UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Alexandre Reeberg de Mello

CONTRIBUTIONS TO SUPPORT VECTOR MACHINE CLASSIFIER

Florianópolis - SC
2019

Alexandre Reeberg de Mello

CONTRIBUTIONS TO SUPPORT VECTOR MACHINE CLASSIFIER

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de doutor em Engenharia de Automação e Sistemas.
Orientador: Prof. Marcelo Ricardo Stemmer, Dr.

Florianópolis - SC
2019

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Mello, Alexandre
   CONTRIBUTIONS TO SUPPORT VECTOR MACHINE CLASSIFIER / Alexandre Mello ; orientador, Marcelo Ricardo Stemmer, 2019.
   123 p.

   Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2019.

   Inclui referências.

   1. Engenharia de Automação e Sistemas. 2. Support Vector Machine. 3. Incremental Learning. I. Ricardo Stemmer, Marcelo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Alexandre Reeberg de Mello

CONTRIBUTIONS TO SUPPORT VECTOR MACHINE CLASSIFIER

O presente trabalho em nível de doutorado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Leandro dos Santos Coelho, Dr.
Universidade Federal do Paraná/Ponticia Universidade Católica do Paraná

Prof. Adilson Gonzaga, Dr.
Universidade de São Paulo

Prof. Jomi Fred Hubner, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de doutor em Engenharia de Automação e Sistemas.

Werner Kraus Junior:53108523953

Assinado de forma digital por Werner Kraus Junior:53108523953
Dados: 2019.09.09 11:09:02 -03'00'

Prof. Werner Kraus Junior, Dr.
Coordenador do Programa

Marcelo Ricardo Stemmer:37609327972

Assinado de forma digital por Marcelo Ricardo Stemmer:37609327972
Dados: 2019.09.06 09:43:32 -03'00'

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Florianópolis - SC, 03 de julho de 2019.

Este trabalho é dedicado a todos que fizeram parte dessa jornada.

AGRADECIMENTOS

Aos meus pais, irmão e familiares por toda a paciência, suporte, dedicação e amor. Aos colegas do departamento que sempre estiveram presentes nas diversas situações, e que tornaram o longo processo de pós-graduação mais leve a agradável. Aos grandes amigos que a pós-graduação me trouxe, Leonardo, Jessica, Sidney, Flávio, Luís, Karila, Fernando e Tadeu. Ao professor Marcelo pela orientação e incentivo durante toda a jornada de pós-graduação. Ao professor Alessandro pela oportunidade de aprendizado e excelente orientação durante o estágio sanduíche na École de Technologie Supérieure, e aos amigos e colegas que fiz durante este período, Jonathan, Luiz Gustavo, Thanh Le, Ronan, Leo, Saypraseuth, Rafael e Sajjad. E principalmente à minha esposa Jessica, meu maior amor e minha inspiração diária. Graças a ela foi possível concluir esta tese, pois sempre esteve ao meu lado, me apoiou em todos os momentos e abriu mão de seus planos para me acompanhar durante o período de sanduíche. Finalmente, à Deus.

RESUMO EXPANDIDO

Nas últimas décadas a área de aprendizado de máquina tem se tornado cada vez mais presente da tecnologia da informação. Com o aumento crescente da quantidade de dados disponíveis, a tarefa de automaticamente descobrir padrões nos dados tem sido realçada pelas pesquisas acadêmicas e do meio industrial. Um método para identificar padrões é estimar uma função que se aproxima do comportamento dos dados disponíveis, e para criar uma função com capacidade de aprendizagem é necessária conter os seguintes tópicos: uma teoria baseada no princípio da indução e a possibilidade de incluir conhecimento passado. Nesta tese o foco é no algoritmo de Máquina de Vetores de Suporte (Support Vector Machine, SVM) para tarefas de classificação de dados, introduzido em meados dos anos 90. Desta forma, é necessário analisar vários aspectos do SVM, incluindo a formulação, métodos de treinamento, o ajuste de hiperparâmetros e a aplicabilidade em conjunto de dados de diferentes tamanhos e finalidades. Baseado nessas análises, nós fizemos três propostas de contribuição. Dada a complexidade computacional do SVM, na primeira contribuição, propõe-se um método para viabilizar o uso do SVM em grandes conjuntos de dados (por exemplo nos casos em que não é possível carregar todo o conjunto de dados na memória disponível) por meio da pré-seleção de instâncias candidatas que são mais provaveis a melhorar a performance do erro de generalização. O objetivo do método é diminuir o tempo dos processos de treinamento e ajuste de hiperparâmetros do SVM degradando o mínimo possível o erro de generalização.

