelo em questão. A Figura 6.3 é um exemplo desse relatório.

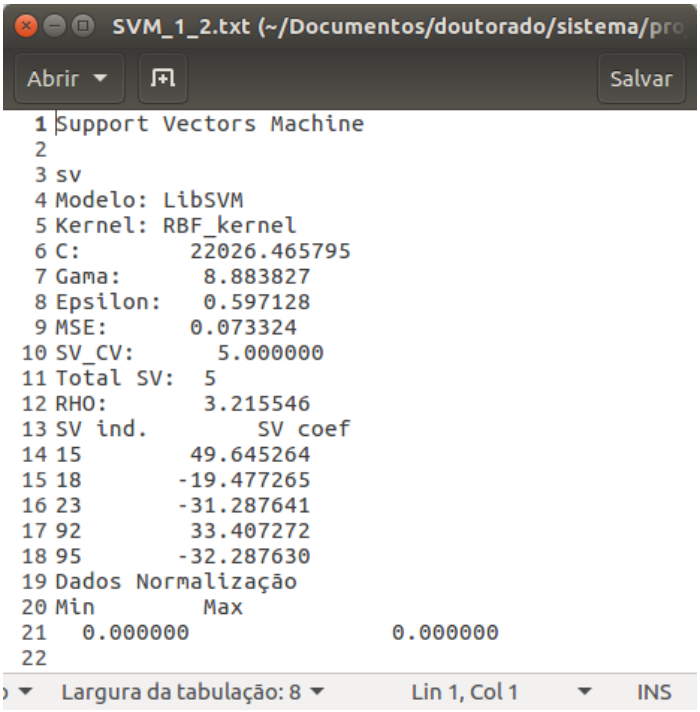


Figura 6.3 – Exemplo de modelo da Fronteira de Pareto.

O relatório *Pareto Front* (Fronteira de Pareto), Figura 6.4, contém os parâmetros do APMT-MODE/MOPSO definidos pelo usuário e os elementos do conjunto de Pareto, assim como seus respectivos objetivos da Fronteira de Pareto. As métricas de qualidade da PF são avaliadas a cada iteração e gravadas no arquivo relatório *Métricas de Qualidade*. As métricas utilizadas para avaliar a Fronteira de Pareto são a cardinalidade do conjunto de soluções não dominadas e o *Spacing*.

Essas métricas são calculadas a cada iteração, permitindo a análise de: (a) convergência do algoritmo e (b) da qualidade das soluções obtidas para cada processo de otimização.

Além dos painéis descritos na Tabela 6.1, o ambiente possui três botões. O botão *carregar dados de treinamento* faz a leitura do arquivo que contém os dados de treinamento. Para carregar os dados de validação, utiliza-se o botão *carregar dados de validação*; esta opção estará disponível se a opção *cross-validation* não estiver ativa. No botão *Salvar relatório*, deve ser escolhido o nome e o local onde os relatórios serão gravados e o botão *rodar* inicia o processo de otimização.

6.1.2 NIOTS: Ambiente Comitê de Máquinas - *Ensembles*

Os métodos de *ensembles* utilizam múltiplos modelos com características distintas para fornecer uma melhor capacidade de generalização, a qual os modelos individualmente possam pro-

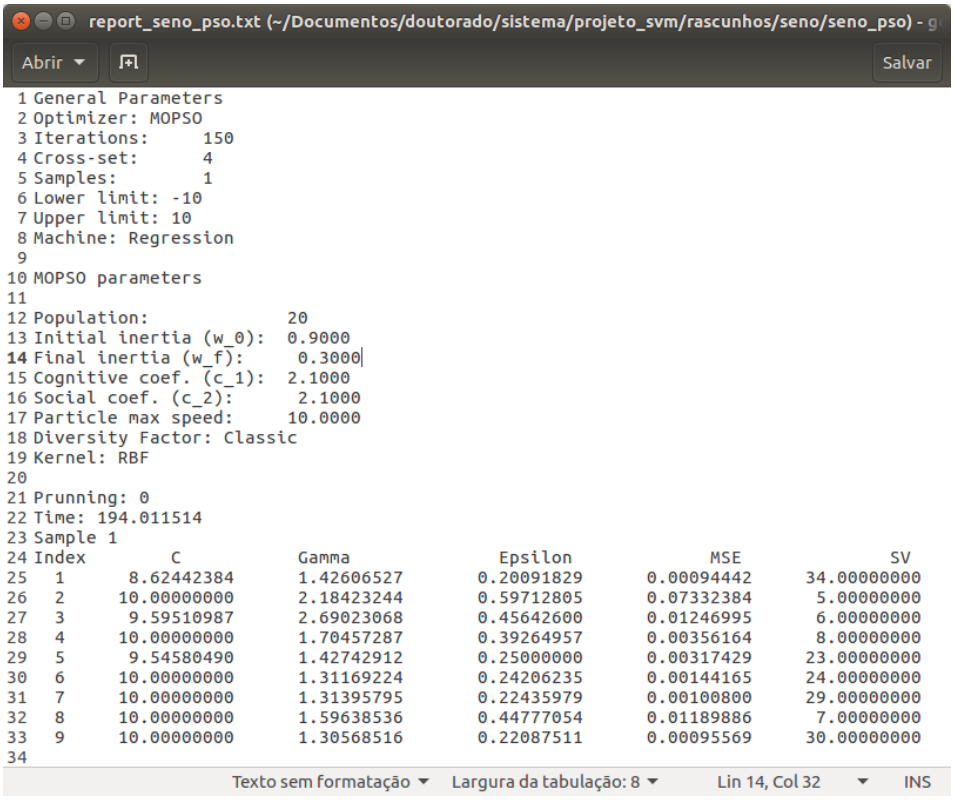


Figura 6.4 – Relatório contendo o Conjunto e a Fronteira de Pareto.

duzir. Os modelos que compõe os *ensembles* diferem-se pela diversidade: (a) dos dados, (b) de parâmetros, (c) estrutural, (d) divisão e conquista, (e) otimização multiobjetivo e (f) *fuzzy ensemble* (REN; ZHANG; SUGANTHAN, 2016a).

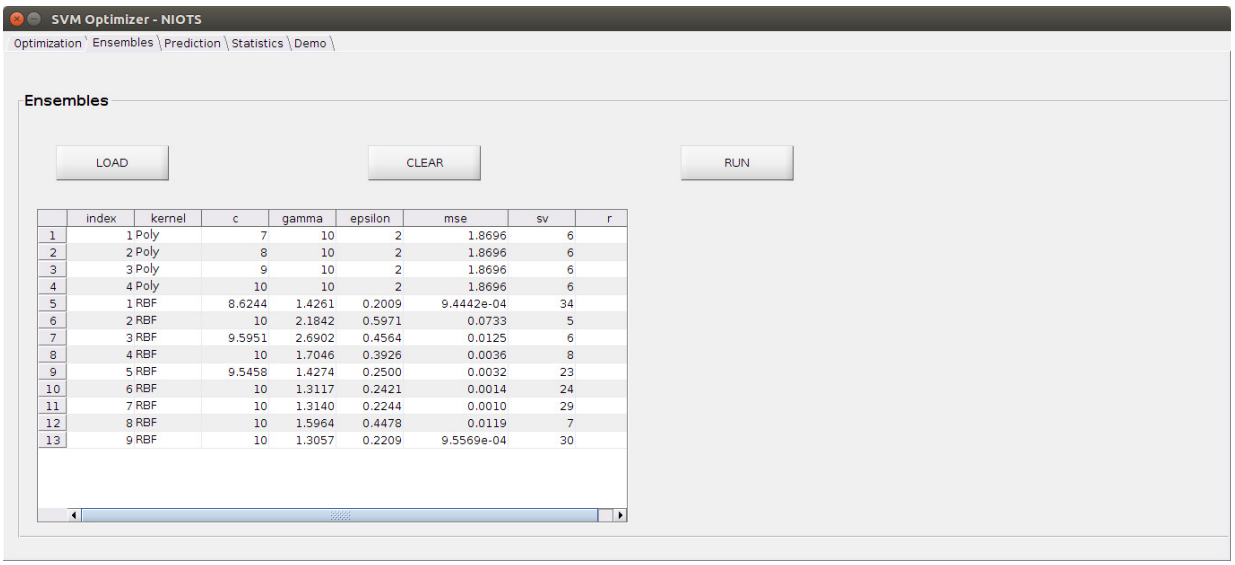
O ambiente *Ensemble* do NIOTS foi desenvolvido para criar *ensembles* baseados na diversidade dos parâmetros obtidos a partir do conjunto de Pareto e na estrutural, em que se pode escolher mais de um tipo de *kernel*. A Figura 6.5 apresenta o ambiente *Ensembles* no qual foram carregadas duas PF com *kernels* RBF e Polinomial.

A leitura das PF é realizada ao acionar o botão *LOAD* no qual deve ser indicado o arquivo da PF gerada pelo ambiente Otimização. Quando o botão *RUN* for acionado, serão solicitados dois arquivos, um para o treinamento dos modelos e outro para validação dos *Ensembles*.

A ferramenta, de fato, treina novamente cada modelo com os hiperparâmetros das Fronteiras de Pareto carregadas e realiza todas as combinações possíveis, como os modelos com *Ensembles* de dois, três e quatro modelos. Ao final, a função *Truncate* é utilizada para determinar o conjunto de *Ensembles* não dominados, considerando o MSE e o número total de vetores de suporte.

O resultado é gravado em um arquivo relatório, na mesma pasta do conjunto de treinamento, indicando os elementos dos *Ensembles* e o quanto cada um foi melhor que o melhor modelo individual pertencente ao *Ensemble*.

Para gerar os *Ensembles*, o NIOTS testa todas as possibilidades de dois, três e quatro elementos que, na prática, para grupos com mais elementos, se torna impraticável, sendo deixado para



	index	kernel	c	gamma	epsilon	mse	sv	r
1	1	Poly	7	10	2	1.8696	6	
2	2	Poly	8	10	2	1.8696	6	
3	3	Poly	9	10	2	1.8696	6	
4	4	Poly	10	10	2	1.8696	6	
5	1	RBF	8.6244	1.4261	0.2009	9.4442e-04	34	
6	2	RBF	10	2.1842	0.5971	0.0733	5	
7	3	RBF	9.5951	2.6902	0.4564	0.0125	6	
8	4	RBF	10	1.7046	0.3926	0.0036	8	
9	5	RBF	9.5458	1.4274	0.2500	0.0032	23	
10	6	RBF	10	1.3117	0.2421	0.0014	24	
11	7	RBF	10	1.3140	0.2244	0.0010	29	
12	8	RBF	10	1.5964	0.4478	0.0119	7	
13	9	RBF	10	1.3057	0.2209	9.5569e-04	30	

Figura 6.5 – Ambiente gráfico NIOTS - Ensembles.

trabalhos futuros o desenvolvimento de um algoritmo Meta-heurístico capaz de definir quais e quantos modelos são necessários para que a complexidade dos *Ensembles* seja minimizada, enquanto a capacidade de generalização seja maximizada. Na Subseção 6.1.3, é descrito o ambiente Predição, no qual os modelos podem ser testados individualmente e gráficos comparativos são gerados.

6.1.3 NIOTS: Ambiente de Predição

No ambiente *Predição*, apresentado na Figura 6.6, podem ser realizados testes com os modelos de funções aproximadoras e dos classificadores gerados pelas soluções dos algoritmos de otimização dos hiperparâmetros. Inicialmente, é necessário escolher entre os modelos *Single-target-SVR* e *Classify-SVM*, em seguida, é definido se os dados que serão lidos possuem ou não rótulos/imagens.

Acionando o botão *Load SV*, uma caixa de diálogo é aberta para que seja indicado o arquivo que contém o modelo a ser avaliado pelo ambiente *Predict*. A Figura 6.3 é um exemplo desse tipo de arquivo que contém todas as informações necessárias para gerar o modelo. Os dados de teste são carregados ao acionar o botão *Features*, que possui o mesmo formato do arquivo de treinamento.

O botão *Prediction* pode ser acionado depois de carregado o arquivo do modelo de SVM/SVR, o arquivo de testes, e escolhido o tipo de modelo que se deseja testar. O acionamento deste retorna o MSE ou AUC para regressores ou classificadores dependendo do modelo em questão. O botão textitPrediction produz, ainda, dois gráficos distintos para as SVRs. A Figura 6.7a confronta as saídas produzidas pelo modelo e as do conjunto de testes e gráficos de resíduos.

Os resíduos possuem informações importantes para análise do modelo. No gráfico dos resíduos, não deve ser observado qualquer padrão, o que significa que o modelo absorveu toda a

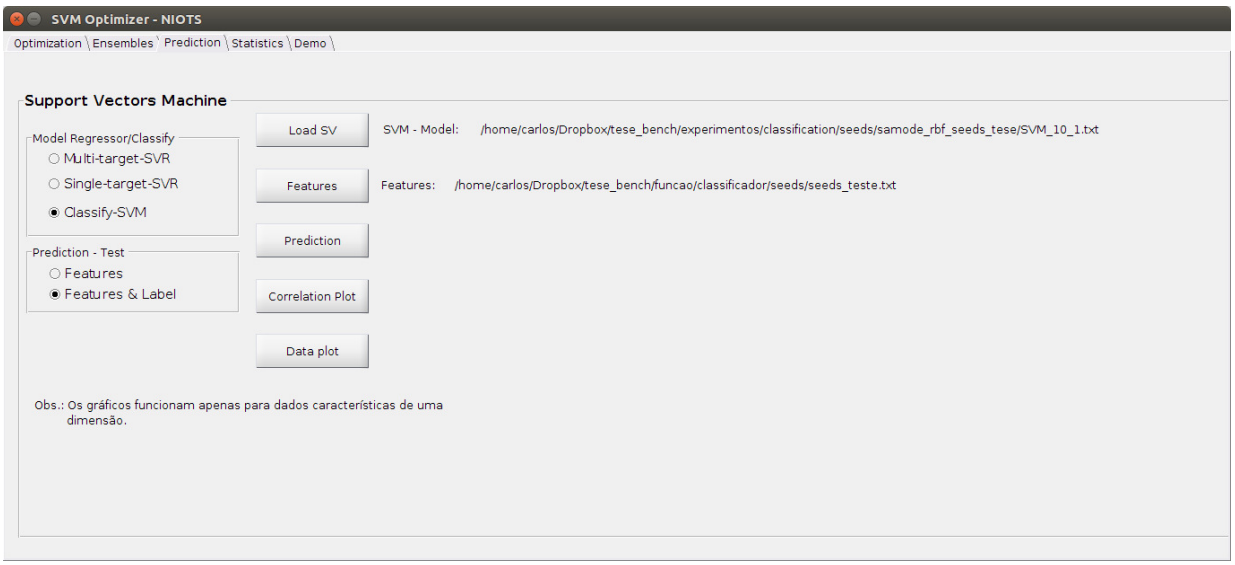
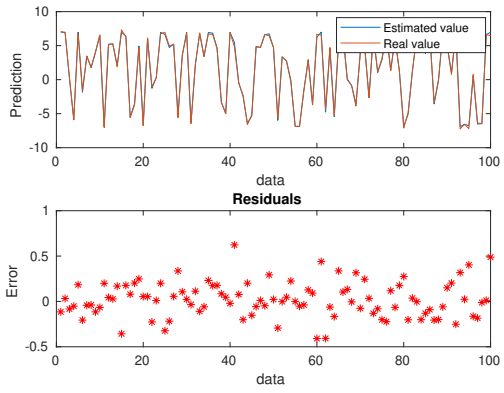
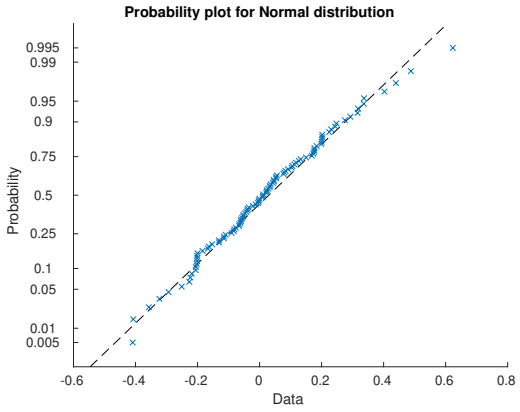


Figura 6.6 – Ambiente gráfico NIOTS - Otimização.

informação disponível e suas diferenças são, em sua maioria, devido a ruídos dos dados. A Figura 6.7b representa o teste de normalidade, neste gráfico, quanto mais próximos os pontos estiverem da linha pontilhada, maior é a normalidade dos resíduos. A normalidade dos resíduos é outro critério utilizado para avaliar a qualidade do regressor em questão.



(a) Gráfico de predição e resíduos.



(b) Teste de normalidade dos resíduos.

Figura 6.7 – Gráficos gerados pela opção *Correlation Plot*.

O gráfico da Figura 6.8 é obtido com o acionamento do botão *Correlation Plot*, no caso de configurada a opção *Single-target-SVR*. No rodapé do gráfico, é impresso o coeficiente de correlação que varia no intervalo $[0, 1]$, quando mais próximo de um, mehor é o ajuste do modelo. No gráfico, pode-se realizar esta análise visualmente, tendo em conta que melhor é a qualidade do modelo em questão, quando os pontos estão mais próximos à reta pontilhada.

Para classificadores, o acionamento do botão *Correlation Plot* gera o gráfico da curva *Receiver Operating Characteristic* - ROC, sendo *Area Under Curve* - AUC a métrica que mede o quanto um classificador é capaz de distinguir entre duas classes. Os valores do AUC variam entre zero

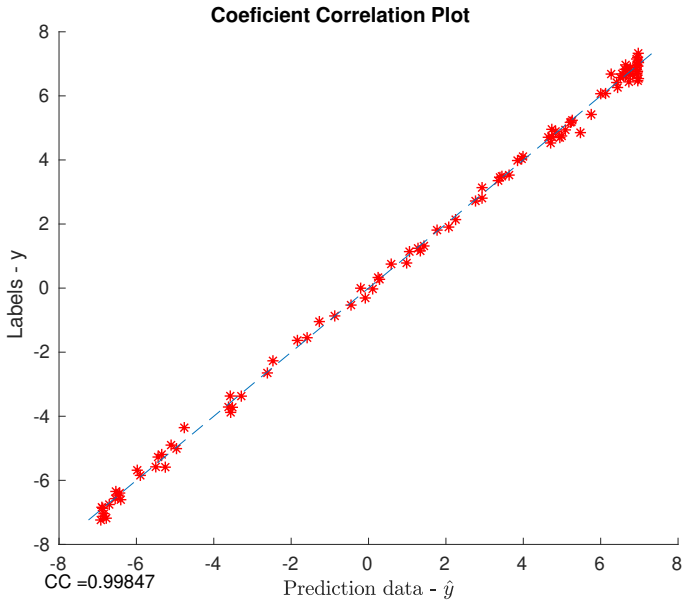


Figura 6.8 – Coeficiente de Correlação.

e um, sendo que, quanto maior o valor da métrica, melhor será o classificador. O modelo é considerado ruim para valores próximos de $0,5$.

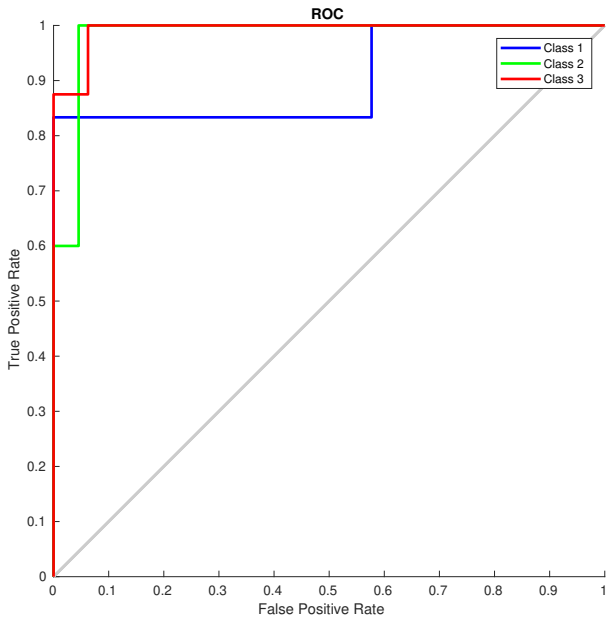


Figura 6.9 – Gráfico da métrica ROC.

Na Subseção 6.1.4, o ambiente *Estatística* possibilita a análise estatísticas dos algoritmos implementados, considerando cada conjunto de treinamento.

6.1.4 NIOTS: Ambiente de Estatística

Quando se trabalha com vários algoritmos de otimização e estes possuem alguns parâmetros a serem configurados, torna-se necessário um estudo estatístico, para que seja possível discernir quais são as opções mais viáveis. O estudo estatístico valida ou não as propostas de algoritmos e suas configurações para determinados problemas. O ambiente *Estatistics* foi implementado para tornar prático e rápido os testes estatísticos realizados.

O ambiente *Estatistics* possui três painéis, o primeiro gera gráficos das métricas *hipervolume*, *cardinalidade do conjunto de não dominados* e *spacing* de cada iteração do conjunto de soluções não dominadas. No painel, têm-se as opções *sample means*, *standard deviation* e *median*, conforme Figura 6.10. O cálculo de cada uma dessas estatísticas é realizado com base em cada iteração das amostras. A opção *Pareto front* gera o gráfico da fronteira de Pareto de todas as amostras.

Neste painel, o botão *Data load* possibilita a leitura do arquivo "metricas.txt", que contém os dados necessários para gerar os gráficos (o botão *Run* gera os gráficos).

O painel *Plot Multi-target* tem os mesmos objetivos e funcionalidades do painel *plot single-target*.

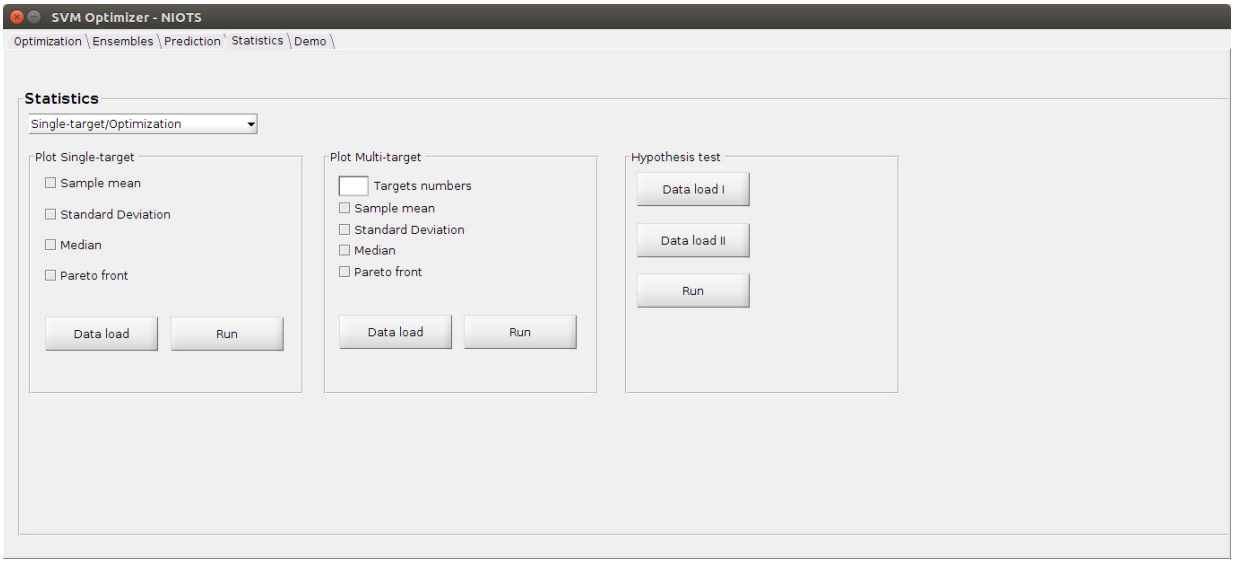


Figura 6.10 – Ambiente gráfico NIOTS - Estatística.

No painel *Hypothesis test*, é possível fazer a leitura de duas séries de dados que serão comparadas. As séries são ajustadas e, então, é aplicado o teste Kolmogorov-Smirnov para verificar se a distribuição das amostas provém ou não de uma distribuição Normal. Caso as duas distribuições advenham de uma distribuição Normal, é aplicado o teste ANOVA e são gerados os gráficos de *Boxplot*. Caso as séries não advenham de uma distribuição Normal, é aplicado o teste de Wilconxon e são gerados os gráfico de *Boxplot*.

6.2 CONCLUSÕES DO CAPÍTULO

Os problemas de otimização multiobjetivo representam melhor o problema de seleção dos hiperparâmetros de uma SVM/SVR, pois consideram um compromisso entre a capacidade de generalização e a complexidade da máquina, além de permitir que o projetista escolha uma solução adequada à sua aplicação.

Para resolver problemas de otimização multiobjetivos multimodais, é comum na literatura o emprego de algoritmos meta-heurísticos. Os algoritmos APMT-MODE e MOPSO em conjunto com as ferramentas LibSVM possuem ampla faixa de possibilidades, o que dificulta a definição de seus parâmetros e escolha da ferramenta. Para tanto, foi desenvolvido o NIOTS, um ambiente gráfico simples que utiliza esses algoritmos e gera um conjunto de relatórios que torna prática a avaliação de uma grande quantidade de informação. O NIOTS possibilita também a validação estatística dos algoritmos implementados e seus parâmetros.

Na versão atual, o NIOTS possibilita a combinação de técnicas novas e outras implementadas em algoritmos mono-objetivo, de modificação de estratégias de exploração e refinamento do espaço de busca do problema de seleção de hiperparâmetros das SVMs, para que seja possível a comparação com outros algoritmos do estado da arte.

7 ESTUDO DE CASOS

A validação dos algoritmos meta-heurísticos APMT-MODE, AP-MODE e MOPSO tem como objetivo demonstrar a capacidade dos algoritmos em resolver diferentes problemas. Os estudos foram realizados com *benchmarks* classificadores e regressores para os *kernels* Gaussiano, polinomial, arco cosseno e Cauchy. Os resultados obtidos foram comparados, através de testes estatísticos não paramétricos, com os resultados obtidos pelo algoritmo NSGA-II.

7.1 INTRODUÇÃO

Para que haja credibilidade na análise do desempenho alcançados pelos algoritmos APMT-MODE, AP-MODE e MOPSO diante do NSGA-II, foram aplicados os testes estatísticos não paramétrico de Friedman aos resultados obtidos por estes algoritmos, em quatro problemas de aproximação de funções e oito classificadores combinados com os *kernels* Gaussiano, Cauchy, Polinomial e arco cosseno.

Os testes estatísticos podem ser categorizados em duas classes: (a) paramétricos e (b) não paramétricos. Os testes paramétricos são utilizados quando as amostras de determinado conjunto de experimentos possuem distribuição normal, independência dos dados e homogeneidade da variância, estes dois últimos segundo García *et al.* (GARCÍA et al., 2008).

A independência dos dados indica a influência que a captação de uma amostra aplica sobre a outra. Neste trabalho, as meta-heurísticas são testadas 32 vezes utilizando os mesmos parâmetros *vide* (Tabela 7.4). Porém todos os processos que envolvem geração de números aleatórios foram realizados de maneira independente, garantindo, assim, a independência entre as amostras.

A homogeneidade da variância requer que as amostras de uma mesma população devem possuir variância iguais.

Assim, os testes não paramétricos são utilizados quando não se pode atestar a normalidade das amostras e/ou a homogeneidade das mesmas. Mesmo quando não são realizados os testes de normalidade, ou não há informação suficiente sobre as amostras, recomenda-se aplicar os testes não paramétricos (DERRAC et al., 2011).

Os testes estatísticos não paramétricos permitem verificar se há diferença significativa entre os resultados obtidos por dois ou mais algoritmos, como também mensurar esta diferença. Estes dados sempre são dados por um nível de significância α, geralmente assumindo os valores $0,05$ ou $0,1$, aqui adotado como $\alpha = 0,05$. Assim, um valor de $p < \alpha$ significa que a hipótese nula é rejeitada e que, pelo menos, uma amostra é estatisticamente diferente da(s) outra(s).

Na comparação de múltiplos algoritmos, deve-se inicialmente identificar (ou inferir) se as soluções obtidas pelos algoritmos em avaliação provêm de uma mesma população, ou seja, se

os algoritmos em questão produzem resultados estatisticamente equivalentes. Essa inferência é realizada através de testes estatísticos de hipótese, em que há duas hipóteses: (a) hipótese nula (H_0), que afirma que não diferença entre as soluções, e (b) hipótese alternativa (H_1), que afirma que há diferença significativa entre os algoritmos avaliados.

Atualmente, a comunidade que tem trabalhado com análise de desempenho de meta-heurísticas opta, nos últimos anos, por usar testes estatísticos não paramétricos. Portanto, foi escolhido o teste de Friedman $N \times N$ para avaliar o desempenho entre os algoritmos APMT-MODE, AP-MODE, MOPSO e NSGA-II, conforme sugerido por Islam et al. (2012).

Na Seção 7.2, são caracterizados os *benchmarks* aplicados às meta-heurísticas APMT-MODE, AP-MODE, MOPSO e NSGA-II.

7.2 DESCRIÇÃO DOS *BENCHMARKS*

Os *benchmarks* adotados para o estudo de caso foram obtidos no repositório UCI (DHEERU; TANISKIDOU, 2017). Entre os trabalhos apresentados na Tabela 4.1, 36% utilizam estes *benchmarks* como referência. Os conjuntos de treinamento (*benchmarks*) dos classificadores e regressores foram escolhidos prezando a diversidade em relação à área de conhecimento e quantidade de entradas, *vide* Tabelas 7.1 e 7.2. Nos classificadores, em relação à quantidade de classes, foram escolhidos seis binários e dois multiclasses, sendo estes últimos o Ecoli e o Iris.

Os *benchmaks* Ecoli e Iris têm uma dificuldade adicional, pois suas classes possuem dados desbalanceados, o que dificulta o treinamento e a atribuição dos hiperparâmetros. Devido ao fato de as SVMs terem sido desenvolvidas para gerar classificadores binários, é necessário adotar estratégias adicionais que empregam vários modelos de SVMs. A ferramenta LibSVM adota a metodologia *One-Against-One* - OAO (CHANG; LIN, 2011). Esta metodologia produz $\frac{k(k+1)}{2}$ modelos de SVMs, para cada *benchmark*, em que k representa o número de classes. A quantidade de vetores de suporte computada, neste caso, é a soma de todos os modelos necessários para dividir as k classes.

As $\frac{k(k+1)}{2}$ SVMs podem possuir parâmetros distintos, como o proposto no trabalho de Lorena e De Carvalho (2008). Entretanto, no NIOTS, todos os modelos de SVMs necessários para distinguir entre as classes recebem os mesmos hiperparâmetros.

Os *benchmarks* regressores foram escolhidos de diferentes áreas. O *1027ESL* é um dado referente à área de recrutamento de pessoas para determinado posto de trabalho de uma empresa, que possui quatro características e uma nota de adequação do candidato ao posto de trabalho.

As entradas do *Auto MPG* consistem em sete características de 398 modelos de carros distintos, em que se deseja predizer o consumo de combustível em galões por milhas percorridas. O *Concrete* tem como objetivo prever a força de compressão do concreto em MPa considerando a quantidade de cimento, agregados, água e outros componentes. Os dados do *Yacht* provêm de

Tabela 7.1 – Descrição dos *benchmarks* classificadores.

Benchmark	**Entradas**	**Classes**	**Número de instâncias**			
			Total	**Treino**	**Validação**	**Teste**
Ecoli	7	8	336	236	50	50
Haberman	3	2	306	214	46	46
Heart	44	2	267	187	40	40
Ionosphere	34	2	351	246	53	53
Iris	4	3	150	105	23	23
Sonar	60	2	208	146	31	31
WDBC	30	2	569	398	85	85

estudos da área de hidrodinâmica, em que seis parâmetros do projeto são as entradas, tendo como saída a ser predita a resistência residual por unidade de peso. O resumo dos dados é apresentado na Tabela 7.2.

Tabela 7.2 – Descrição dos *benchmarks* regressores.

Benchmark	**Entradas**	**Número de instâncias**			
		Total	**Treino**	**Validação**	**Teste**
1027 ESL	4	488	342	73	73
Auto MPG	7	398	279	60	60
Concrete	8	1030	721	155	155
Yacht	6	308	216	46	46

No processo de modelagem matemática de problemas utilizando SVM/SVRs, deve-se definir os *kernels a priori*, os quais possuem diferentes quantidades de parâmetros; devido às características de cada um deles, cada *kernel* gera um PSP diferente para cada conjunto de dados.

Este comportamento pode ser observado nos gráficos gerados pelo GS no Apêndice B. O conjunto de problemas utilizados neste trabalho é composto pela combinação dos *kernels* com os diferentes conjuntos de dados, em que cada combinação gera um PSM diferente. O total de 44 problemas é dado por todas as combinações possíveis das entradas da coluna *kernel* e as da coluna *Benchmarks*, da Figura 7.1.

As equações das funções *kernel* polinomial, Gaussiano, arco cosseno e Cauchy são apresentadas na Tabela 7.3. O *kernel* arco cosseno não possui parâmetros; este foi proposto por Youngmin Cho and Lawrence K. (CHO; SAUL, 2009) para obtenção de redes profundas com emprego de SVMs.

O *kernel* Cauchy é um tipo de RBF, que calcula a norma da diferença entre os vetores de suporte e o dado de entrada do modelo, tendo o parâmetro c a ser ajustado. O *kernel* Cauchy possui a vantagem computacional de não empregar a operação exponencial, assim como por não possuir restrições numéricas (por gerar valores limitados superiormente), tendo em conta que sua imagem está confinada ao intervalo $[0, 1]$. O mesmo ocorre com o *kernel* Gaussiano. Ou seja, tendo $\mathbf{x} = \mathbf{y}$, implica em $K(\mathbf{x}, \mathbf{y}) = 1$, sendo K o *kernel* Gaussiano, indicando similaridade máxima entre os vetores $\mathbf{x}$ e $\mathbf{y}$.

Algoritmos	*Kernels*	*Benchmarks*
MOPSO	Gaussiano	Ecoli
APMT-MODE	Polinomial	Haberman Heart Ionosphere Iris
AP-MODE	Acos	Sonar WDBC 1027 ESL
GS		Auto Concrete
NSGA-II	Cauchy	Yacht

Figura 7.1 – Metodologia aplicada para gerar os problemas de estudo de casos e algoritmos.

Tabela 7.3 – Kernels disponíveis no NIOTS.

Kernel	**Função** $K(\mathbf{x}, \mathbf{y})$	**Parâmetros**
Polinomial	$K(\mathbf{x}, \mathbf{y}) = (\beta \mathbf{x} \cdot \mathbf{y} + k)^d$	β, κ, d
RBF Gaussiano	$K(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\gamma}\right)$	γ
Arco cosseno	$K(\mathbf{x}, \mathbf{y}) = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}\right)$	-
Cauchy	$K(\mathbf{x}, \mathbf{y}) = \frac{1}{1+\frac{\|\mathbf{x}-\mathbf{y}\|}{\lambda^2}}$	λ

O *kernel* polinomial possui os parâmetros β, κ e d, sendo adotado o valor igual a um para os dois primeiros. O parâmetro d indica o grau do polinômio e foi limitado a valores inteiros entre $[1, 20]$ para todos os casos. Este polinômio possui a vantagem computacional de executar apenas operações de soma e multiplicação, mas, mesmo com parâmetros adequados, não apresenta bons resultados para todos os tipos de problemas.

Ao considerar todas as combinações possíveis, têm-se quarenta e quatro problemas distintos para a execução do estudo de casos. Segundo Derrac *et al.* (DERRAC et al., 2011), para aplicação do teste não paramétrico de Friedman $N \times N$, os limites indicados são $2 \cdot k \leq n \leq 8 \cdot k$, em que k é a quantidade de problemas e n o número de algoritmos a serem avaliados. Na Seção 7.3, são apresentados os resultados do teste de Friedman, assim como os testes *post hoc* de Nemenyi, Shaffer e Bergmann empregados na validação dos algoritmos meta-heurísticos desenvolvidos.

7.3 RESULTADOS E DISCUSSÃO

Para validar o desempenho dos algoritmos meta-heurísticos APMT-MODE, AP-MODE, MOPSO e NSGA-II, cada algoritmo resolveu os 44 problemas 32 vezes, com os mesmos parâmetros, con-

forme descrito na Tabela 7.4. Para todos os problemas em questão, foram empregados os mesmos parâmetros.

Tabela 7.4 – Configuração dos parâmetros dos algoritmos APMT-MODE, AP-MODE, MOPSO, NSGA-II.

Parâmetros	**Configuração**
C	$[-10; 10]$
γ	$[-10; 10]$
λ	$[-10; 10]$
d	$[1; 10]$
ϵ	$[0,0001; 2]$
População	20
Iterações	100
Distribuição	Uniforme
Adição de diversidade	Clássica

A configuração dos parâmetros do MOPSO foi a mesma empregada para todos os problemas, e os valores estão apresentados na Tabela 7.5.

Tabela 7.5 – Parâmetros empregados para o algoritmo MOPSO.

Parâmetros	**MOPSO**
w_0	$0,9$
w_f	$0,1$
c_1	$2,1$
c_2	$2,1$
V_{max}	10

Os parâmetros w_0, w_f são os fatores de inércia inicial e final respectivamente, que decaem linearmente do valor inicial ao final, controlando a transição da fase de exploração para a fase de refinamento. Os parâmetros c_1 e c_2 são os coeficientes social e individual respectivamente, em que foram adotados valores iguais para estes parâmetros, tendo em conta as formas das funções objetivo, que envolvem características distintas para um mesmo problema. Assim como os parâmetros gerais, estes foram empregados em todos os problemas deste estudo.

Os algoritmos meta-heurísticos APMT-MODE e AP-MODE foram configurados com os valores $Cr = 0,4$ e $F = 0,7$ nas primeiras cinco iterações, e, então, a cada cinco iterações seguintes, os parâmetros são ajustados, de acordo com a metodologia apresentado na Seção 5.2.

Os parâmetros do algoritmo NSGA-II foram definidos como $Cr = 0,9$ e $M = 0,1$, em que os parâmetros Cr e M são a probabilidade de ocorrerem a recombinação e a mutação sobre um indivíduo, respectivamente.

A métrica utilizada para avaliar a qualidade das fronteiras Pareto obtidas nos experimentos foi o *Inverted Generational Distance* (IGD), que mensura o quão próximo o melhor conjunto de soluções não dominantes (gerados pelas meta-heurísticas) está da fronteira de Pareto. Entretanto, para os problemas abordados neste trabalho, não há uma fronteira de Pareto conhecida. Assim, para cada um dos problemas, os conjuntos não dominantes, gerados por todos os algoritmos em

avaliação, foram concatenados em uma única matriz. Deste conjunto de soluções, as que são não dominantes foram adotadas como fronteira de Pareto.

Para aplicação da métrica IGD, é necessário que a fronteira de Pareto e o conjunto de soluções, que está sendo avaliado, tenham a mesma cardinalidade. Então, entre os dois conjuntos em questão, são eliminadas soluções aleatoriamente escolhidas do conjunto com maior cardinalidade. Uma vez a cardinalidade dos conjuntos são iguais, pode-se calcular o IGD de cada um dos 32 experimentos de cada problema, para cada uma das meta-heurística, assim como suas respectivas medianas *vide* (Tabela 7.6). Dessa maneira, pode-se aplicar os testes de hipóteses para verificar se as meta-heurísticas produzem resultados equivalentes ou não.

Ao invés de estipular um nível de significância α *a priori*, é calculado o menor nível de significância que resulta na rejeição de H_0. Esta é a definição do *p-value*, o qual é a probabilidade de obter um resultado ao menos tão extremo quanto um que é realmente observado, assumindo que H_0 é verdadeiro (DERRAC et al., 2011). Esta informação é útil para definir o quanto é significativo o resultado do teste.

Assim, o *p-value* possui características qualitativa e quantitativa, a respeito das hipóteses. Derrac *et al.* recomendam que, apesar de os testes *post hoc* poderem ser empregados, mesmo se a hipótese nula não for rejeitada, é aconselhável que seja verificada a rejeição antes dos testes *post hoc* (DERRAC et al., 2011).

Os algoritmos são classificados em ordem ascendente, de acordo com seu resultado para cada problema *vide* (Tabela 7.6). Então, é calculado o *rank* apresentado na Tabela 7.7, consistindo na média do *rank* de cada um dos algoritmos para todos os problemas. No caso de igualdade entre os resultados dos algoritmos meta-heurísticos para o mesmo problema, toma-se a média do *rank* entre eles.

O *p-value* calculado pelo teste de Friedman é $1,9479 \times 10^{-4}$, para as medianas apresentadas na Tabela 7.6. Sendo assim, rejeita-se a hipótese nula de que todos os algoritmos produzem soluções provenientes de uma mesma população, com significância menor que a adotada. Ou seja, pelo menos um, entre os algoritmos APMT-MODE, AP-MODE, MOPSO e NSGA-II, é diferente estatisticamente dos demais.

Para verificar quais algoritmos são diferentes e quais são os melhores, aplica-se o teste *post hoc* de Nemenyi, Shaffer e Bergmann, obtendo os seguintes *p-values* ajustados, conforme Tabela 7.8.

Pelos *p-values* apresentados na Tabela 7.8, pode-se concluir que os algoritmos APMT-MODE, AP-MODE e MOPSO são melhores que o NSGA-II com confiança superior a 95%. Entretanto, entre os algoritmos APMT-MODE, AP-MODE e MOPSO, estatisticamente não há diferença, pois, para todos os testes os *p-values* são maiores que significância ($\alpha = 0,05$).

Entretanto, pelo *ranking* do teste de Friedman, tem-se o APMT-MODE com *ranking* igual a $2,1023$, que, portanto, possui melhor desempenho entre as demais meta-heurísticas, para os problemas apresentados.

Tabela 7.6 – Mediana da métrica IGD obtido pelo APMT-MODE, AP-MODE, MOPSO e NSGA-II.

Problemas	**MOPSO**	**APMT**	**AP**	**NSGA_II**
Ecoli Gaussiano	5,1717	16,4054	6,0183	19,8498
Ecoli Polinomial	0,0000	0,0000	0,0000	43,4265
Ecoli Cauchy	2,9954	1,7243	2,1251	6,1783
Ecoli Acos	0,0000	0,0000	0,0000	0,5000
Haberman Gaussiano	6,0061	2,0519	3,0375	7,3416
Haberman Polinomial	3,6270	3,6270	3,6626	16,2116
Haberman Cauchy	0,5061	0,5000	0,5127	6,1221
Haberman Acos	0,5000	0,0000	0,0000	0,0000
Heart Gaussiano	8,4210	8,2322	8,2041	6,7485
Heart Polinomial	9,2874	0,0077	0,7310	7,1400
Heart Cauchy	6,2355	5,8307	6,2786	7,4164
Heart Acos	0,0000	0,0000	0,0000	4,5209
Ionosphere Gaussiano	12,2555	7,0231	7,5805	13,1900
Ionosphere Polinomial	14,9648	14,2454	14,7332	26,9723
Ionosphere Cauchy	15,0619	7,4617	7,3251	24,2214
Ionosphere Acos	0,0000	0,0000	0,0000	1,0000
Iris Gaussiano	0,5000	0,5000	0,5000	2,5495
Iris Polinomial	0,5000	0,5000	0,5000	14,0000
Iris Cauchy	0,5000	0,5000	0,2500	3,5000
Iris Acos	0,0000	0,0000	0,0000	0,0000
Sonar Gaussiano	11,0250	3,0601	6,4140	23,5031
Sonar Polinomial	29,1204	16,6917	25,2492	22,9916
Sonar Cauchy	24,0031	17,7580	2,0000	21,5031
Sonar Acos	0,0000	0,0000	0,0000	0,5000
WDBC Gaussiano	6,6058	3,3089	3,2885	39,1854
WDBC Polinomial	0,0000	0,2500	0,5000	16,4953
WDBC Cauchy	2,5248	1,7906	2,7990	20,2326
WDBC Acos	0,5000	0,0000	0,5000	0,0000
1027 ESL Gaussiano	4,1240	3,9100	3,5663	8,2754
1027 ESL Polinomial	26,4267	28,4892	25,6151	45,7355
1027 ESL Cauchy	4,3918	3,8096	14,4421	7,9575
1027 ESL Acos	64,0838	76,4435	77,8783	110,1342
Auto Gaussiano	122,6570	184,3307	180,8372	173,6212
Auto Polinomial	77,4498	111,8396	101,4027	307,1437
Auto Cauchy	129,4332	137,8383	194,0885	156,2520
Auto Acos	184,6070	204,3442	193,9615	145,2688
Concrete Gaussiano	73,8804	84,3934	78,3859	80,0197
Concrete Polinomial	124614,2556	152749,4264	380137,0964	88400,9141
Concrete Cauchy	32,6706	32,0196	31,0844	67,9231
Concrete Acos	20,6967	21,4341	21,2940	22,5134
Yacht Gaussiano	136,2193	132,2977	133,7860	102,0533
Yacht Polinomial	1994,6831	2223,1223	2224,2784	2405,5000
Yacht Cauchy	167,9579	143,9500	147,3436	117,5682
Yacht Acos	11,9348	12,8520	12,7795	19,4802

Tabela 7.7 – Média dos *rankings* obtidos pelos algoritmos.

Algoritmo	Ranking
APMT-MODE	2,1023
AP-MODE	2,2841
MOPSO	2,3864
NSGA-II	3,2273

Tabela 7.8 – Os *p-values* ajustados obtidos para testes *post hoc* de Nemenyi, Shaffer e Bergmann.

Hipóteses	p não ajustado	p-Neme	p-Shaf	p-Berg
APMT vs. NSGA-II	0,000044	0,000262	0,000262	0,000262
AP vs. NSGA-II	0,000611	0,003665	0,001833	0,001833
MOPSO vs. NSGA-II	0,002249	0,013496	0,006748	0,004499
MOPSO vs. APMT	0,302000	1,000000	0,906001	0,906001
APMT vs. AP	0,508883	1,000000	1,000000	0,906001
MOPSO vs. AP	0,710209	1,000000	1,000000	0,906001

Na Seção 7.4, é realizada uma análise da complexidade das operações matemáticas (em termos de tempo de execução) entre modelos de SVMs, gerados a partir de uma classe de *kernels* Hermitianos e SVMs geradas por *kernels* Gaussianos e polinomial com parâmetros ajustados pelos APMT-MODE.

7.4 ANÁLISE DA COMPLEXIDADE DAS OPERAÇÕES MATEMÁTICAS DOS *KERNELS*

As funções geralmente usadas como *kernels* são funções de base radial (RBFs, principalmente Gaussianas), funções polinomiais e sigmoides (BURGES, 1998). No entanto, cada *kernel* é mais adequado a uma aplicação específica, em que os critérios de escolha são: (a) baixa complexidade, (b) alta capacidade de generalização e (c) poucos parâmetros a serem ajustados. Nesse sentido, trabalhos anteriores demonstram que cada conjunto de dados possui um *kernel* adequado que satisfaz estes requisitos (GUO et al., 2015; MOGHADDAM; HAMIDZADEH, 2016; ZANATY; AFIFI, 2018; CHO; SAUL, 2009; DIBIKE et al., 2001).

Moghaddam e Hamidzadeh (2016) propõem o *kernel* polinomial ortogonal de Hermite, que possui uma boa capacidade de generalização, com um baixo número de SVs e um menor esforço computacional quando comparado com os *kernels* Gaussiano, Wavelet e Chebyshev, para uma lista de *benchmarks* apresentados na Tabela 7.9. Além disso, uma combinação do *kernel* Hermitiano com outros *kernels* tem mostrado alcançar resultados ainda melhores.

O *kernel* Hermite-Chebyshev (H-C) apresenta um número menor de SVs, enquanto o Hermite-Gaussiano (H-G) tem uma taxa de erro menor em todos os *benchmarks* estudados. Em ambos os casos, para os *kernels* híbridos, um esforço computacional mais alto é notado. Os *kernels* propos-

tos em Moghaddam e Hamidzadeh (2016) juntamente com os *kernels* polinomiais e Gaussianos são apresentados na Tabela 7.9.

Tabela 7.9 – Funções *Kernel*.

Kernel	**Função** $K(\mathbf{x}, \mathbf{y})$	**Parâmetros**
Polinomial	$K(\mathbf{x}, \mathbf{y}) = (\beta \mathbf{x} \cdot \mathbf{y} + \kappa)^d$	β, κ, d
Gaussiano	$K(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\gamma}\right)$	γ
Hermite	$K(\mathbf{x}, \mathbf{y}) = \prod_{j=1}^{f} \sum_{i=0}^{n} He_i(x_j) He_i(y_j)$	n
H-G	$K(\mathbf{x}, \mathbf{y}) = exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\gamma}\right) + \prod_{j=1}^{f} \sum_{i=0}^{n} He_i(x_j) He_i(y_j)$	n, γ
H-C	$K(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=0}^{n} T_i(\mathbf{x}) T_i^t(\mathbf{y})}{\sqrt{a - \langle \mathbf{x}, \mathbf{y} \rangle}} + \prod_{j=1}^{f} \sum_{i=0}^{n} He_i(x_j) He_i(y_j)$	n, a

O *kernel* polinomial pode ser simplificado tornando β e κ igual a um, para algumas aplicações específicas, como mostrado por Dibike et al. (2001). Nesse caso, há apenas um parâmetro inteiro d a ser ajustado para produzir resultados satisfatórios.

Tabela 7.10 – Características do *benchmarks* classificadores.