A segunda contribuição está relacionada com a tarefa de ajuste de hyperparâmetros, que dependendo do caso pode ser demorada. A utilização do SVM com uma função *kernel* fornece uma liberdade que viabiliza aplicar o método em muitas situações, contudo, a escolha dos hiperparâmetros pode ser uma desvantagem. Desta forma, nós propõe-se um método para ajuste dos hiperparâmetros que utiliza um mínimo local pré-definido como critério de parada e possui um mecanismo para escapar de eventuais locais mínimos. Muitas tentativas foram realizadas para abordar os problemas do SVM relacionados ao custo computacional, aprendizado incremental e consumo de memória relacionado a ambos. Na última contribuição, introduz-se uma nova variação do SVM com uma complexidade computacional menor comparado ao SVM original, que é capaz de lidar com procedimentos incrementais e decrementais (sem a necessidade de retreinar completamente), e é eficiente no gerenciamento de memória.

Objetivos

A fim de lidar com grandes conjuntos de dados nos problemas de aprendizado nós propõe-se um método de amostragem passiva que seleciona um sub-conjunto dos conjuntos de treinamento disponível diminuindo a necessidade de recurso computacional, porém, mantendo a capacidade de generalização. Os resultados do protocolo experimental mostra que o método proposto pré-seleciona instâncias que tem mais chance de serem vetores de suporte (e seus respectivos vizinhos) mesmo em um espaço reduzido, logo, não compromete a capacidade de generalização. Os métodos de otimização de caixa-preta (*blackbox*) baseados em meta-heurística ou por busca

extensiva podem ser utilizados para ajuste de hiperparâmetros, contudo não fornecem propriedades de convergência matemática e um critério dinâmico de parada, o que pode resultar em resultados sub-ótimos. Desta maneira, propõe-se um método para seleção de hiperparâmetros baseado em métodos de otimização por busca direcionada. O método proposto fornece ao usuário uma flexibilidade ao permitir a escolha de parâmetros a fim de explorar diferentes estratégias e situações. Os experimentos mostram que o método proposto é menos suscetível a resultados sub-ótimo. Por fim, introduz-se uma nova variante do SVM adequada para o aprendizado incremental e decremental que integra elementos do SVM, da variação do SVM gêmeo (*Twin SVM*) e da teoria difusa (*fuzzy*). O método proposto mantém a flexibilidade do SVM e adiciona os procedimentos de incremento e decremento. O conceito difuso incorporado realça a resistência a ruído e tende a melhorar a generalização do método proposto quando utilizamos o modelo linear. A etapa incremental pode ser utilizada em diferentes quantidades, e o procedimento decremental controla a complexidade do modelo. Segundo os resultados apresentados, o método possui uma capacidade de generalização competitiva comparada aos outros métodos de aprendizado incremental baseado em SVM.

Metodologia

Para o desenvolvimento da tese foi realizada uma pesquisa exploratória a fim de compreender as limitações do SVM, logo ter capacidade de formular as hipóteses. Uma pesquisa empírica foi realizada para analisar de maneira mais profunda os métodos por meio de uma pesquisa bibliográfica por meio de uma revisão sistemática. Define-se o escopo da tese como uma variante do SVM adequada para conjuntos de dados contínuos e de larga escala, com uma solução eficiente para ajustes de hiperparâmetros. Uma pesquisa quantitativa foi conduzida com o propósito de validação para comparar diretamente os métodos propostos com trabalhos relacionados utilizando conjuntos de dados *benchmark* ou criados a fim de analisar aspectos individuais, analisando as saídas de forma numérica (como a exatidão), a complexidade computacional, o tempo de processamento e o consumo de memória RAM. Questiona-se a validação em relação aos resultados do protocolo experimental e a aplicabilidade em situações reais, e foram desenvolvidos múltiplos procedimentos experimentais para suavizar a incerteza da validação interna e avaliar as variáveis com parâmetros numéricos.