Índice	***Benchmark***	**Classes**	**Instâncias**	**Entradas**
1	Ecoli	8	336	7
2	Glass	7	214	9
3	Haberman	2	306	3
4	Heart	2	267	44
5	Ionosphere	2	351	34
6	Iris	3	150	4
7	Liver	2	345	6
8	Pendigits	10	3498	36
9	Satimage	6	4435	36
10	Sonar	2	208	60
11	Wdbc	2	569	30

O objetivo ao usar *benchmarks* de classificação é demonstrar que uma escolha adequada de hiperparâmetros pode produzir melhores soluções usando *kernels* clássicos e mais simples do que os apresentados em (MOGHADDAM; HAMIDZADEH, 2016). As características dos *benchmarks* são apresentadas na Tabela 7.10. Para uma comparação justa, como descrito em (MOGHADDAM; HAMIDZADEH, 2016), a estratégia *cross-validation* (com dez pastas) foi aplicada em todos os experimentos.

Cada problema de *benchmark* foi resolvido empregando os *kernels* polinomiais e Gaussianos e usando o algoritmo APMT-MODE. Para cada *benchmark* e para cada *kernel*, uma frontreira de Pareto foi gerada com um conjunto de Pareto correspondente. A Tabela 7.11 apresenta o conjunto de Pareto (C, D), junto com sua respectiva fronteira de Pareto ($Error$, $\#SV$), para o *kernel* polinomial do *benchmark* Ecoli (como exemplo de solução gerado pelo APMT-MODE). Para melhor comparar esses resultados com os de (MOGHADDAM; HAMIDZADEH, 2016), para cada *benchmark*, foi escolhida uma solução com erro similar, e um menor número de SVs, a partir de cada PF. No caso particular da Tabela 7.11, o primeiro resultado foi selecionado.

Tabela 7.11 – Fronteira de Pareto e conjunto de Pareto para o *benchmark* Ecocli com *kernel* polinomial.

Índice	**C**	**Grau**	**Erro**	**#SV**
1	2,50	1	12,4	126
2	10,0	1	14,8	99
3	1,75	2	12,7	112
4	-1,25	7	13,3	104
5	7,00	1	14,5	101
6	6,25	1	14,2	102
7	5,50	1	13,9	103
8	-2,75	6	13,9	103

Os *kernels* Hermite, H-C e H-G foram escolhidos para serem comparados com os *kernels* Gaussianos e polinomiais, pois são eles que mostram os melhores desempenhos sobre os demais *kernels* apresentados por Moghaddam e Hamidzadeh (2016). A Tabela 7.12 apresenta os resultados dos hiperparâmetros otimizados definidos pelo APMT-MODE para os *kernels* polinomiais e Gaussianos, em comparação com os resultados empregando os *kernels* Hermite, H-C e H-G em (MOGHADDAM; HAMIDZADEH, 2016).

Pode-se observar que, na maioria dos resultados obtidos pelo algoritmo APMT-MODE, o erro ou é menor ou, no máximo, 19% maior que os resultados anteriores, exceto pelo *benchmark* Satimage. Por outro lado, todos os resultados do APMT-MODE apresentam uma redução de 15% a 85% na quantidade de vetores de suporte (#SVs), sendo a única exceção o *benchmark* Sonar em que os resultados são equivalentes.

Outra maneira de comparar cada solução é medir seu esforço computacional. Essa métrica tem alta importância para o consumo de energia e desempenho em tempo real, especialmente em sistemas embarcados. Neste caso, cada configuração SVM e *kernel* foi implementada em *linguagem C*, compilada e executada em um processador ARM (Raspberry Pi 3). Os resultados para cada configuração são apresentados na Tabela 7.12 e também são mostrados na Figura 7.2, onde o tempo total de cálculo para cada *kernel* é exibido como uma quantidade relativa ao maior tempo de cada *benchmark*.

A partir da Figura 7.2, pode-se observar que tanto os *kernels* polinomiais quanto os Gaussianos, com parâmetros definidos pelo APMT-MODE, representam uma pequena fração do tempo total quando comparados aos *kernels* apresentados em (MOGHADDAM; HAMIDZADEH, 2016). Essa redução de tempo deve-se à sua menor complexidade matemática e também à menor quan-

Tabela 7.12 – Taxa de erro e #SVs dos *kernels* Hermite, H-C e H-G, Gaussian e polinomial, onde #SV é a cardinalidade do conjunto de vetores de suporte (MOGHADDAM; HAMIDZADEH, 2016). Os melhores resultados são destacados em negrito.

Benchmark	***Kernels* Hermite (MOGHADDAM; HAMIDZADEH, 2016)**						**APMT-MODE**			
	Hermite		**H-C**		**H-G**		**Gaussiano**		**Polinomial**	
	#SV	**Erro**	**#SV**	**Erro**	**#SV**	**Erro**	**#SV**	**Erro**	**#SV**	**Erro**
Ecoli	278	11,89	273	13,10	161	**10,09**	**117**	11,81	126	12,42
Glass	153	28,30	148	30,10	151	**26,10**	102	28,09	**95**	26,19
Haberman	163	24,09	167	27,16	161	24,09	142	25,33	**137**	**24,00**
Heart	135	18,27	131	19,47	132	**17,07**	94	17,69	**89**	18,46
Ionosphere	104	6,21	92	4,14	107	**4,01**	**79**	4,57	79	7,28
Iris	75	3,40	107	4,10	101	3,10	12	2,66	**11**	**2,00**
Liver	293	26,11	287	**24,28**	291	24,90	139	26,66	**133**	26,66
Pendigits	2543	24,18	2876	27,10	2760	25,72	332	0,74	**287**	**0,28**
Satimage	3711	8,26	3432	**5,17**	3510	9,42	1256	7,83	**1062**	9,09
Sonar	75	14,09	**73**	16,19	79	13,19	80	**12,50**	73	15,50
Wdbc	75	3,69	70	4,12	79	**2,18**	**48**	2,60	54	2,79

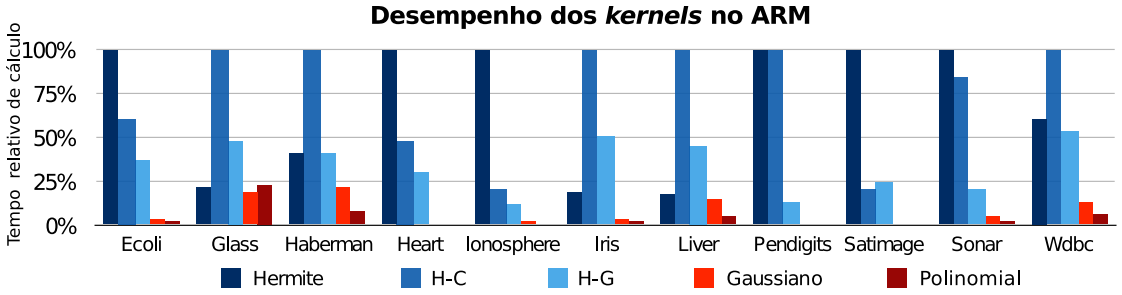


Figura 7.2 – Comparação do esforço computacional entre diferentes *kernels* para cada *benchmark*.

tidade de SVs que os modelos possuem, evidenciando que o APMT-MODE é capaz de gerar modelos eficientes, quando considerados os critérios complexidade e capacidade de generalização, frente a *kernels* mais complexos computacionalmente.

Dos resultados apresentados na geração de modelos com os *kernels* Gaussiano e polinomial, conclui-se que o APMT-MODE é capaz de encontrar parâmetros para os modelos de SVM/SVRs, que possam ser empregados em situações práticas em que há limitações de *hardware* e tempo de processamento, como, por exemplo, geração de modelos para monitoramento da penetração e largura de processos de solda *vide* (Seção 7.6).

7.5 ESTUDO COMPARATIVO ENTRE APMT-MODE E *GRID-SEARCH* - GS

A meta-heurística desenvolvida, APMT-MODE, foi aplicada para obter modelos de SVRs e comparar seus resultados obtidos com a técnica de busca exaustiva GS *vide* (NARZISI, 2007; FRIEDRICHS; IGEL, 2005a) e Seção 3.1. Para este estudo, foram empregados os *kernels* polinomial e Gaussiano. Os conjuntos de dados foram divididos em conjuntos de treinamento, validação e teste com cardinalidade de 70%, 15% e 15%, respectivamente.

O GS consiste em uma busca exaustiva no espaço das variáveis de decisão, em que uma variável de decisão está associada a um eixo do espaço. Quatro *benchmarks* foram escolhidos para comparação e, em todos os casos, o espaço de busca é definido pelos intervalos $C = \{c \in \mathbb{R} : -10 \leq c \leq 10\}$, $\Gamma = \{\gamma \in \mathbb{R} : -10 \leq \gamma \leq 10\}$ e $D = \{d \in \mathbb{Z} : 0 < d \leq 10\}$ com incrementos de $0,25$, $0,25$ e 1, respectivamente.

Os conjuntos, $C \times D$ e $C \times \Gamma$, resultantes do produto cartesiano, são as malhas para os *kernels* polinomiais e Gaussianos, respectivamente. A malha resultante para o *kernel* Gaussiano possui 6400 pontos, ou seja, foram gerados este número de modelos igualmente distribuídos pelo espaço de busca. Para o *kernel* polinomial, a malha foi gerada pelo mesmo procedimento, entretanto, o grau do polinômio é inteiro, resultando em uma malha com 800 pontos.

O PSP, modelado como MOOP, para os regressores, possuem três variáveis de decisão, o que torna a busca, pelo GS, impraticável. Portanto, o parâmetro ϵ-*insensitive* foi pré-ajustado para o valor médio encontrado previamente pelo APMT-MODE, em cada caso. A Figura 7.3 representa as superfícies das funções objetivos E_e e C_{SV}, para o *benchmark* 1027 ESL, para um ϵ-*insensitive* fixo e igual a um.

Na Figura 7.3, pode-se notar as diferentes características das funções objetivos, que possuem regiões de nível, de rugosidade e desníveis abruptos. Os asteriscos representam a fronteira de Pareto encontrada pelo GS, considerando os 800 modelos gerados pelo GS (pontos da malha).

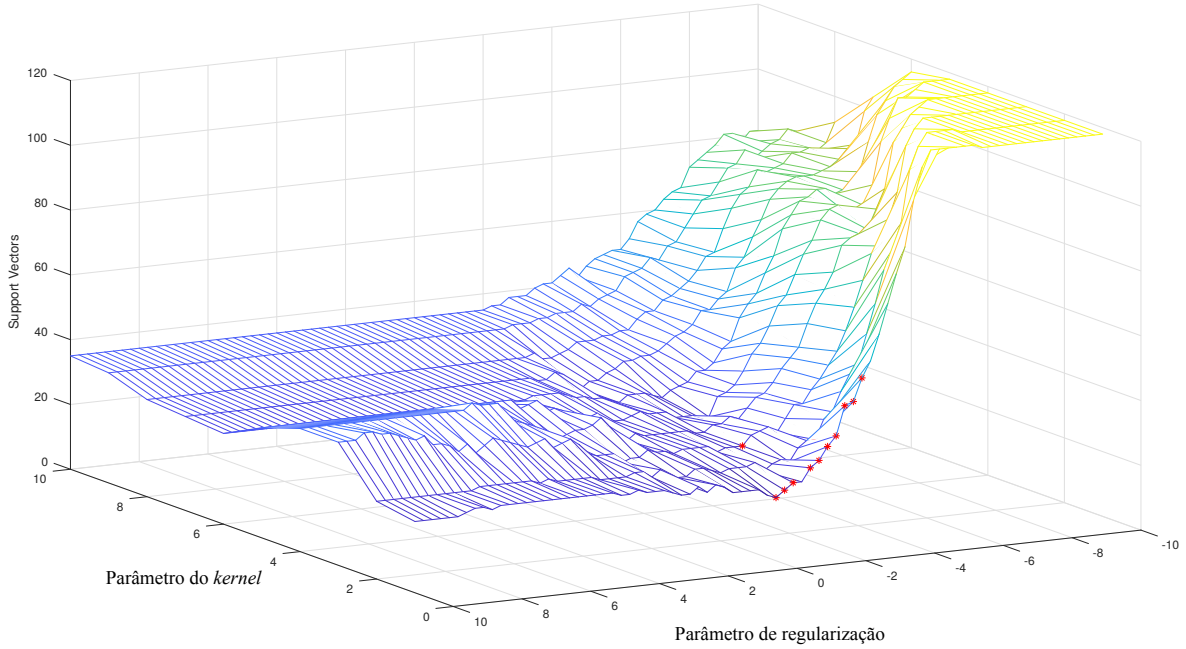
A metodologia empregada para a obtenção da Figura 7.4 foi a mesma da Figura 7.3. Observa-se na Figura 7.4 que a mesma possui várias regiões de nível e alguns mínimos locais, alguns vales, que se apresentam com formas bem distintas da Figura 7.3. Os gráficos estão na mesma orientação, permitindo, assim, a comparação da localização entre os pontos (asteriscos) que representam a fronteira de Pareto.

Destaca-se, ainda, na Figura 7.4, que as superfícies das duas funções objetivos são consideravelmente distintas em relação às superfícies apresentadas pela Figura 7.3, mais uma vez evidenciando a complexidade do PSP e a necessidade de se desenvolverem meta-heurísticas que encontrem soluções de boa qualidade para diversos *benchmarks*, combinados com distintos *kernels*.

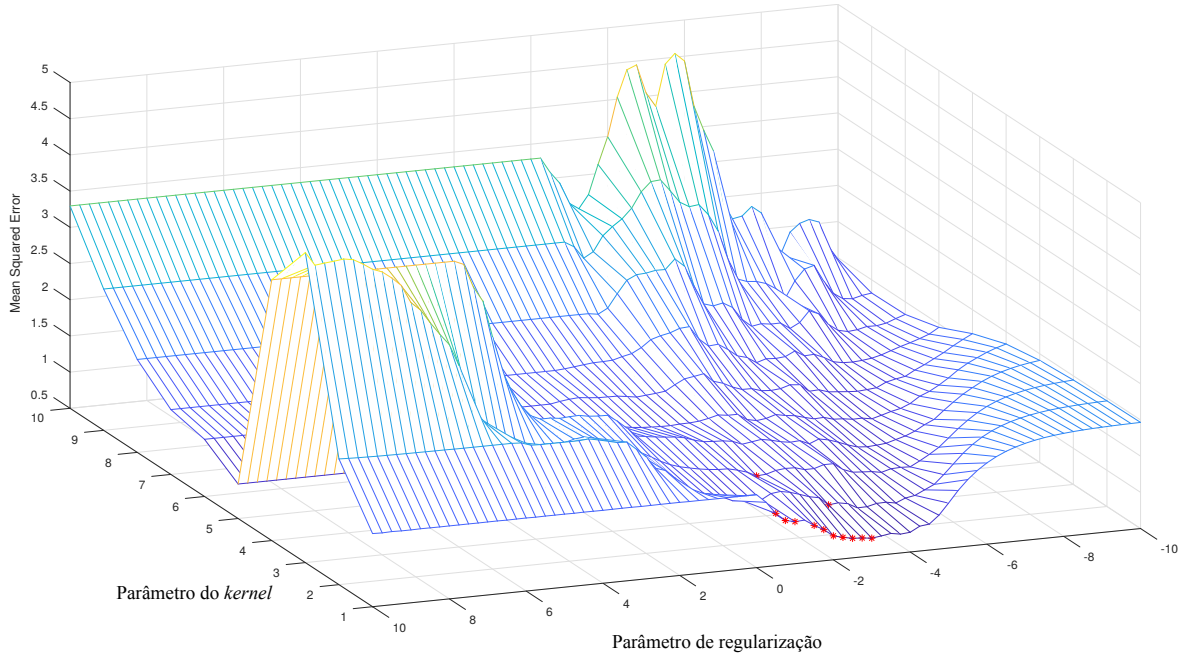
As funções objetivos $C_{SV}(\omega) = (\#SV)$ e $E_e(\omega) = MSE$, como descrito na Seção 3.1.1, são empregadas para ambos, APMT-MODE e GS. A fronteira de Pareto do GS é definida pela função *Truncate*, considerando todos os modelos gerados pelos parâmetros da malha. Os resultados de cada método são mostrados nas figuras 7.5 e 7.6, em que a PF gerada pelo GS é representada pelos asteriscos em azul e a gerada pelo APMT-MODE pelos asteriscos em vermelho.

Os melhores resultados de cada PF estão resumidos em Tabela 7.13. Como pode ser observado, os PFs encontrados pelo algoritmo APMT-MODE apresentam maior proximidade com a origem do sistema de eixo do espaço objetivo, melhor distribuição e maior cardinalidade, o que evidencia, graficamente, os melhores resultados obtido pelo APMT-MODE.

Adicionalmente, observou-se experimentalmente que o APMT-MODE alcança melhores re-



(a) Espaço de busca da função objetivo SV.

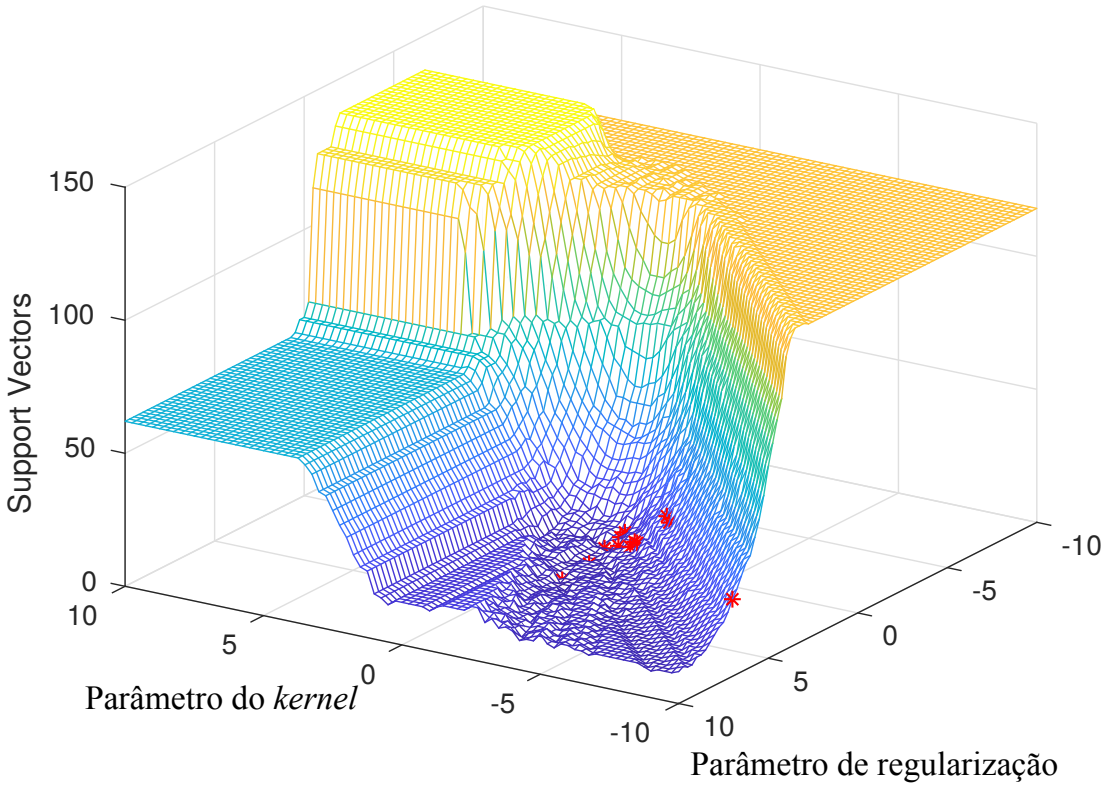


(b) Espaço de busca da função objetivo MSE.

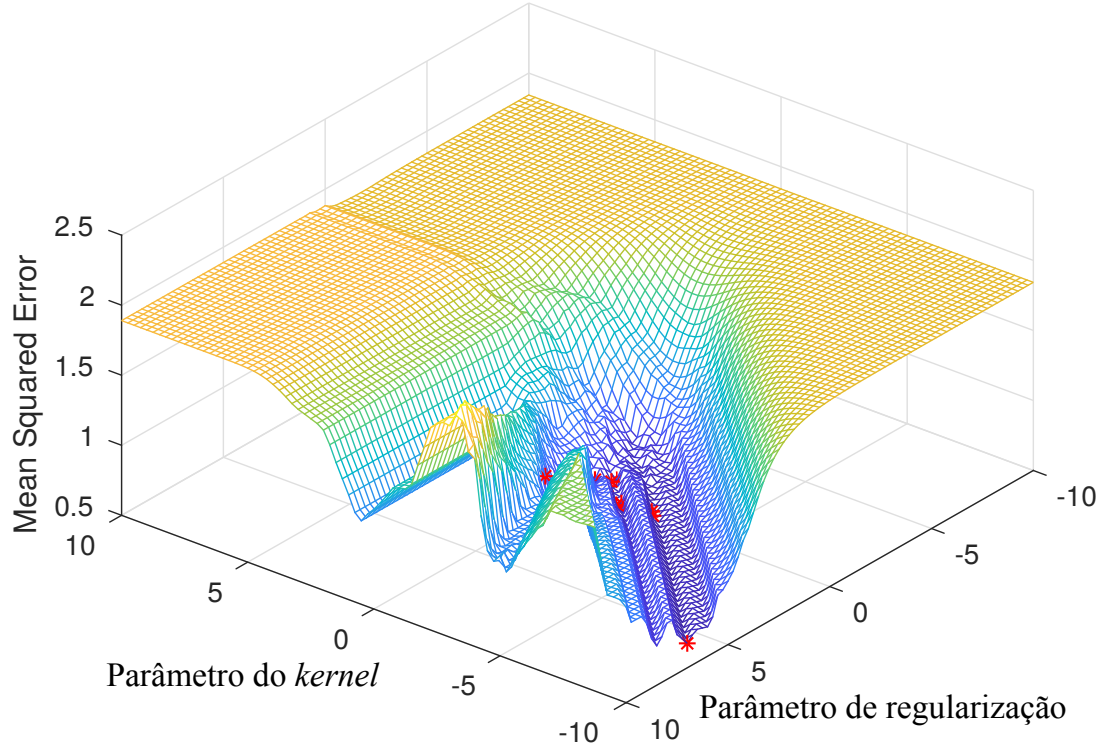
Figura 7.3 – Espaço de busca *benchmark* 1027 ESL com *kernel* polinomial.

sultados utilizando pelo menos 25% menos avaliações de função objetivo para o *kernel* polinomial e 43% menos avaliações para o *kernel* Gaussiano se comparado à malha GS. Parte deste resultado pode ser explicada devido ao tamanho do passo fixo usado na malha GS. Uma pesquisa mais detalhada usando o método GS aumentaria significativamente o esforço computacional necessário para obter a solução e poderia se tornar inviável.

Na Seção 7.6, é apresentado um estudo de caso em que foram gerados modelos SVRs em-



(a) Espaço de busca da função objetivo SV.



(b) Espaço de busca da função objetivo MSE.

Figura 7.4 – Espaço de busca *benchmark* 1027 ESL com *kernel* polinomial.

pregados no controle do processo de soldagem. Estes estudos visam mostrar a capacidade do APMT-MODE gerar bons modelos para aplicações reais.

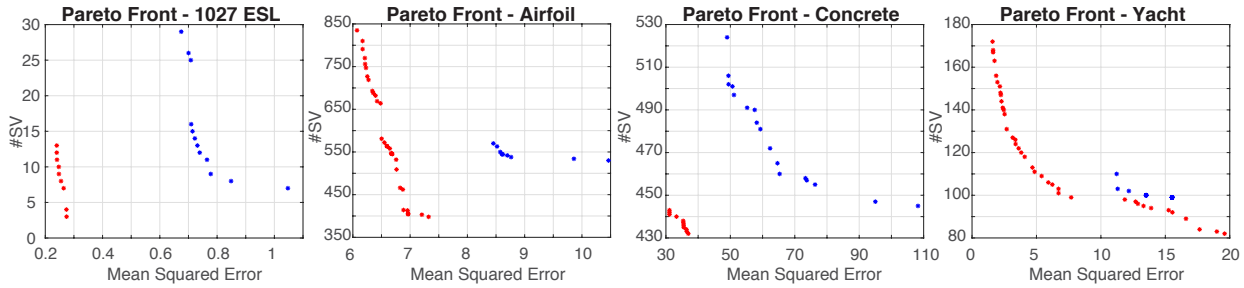


Figura 7.5 – Gráfico comparativo, da fronteira de Pareto obtida, entre o APMT-MODE (asteriscos vermelhos) e o GS (asteriscos azuis), para *benchmarks* usando o *kernel* Gaussiano.

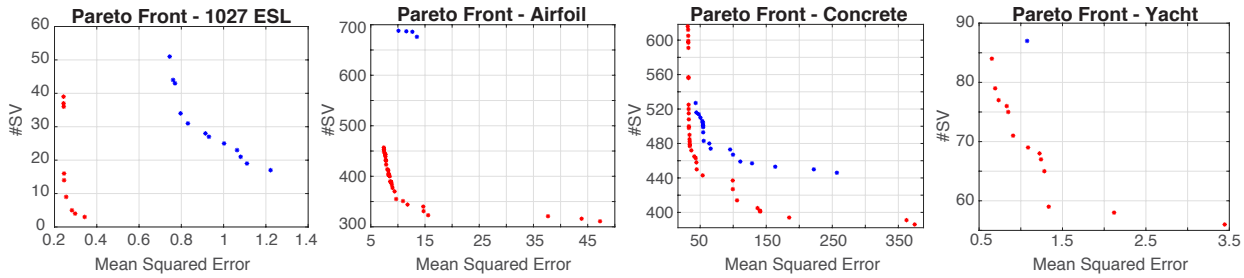


Figura 7.6 – Gráfico comparativo, da fronteira de Pareto obtida, entre o APMT-MODE (asteriscos vermelhos) e o GS (asteriscos azuis), para *benchmarks* usando o *kernel* polinomial.

Tabela 7.13 – Resultados para *benchmarks* de regressão. As entradas em negrito destacam os melhores resultados.

Benchmark	**Inst.**	**Entradas**	**GS**				**APMT-MODE**			
			Gaussiano		**Pol.**		**Gaussiano**		**Poli.**	
			MSE	**#SV**	**MSE**	**#SV**	**MSE**	**#SV**	**MSE**	**#SV**
1027 ESL	489	4	0,67	29	0,70	32	**0,24**	**11**	**0,24**	39
Airfoil	1503	5	8,45	570	9,85	481	**7,32**	**398**	7,40	457
Concrete	1030	8	48,9	524	43,16	527	**31,10**	**441**	31,17	616
Yacht	308	6	15,53	99	1,07	87	1,59	172	**0,65**	**84**

7.6 APLICAÇÃO DO APMT-MODE NO PROBLEMA DE GERAÇÃO DE MODELOS PARA CONTROLE PROCESSOS DE SOLDA

Para uma validação final do algoritmo APMT-MODE, foi selecionado um estudo de caso do preditor de geometria do cordão de solda apresentado em (BESTARD et al., 2018). Neste contexto, modelos do processo de soldagem são usados para prever a penetração e a largura do cordão de solda, com base em um método de fusão de sensor termográfico que monitora a geometria do cordão de solda para controlar um processo os parâmetros do *Gas Metal Arc Welding* - GMAW.

Tanto a profundidade quanto a largura são essenciais para controlar os parâmetros do processo de soldagem visando obter um cordão de solda de boa qualidade *vide* (Figura 7.7, com exemplo de um cordão de solda). Alguns aspectos que tornam o modelo de regressão para este problema desafiador são seu comportamento altamente não linear, bem como a necessidade de uma previsão de resposta rápida para o controle em tempo real de sua dinâmica.

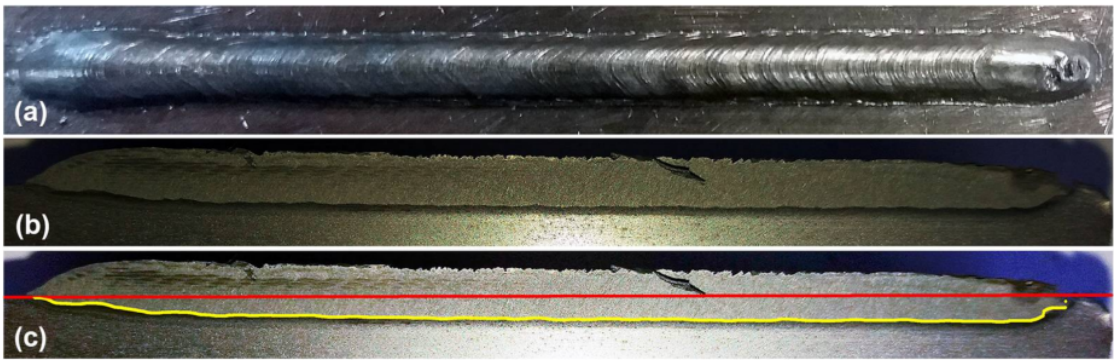


Figura 7.7 – Cordão de solda produzido para obtenção dos dados de validação e treinamento dos modelos SVRs. Tomado de Bestard et al. (2018).

Para o controle em tempo real do processo de soldagem, foram gerados os modelos de predição da penetração e largura do cordão de solda. Em ambos os casos, o vetor de entrada do modelo é composto de oito variáveis de decisão: (a) o pico termográfico (T_p), (b) a base termográfica (T_b), (c) a área termográfica (T_a), (d) o volume termográfico (T_v), (e) a largura termográfica (T_w), (f) a medição da corrente e tensão de soldagem (iu) na corrente (nT), (g) a amostra anterior da corrente e voltagem ($nT-T$), e (h) o valor estimado anterior $\hat{DW}(nT-T)$. O símbolo T é o tempo da amostra e n é o número da amostra (BESTARD et al., 2018).

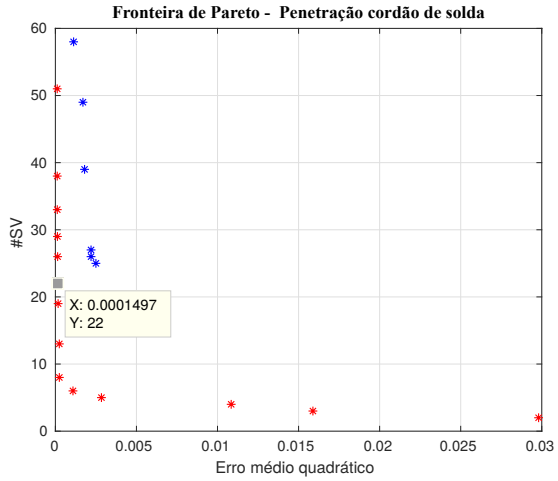
O APMT-MODE desenvolvido, descrito na Seção 5.2, foi usado para encontrar modelos que podem prever a penetração e a largura do cordão de solda com o MSE e a complexidade mínima em sua implementação de *hardware*. Assim, foi escolhido o *kernel* polinomial pelo fato de sua arquitetura de *hardware* ser mais simples e rápida. Os modelos SVR foram treinados, validados e testados com dois conjuntos de dados, cada um com 607 pontos de dados, um para a previsão de profundidade e outro para a previsão de largura.

O processo de treinamento produziu duas fronteiras de Pareto, e seus resultados são mostrados na Figura 7.8, em que os dados etiquetados significam os resultados escolhidos para implementar em *hardware* o estudo de caso.

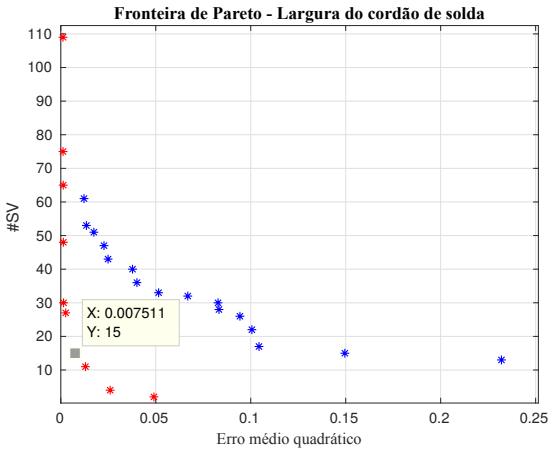
Para selecionar a melhor solução de cada PF, dois requisitos precisavam ser atendidos. O primeiro diz respeito à restrição do processo de soldagem quanto ao erro máximo de estimativa para a profundidade do cordão de solda de $5,0 \times 10^{-4}$ e largura de $1,28 \times 10^{-1}$. O segundo requisito é a quantidade SV, para um dado *kernel*, possíveis de serem implementados em paralelos na plataforma FPGA escolhida para uso em tempo real, como será descrito na Seção 7.6.1, que é no máximo 50 *kernels* polinomiais.

Os modelos escolhidos são destacados nas Figuras 7.8a e 7.8b. Essas soluções têm MSEs de $58,08\%$ e $76,7\%$ inferiores ao Perceptron NN projetado por (BESTARD et al., 2018) para profundidade e largura do cordão de solda, respectivamente. As Figuras 7.9a e 7.9b representam o comportamento dos modelos em relação ao conjunto de dados de teste, para predição de profundidade e largura, respectivamente.

A qualidade dos modelos pode ser observada no gráfico de resíduos *vide* (Figura 7.9a), que

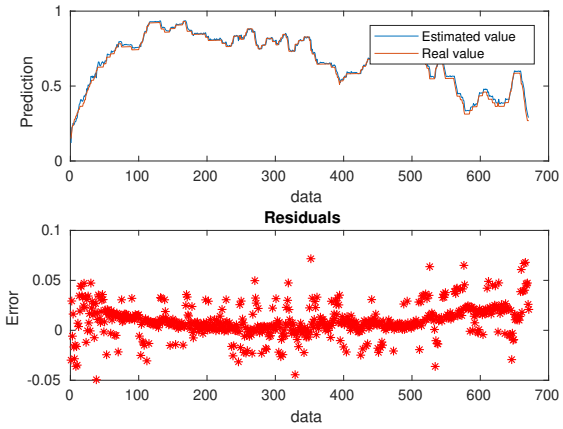


(a) Fronteira de Pareto do modelo de penetração.

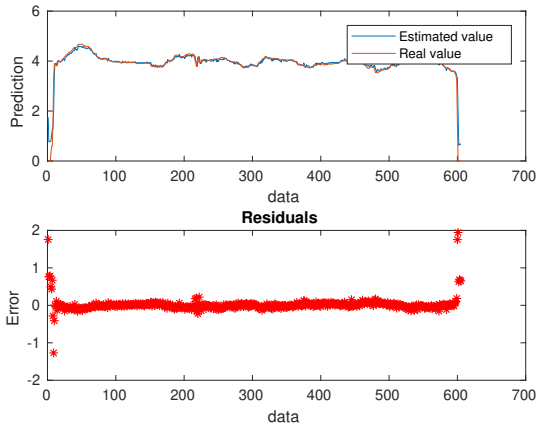


(b) Fronteira de Pareto do modelo de largura.

Figura 7.8 – Gráficos comparativos entre as fronteiras de Pareto do APMT-MODE e GS. APMT-MODE (asteriscos vermelhos) e GS (asteriscos azuis).



(a) Modelo preditor da penetração do cordão de solda.



(b) Modelo preditor da largura do cordão de solda.

Figura 7.9 – Gráficos da precisão e resíduos da penetração e largura do cordão de solda.

deve ser uma distribuição normal com uma média de zero e desvio-padrão mínimo. No modelo de profundidade do cordão de solda, alguns resíduos apresentam algum padrão e não passam no teste de normalidade. Isto se deve ao fato de que esses resíduos foram feitos sobre o conjunto de testes e não sobre o conjunto de treinamento. Mesmo assim, esses resultados são bons, mostrando que os modelos têm uma boa capacidade de generalização.

O modelo de largura do cordão de solda não tem padrão no gráfico de resíduos e passa no teste de normalidade sobre o conjunto de teste. Os pontos distantes de zero são *outliers* devido à alta instabilidade no início e no final do processo de soldagem.

O coeficiente de correlação (R^2) mede a relação estatística entre os valores previstos e rotulados. Os modelos de profundidade e largura do cordão de solda têm $R^2 = 0,988$ e $R^2 = 0,916$,

respectivamente, e, assim, em ambos os casos, estão próximos de um, produzindo resultados satisfatórios.

7.6.1 Resultados da implementação de *hardware* em *Field Programmable Gate Array* - FPGA do modelo SVR para previsão da penetração e largura do cordão de solda

Para uma validação final do algoritmo APMT-MODE, foi selecionado um estudo de caso do preditor de geometria do cordão de solda apresentado em (BESTARD et al., 2018). Nesse contexto, modelos do processo de soldagem são usados para prever a penetração e a largura do cordão de solda, com base em um método de fusão de sensor termográfico que monitora a geometria do cordão de solda, para controlar um processo os parâmetros do *Gas Metal Arc Weilding* - GMAW.

A implementação em *hardware* do modelo SVR é baseada na Equação 2.42 e o *kernel* polinomial. A arquitetura é desenvolvida em *Very High Description Language* - VHDL e seu módulo superior consiste em blocos de *kernel* paralelos (KBs) seguidos por uma unidade *Multiply-Accumulate* - MAC controlada por uma *Finite State Machine* - FSM, como mostrado na Figura 7.10. As operações aritméticas são implementadas com bibliotecas de ponto flutuante de largura de bit ajustáveis baseadas no padrão IEEE 754. Essas bibliotecas são descritas em (NOZ et al., 2009) e (NOZ et al., 2010) e oferecem melhor precisão e maior alcance dinâmico adequado para números reais pequenos e grandes em comparação com a aritmética de inteiros de ponto fixo ou simples.

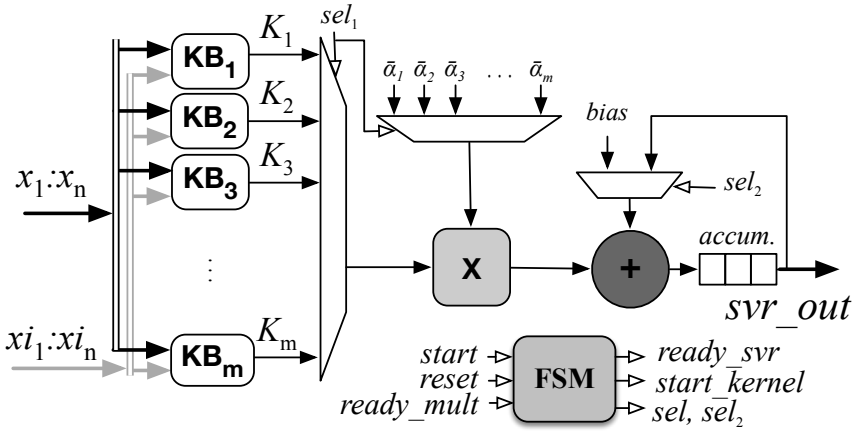


Figura 7.10 – Arquitetura em *hardware* do modelo SVR.

O módulo do *kernel* polinomial usa um multiplicador e um somador, em conjunto com multiplexadores e um FSM interno para controlar o processo conforme apresentado na Figura 7.11.

Além disso, uma ferramenta de configuração de arquitetura de *hardware* para o SVM foi desenvolvida para gerar módulos automaticamente, com um número variável de blocos de *kernel* paralelos, entradas, pesos e *bias*.

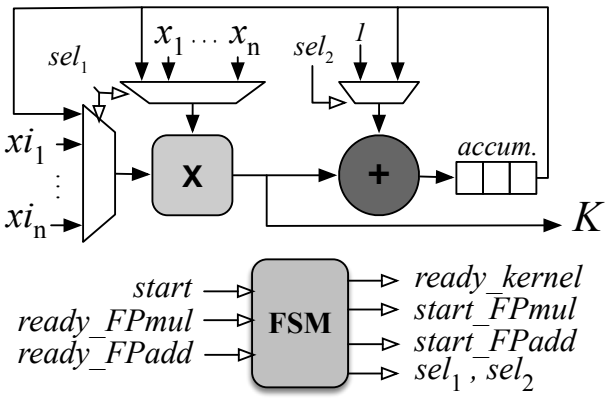


Figura 7.11 – Arquitetura do bloco *kernel* polinomial.

A Tabela 7.14 apresenta os resultados da síntese de FPGA dos preditores de SVR de penetração e largura em comparação com a arquitetura similar descrita em (BESTARD et al., 2018). Para obter estes resultados, foi usada a ferramenta Quartus-II da Altera (INTEL, 2012), a plataforma empregada é um Altera Cyclone V (5CSEMA5F31C6N). Pode-se notar que o SVR usando o *kernel* polinomial usa pelo menos 43% menos recursos para o preditor de profundidade e pelo menos 59% menos recursos para o preditor de largura. No desempenho de computação, o SVR também é 60% e 105% mais rápido para os preditores de profundidade e largura, respectivamente.

Tabela 7.14 – Resultados de síntese de *hardware* comparando a arquitetura SVR com o Perceptron NN de (BESTARD et al., 2018).

Arquitetura	**FP Precisão**	**Uso (ALMs) (Elementos/%)**	**Uso (DSPs) (Elementos/%)**	**Frequência (MHz)**	**Tempo (μs)**
Perceptron NN	32 bits	19,809 / 62%	24 / 21%	117.08	1.71
(12 neurônios)	27 bits	16,615 / 52%	24 / 21%	125.77	1.59
	24 bits	14,233 / 44%	24 / 21%	130.19	1.54
SVR Poli. Pen	32 bits	11,238 / 35%	23 / 20%	105.59	0.98
(22 SVs)	27 bits	8,890 / 28%	23 / 20%	105.53	0.99
	24 bits	7,758 / 24%	23 / 20%	112.60	0.92
SVR Poli. Larg	32 bits	7,980 / 25%	16 / 14%	103.47	0.80
(15 SVs)	27 bits	6,439 / 20%	16 / 14%	107.77	0.77
	24 bits	5,359 / 16%	16 / 14%	108.30	0.75

Na Seção 7.7, são apresentadas as conclusões do capítulo.

7.7 CONCLUSÃO DO CAPÍTULO

Os algoritmos APMT-MODE, AP-MODE, MOPSO e NSGA-II foram submetidos a uma série de problemas com características distintas. Estes são resultantes da combinação de *benchmarks*,

classificadores e regressores, conjuntamente aos *kernels*: gaussiano, polinomial, arco cosseno e Cauchy.

A partir do teste de Friedman, conclui-se que os algoritmos APMT-MODE, AP-MODE, MOPSO e NSGA-II produzem resultados estatisticamente diferentes entre eles, enquanto que os testes *post hoc* identificam quais possuem melhor desempenho. Os algoritmos APMT-MODE, AP-MODE e MOPSO são estatisticamente iguais e estes se diferem do algoritmo NSGA-II. A Tabela 7.7 apresenta o APMT-MODE com melhor desempenho para este grupo de problemas em relação aos demais algoritmos estudados.

O fato de os algoritmos APMT-MODE e AP-MODE serem mais eficientes que o NSGA-II pode ser explicado devido à natureza diversa dos problemas estudados. Cada problema também possui, como característica, diversos formatos de superfície das funções objetivo, como por exemplo, regiões de nível, rugosidade e desníveis abruptos. Essas características combinadas acrescentam significativamente a complexidade em se aproximar da fronteira de Pareto.

Entretanto, estatisticamente, o MOPSO implementado não possui diferença nos resultados entre os algoritmos APMT-MODE e AP-MODE, mas, para gerar soluções mais competitivas, deve-se ajustar seus parâmetros de acordo com o problema em questão. O estudo realizado em Santos et al. (2017) apresenta resultados satisfatórios para parâmetros adequadamente ajustados para o problema de controle preditivo não linear quando utilizado o MOPSO, enquanto que o APMT-MODE e AP-MODE possuem maior chance de adaptarem-se às situações diversas apresentada pelo problema de seleção de parâmetros.

Os resultados da Seção 7.4 mostram que, na maioria dos casos, as soluções apresentaram menores erros e menor número de vetores de suporte, em comparação aos *kernels* de referência em (MOGHADDAM; HAMIDZADEH, 2016). Além disso, todas as soluções apresentadas têm pelo menos quatro vezes menos esforço computacional. Os resultados da regressão mostram melhorias semelhantes, comparados ao método de busca GS para o treinamento de SVR.

Finalmente, um estudo de caso de um preditor de geometria de cordão de soldagem baseado em um SVR é desenvolvido e comparado com um projeto de referência que usa um Perceptron NN apresentado em (BESTARD et al., 2018). Uma arquitetura FPGA do SVR é implementada e, mais uma vez, os resultados mostram melhor precisão e menor uso de recursos de *hardware* e um desempenho até duas vezes mais rápido.

Diante do exposto, a meta-heurística APMT-MODE mostrou-se uma ferramenta capaz de gerar soluções de boa qualidade usando *kernels* clássicos e simples computacionalmente, quando considerados os quesitos precisão e simplicidade.

8 CONCLUSÕES

As SVM/SVRs são uma técnica de aprendizagem de máquinas em que o modelo de treinamento é baseado em margens largas, garantindo a minimização do risco empírico e a capacidade de generalização simultaneamente, além desta característica, o modelo de treinamento consiste em um problema de otimização quadrática e convexa. Portanto, o treinamento das SVM/SVRs pode ser realizado por meio de métodos que empregam o gradiente sem o risco de encontrar soluções que são mínimos locais. Por isso, conforme descrito no Capítulo 4, as SVM/SVRs têm sido amplamente empregadas em identificação de padrões, predição.

Verificou-se que o algoritmo GS exige um grande esforço computacional para encontrar as soluções de boa qualidade para o problema de seleção de parâmetros e, mesmo assim, devido à granularidade da malha, este não é capaz de encontrar soluções compatíveis às encontradas pelos algoritmos APMT-MODE, MOPSO e AP-MODE. Para as SVRs que possuem o parâmetro ϵ-*insenstive*, além do parâmetro de regularização e dos parâmetros do *kernels*, o GS torna-se impraticável, tendo em vista que sua complexidade é $O(n^m)$, em que n é a granularidade da malha e m é o número de variáveis de decisão.

O PSP foi modelado como um MOOP, logo o emprego da metodologia multiobjetivo mostrou uma técnica em que, embora aumente a complexidade do PSP, os resultados oferecem ao projetista flexibilidade na escolha de uma solução de qualidade que seja adequada a sua aplicação, pois cada uma possui restrições distintas relacionada à complexidade e à precisão das SVM/SVRs.

Como consequência da primeira hipótese, levantada na Seção 1.3, foram desenvolvidos os algoritmos meta-heurísticos APMT-MODE, AP-MODE e MOPSO, os quais produzem resultados estatisticamente equivalentes, para o grupo de problemas estudados. Entretanto o APMT-MODE, por ter um *ranking* melhor, pode ser considerado mais adequado para encontrar boas soluções do problema.

A segunda hipótese, Seção 1.3, refere-se à não existência de algoritmos meta-heurísticos multiobjetivos para sintonização de hiperparâmetros de SVRs. No Capítulo 4, foi confirmada a hipótese. O APMT-MODE encontra um conjunto de soluções não dominantes satisfatório para diversos modelos *benchmarks* de SVRs, assim como modelos para o processo de soldagem (com fortes não linearidades). Portanto, a eficiência do algoritmo meta-heurístico APMT-MODE é comprovada.

As técnicas de adição de diversidade OBL e AR foram implementadas com o objetivo de aumentar a capacidade de exploração dos algoritmos APMT-MODE, AP-MODE e MOPSO, entretanto o OBL exige um esforço computacional proibitivo, pois, a cada iteração, o número de avaliações da função custo dobra. A técnica Atrativo Repulsivo, embora não acrescente esforço computacional, configura dois parâmetros extra, os quais são dependentes do problema em questão.

O modelo de treinamento das SVM/SVRs possui a vantagem de depender somente do produto interno entre vetores característicos que compõem os dados de treinamento, sendo assim, é possível o emprego do truque do *kernel*, para levar os dados do espaço original (onde são não linearmente separáveis) para um espaço de dimensão maior (onde são linearmente separáveis). Entretanto a capacidade de generalização e a complexidade das SVM/SVRs dependem, além dos hiperparâmetros da escolha adequada, para cada conjunto de treinamento, do *kernel* a ser empregado. Esta escolha acrescenta uma dificuldade adicional à definição do modelo de treinamento, pois alguns *kernels* possuem operações matemática complexas, problema de instabilidade numérica e/ou vários parâmetros a serem configurados.

Os estudos desenvolvidos na Seção 7.4 mostraram que o algoritmo meta-heurístico APMT-MODE juntamente com a flexibilidade dada pelo sistema NIOTS, que permite avaliar diferentes tipos de *kernels*, mostrou-se capaz de resolver diversos PSP. Adicionalmente, conclui-se que *kernels* clássicos podem ser eficientes para sistemas com restrições de recursos de *hardware*, atendendo, a terceira hipótese levantada na Seção 1.3.

O desempenho das SVM/SVRs está fortemente relacionado com o PSP, que possui mínimos locais, regiões de nível, características que dificultam a determinação dos hiperparâmetros adequados. Adicionalmente, cada problema (conjunto de treinamento e *kernel*) possui suas especificidades e, portanto, definir um método analítico para resolvê-lo ou mesmo configurar uma meta-heurística é uma tarefa que exige esforço considerável e conhecimento sobre o tema. Portanto, o AP-MODE e o APMT-MODE são capazes de resolver o PSP modelado como MOOP, devido às suas características adaptativas associadas à capacidade de exploração do algoritmo DE, e, portanto, atendendo à quarta hipótese levantada na Seção 1.3.

O sistema NIOTS foi implementado com o objetivo de tornar intuitivo o uso de todas as técnicas relacionada a encontrar um conjunto de soluções não dominantes o mais próximo possível da fronteira de Pareto para o problema de seleção de parâmetros. No Capítulo 6, tem-se a descrição de todas as funcionalidades disponíveis no NIOTS. Com esta ferramente, é possível testa vários tipos de *kernels*, técnicas de adição de diversidade, escolha entre os algoritmos APM-MODE, AP-MODE, MOPSO e de GS. Os relatórios gerados permitem a análise posterior dos resultados e o uso para geração de modelos em outras linguagens de programação.