Resultados e discussão

Através da validações realizadas em quatro conjuntos de dados públicos, o método proposto para reduzir o tempo de processamento através da amostragem passiva apresentou resultados similares comparado ao método sem amostragem, confirmando a pouca degradação do modelo. A principal contribuição do método está na possibilidade de realizar a amostragem passiva para determinar do sub-conjunto em um espaço reduzido.

Em testes realizados em diversos conjutos de dados, o método proposto para ajuste de hiperparâmetros apresentou resultados mais consistentes comparados à outros métodos da literatura, isto é, uma convergência mais efetiva que requer um menor

número de avaliações da função objetivo. Os diferenciais do método proposto é a existência de um critéria dinâmico de parada, um método para escapar de mínimos locais indesejados, e a integração com um *framework* já existente.

Testes realizados em diversos conjuntos de dados da variante que integra elementos do SVM, da variação do SVM gêmeo (*Twin SVM*) e da teoria difusa (*fuzzy*) apresentaram um tempo de treinamento menor comparado ao SVM clássica, porém com a capacidade de incrementar e decrementar novos elementos sem a necessidade de retreinamento. O conceito difuso apresentou uma resistência à pequenos ruídos, e a performance utilizando um kernel aproximado aproximou-se de modelos não-lineares.

Considerações finais

Nesta tese apresentamos metodologias que visam melhorar as questões de escalabilidade, eficiência computacional e performance de generalização do SVM ou de uma variante, e todos os métodos propostos são adequados para serem utilizados em cenários reais.

Palavras-chave: Máquina de Vetores de Suporte. Aprendizado de Máquina. Máquina de Vetores de Suporte Gêmea. Inteligência Artificial.

RESUMO

Nas últimas décadas a área de aprendizado de máquina tem se tornado cada vez mais presente da tecnologia da informação. Com o aumento crescente da quantidade de dados disponíveis, a tarefa de automaticamente descobrir padrões nos dados tem sido realçada pelas pesquisas acadêmicas e do meio industrial. Um método para identificar padrões é estimar uma função que se aproxima do comportamento dos dados disponíveis, e para criar uma função com capacidade de aprendizagem é necessária conter os seguintes tópicos: uma teoria baseada no princípio da indução e a possibilidade de incluir conhecimento passado. Nesta tese o foco é no algoritmo de Máquina de Vetores de Suporte (Support Vector Machine, SVM) para tarefas de classificação de dados, introduzido em meados dos anos 90. Desta forma, é necessário analisar vários aspectos do SVM, incluindo a formulação, métodos de treinamento, o ajuste de hiperparâmetros e a aplicabilidade em conjunto de dados de diferentes tamanhos e finalidades. Baseado nessas análises, nós fizemos três propostas de contribuição. Dada a complexidade computacional do SVM, na primeira contribuição, propõe-se um método para viabilizar o uso do SVM em grandes conjuntos de dados (por exemplo nos casos em que não é possível carregar todo o conjunto de dados na memória disponível) por meio da pré-seleção de instâncias candidatas que são mais provaveis a melhorar a performance do erro de generalização. O objetivo do método é diminuir o tempo dos processos de treinamento e ajuste de hiperparâmetros do SVM degradando o mínimo possível o erro de generalização. A segunda contribuição está relacionada com a tarefa de ajuste de hyperparâmetros, que dependendo do caso pode ser demorada. A utilização do SVM com uma função *kernel* fornece uma liberdade que viabiliza aplicar o método em muitas situações, contudo, a escolha dos hiperparâmetros pode ser uma desvantagem. Desta forma, nós propõe-se um método para ajuste dos hiperparâmetros que utiliza um mínimo local pré-definido como critério de parada e possui um mecanismo para escapar de eventuais locais mínimos.

Muitas tentativas foram realizadas para abordar os problemas do SVM relacionados ao custo computacional, aprendizado incremental e consumo de memória relacionado a ambos. Na última contribuição, introduz-se uma nova variação do SVM com uma complexidade computacional menor comparado ao SVM original, que é capaz de lidar com procedimentos incrementais e decrementais (sem a necessidade de retreinar completamente), e é eficiente no gerenciamento de memória. A fim de lidar com grandes conjuntos de dados nos problemas de aprendizado nós propõe-se um método de amostragem passiva que seleciona um sub-conjunto dos conjuntos de treinamento disponível diminuindo a necessidade de recurso computacional, porém, mantendo a capacidade de generalização. Os resultados do protocolo experimental mostra que o método proposto pré-seleciona instâncias que tem mais chance de serem vetores de suporte (e seus respectivos vizinhos) mesmo em um espaço reduzido, logo, não compromete a capacidade de generalização.