No Capítulo 7, o teste estatístico de Friedman, juntamente com os testes *post hoc*, demonstrou que os algoritmos propostos APMT-MODE, AP-MODE e MOPSO não possuem diferença entre si, mas produzem resultados significativamente melhores que os apresentados para o NSGA-II. O NSGA-II é um algoritmo de otimização multiobjetivo clássico muito utilizado na literatura para solução de problemas MOOP *vide* (Tabela 4.1), o que ressalta a contribuição dos algoritmos propostos neste trabalho.

A métrica hipervolume não se apresentou eficiente para o problema de seleção de parâmetros modelada como MOOP, pois a métrica indica convergência prematura dos algoritmos o que não se verifica, de fato, quando observada o conjunto não dominante nas iterações iniciais, este fato se deve ao fato de o hipervolume medir a diversidade e a distância do ponto *nadir* em um escalar.

Uma hipótese provável para este comportamento deve-se, por exemplo, à diminuição da diversidade do conjunto de soluções não dominantes enquanto que este se afasta do *nadir*, isto mantém o hipervolume constante ou com variações infinitesimais.

Um conjunto de soluções não dominantes de hiperparâmetros de qualidade tem-se mostrado essencial para obtenção de modelos de SVM/SVRs com alta capacidade de generalização e baixa complexidade. Portanto, o PSP mantém-se em aberto devido à sua complexidade, à diversidade, ao desenvolvimento de novos *kernels*. Na Seção 8.1, são listados os trabalhos futuros.

8.1 TRABALHOS FUTUROS

Os algoritmos desenvolvidos para encontrar soluções para o problema de seleção de parâmetros mostraram-se eficientes na obtenção de diferentes tipos de modelos para *benhcmarks* e para aplicações. Entretanto, na elaboração deste trabalho, surgiram algumas propostas que são deixadas como sugestão de trabalhos futuros, divididos em projeto de SVM/SVRs e aplicações.

8.1.1 Projeto de SVM/SVRs

- Os comitês de máquinas são conjuntos de modelos de reconhecimento de padrões com características distintas empregados para um propósito comum. Escolher quais e quantas máquinas são necessárias para esse propósito, visando minimizar a complexidade do comitê de máquinas e maximizar a capacidade de generalização, é um problema complexo. Este pode ser abordado com o uso de SVM/SVRs de uma fronteira de Pareto, em que deve ser definido quais e quantas máquinas são necessárias para esta finalidade, visando aos objetivos citados.

- As SVMs foram desenvolvidas para classificação de dados de classes binárias, para conjuntos de treinamento multiclasses, deve-se empregar técnicas, como o OAO, em que são necessárias $k(k+1)/2$ SVMs, neste caso, pode-se atribuir hiperparâmetros às máquinas individualmente para se obter resultados mais eficientes. Esta estratégia têm um aumento considerável no esforço computacional dos algoritmos, pois manipular as variáveis de decisão, no problema de seleção de parâmetros, consiste em uma fração insignificante de esforço computacional quando comparado com a avaliação da função custo. Assim, propõe-se com trabalhos futuros adaptar os algoritmos desenvolvidos para esta tarefa.

- A tarefa de definir os *kernel* adequado para cada tipo de dados de treinamento, mesmo com o desenvolvimento do NIOTS, fica a cargo do projetista. Esta tarefa possui considerável importância, pois existem inúmeros *kernels* na literatura, cada um adequado ao tipo de problemas com diferentes quantidades de parâmetros a ser definido. Este problema é particularmente complexo, pois primeiro deve-se adotar um *kernel* e depois os hiperparâmetros, sendo um problema que naturalmente deve ser resolvido em duas etapas.

- Youngmin Cho e Lawrence K. propõem uma técnica e condições para que as SVMs possam se comportar como redes profundas, mantendo as características quadrática e convexidade do problema de treinamento destas (CHO; SAUL, 2009). Por outro lado, ainda deixa em aberto o problema de definir qual a profundidade da rede assim como a combinação de *kernels* e seus parâmetros. Dessa maneira, essa tarefa também pode ser abordada por um algoritmos meta-heurísticos apropriados, com a finalidade de encontrar os parâmetros de redes profundas baseadas em SVMs.

- A combinação linear de *kernels* com coeficientes, parcelas e *kernels* adequados produz SVM/SVRs com alta capacidade de generalização. Entretanto ajustar os coeficientes, definir os *kernels* e os seus parâmetros e a quantidade de parcelas da combinação linear consiste em um problema complexo. Assim, desenvolver uma meta-heurística para resolvê-lo poderá produzir modelos para classificadores e regressores que possuam baixa complexidade e alta capacidade de generalização.

- *Free Simulation* dados de validação.

8.1.2 Aplicações de SVM/SVRs

- Elaboração de modelos aproximadores de funções para estimar produção de soja e milho baseado na adubação de características do solo. Essa aplicação está associada à agricultura de precisão.

- Modelagem do processo de soldas, para controle da corrente e tensão.

- Predição de porosidade e resistência no processo de obtenção de materiais porosos, baseado nos parâmetros do experimentos.

- Predição de tempo de reconfiguração parcial de estrutura de *hardwares* reprogramáveis *online*.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRAHAM, A.; JAIN, L. Evolutionary multiobjective optimization. In: ABRAHAM, A.; JAIN, L.; GOLDBERG, R. (Ed.). *Evolutionary Multiobjective Optimization*. [S.l.]: Springer London, 2005, (Advanced Information and Knowledge Processing). p. 1–6. ISBN 978-1-85233-787-2.

AGARWAL, P.; MEHTA, S. Nature-Inspired Algorithms: State-of-Art, Problems and Prospects. *International Journal of Computer Applications*, v. 100, n. 14, p. 14–21, 2014. ISSN 09758887. Disponível em: <http://research.ijcaonline.org/volume100/number14/pxc3898331.pdf>.

AHMAD, A. et al. A review on applications of ann and svm for building electrical energy consumption forecasting. *Renewable and Sustainable Energy Reviews*, Pergamon, v. 33, n. 1, p. 102–109, 2014.

ALBERTO, I.; MATEO, P. A crossover operator that uses pareto optimality in its definition. *Top*, Springer, v. 19, n. 1, p. 67–92, 2011.

ALI, M.; SIARRY, P.; PANT, M. An efficient differential evolution based algorithm for solving multi-objective optimization problems. *European journal of operational research*, Elsevier, v. 217, n. 2, p. 404–416, 2012.

ALMASI, O. N.; KHOOBAN, M. H. A parsimonious svm model selection criterion for classification of real-world data sets via an adaptive population-based algorithm. *Neural Computing and Applications*, v. 30, n. 11, p. 3421–3429, Dec 2018. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-017-2930-y>.

ALPAYDIN, E. *Introduction to machine learning*. [S.l.]: MIT press, 2014.

ALWAN, H. B.; KU-MAHAMUD, K. R. Mixed-variable ant colony optimisation algorithm for feature subset selection and tuning support vector machine parameter. *International journal of bio-inspired computation*, Inderscience Publishers (IEL), v. 9, n. 1, p. 53–63, 2017.

ANAM, K.; AL-JUMAILY, A. Swarm-based extreme learning machine for finger movement recognition. In: IEEE. *Biomedical Engineering (MECBME), 2014 Middle East Conference on*. [S.l.], 2014. p. 273–276.

ANAM, K. et al. Swarm-based Extreme Learning Machine for Finger Movement Recognition. 2014.

AYYAZ, M.; JAVED, I.; MAHMOOD, W. Handwritten character recognition using multiclass svm classification with hybrid feature extraction. *Pakistan Journal of Engineering and Applied Sciences*, v. 0, n. 0, 2016. ISSN 2415-0584. Disponível em: <http://journal.uet.edu.pk/ojs_old/index.php/pjeas/article/view/154>.

BACK, T.; FOGEL, D. B.; MICHALEWICZ, Z. (Ed.). *Handbook of Evolutionary Computation*. 1st. ed. Bristol, UK, UK: IOP Publishing Ltd., 1997. ISBN 0750303921.

BADER, J. M. *Hypervolume-based search for multiobjective optimization: theory and methods*. [S.l.]: Johannes Bader, 2010.

BAMAKAN, S. M. H.; WANG, H.; RAVASAN, A. Z. Parameters Optimization for Nonparallel Support Vector Machine by Particle Swarm Optimization. *Procedia Computer Science*, The Author(s), v. 91, n. Itqm, p. 482–491, 2016. ISSN 18770509. Disponível em: <http://dx.doi.org/10.1016/j.procs.2016.07.125>.

BEJINARIU, S.-I.; LUCA, R.; COSTIN, H. Metaheuristic algorithms based multi-objective optimization for image segmentation. In: IEEE. *2018 International Conference and Exposition on Electrical And Power Engineering (EPE)*. [S.l.], 2018. p. 0438–0443.

BESTARD, G. A. et al. Sensor fusion to estimate the depth and width of the weld bead in real time in gmaw processes. *Sensors*, v. 18, n. 4, p. 962, 2018. ISSN 1424-8220.

BIRATTARI, M. et al. Classification of metaheuristics and design of experiments for the analysis of components. *Teknik Rapor, AIDA-01-05*, Citeseer, 2001.

BLACKWELL, T.; KENNEDY, J. Impact of communication topology in particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, 2018.

BONESSO, D. *Estimação dos Parâmetros do Kernel em um Classificador SVM na Classificação de Imagens Hiperespectrais em uma Abordagem Multiclasse*. 108 p. Tese (Doutorado), 2013.

BRERETON, R. G.; LLOYD, G. R. Support vector machines for classification and regression. *The Analyst*, v. 135, n. 2, p. 230–267, 2010. ISSN 0003-2654.

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

CARBONERO-RUZ, M. et al. A two dimensional accuracy-based measure for classification performance. *Information Sciences*, v. 382-383, p. 60 – 80, 2017. ISSN 0020-0255. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025516319181>.

CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, v. 2, p. 27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.

Chatelain, C. et al. Multi-objective optimization for svm model selection. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. [S.l.: s.n.], 2007. v. 1, p. 427–431. ISSN 1520-5363.

CHATELAIN, C. et al. Multi-objective optimization for SVM model selection. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, v. 1, p. 427–431, 2007. ISSN 15205363.

Chatelain, C. et al. Multi-objective optimization for svm model selection. In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. [S.l.: s.n.], 2007. v. 1, p. 427–431. ISSN 1520-5363.

CHEN, P. et al. An improved SVM classifier based on double chains quantum genetic algorithm and its application in analogue circuit diagnosis. *Neurocomputing*, Elsevier, v. 211, p. 202–211, 2016. ISSN 18728286. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2015.12.131>.

CHENGLIN, Z. et al. Fault diagnosis of sensor by chaos particle swarm optimization algorithm and support vector machine. *Expert Systems with Applications*, Elsevier Ltd, v. 38, n. 8, p. 9908–9912, 2011. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2011.02.043>.

CHERKASSKY, V. Practical selection of SVM parameters and noise estimation for SVM regression. *Neural Networks*, v. 17, p. 113–126, 2004.

CHO, Y.; SAUL, L. K. Kernel methods for deep learning. In: BENGIO, Y. et al. (Ed.). *Advances in Neural Information Processing Systems 22*. Curran Associates, Inc., 2009. p. 342–350. Disponível em: <http://papers.nips.cc/paper/3628-kernel-methods-for-deep-learning.pdf>.

CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, v. 6, n. 1, p. 58–73, Feb 2002. ISSN 1089-778X.

COELLO, C. A. C. et al. A Hybrid Surrogate-Based Approach for Evolutionary Multi-Objective Optimization. In: *2013 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2013. p. 2548–2555. ISBN 9781479904549.

COELLO, G. B. L. C. A. C.; VELDHUIZEN, D. A. V. *Evolutionary Algorithms for Solving Multi-Objective Problems*. USA: Springer Science, 2007. ISBN 978-0-387-33254-3.

CORREIA, M. B. A Study of redundancy and neutrality in evolutionary optimization. *Evolutionary computation*, v. 21, n. 3, p. 413–443, 2013. ISSN 1063-6560.

CORTES, C.; VAPNIK, V. Support vector networks. *Machine Learning*, v. 20, n. 3, p. 273–297, 1995. ISSN 08856125. Disponível em: <http://link.springer.com/10.1007/BF00994018>.

DAS, S.; SUGANTHAN, P. N. Differential evolution: a survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, IEEE, v. 15, n. 1, p. 4–31, 2011.

DAS, S.; SUGANTHAN, P. N. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, v. 15, n. 1, p. 4–31, 2011. ISSN 1089778X.

DAS, S. P.; PADHY, S. A novel hybrid model using teaching–learning-based optimization and a support vector machine for commodity futures index forecasting. *International Journal of Machine Learning and Cybernetics*, v. 9, n. 1, p. 97–111, Jan 2018. ISSN 1868-808X. Disponível em: <https://doi.org/10.1007/s13042-015-0359-0>.

DEB, K.; JAIN, H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 18, n. 4, p. 577–601, 2014.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, v. 1, n. 1, p. 3 – 18, 2011. ISSN 2210-6502. Disponível em: <http://www.sciencedirect.com/science/article/pii/S2210650211000034>.

DHEERU, D.; TANISKIDOU, E. K. *UCI Machine Learning Repository*. 2017. Disponível em: <http://archive.ics.uci.edu/ml>.

DIBIKE, Y. B. et al. Model induction with support vector machines: Introduction and applications. *Journal of Computing in Civil Engineering*, v. 15, n. 3, p. 208–216, 2001.

Dos Santos, G. S. et al. Least squares support vector machines with tuning based on chaotic differential evolution approach applied to the identification of a thermal process. *Expert Systems with Applications*, Elsevier Ltd, v. 39, n. 5, p. 4805–4812, 2012. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2011.09.137>.

FAN, Q.; ZHANG, Y. Self-adaptive differential evolution algorithm with crossover strategies adaptation and its application in parameter estimation. *Chemometrics and Intelligent Laboratory Systems*, Elsevier, v. 151, p. 164–171, 2016.

FARIS, H. et al. A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. *Neural Computing and Applications*, v. 30, n. 8, p. 2355–2369, Oct 2018. ISSN 1433-3058. Disponível em: <https://doi.org/10.1007/s00521-016-2818-2>.

FARIS, H. et al. A multi-verse optimizer approach for feature selection and optimizing svm parameters based on a robust system architecture. *Neural Computing and Applications*, Springer, v. 30, n. 8, p. 2355–2369, 2018.

FORTIN, F.-A.; PARIZEAU, M. Revisiting the nsga-ii crowding-distance computation. In: ACM. *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. [S.l.], 2013. p. 623–630.

FRANçA, F. O. de. *Algoritmos bio-inspirados aplicados à otimização dinâmica*. Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e de Computação - Unicamp, 2005.

FRIEDRICHS, F.; IGEL, C. Evolutionary tuning of multiple {SVM} parameters. *Neurocomputing*, v. 64, p. 107–117, 2005. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231204005223>.

FRIEDRICHS, F.; IGEL, C. Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, v. 64, n. 1-4 SPEC. ISS., p. 107–117, 2005. ISSN 09252312.

GANAPATHIRAJU, A.; HAMAKER, J.; PICONE, J. Applications of support vector machines to speech recognition. *Signal Processing, IEEE Transactions on*, v. 52, n. 8, p. 2348–2355, Aug 2004. ISSN 1053-587X.

GANAPATHIRAJU, A. et al. Applications of Support Vector Machines to Speech Recognition. v. 52, n. 8, p. 2348–2355, 2004.

GANG, R.; ZHUPING, Z. Traffic safety forecasting method by particle swarm optimization and support vector machine. *Expert Systems with Applications*, Elsevier Ltd, v. 38, n. 8, p. 10420–10424, 2011. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2011.02.066>.

García Nieto, P. J. et al. A hybrid wavelet kernel SVM-based method using artificial bee colony algorithm for predicting the cyanotoxin content from experimental cyanobacteria concentrations in the Trasona reservoir (Northern Spain). *Journal of Computational and Applied Mathematics*, Elsevier B.V., v. 309, p. 587–602, 2015. ISSN 03770427. Disponível em: <http://dx.doi.org/10.1016/j.cam.2016.01.045>.

García Nieto, P. J. et al. A hybrid PSO optimized SVM-based model for predicting a successful growth cycle of the Spirulina platensis from raceway experiments data. *Journal of Computational and Applied Mathematics*, Elsevier B.V., v. 291, p. 293–303, 2016. ISSN 03770427. Disponível em: <http://dx.doi.org/10.1016/j.cam.2015.01.009>.

GARCÍA, S. et al. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *Journal of Heuristics*, v. 15, n. 6, p. 617, May 2008. ISSN 1572-9397. Disponível em: <https://doi.org/10.1007/s10732-008-9080-4>.

GASPAR, P.; CARBONELL, J.; OLIVEIRA, J. L. On the parameter optimization of Support Vector Machines for binary classification. *Journal of integrative bioinformatics*, v. 9, n. 3, p. 201, 2012. ISSN 1613-4516. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/22829572>.

GHORBANI, M.; ZARGAR, G.; JAZAYERI-RAD, H. Prediction of asphaltene precipitation using support vector regression tuned with genetic algorithms. *Petroleum*, Elsevier Ltd, v. 2, n. 3, p. 1–6, 2016. ISSN 24056561. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S2405656116300876>.

GOMES, T. A. F. et al. Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing*, v. 75, p. 3–13, 2012. ISSN 09252312.

GONG, M. et al. *Bio-inspired Computing – Theories and Applications: 11th International Conference, BIC-TA 2016, Xi'an, China, October 28-30, 2016, Revised Selected Papers*. Springer Singapore, 2017. (Communications in Computer and Information Science, pt. 2). ISBN 9789811036149. Disponível em: <https://books.google.com.br/books?id=IXfbDQAAQBAJ>.

GRIVA, I.; NASH, S.; SOFER, A. *Linear and Nonlinear Optimization: Second Edition*. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009. ISBN 9780898717730. Disponível em: <https://books.google.com.br/books?id=uOJ-Vg1BnKgC>.

GRIVA, I.; NASH, S. G.; SOFER, A. *Linear and Nonlinear Optimization (2. ed.)*. [S.l.]: SIAM, 2008. I-XXII, 1-742 p. ISBN 978-0-89871-661-0.

GUI, G. et al. Data-driven support vector machine with optimization techniques for structural health monitoring and damage detection. *KSCE Journal of Civil Engineering*, v. 21, n. 2, p. 523–534, Feb 2017. ISSN 1976-3808. Disponível em: <https://doi.org/10.1007/s12205-017-1518-5>.

GUO, X. C. et al. A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. *Neurocomputing*, v. 71, n. 16-18, p. 3211–3215, 2008. ISSN 09252312.

GUO, Y. M. et al. Fast prediction with sparse multikernel ls-svr using multiple relevant time series and its application in avionics system. *Mathematical Problems in Engineering*, Hindawi, v. 2015, 2015.

GUYON, I. et al. Gene Selection for Cancer Classification using Support Vector Machines. *Mach. Learn.*, v. 46, n. 1-3, p. 389–422, 2002. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1012487302797>.

HILL, B. K. e D. R. *Introdução à Álgebra Linear com Aplicações*. Oitava edicação. Rio de Janeiro: LTC, 2006. 607 p.

HOFMANN, T.; SCHöLKOPF, B.; SMOLA, A. J. Kernel methods in machine learning. *Annals of Statistics*, v. 36, n. 3, p. 1171–1220, 2008. ISSN 00905364.

HOLLAND, J. H. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press*, 1975.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262082136.

HSU, C.; LIN, C. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, v. 13, n. 2, p. 415–425, 2002. ISSN 1045-9227. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=991427>.

HUANG, C. L.; DUN, J. F. A distributed PSO-SVM hybrid system with feature selection and parameter optimization. *Applied Soft Computing Journal*, v. 8, n. 4, p. 1381–1391, 2008. ISSN 15684946.

HUANG, C. L.; WANG, C. J. A GA-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with Applications*, v. 31, n. 2, p. 231–240, 2006. ISSN 09574174.

Huang, V. L. et al. Multi-objective optimization using self-adaptive differential evolution algorithm. In: *2009 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2009. p. 190–194. ISSN 1089-778X.

IGEL, C. Multi-objective model selection for support vector machines. *Lecture Notes in Computer Science*, v. 3410, p. 534–546, 2005.

INTEL. *Quartus II Web Edition*. 2012. Disponível em: <https://www.intel.com/content/www/us/en/programmable/downloads/software/quartus-ii-we/121.html>.

Islam, S. M. et al. An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 42, n. 2, p. 482–500, April 2012. ISSN 1083-4419.

JACK, L.; NANDI a.K. Fault Detection Using Support Vector Machines and Artificial Neural Networks, Augmented By Genetic Algorithms. *Mechanical Systems and Signal Processing*, v. 16, n. 2-3, p. 373–390, 2002. ISSN 08883270. Disponível em: <http://dx.doi.org/10.1006/mssp.2001.1454>.

JAFARI, A. et al. An eeg artifact identification embedded system using ica and multi-instance learning. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. [S.l.: s.n.], 2017. p. 1–4. ISSN 2379-447X.

JI, Y. et al. An EnKF-based scheme to optimize hyper-parameters and features for SVM classifier. *Pattern Recognition*, Elsevier, v. 62, p. 202–213, 2017. ISSN 00313203. Disponível em: <http://dx.doi.org/10.1016/j.patcog.2016.08.014>.

JI, Y. et al. An enkf-based scheme to optimize hyper-parameters and features for svm classifier. *Pattern Recognition*, v. 62, p. 202 – 213, 2017. ISSN 0031-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0031320316302278>.

JIANG, S. et al. Consistencies and contradictions of performance metrics in multiobjective optimization. *IEEE Transactions on Cybernetics*, v. 44, n. 12, p. 2391–2404, 2014. ISSN 21682267.

JR, I. F. et al. A Brief Review of Nature-Inspired Algorithms for Optimization. *ELEKTROTEHNISKI VESTNIK*, v. 80, n. 3, p. 1–7, 2013. ISSN 00135852.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Neural Networks, 1995. Proceedings., IEEE International Conference on*. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4.

KIRKPATRICK, S. et al. Optimization by simulated annealing. *science*, Washington, v. 220, n. 4598, p. 671–680, 1983.

KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. *Informatica*, v. 31, p. 249–268, 2007. ISSN 09226389.

LI, J. et al. Optimal Dissolved Gas Ratios Selected by Genetic Algorithm for Power Transformer Fault Diagnosis Based on Support Vector Machine. *IEEE Transactions on Dielectrics and Electrical Insulation*, v. 23, n. 2, p. 1198–1206, 2016.

LI, K. et al. Stable matching-based selection in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, IEEE, v. 18, n. 6, p. 909–923, 2014.

LI, Q. et al. Parallel multitask cross validation for support vector machine using gpu. *Journal of Parallel and Distributed Computing*, Elsevier, v. 73, n. 3, p. 293–302, 2013.

LI, W.; LIU, L.; GONG, W. Multi-objective uniform design as a SVM model selection tool for face recognition. *Expert Systems With Applications*, Elsevier Ltd, v. 38, n. 6, p. 6689–6695, 2011. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2010.11.066>.

LIMA, C. A. de M. *Comitê de máquinas: uma abordagem unificada empregando máquinas de vetores-suporte*. Tese (Tese), 2004.

LIN, Q. et al. A novel multi-objective particle swarm optimization with multiple search strategies. *European Journal of Operational Research*, Elsevier, v. 247, n. 3, p. 732–744, 2015.

LIN, S.-w. et al. Parameter determination of support vector machine and feature selection using simulated annealing approach. *Applied Soft Computing Journal*, v. 8, p. 1505–1512, 2008.

LIN, S.-W. et al. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert Systems with Applications*, v. 35, n. 4, p. 1817–1824, 2008. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417407003752>.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, v. 14, n. 2, p. 43–67, 2007.

LORENA, A. C.; CARVALHO, A. C. P. L. F. D. Introdução à Máquinas de Vetores de Suporte. *Relatórios técnicos do ICMC*, p. 66, 2003.

LORENA, A. C.; De Carvalho, A. C. P. L. F. Evolutionary tuning of SVM parameter values in multiclass problems. *Neurocomputing*, v. 71, n. 16-18, p. 3326–3334, 2008. ISSN 09252312.

LU, Y. et al. A new hybrid algorithm for bankruptcy prediction using switching particle swarm optimization and support vector machines. *Discrete Dynamics in Nature and Society*, Hindawi Publishing Corporation, v. 501, p. 294930, 2015.

LUTS, J. et al. A tutorial on support vector machine-based methods for classification problems in chemometrics. *Analytica Chimica Acta*, Elsevier B.V., v. 665, n. 2, p. 129–145, 2010. ISSN 00032670. Disponível em: <http://dx.doi.org/10.1016/j.aca.2010.03.030>.

MAHMOUDI, N.; OROUJI, H.; FALLAH-MEHDIPOUR, E. Integration of Shuffled Frog Leaping Algorithm and Support Vector Regression for Prediction of Water Quality Parameters. *Water Resources Management*, Water Resources Management, v. 30, n. 7, p. 2195–2211, 2016. ISSN 15731650. Disponível em: <http://dx.doi.org/10.1007/s11269-016-1280-3>.

MASTINU, E. et al. Embedded system for prosthetic control using implanted neuromuscular interfaces accessed via an osseointegrated implant. *IEEE transactions on biomedical circuits and systems*, IEEE, v. 11, n. 4, p. 867–877, 2017.

MATEO, P. M.; ALBERTO, I. A mutation operator based on a pareto ranking for multi-objective evolutionary algorithms. *Journal of Heuristics*, Springer, v. 18, n. 1, p. 53–89, 2012.

MEZA, J. et al. MOVPSO: Vortex Multi-Objective Particle Swarm Optimization. *Applied Soft Computing*, Elsevier B.V., v. 52, p. 1042–1057, 2016. ISSN 15684946. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1568494616304859>.

MEZA, J. et al. Movpso: vortex multi-objective particle swarm optimization. *Applied Soft Computing*, Elsevier, v. 52, p. 1042–1057, 2017.

MIN, H. Simultaneous Feature with Support Vector Selection and Parameters Optimization Using GA-Based SVM Solve the Binary Classification. p. 426–433, 2016.

MINH, H. Q.; NIYOGI, P.; YAO, Y. Mercer's theorem, feature maps, and smoothing. In: *Learning theory*. [S.l.]: Springer, 2006. p. 154–168.

MIRANDA, P. B. et al. Multi-objective optimization and meta-learning for svm parameter selection. In: IEEE. *Neural Networks (IJCNN), The 2012 International Joint Conference on*. [S.l.], 2012. p. 1–8.

MIRANDA, P. B. C.; PRUDÊNCIO, R. B. C. Active Testing for SVM Parameter Selection. In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. Dallas, TX: [s.n.], 2013. p. 1–8.

MIRANDA, P. B. C. et al. A hybrid meta-learning architecture for multi-objective optimization of SVM parameters. *Neurocomputing*, Elsevier, v. 143, p. 27–43, 2014. ISSN 18728286. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2014.06.026>.

MIRANDA, P. B. C. et al. Multi-objective optimization and Meta-learning for SVM parameter selection. *Proceedings of the International Joint Conference on Neural Networks*, p. 10–15, 2012. ISSN 2161-4393.

MIRJALILI, S. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, Springer, v. 27, n. 4, p. 1053–1073, 2016.

MIRJALILI, S. et al. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, Elsevier, v. 47, p. 106–119, 2016.

MOGHADDAM, V. H.; HAMIDZADEH, J. New hermite orthogonal polynomial kernel and combined kernels in support vector machine classifier. *Pattern Recognition*, v. 60, p. 921 – 935, 2016. ISSN 0031-3203.

NAGARAJ, K.; SRIDHAR, A. A predictive system for detection of bankruptcy using machine learning techniques. *arXiv preprint arXiv:1502.03601*, 2015.

NARZISI, G. An experimental multi-objective study of the svm model selection problem. *Courant Inst. Math. Sci*, 2007.

NARZISI, G. An Experimental Multi-Objective Study of the SVM Model Selection problem. n. C, p. 1–11, 2008.

NIU, D.; DAI, S. A short-term load forecasting model with a modified particle swarm optimization algorithm and least squares support vector machine based on the denoising method of empirical mode decomposition and grey relational analysis. *Energies*, v. 10, n. 3, 2017. ISSN 1996-1073. Disponível em: <http://www.mdpi.com/1996-1073/10/3/408>.

NIU, X. et al. Investigation of ANN and SVM based on limited samples for performance and emissions prediction of a CRDI-assisted marine diesel engine. *Applied Thermal Engineering*, Elsevier Ltd, v. 111, p. 1353–1364, 2016. ISSN 13594311. Disponível em: <http://dx.doi.org/10.1016/j.applthermaleng.2016.10.042>.

NOZ, D. M. et al. Tradeoff of FPGA design of floating-point transcendental functions. In: *17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*. [S.l.: s.n.], 2009. p. 239–242.

NOZ, D. M. et al. FPGA based floating-point library for cordic algorithms. In: *Programmable Logic Conference (SPL), VI Southern*. Ipojuca, Brazil: [s.n.], 2010. p. 55–60.

OKABE, T.; JIN, Y.; SENDHOFF, B. A critical survey of performance indices for multi-objective optimisation. In: IEEE. *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.* [S.l.], 2003. v. 2, p. 878–885.

PHAN, A. V.; NGUYEN, M. L.; BUI, L. T. Feature weighting and SVM parameters optimization based on genetic algorithms for classification problems. *Applied Intelligence*, Applied Intelligence, 2016. ISSN 0924-669X. Disponível em: <http://link.springer.com/10.1007/s10489-016-0843-6>.

POLI, R. Analysis of the publications on the applications of particle swarm optimisation. 2008.

RAHNAMAYAN, S.; TIZHOOSH, H. R.; SALAMA, M. M. Opposition versus randomness in soft computing techniques. *Applied Soft Computing*, Elsevier, v. 8, n. 2, p. 906–918, 2008.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Gsa: A gravitational search algorithm. *Information Sciences*, v. 179, n. 13, p. 2232 – 2248, 2009. ISSN 0020-0255. Special Section on High Order Fuzzy Sets. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020025509001200>.

RASTGOUFARD, S.; DIMITRIOS, C. Parameter Selection of Multi-Class SVM with Evolutionary Optimization Methods for Static Security Evaluation in Power Systems. In: *2016 IEEE Electrical Power and Energy Conference (EPEC)*. Ottawa, ON: [s.n.], 2016. p. 1–6. ISBN 9781509019199.

REN, Y.; ZHANG, L.; SUGANTHAN, P. N. Ensemble classification and regression-recent developments, applications and future directions. *IEEE Comp. Int. Mag.*, v. 11, n. 1, p. 41–53, 2016.

REN, Y.; ZHANG, L.; SUGANTHAN, P. N. Review Article Ensemble Classification and Regression—Recent Developments, Applications and Future Directions. *IEEE Computational Intelligence Magazine*, n. February, p. 41–53, 2016.

RICHARDS, M.; VENTURA, D. A. Dynamic sociometry in particle swarm optimization. Atlantis Press, 2003.

RIGET, J.; VESTERSTRØM, J. S. A diversity-guided particle swarm optimizer-the arpso. *Dept. Comput. Sci., Univ. of Aarhus, Aarhus, Denmark, Tech. Rep*, Citeseer, v. 2, p. 2002, 2002.

RIQUELME, N.; LÜCKEN, C. V.; BARAN, B. Performance metrics in multi-objective optimization. In: IEEE. *2015 Latin American Computing Conference (CLEI)*. [S.l.], 2015. p. 1–11.

Santos, C. E. et al. A svm optimization tool and fpga system architecture applied to nmpc. In: *2017 30th Symposium on Integrated Circuits and Systems Design (SBCCI)*. [S.l.: s.n.], 2017. p. 96–102.

SHAMSHIRBAND, S. et al. Support vector machine-based exergetic modelling of a di diesel engine running on biodiesel-diesel blends containing expanded polystyrene. *Applied Thermal Engineering*, v. 94, p. 727–747, 2016. ISSN 13594311.

SHAOWU, Z. et al. Parameters selection of SVM for function approximation based on Differential Evolution. *Proceedings on Intelligent Systems and Knowledge Engineering (ISKE2007)*, 2007. Disponível em: <http://www.atlantis-press.com/php/paper-details.php?id=1270>.

SHAWE-TAYLOR, J.; SUN, S. A review of optimization methodologies in support vector machines. *Neurocomputing*, Elsevier, v. 74, n. 17, p. 3609–3618, 2011. ISSN 09252312. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2011.06.026>.

SHERIN, B. M. GA Based Selection and Parameter Optimization for an SVM Based Underwater Target Classifier. In: *2015 International Symposium on Ocean Electronics (SYMPOL)*. [S.l.: s.n.], 2015. p. 1–7.

SHIH-YUAN, C. et al. Cross-searching strategy for multi-objective particle swarm optimization. *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, p. 3135–3141, 2007.

SHIN, K.-S.; LEE, T. S.; KIM, H.-j. An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, v. 28, n. 1, p. 127–135, 2005. ISSN 09574174.

SIPSER, M. *Introduction to the Theory of Computation*. [S.l.]: Thomson Course Technology Boston, 2006. v. 2.

SMOLA, A.; VAPNIK, V. Support vector regression machines. *Advances in neural information processing systems*, v. 9, p. 155–161, 1997.

SMOLA, A. J. et al. Advances in Large Margin Classi ers. *Advances*, 1999.

SMOLA, A. J. et al. *Regression estimation with support vector learning machines*. Tese (Doutorado) — Master's thesis, Technische Universität München, 1996.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and computing*, Springer, v. 14, n. 3, p. 199–222, 2004.

SOARES, C.; BRAZDIL, P. B.; KUBA, P. A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, v. 54, n. 3, p. 195–209, 2004. ISSN 08856125.

STORN, R.; PRICE, K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, v. 11, n. 4, p. 341–359, 1997. ISSN 1573-2916. Disponível em: <http://dx.doi.org/10.1023/A:1008202821328>.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, Springer, v. 11, n. 4, p. 341–359, 1997.

SUBASI, A.; KEVRIC, J.; CANBAZ, M. A. Epileptic seizure detection using hybrid machine learning methods. *Neural Computing and Applications*, Springer, p. 1–9, 2017.

SUKAWATTANAVIJIT, C.; CHEN, J.; ZHANG, H. GA-SVM Algorithm for Improving Land-Cover Classification Using SAR and Optical Remote Sensing Data. *IEEE Geoscience and Remote Sensing Letters*, v. 14, n. 3, p. 284–288, 2017.

SUN, J.; LAI, C.; WU, X. *Particle Swarm Optimisation: Classical and Quantum Perspectives*. CRC Press, 2016. (Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series). ISBN 9781439835777. Disponível em: <https://books.google.dk/books?id=P2HRBQAAQBAJ>.

SUTTORP, T.; IGEL, C. Multi-objective optimization of support vector machines. *Multi-objective machine learning*, v. 16, p. 199–220, 2006.

TANG, X. et al. Multi-fault classification based on support vector machine trained by chaos particle swarm optimization. *Knowledge-Based Systems*, Elsevier B.V., v. 23, n. 5, p. 486–490, 2010. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2010.01.004>.

THARWAT, A.; MOEMEN, Y. S.; HASSANIEN, A. E. Classification of toxicity effects of biotransformed hepatic drugs using whale optimized support vector machines. *Journal of Biomedical Informatics*, Elsevier Inc., v. 68, p. 132–149, 2017. ISSN 15320464. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1532046417300515>.

THASEEN, I. S.; KUMAR, C. A. Intrusion detection model using fusion of chi-square feature selection and multi class svm. *Journal of King Saud University - Computer and Information Sciences*, v. 29, n. 4, p. 462 – 472, 2017. ISSN 1319-1578. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1319157816300076>.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern Recognition (Fourth Edition)*. Fourth edition. Boston: Academic Press, 2009. 151 - 260 p. ISBN 978-1-59749-272-0. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9781597492720500062>.

THEODORIDIS, S. et al. *Introduction to Pattern Recognition: A Matlab Approach: A Matlab Approach*. [S.l.]: Academic Press, 2010.

TIZHOOSH, H. R. Opposition-based learning: A new scheme for machine intelligence. In: *CIMCA/IAWTIC*. [S.l.: s.n.], 2005. p. 695–701.

ULUNGU, E. L.; TEGHEM, J. Multi-objective combinatorial optimization problems: A survey. *Journal of Multi???Criteria Decision Analysis*, v. 3, n. 2, p. 83–104, 1994. ISSN 10991360.

Von L u cken, C.; BARáN, B.; BRIZUELA, C. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, v. 58, n. 3, p. 707–756, 2014. ISSN 15732894.

WAGNER, T.; TRAUTMANN, H. Online convergence detection for evolutionary multi-objective algorithms revisited. In: IEEE. *IEEE Congress on Evolutionary Computation*. [S.l.], 2010. p. 1–8.

WANG, Y.; YANG, Y. Particle swarm with equilibrium strategy of selection for multi-objective optimization. *European Journal of Operational Research*, Elsevier B.V., v. 200, n. 1, p. 187–197, 2010. ISSN 03772217. Disponível em: <http://dx.doi.org/10.1016/j.ejor.2008.12.026>.

WU, Q. Hybrid forecasting model based on support vector machine and particle swarm optimization with adaptive and Cauchy mutation. *Expert Systems with Applications*, v. 38, n. 8, p. 9070–9075, 2011. ISSN 09574174.

WU, Q.; LAW, R. Cauchy mutation based on objective variable of Gaussian particle swarm optimization for parameters selection of SVM. *Expert Systems with Applications*, v. 38, n. 6, p. 6405–6411, 2011. ISSN 09574174.

XING, B.; GAO, W.-J. Introduction to computational intelligence. In: ____. *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms*. Cham: Springer International Publishing, 2014. p. 3–17. ISBN 978-3-319-03404-1. Disponível em: <http://dx.doi.org/10.1007/978-3-319-03404-1_1>.

XUE, F.; SANDERSON, A. C.; GRAVES, R. J. Pareto-based multi-objective differential evolution. In: *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*. [S.l.: s.n.], 2003. v. 2, p. 862–869 Vol.2.

YANG, X. *Cuckoo Search and Firefly Algorithm: Theory and Applications*. Springer International Publishing, 2013. (Studies in Computational Intelligence). ISBN 9783319021409. Disponível em: <https://books.google.dk/books?id=pp-rngEACAAJ>.

YAO, P.; XUE, J.; ZHOU, K. Study on the wire feed speed prediction of double-wire-pulsed MIG welding based on support vector machine regression. *The International Journal of Advanced Manufacturing Technology*, p. 2107–2116, 2015. ISSN 0268-3768. Disponível em: <http://link.springer.com/10.1007/s00170-015-7039-9>.

YE, F.; LOU, X. Y.; SUN, L. F. An improved chaotic fruit fly optimization based on a mutation strategy for simultaneous feature selection and parameter optimization for svm and its applications. *PloS one*, Public Library of Science, v. 12, n. 4, p. e0173516, 2017.

YI, W. et al. An intelligent algorithm of Support Vector Regression parameters optimization in soft measurements. *Proceedings of the 19th International Conference on Soft Computing and Measurements, SCM 2016*, n. 1, p. 404–406, 2016.

YOU, D. Y.; GAO, X. D.; KATAYAMA, S. Multisensor Fusion System for Monitoring High-Power Disk Laser Welding Using Support Vector Machine. *Ieee Transactions on Industrial Informatics*, v. 10, n. 2, p. 1285–1295, 2014.

ZANATY, E.; AFIFI, A. Generalized hermite kernel function for support vector machine classifications. *International Journal of Computers and Applications*, Taylor & Francis, p. 1–9, 2018.

ZAPOTECAS-MARTÍNEZ, S.; GARCÍA-NÁJERA, A.; LÓPEZ-JAIMES, A. Multi-objective grey wolf optimizer based on decomposition. *Expert Systems with Applications*, Elsevier, v. 120, p. 357–371, 2019.

ZENG, N. et al. A new switching-delayed-pso-based optimized svm algorithm for diagnosis of alzheimer's disease. *Neurocomputing*, v. 320, p. 195 – 202, 2018. ISSN 0925-2312. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925231218310531>.

ZHANG, W. et al. Short-term wind speed forecasting based on a hybrid model. *Applied Soft Computing Journal*, Elsevier Ltd, v. 13, n. 7, p. 3225–3233, 2017. ISSN 15684946. Disponível em: <http://dx.doi.org/10.1016/j.energy.2016.10.040>.

ZHANG, X.; WANG, J.; ZHANG, K. Short-term electric load forecasting based on singular spectrum analysis and support vector machine optimized by cuckoo search algorithm. *Electric Power Systems Research*, v. 146, p. 270 – 285, 2017. ISSN 0378-7796. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378779617300445>.

ZHANG, X. Z. X.; GUO, Y. G. Y. Optimization of SVM Parameters Based on PSO Algorithm. *2009 Fifth International Conference on Natural Computation*, v. 1, p. 536–539, 2009.

ZHAO, S.-Z. et al. Dynamic multi-swarm particle swarm optimizer with harmony search. *Expert Systems with Applications*, v. 38, n. 4, p. 3735 – 3742, 2011. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417410009899>.

ZHAO, Z. et al. A differential evolution algorithm with self-adaptive strategy and control parameters based on symmetric latin hypercube design for unconstrained optimization problems. *European Journal of Operational Research*, v. 250, n. 1, p. 30 – 45, 2016. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221715009698>.

ZIEN, a. et al. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics (Oxford, England)*, v. 16, n. 9, p. 799–807, 2000. ISSN 1367-4803.

ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms — a comparative case study. In: EIBEN, A. E. et al. (Ed.). *Parallel Problem Solving from Nature — PPSN V*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 292–301. ISBN 978-3-540-49672-4.

APÊNDICES

A OBTENÇÃO DO MODELO DE TREINAMENTOS DAS SVM

O treinamento de uma SVM visa encontrar uma função $f(\mathbf{x})$ entre margens que são definidas a partir de pontos de treinamento específicos de forma que estas separem os dados de treinamento com erro mínimo e distância máxima entre elas. Considerando estes critérios, o treinamento foi modelado como um problema de otimização quadrática. Esta última característica garante que o problema de treinamento seja convexo e, portanto, possua solução única. Neste Apêndice, é apresentada cada etapa do desenvolvimento que explora os conceitos envolvidos na obtenção problema de treinamento.

Os pontos que estão sobre as margens (*vide* Figura 2.2) satisfazem a Equação A.1, que pode ser escrita na forma do Sistema A.2.

$$|\mathbf{w} \cdot \mathbf{x} + b| = 1, \tag{A.1}$$

$$\begin{cases} \mathbf{w} \cdot \mathbf{x} + b \geqslant +1 & \text{se} \quad y_i = +1 \\ \mathbf{w} \cdot \mathbf{x} + b \leqslant -1 & \text{se} \quad y_i = -1. \end{cases} \tag{A.2}$$

Combinando as equações do Sistema A.2, obtemos a Restrição A.3, garantindo que não haja erro empírico na classificação. Os pontos que satisfazem as equações A.4 e A.5 estão sobre as margens H_1 e H_2.

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0, \tag{A.3}$$

$$H_1 : \mathbf{w} \cdot \mathbf{x}_i + b = +1, \tag{A.4}$$

$$H_2 : \mathbf{w} \cdot \mathbf{x}_i + b = -1, \tag{A.5}$$

Considerando que a distância entre a origem do sistema de eixos e as margens é dada por:

$$d(O, H_1) = \frac{|b - 1|}{\|\mathbf{w}\|}, \tag{A.6}$$

$$d(O, H_2) = \frac{|b + 1|}{\|\mathbf{w}\|}. \tag{A.7}$$

Portanto, a distância entre as margens é dada por

$$\begin{aligned} d(H_1, H_2) &= |d(O, H_1) - d(O, H_1)| \\ d(H_1, H_2) &= \left| \frac{|b-1|}{\|\mathbf{w}\|} - \frac{|b+1|}{\|\mathbf{w}\|} \right| \\ d(H_1, H_2) &= \left| \frac{|b-1-b-1|}{\|\mathbf{w}\|} \right| \\ d(H_1, H_2) &= \left| \frac{|-2|}{\|\mathbf{w}\|} \right| \\ d(H_1, H_2) &= \frac{2}{\|\mathbf{w}\|}. \end{aligned} \tag{A.8}$$

As margens máximas são encotradas minimizando $\|\mathbf{w}\|$ na Equação A.8, sujeito à restrição da Equação A.3, gerando, assim, o problema de otimização A.9 nas variáveis $\mathbf{w}$ e b.

$$\begin{aligned} &\text{Min} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 \\ &s.a. \\ &y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geqslant 0 \qquad i = 1, ..., N, \end{aligned} \tag{A.9}$$

em que N é a cardinalidade do conjunto de treinamento.

O Problema 2.6 é simplificado eliminando as restrições originais do problema e acrescentando as variáveis de Lagrange, obtendo o problema de otimização A.10.

$$\begin{aligned} &\text{Min} \quad L_p(\mathbf{w}, b, \alpha_i) = \tfrac{1}{2}\|\mathbf{w}\|^2 - \textstyle\sum_{i=1}^{N} \alpha_i (y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \\ &s.a. \\ &\alpha_i \geqslant 0 \qquad i = 1, ..., N. \end{aligned} \tag{A.10}$$

Os pontos críticos da função objetivo do problema de otimização A.10 são encontrados fazendo $\nabla L_p = 0$:

$$\begin{aligned} \frac{\partial L_p}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0 \\ \mathbf{w} &= \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \end{aligned} \tag{A.11}$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{N} \alpha_i y_i = 0. \tag{A.12}$$

Expandindo A.10,

$$L_p(\mathbf{w}, b, \alpha_i) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{w} \cdot \mathbf{x}_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i \tag{A.13}$$

e, substituindo A.11 e A.12 em A.13, obtém-se:

$$L_d(\alpha_i) = \frac{1}{2}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right) - \sum_{i=1}^{N} \alpha_i y_i \left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right) \cdot \mathbf{x}_i - b \sum_{i=1}^{N} \alpha_i y_i + \sum_{i=1}^{N} \alpha_i. \tag{A.14}$$

Desenvolvendo o produto $\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right)$, tem-se:

$$(\alpha_1 y_1 \mathbf{x}_1 + \alpha_2 y_1 \mathbf{x}_2 + ... + \alpha_n y_n \mathbf{x}_n) \cdot (\alpha_1 y_1 \mathbf{x}_1 + \alpha_2 y_1 \mathbf{x}_2 + ... + \alpha_n y_n \mathbf{x}_n).$$

O produto escalar goza da propriedade distributiva em relação à soma de vetores, portanto:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j). \tag{A.15}$$

Desenvolvendo $\sum_{i=1}^{N} \alpha_i y_i \left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right) \cdot \mathbf{x}_i$ tem-se:

$$\begin{aligned}
&\alpha_1 y_1 \left[(\alpha_1 y_1 \mathbf{x}_1 + \alpha_1 y_1 \mathbf{x}_1 + ... + \alpha_N y_N \mathbf{x}_N) \cdot \mathbf{x}_1\right] \\
&+\alpha_2 y_2 \left[(\alpha_1 y_1 \mathbf{x}_1 + \alpha_1 y_1 \mathbf{x}_1 + ... + \alpha_N y_N \mathbf{x}_N) \cdot \mathbf{x}_2\right] \\
&+...\alpha_N y_N \left[(\alpha_1 y_1 \mathbf{x}_1 + \alpha_1 y_1 \mathbf{x}_1 + ... + \alpha_N y_N \mathbf{x}_n) \cdot \mathbf{x}_N\right].
\end{aligned}$$

A expressão anterior pode ser expandida aplicando a propriedade distributiva em relação à soma de vetores e a propriedade associativa em relação à multiplicação por escalar. Concatenando e usando a notação de somatórios, obtém se:

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j). \tag{A.16}$$

Para obter a Equação A.17, substituem-se as expressões A.15 e A.16 convenientemente em A.14.

$$L_d(\alpha_i) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) - \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + \sum_{i=1}^{N} \alpha_i. \tag{A.17}$$

Obtendo o Problema A.18 conhecido como dual de Wolf, nas variáveis α_i

$$\begin{array}{ll} \text{Max} & L_d(\alpha_i) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \\ \text{s.a} & \\ & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & \alpha_i \geqslant 0 \qquad\qquad i = 1, ..., N. \end{array} \tag{A.18}$$

No problema A.18, a função L_d é quadrática restrita a uma equação linear, ambas convexas, sendo que essa característica é condição necessária e suficiente (segundo o teorema de KKT) para solução ótima única. Essa característica garante que o classificador obtido por este método seja o melhor global, dado um conjunto de treinamento e os parâmetros do modelo (GRIVA; NASH; SOFER, 2009).

Resolvendo o problema A.18, têm-se os valores ótimos de α_i^*, com os quais calcula-se o $\mathbf{w}^*$ a partir da Equação A.19.

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i. \tag{A.19}$$

O b^* é calculado implicitamente isolando-o na equação $\alpha_i^*[y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1] = 0$, como mostrado a seguir:

$$\begin{aligned} \alpha_i^*[y_i(\mathbf{w}^* \cdot \mathbf{x}_i + b^*) - 1] &= 0 \\ \alpha_i^* y_i \mathbf{w}^* \cdot \mathbf{x}_i + \alpha_i^* y_i b^* - y_i &= 0 \\ \alpha_i^* y_i b^* &= -\alpha_i^* y_i \mathbf{w}^* \cdot \mathbf{x}_i + y_i \\ b^* &= -\mathbf{w}^* \cdot \mathbf{x}_i + \frac{1}{y_i}. \end{aligned}$$

Como as margens são equidistantes do hiperplano classificador e o termo b^* determina uma translação em relação a origem, toma-se sua média, assim tem-se:

$$b^* = \frac{1}{\#SV} \sum_{i=1}^{\#SV} \left(\frac{1}{y_i} - \mathbf{w}^* \cdot \mathbf{x}_i \right). \tag{A.20}$$

Utilizando os resultado das equações A.20 e A.19, obtém-se a função classificadora (em função dos α^*), cujos índices pertencem ao conjunto $SV = \{\alpha^* : \alpha^* \neq 0\}$, conforme Equação A.21

$$f(\mathbf{x}) = sgn \left(\sum_{i=1}^{SV} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b \right). \tag{A.21}$$

A função A.21 é capaz de classificar corretamente todos os dados do conjunto de treinamento com a melhor generalização possível, mas, no caso de os problemas não serem linearmente separáveis, pode-se relaxar as restrições do problema de otimização A.9. Este caso é tratado na Seção A.1.