Os métodos de otimização de caixa-preta (*blackbox*) baseados em meta-heurística ou por busca extensiva podem ser utilizados para ajuste de hiperparâmetros, contudo não fornecem propriedades de convergência matemática e um critério dinâmico de parada, o que pode resultar em resultados sub-ótimos. Desta maneira, propõe-se um método para seleção de hiperparâmetros baseado em métodos de otimização por busca direcionada. O método proposto fornece ao usuário uma flexibilidade ao permitir a escolha de parâmetros a fim de explorar diferentes estratégias e situações. Os experimentos mostram que o método proposto é menos suscetível a resultados subótimo. Por fim, introduz-se uma nova variante do SVM adequada para o aprendizado incremental e decremental que integra elementos do SVM, da variação do SVM gêmeo (*Twin SVM*) e da teoria difusa (*fuzzy*). O método proposto mantém a flexibilidade do SVM e adiciona os procedimentos de incremento e decremento. O conceito difuso incorporado realça a resistência a ruído e tende a melhorar a generalização do método proposto quando utilizamos o modelo linear. A etapa incremental pode ser utilizada em diferentes quantidades, e o procedimento decremental controla a complexidade do modelo. Segundo os resultados apresentados, o método possui uma capacidade de generalização competitiva comparada aos outros métodos de aprendizado incremental baseado em SVM.

Para o desenvolvimento da tese foi realizada uma pesquisa exploratória a fim de compreender as limitações do SVM, logo ter capacidade de formular as hipóteses. Uma pesquisa empírica foi realizada para analisar de maneira mais profunda os métodos por meio de uma pesquisa bibliográfica por meio de uma revisão sistemática. Define-se o escopo da tese como uma variante do SVM adequada para conjuntos de dados contínuos e de larga escala, com uma solução eficiente para ajustes de hiperparâmetros. Uma pesquisa quantitativa foi conduzida com o propósito de validação para comparar diretamente os métodos propostos com trabalhos relacionados utilizando conjuntos de dados *benchmark* ou criados a fim de analisar aspectos individuais, analisando as saídas de forma numérica (como a exatidão), a complexidade computacional, o tempo de processamento e o consumo de memória RAM. Questiona-se a validação em relação aos resultados do protocolo experimental e a aplicabilidade em situações reais, e foram desenvolvidos múltiplos procedimentos experimentais para suavizar a incerteza da validação interna e avaliar as variáveis com parâmetros numéricos.

Nesta tese apresentamos metodologias que visam melhorar as questões de escalabilidade, eficiência computacional e performance de generalização do SVM ou de uma variante, e todos os métodos propostos são adequados para serem utilizados em cenários reais.

Palavras-chave: Máquina de Vetores de Suporte. Aprendizado de Máquina. Máquina de Vetores de Suporte Gêmea. Inteligência Artificial.

ABSTRACT

Over the past decade machine learning has become one of the mainstays of the information technology, and with the increasing amount of available data the automatically discover of patterns in data has become one of the main tasks in the field. A method to find these patterns is to estimate a data-based function and to build a successful learning function it must include: a solid theoretical background based on an induction principle, the possibility to include prior knowledge, and an efficient way to apply it in practice. In this thesis, we will focus on the Support Vector Machine (SVM) algorithm for classification tasks, which has been introduced in the mid-90s and has become popular in academic and industrial communities since then. We will analyze many aspects of the SVM, including its formulation, the training method, the hyperparameters tuning, and applicability on a range of different datasets. Based on the analysis, we propose three contributions. Given the computational complexity of the SVM, in the first contribution, we propose a method to make viable the usage of the SVM for large datasets (i.e., cases that the dataset does not fit on available memory) by preselecting instance candidates that are more suitable to enhance the generalization error performance. The method goal is to decrease the training and hyperparameter tuning procedures time degrading as less as possible the generalization error.

The second contribution is related to the hyperparameter tuning task, that depending on the case may be time-consuming. The use of the SVM with a kernel function gives a freedom that allows to apply the method in many situations, however, choosing the hyperparameters is a downside. In this way, we propose a hyperparameters tuning method with mathematical convergence properties that uses a predefined local minima as stopping criteria and have a mechanism to escape from undesired local minima.