A.1 Margens Suaves

Alguns conjuntos de dados possuem rótulos que não são linearmente separáveis, então nestes casos, as restrições do Problema de Otimização A.9 são relaxadas, permitindo que alguns dados estejam entre as margens, ou mesmo, sejam classificados errados no conjunto de treinamento. Entretanto estes dados devem penalizar a função objetivo por um peso a ser definido a *priori* denominado *parâmetro de regularização*. Este processo aumenta a distância entre as margens e desde que o parâmetro de regularização seja adequadamente definido melhora a capacidade de generalização da SVM.

A.1.1 SVMs de Margens Suaves

Nas situações em que os dados são não linearmente separáveis, mas que podem ser divididos em duas classes, por um hiperplano, adota-se um classificador que aceita tais erros. Estes erros são controlados por um parâmetro denominado *parâmetro de regularização*, sendo esses classificadores conhecidos como *SVMs de margens suaves*.

No problema A.9, adiciona-se a variável de folga ξ_i, tornando o modelo tolerável a erros empíricos, assim como a dados entre as margens, possibilitando deste modo que as SVMs classifiquem dados não linearmente separáveis. Na Figura 2.2, a estrela acima e a direita de F é um exemplo de erro empírico penalizada por ξ_i, enquanto que círculo azul representa um caso de dado classificado corretamente entre as margens penalizado por ξ_j. O problema de otimização primal para SVMs de margens suaves é pela Equação A.22

$$\begin{aligned} &\text{Min} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &s.a. \\ &y_i(\mathbf{w}\cdot\mathbf{x}_i + b) + \xi_i - 1 \geqslant 0 \\ &\xi_i \geqslant 0. \end{aligned} \tag{A.22}$$

No Modelo A.22, o parâmetro C é o parâmetro de regularização, que pondera o erro empírico cometido pela máquina.

O parâmetro de regularização (C) deve ser determinado pelo projetista da máquina *a priori*, sendo que a escolha correta do valor deste parâmetro determina o compromisso que uma SVM possui entre sua complexidade e a capacidade de generalização. Caso o valor de C tenda ao infinito, a SVM obtida se aproxima do modelo de margens rígidas, valorizando dados muito específicos ou, até mesmo, ruídos e *outliers*, comuns em dados reais. Por outro lado, se o valor de C tende a zero, a SVM permitirá tantos erros que não será capaz de classificar os dados, assim possuindo baixa capacidade de generalização.

As SVMs de margens suaves também utilizam o problema dual. Inicialmente, eliminam-se as restrições do do Problema A.22 utilizando a técnica dos multiplicadores de Lagrange e, então,

encontra os pontos críticos de L_p, Problema A.23

$$\begin{array}{l}\textbf{Min} \quad L_p(\mathbf{w}, b, \alpha_i, \mu_i) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\mathbf{w}\cdot\mathbf{x}_i + b) + \xi_i - 1) - \sum_{i=1}^{N}\mu_i\xi_i \\ s.a. \\ \alpha_i, \mu_i \geqslant 0 \qquad i = 1, ..., N.\end{array} \tag{A.23}$$

Para determinar os pontos críticos do problema A.23, calcula-se $\nabla L_p = 0$, obtendo:

$$\begin{aligned}\frac{\partial L_p}{\partial \mathbf{w}} &= \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i = 0 \\ \mathbf{w} &= \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i = 0 \qquad \text{(A.24)}\end{aligned}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{A.25}$$

$$\begin{aligned}\frac{\partial L_p}{\partial \xi_i} &= -\alpha_i + C - \mu_i = 0 \\ \mu_i &= C - \alpha_i \qquad \text{(A.26)} \\ 0 \leqslant \alpha_i &\leqslant C. \qquad \text{(A.27)}\end{aligned}$$

Substituindo a Equação A.26 em L_p, tem-se

$$\begin{aligned}
L_d(\alpha_i) &= \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i \mathbf{w}\cdot\mathbf{x}_i - b\sum_{i=1}^{N}\alpha_i y_i - \sum_{i=1}^{N}\alpha_i\xi_i \\
&- \sum_{i=1}^{N}\mu_i\xi_i + \sum_{i=1}^{N}\alpha_i \\
L_d(\alpha_i) &= \frac{1}{2}\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\right)\cdot\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\right) + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i\left(\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\right)\cdot\mathbf{x}_i \\
&- b\sum_{i=1}^{N}\alpha_i y_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\mu_i\xi_i + \sum_{i=1}^{N}\alpha_i \\
L_d(\alpha_i) &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) - \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) + C\sum_{i=1}^{N}\xi_i \\
&+ \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}(C-\alpha_i)\xi_i \\
L_d(\alpha_i) &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) - \sum_{i=1}^{N}(C-\alpha_i)\xi_i + \sum_{i=1}^{N}(C-\alpha_i)\xi_i + \sum_{i=1}^{N}\alpha_i \\
L_d(\alpha_i) &= \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j). \qquad \text{(A.28)}
\end{aligned}$$

A partir das equações A.28, A.26 e A.25, tem-se o seguinte problema de otimização quadrático, que depende de α_i e ξ_i

$$\begin{array}{ll}
\text{Max} & L_d(\alpha_i,\xi_i) = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) \\
\text{s.a} & \\
\sum_{i=1}^{N}\alpha_i y_i = 0 & \\
\xi_i \geqslant 0 & i = 1,...,N \\
0 \leqslant \alpha_i \leqslant C & i = 1,...,N.
\end{array} \qquad \text{(A.29)}$$

O teorema de KKT enuncia condições necessárias e suficientes para identificar a solução ótima de um problema convexo de otimização convexo, para o Problema A.23, têm-se as seguintes condições:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0, \tag{A.30}$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i = 0, \tag{A.31}$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \tag{A.32}$$

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geqslant 0, \tag{A.33}$$

$$\xi_i \geqslant 0, \tag{A.34}$$

$$\alpha_i \geqslant 0, \tag{A.35}$$

$$\mu_i \geqslant 0, \tag{A.36}$$

$$\alpha_i\{y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\} = 0, \tag{A.37}$$

$$\mu_i \xi_i = 0. \tag{A.38}$$

As equações de A.30 - A.32 referem-se ao cálculo de $\nabla L_p = 0$, ou seja, o ponto ótimo deve ser um ponto crítico; enquanto que as inequações A.33 e A.34 são as restrições do problema, que devem ser atendidas, e α e μ são as variáveis Lagrangeanas. As equações A.37 e A.38 são chamadas de *folga complementar*, indicando quais restrições e variáveis Lagrangeanas estão ativas. Nas equações de folga complementar, se as variáveis Lagrangeanas são nulas e as restrições estão ativas, então, estas restrições são degeneradas (GRIVA; NASH; SOFER, 2009).

A função objetivo assim como as restrições do problema A.29, são contínuas, diferenciáveis e convexas. Neste caso, as condições de Karush-Kuhn-Tucker - KKT são necessárias e suficientes para existência de ponto ótimo único. Estas características do problema, juntamente com as condições de KKT, garantem que a SVM obtida a partir do conjunto de treinamento (e definido um parâmetro de regularização) seja ótima global. Neste contexto, fica em aberto a escolha do parâmetro de regularização C.

Determinados os valores ótimos de α_i^*, as variáveis $\mathbf{w}^*$ e $\mathbf{b}^*$ são determinadas pelas equações 2.8 e 2.9, respectivamente. As variáveis ξ_i não têm influência na equação ótima do hiperplano; entretanto é um indicativo do erro empírico da máquina. Considerando a equação A.37 em que os $\alpha_i \neq 0$, tem-se:

$$\begin{aligned} y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i &= 0 \\ \xi_i &= 1 - y_i(\mathbf{x}_i \cdot \mathbf{w} + b) \\ \xi_i &= \max\{0, 1 - y_i(\mathbf{x}_i \cdot \mathbf{w} + b)\}. \end{aligned} \tag{A.39}$$

A condição na Equação A.39 faz-se necessária, pois, no caso de valores classificados corretamente, a variável ξ_i pode assumir valores negativos.

Uma vez determinados os α^* a função classificadora é dada pela função sinal 2.14:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{\#SV} \alpha_i^* y_i(\mathbf{x}_i \cdot \mathbf{x}) + b^*\right), \tag{A.40}$$

em que $SV = \{\alpha^* : \alpha^* \neq 0\}$.

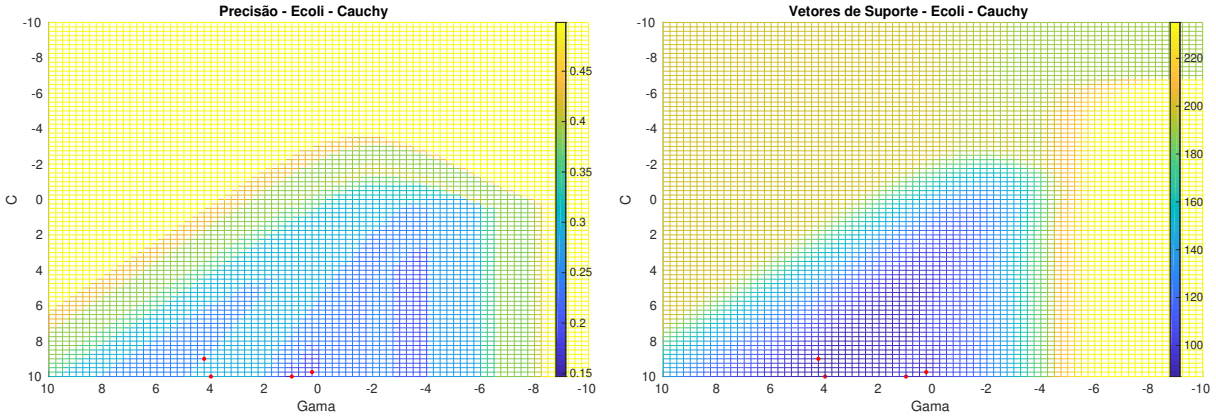
Para classificar dados em que as margens suaves não oferece flexibilidade o suficiente, emprega-se uma técnica denominada *kernel trick* em que os dados são levados ao um espaço de dimensão maior de modo que neste os dados são linearmente separáveis e, portanto, a técnica de margens suaves possa ser empregada sem que o modelo perca sua convexidade. A técnica do *kernel trick* é detalhada na Subseção 2.2.3.

B GRÁFICOS DAS FUNÇÕES OBJETIVO OBTIDOS PELO GS

O problema de seleção de parâmetros das SVMs são conhecidos por serem multimodais, sensíveis à variação dos hiperparâmetros e contraditórios quando considerados os objetivos complexidade e precisão. Adicionalmente, cada combinação de *kernel* e *benchmark* produz uma função objetivo para a precisão e outra para a complexidade, distintas entre cada combinação.

O Grid Search foi utilizado para gerar os gráficos apresentados da Figura B.1 a Figura B.21. O domínio de cada uma é formado por uma malha obtida pela variação do parâmetro de regularização e do parâmetro do *kernel* sendo que, para cada intersecção da malha, é gerado um modelo e, então, avalia-se a sua precisão e complexidade.

Os pontos marcados nos gráficos são as soluções não dominantes encontradas pelo GS, assim, como Capítulo 7, os intervalos de busca para todos os problemas são iguais, com exceção dos problemas com kernel polinomial em que d e definido como $\{d \in \mathcal{Z} : d \in [1, 10]\}$.

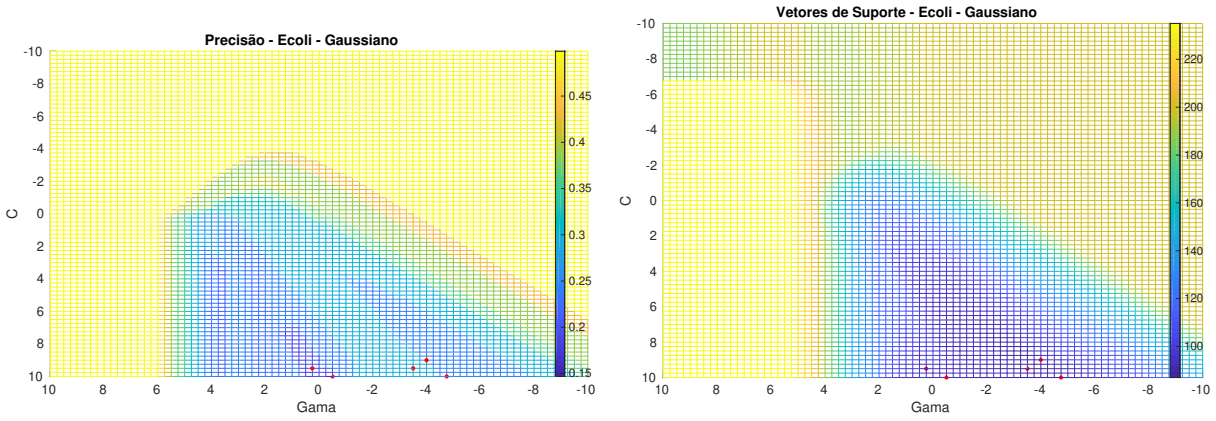


(a) Função precisão.

(b) Função vetores de suporte.

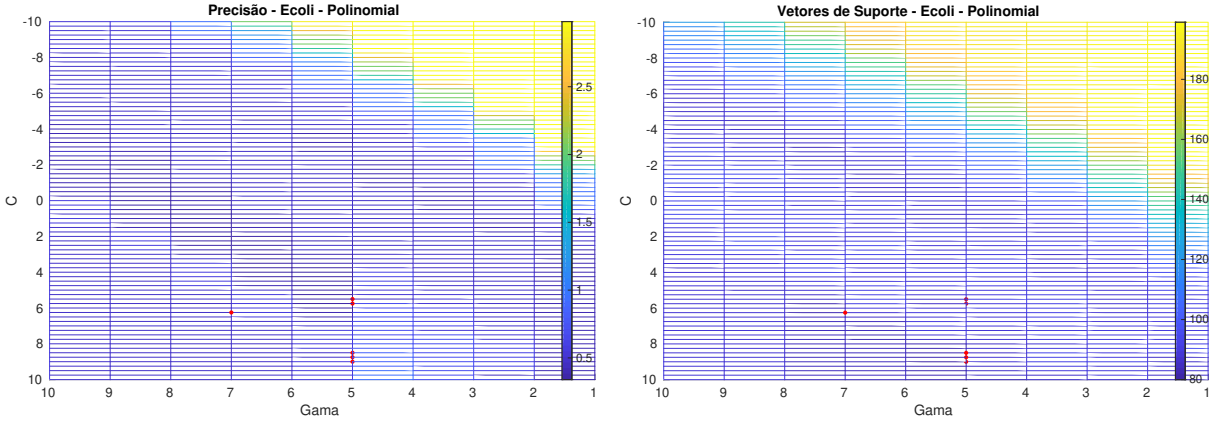
Figura B.1 – Funções objetivo do benchmark Ecoli com kernel Cauchy.

Os gráficos das funções objetivos precisão e complexidade do *benchmark* Ecoli apresentam



(a) Função precisão. (b) Função vetores de suporte.

Figura B.2 – Funções objetivo do benchmark Ecoli com kernel Gaussiano.



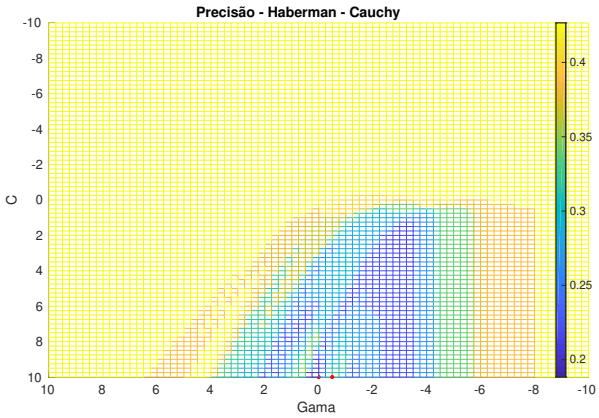
(a) Função precisão. (b) Função vetores de suporte.

Figura B.3 – Funções objetivo do benchmark Ecoli com kernel Polinomial.

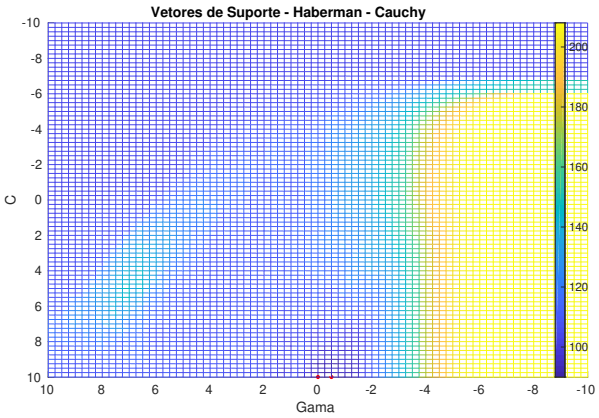
superfícies com regiões de mínimos distintos. Este fato fica evidente pela disposição das soluções não dominantes entre todos os modelos gerados. Na Figura B.1a, há dois pontos na região azul mais escuro (próximo ao gama igual a zero) e três na região azul com tonalidade mais escura (próximo ao gama igual a -4). Este padrão repete-se para as figuras B.2 e B.3.

Em todas as figuras da Seção B, pode ser observado o mesmo padrão como descrito para o *benchmark* Ecoli e os kernels Cauchy, gaussiano e polinomial. As regiões de nível e posições diferentes do espaço de busca em cada caso e as regiões com rugosidade e vales evidenciam a complexidade do problema. Observa-se que, mesmo por um processo de busca exaustiva, a fronteira de Pareto obtida é pobre em cardinalidade e distribuição entre os pontos.

Para os problema com *benchmark* regressores, foi utilizada a mesma metodologia empregada para os *benchmarks* classificadores, com a diferença, que aqueles possuem um parâmetro extra, com três parâmetros a ser configurado que inviabilizam a sua representação gráfica juntamento com a imagem das funções objetivo, mas, para cada valor distinto de épsilon, existe um gráfico com características semelhantes às apresentadas pelos classificadores.

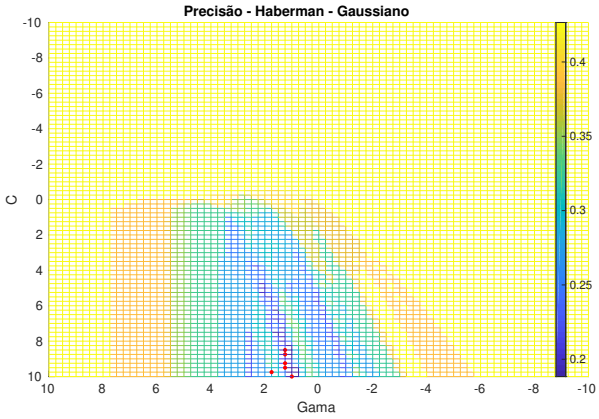


(a) Função precisão.

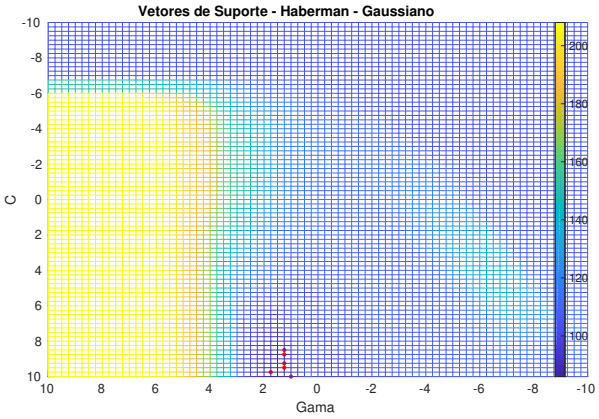


(b) Função vetores de suporte.

Figura B.4 – Funções objetivo do benchmark Haberman com kernel Cauchy.

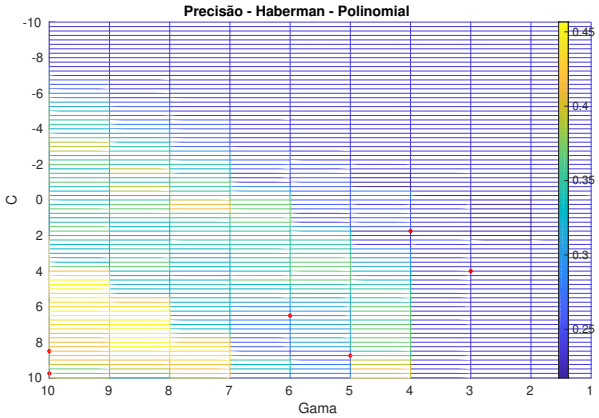


(a) Função precisão.

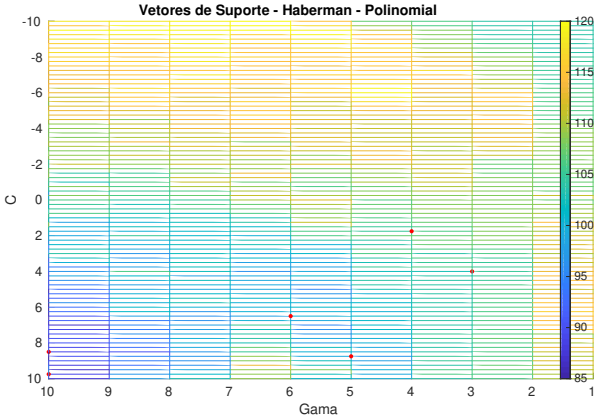


(b) Função vetores de suporte.

Figura B.5 – Funções objetivo do benchmark Haberman com kernel Gaussiano.

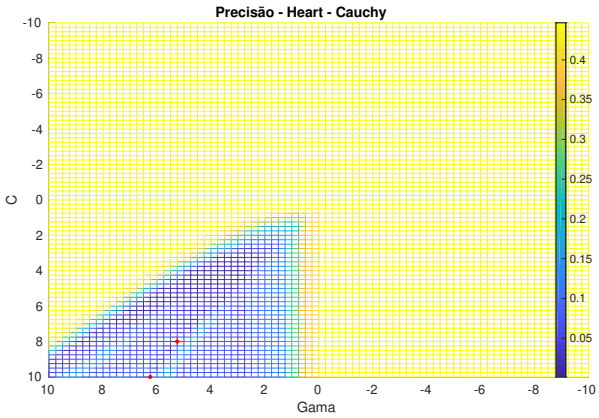


(a) Função precisão.

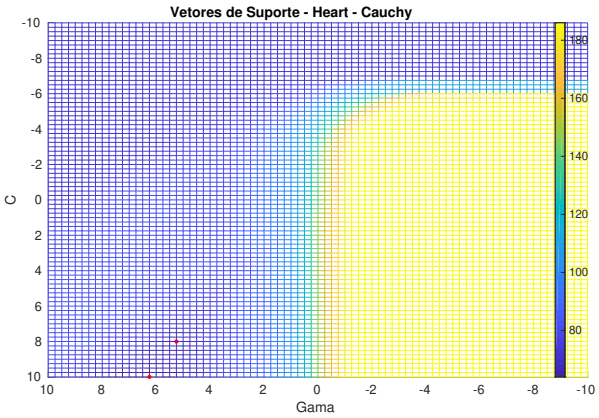


(b) Função vetores de suporte.

Figura B.6 – Funções objetivo do benchmark Haberman com kernel Polinomial.

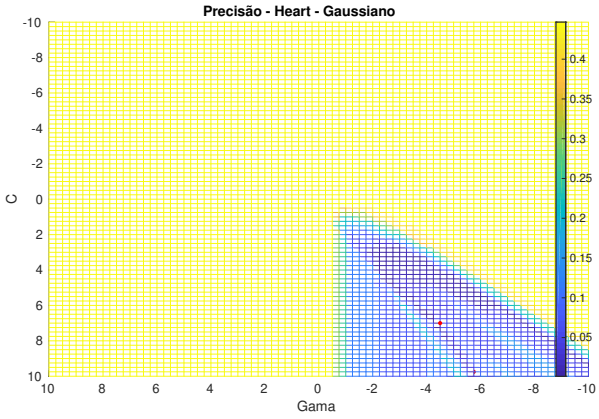


(a) Função precisão.

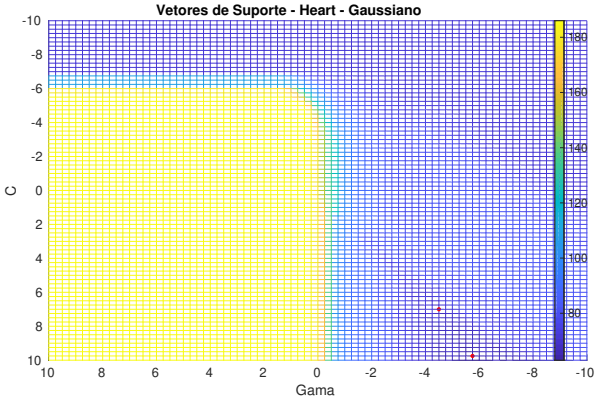


(b) Função vetores de suporte.

Figura B.7 – Funções objetivo do benchmark Heart com kernel Cauchy.

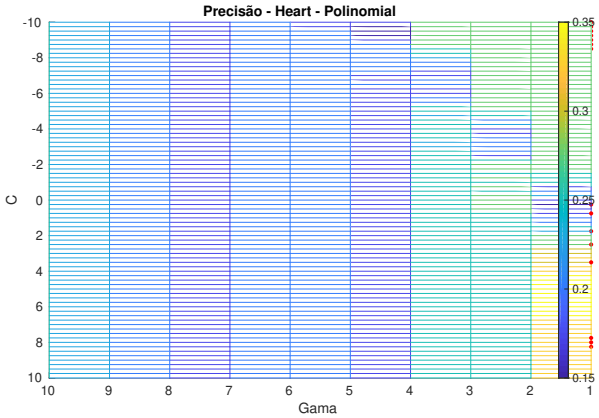


(a) Função precisão.

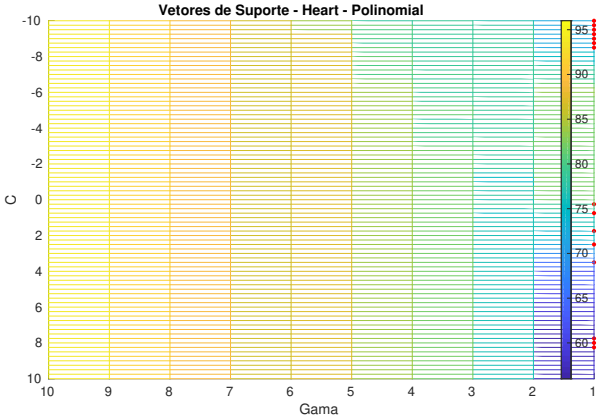


(b) Função vetores de suporte.

Figura B.8 – Funções objetivo do benchmark Heart com kernel Gaussiano.

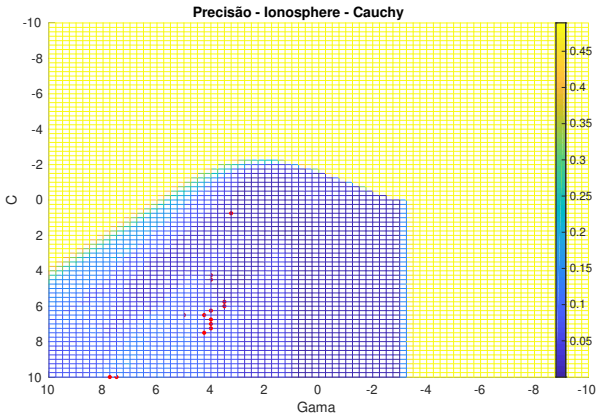


(a) Função precisão.

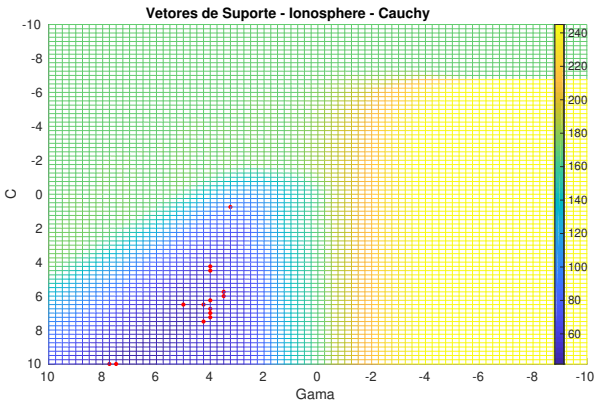


(b) Função vetores de suporte.

Figura B.9 – Funções objetivo do benchmark Heart com kernel Polinomial.

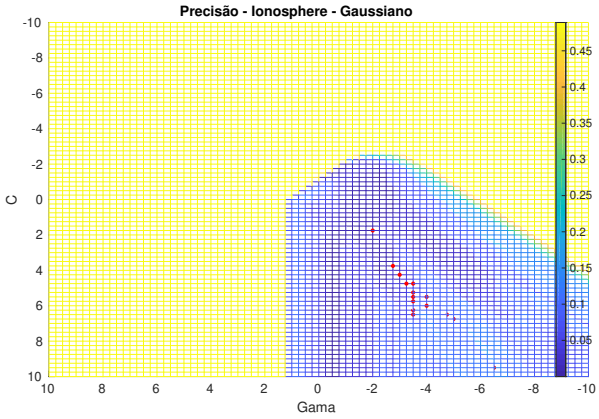


(a) Função precisão.

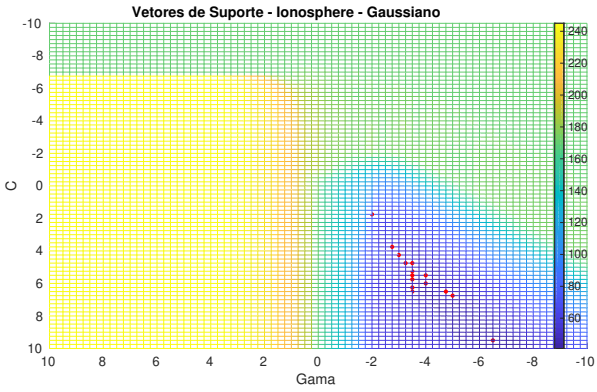


(b) Função vetores de suporte.

Figura B.10 – Funções objetivo do benchmark Ionosphere com kernel Cauchy.

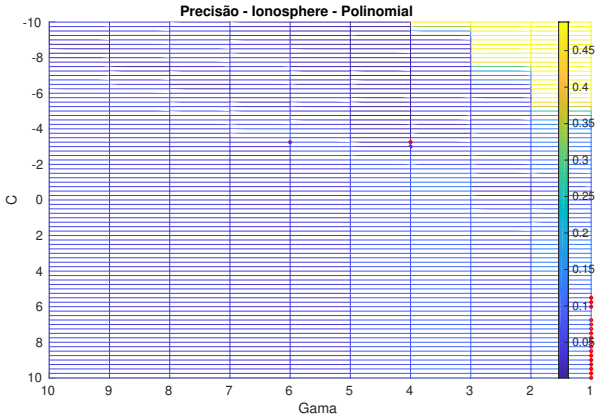


(a) Função precisão.

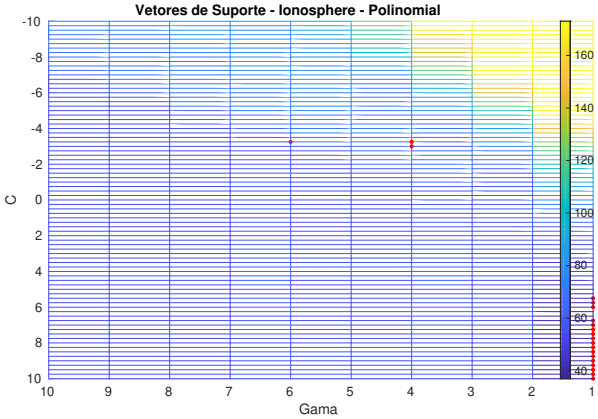


(b) Função vetores de suporte.

Figura B.11 – Funções objetivo do benchmark Ionosphere com kernel Gaussiano.

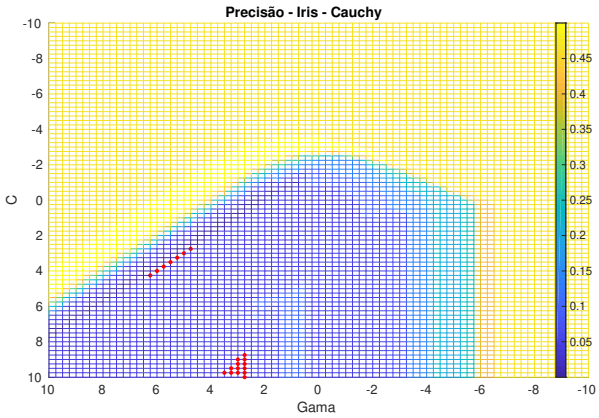


(a) Função precisão.

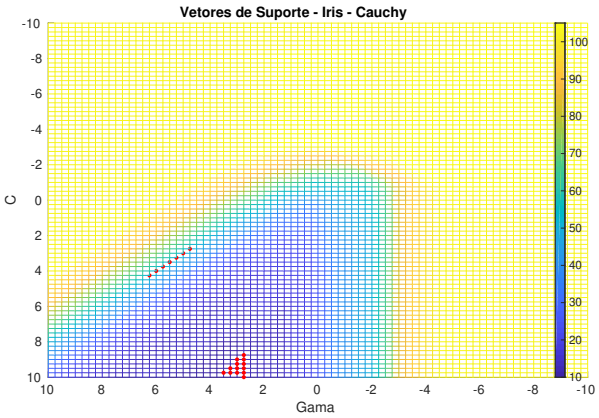


(b) Função vetores de suporte.

Figura B.12 – Funções objetivo do benchmark Ionosphere com kernel Polinomial.

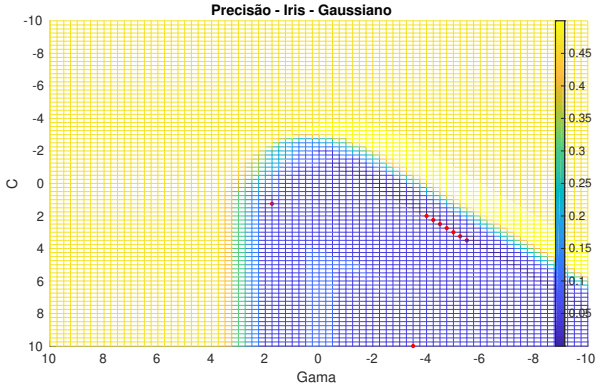


(a) Função precisão.

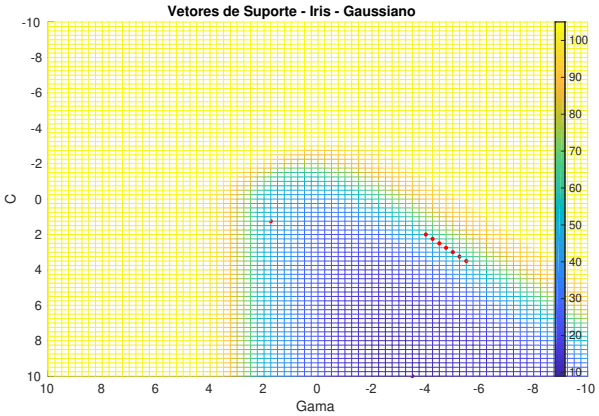


(b) Função vetores de suporte.

Figura B.13 – Funções objetivo do benchmark Iris com kernel Cauchy.

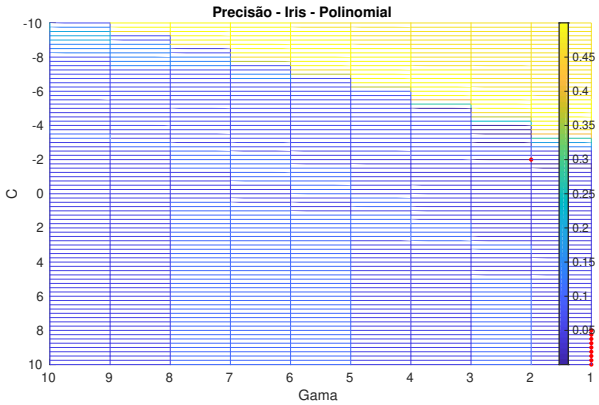


(a) Função precisão.

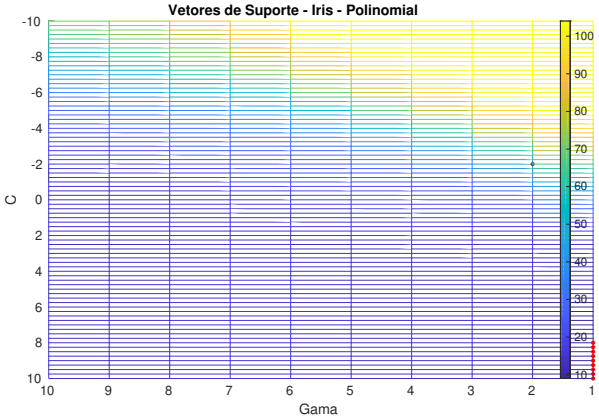


(b) Função vetores de suporte.

Figura B.14 – Funções objetivo do benchmark Iris com kernel Gaussiano.

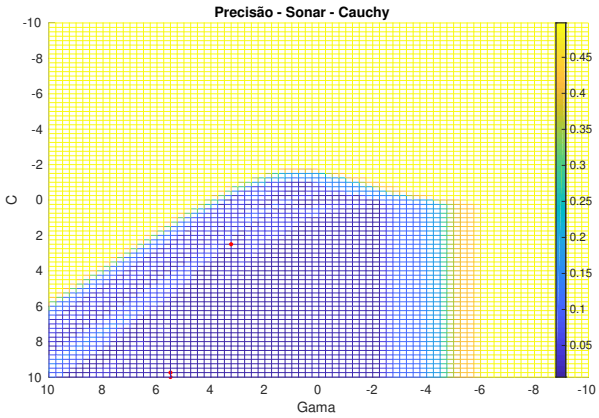


(a) Função precisão.

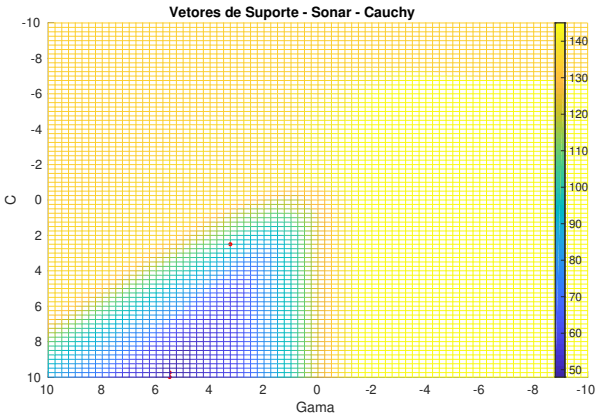


(b) Função vetores de suporte.

Figura B.15 – Funções objetivo do benchmark Iris com kernel Polinomial.

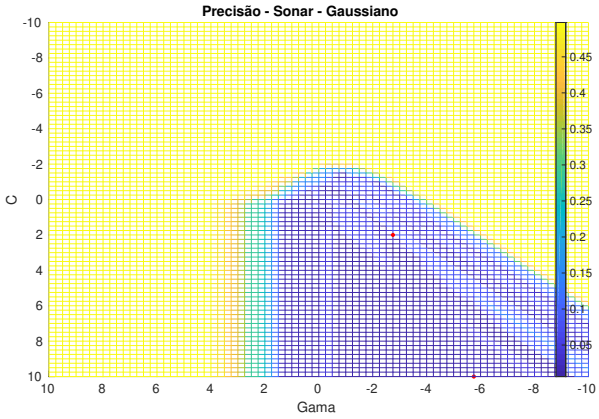


(a) Função precisão.

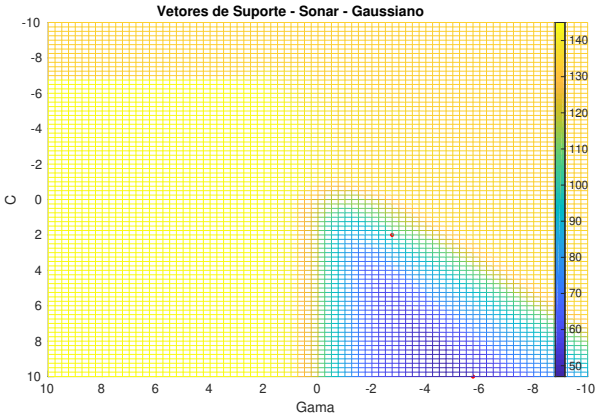


(b) Função vetores de suporte.

Figura B.16 – Funções objetivo do benchmark Sonar com kernel Cauchy.

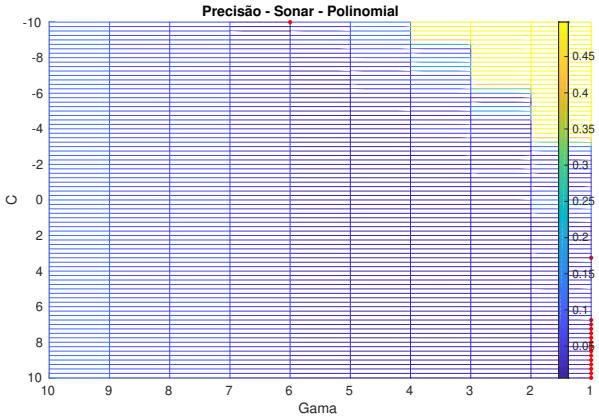


(a) Função precisão.

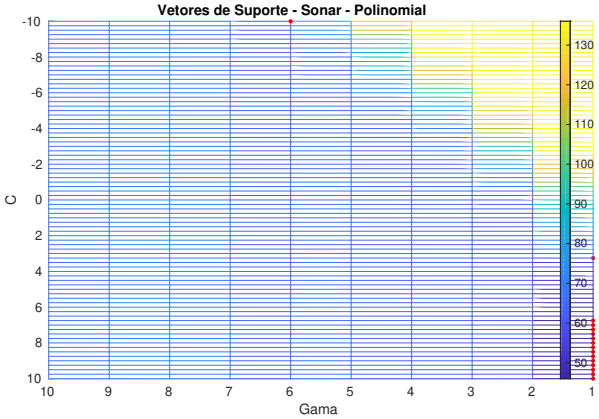


(b) Função vetores de suporte.

Figura B.17 – Funções objetivo do benchmark Sonar com kernel Gaussiano.

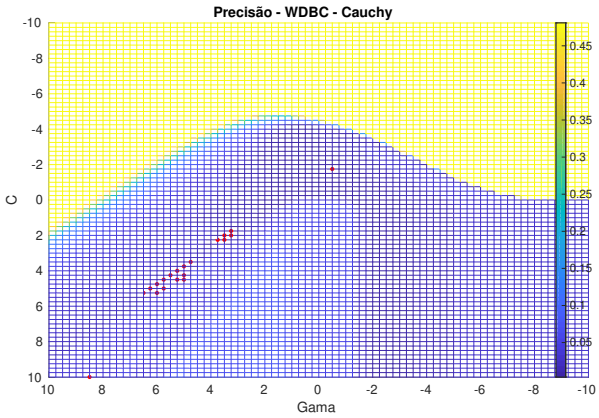


(a) Função precisão.

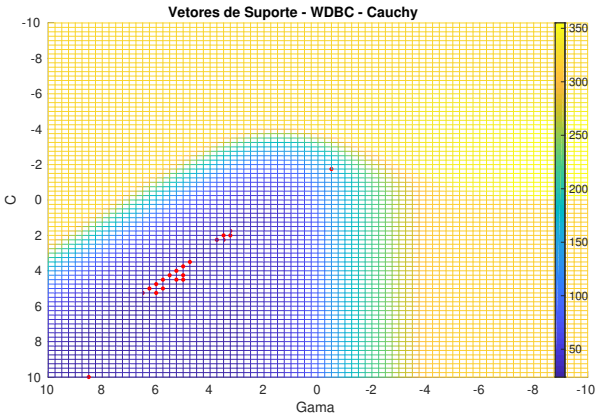


(b) Função vetores de suporte.

Figura B.18 – Funções objetivo do benchmark Sonar com kernel Polinomial.

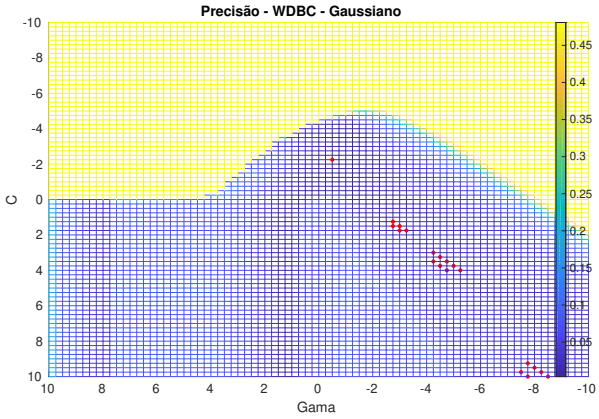


(a) Função precisão.

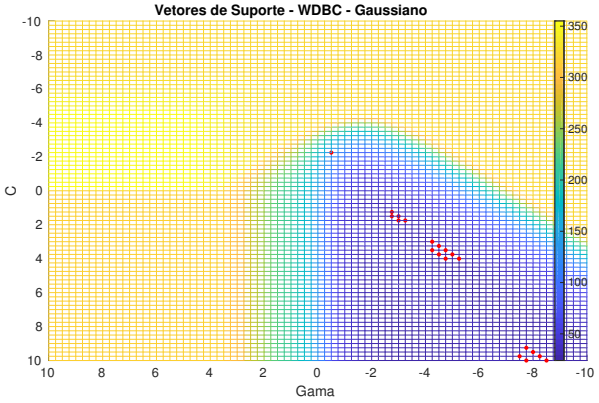


(b) Função vetores de suporte.

Figura B.19 – Funções objetivo do benchmark WDBC com kernel Cauchy.

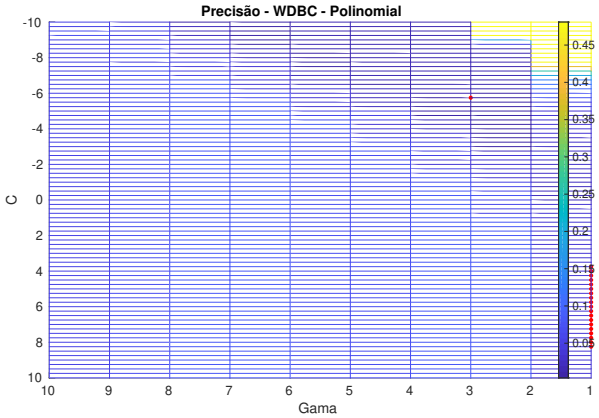


(a) Função precisão.

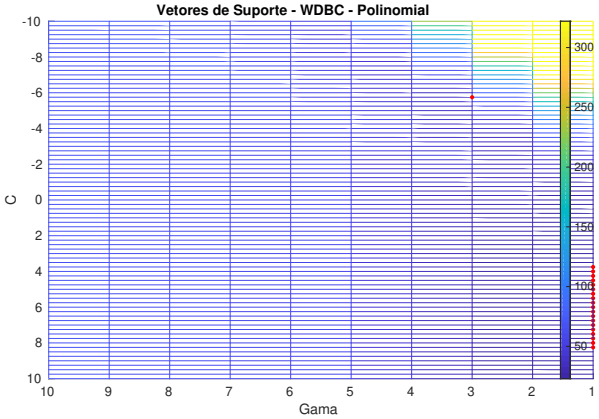


(b) Função vetores de suporte.

Figura B.20 – Funções objetivo do benchmark WDBC com kernel Gaussiano.



(a) Função precisão.



(b) Função vetores de suporte.

Figura B.21 – Funções objetivo do benchmark WDBC com kernel Polinomial.

C TRABALHOS PUBLICADOS EM CONGRESSOS INTERNACIONAIS

- SANTOS, C. E.; SAMPAIO, R. C.; AYALA, H.; COELHO, L. dos S.; JACOBI, R.; LLANOS, C. H. A SVM optimization tool and FPGA system architecture applied to NMPC. In: 30th Symposium on Integrated Circuits and Systems Design. Fortaleza, CE, Brazil: [s.n.], 2017.

- SAMPAIO, R. C.; SANTOS, C. E.; JACOBI, R.; LLANOS, C. H.; COELHO, L. dos S.; AYALA, H. Support vector regression based nonlinear model predictive control on FPGA. In: 24th ABCM International Congress of Mechanical Engineering - COBEM. Curitiba, PR, Brazil: [s.n.], 2017.

- PASSOS, J.C.P.; SANTOS, C. E.; SAMPAIO, R. C.; COELHO, L. dos S.; LLANOS, C. H. Sizing Optimization of a Exoskeleton Structure Utilizing Finite Element Analysis and Multi-Objective Search. In: XXII Congresso Brasileiro de Automática - CBA. João Pessoa, PB, Brazil:[s.n.], 2018.

- SANTOS, E. J. R.; SANTOS, C. E.; LLANOS, C. H.; MOTTA, M. A. S. Aplicação de Técnicas de Inteligência Artificial para Predição de Precipitação em Belém-PA. In: XX Congresso Brasileiro de Meteorologia - CBMET. Maceió, AL, Brasil: [s.n.], 2018.

D TRABALHOS SUBMETIDOS PARA *JOURNAL INTERNACIONAL*

- ANACONA-MOSQUERA, O.E.;SANTOS, C. E.; CABRAI, F.; SAMPAIO, R. C.; TEODORO, G.; JACOBI, R.;LLANOS, C. H. Submetido a: IEEE Design & Test, 2018.