Many attempts have been made to tackle the problems of the SVM computational cost, incremental learning, and memory consumption related to both. In the last contribution, we introduce a novel SVM-variant with a smaller computational complexity compared to the original SVM, which is able to deal with incremental and decremental procedures (without the need for full retraining) and being memory efficient.

We perform exploratory research to understand the SVM limitations, so we are able to formulate our hypothesis. We did empirical research to further analyze the proposed methods through bibliography research over a systematic review. We set the thesis scope as an SVM-variant suitable for continuous and large-scale datasets with an effective hyperparameters tuning solution. We conduct quantitative research with a proposal validation that directly compares the proposed methods with related work using benchmark or controlled generated datasets, analyzing the numerical outputs as accuracy, computational complexity, processing time, and RAM consumption. We question the validation regarding the results of the experimental protocol and the applicability in real-world situations. We develop multiple experimental procedures to mitigate the uncertainty of the internal validation, and we evaluate the variables with numerical parameters.

To deal with large datasets in learning problems we propose a passive sampling that selects a subset from the available training data that can be handled within time an computational resource constraints, maintaining the generalization capability. The experimental protocol results shows that the proposed method preselects instances that are likely to be support vectors (and its neighbors) even in a reduced space, thus, it does not compromise much the generalization capability.

Most common methods for hyperparameters tuning does not provide mathematical convergence properties and dynamic stopping criteria, leading to sub-optimal outcomes, in this way, the proposed model selection technique leads to an outcome less susceptible to sub-optimal results and requires inferior processing time compared to frequently used methods. The prosed method gives the user a flexibility of choosing parameters to explore different strategies and situations, and the experimental shows that the method is more likely to require less function evaluations to reach good set of hyperparameters.

We introduced a novel SVM-variant suitable for incremental and decremental learning that integrates elements from many methods from literature. The proposed method keeps the SVM flexibility and adds incremental and decremental procedures. The fuzzy concept incorporated enhances noise-resistance and generalization performance when using the linear model. The incremental step can run with different batch sizes, and the decremental procedure controls the model complexity. According to the experimental results, the method presents a competitive generalization capability compared to the best methods available.

In this thesis, we present methodologies that address the issues of improving scalability, computational and data efficiency, and generalization performance of the SVM or a variant, and all proposed methods are suitable to be used in real-world scenarios.

Keywords: Support Vector Machine. Incremental Learning. Machine Learning. Twin Support Vector Machine. Artificial Intelligence.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BADS	Bayesian Adaptive Direct Search
BBO	Black-Box Optimization
BO	Bayesian Optimization
CV	Cross Validation
DAG	Directed Acyclic Graph
DCD	Dual Coordinate Descent
DG	Delaunay Graph
DGs	Delaunay Graphs
ECOC	Error-Correcting Output Codes
FBTWSVM	Fuzzy Bounded Twin Support Vector Machine
GEPSVM	Generalized Eigenvalue Proximal Support Vector Machine
GPS	Generalized Pattern Search
GS	Grid Search
GSVM	Generalized Support Vector Machine
IELM	Incremental Extreme Learning Machine
ILVQ	Incremental Learning Vector Quantization
ISVM	Incremental Support Vector Machine
ITWSVM	Improved Twin Support Vector Machine
I-TWSVM	Incremental Twin Support Vector Machine
KKT	Karush–Kuhn–Tucker Complementary Conditions
LDA	Linear Discriminant Analysis
LPSVM	Linear Programming Support Vector Machine
LS-SVM	Least Squares Support Vector Machine
LS-TWSVM	Least Square Twin Support Vector Machine
LSVM	Lagrangian Support Vector Machine
MADS	Mesh Adaptive Direct Search
NFL	No Free Lunch
NM	Nelder-Mead
NOMAD	Nonlinear Optimization by Mesh Adaptive Direct Search
ODAG-LSTSVM	Optimal Directed Acyclic Graph to the Least Squares Twin Support Vector Machine
ORF	Online Random Forest
Ortho-MAD2S	Ortho Mesh Adaptive Direct Dual Search
OTWISVM	On-line Twin Independent Vector Machine
OVO	One-Vs-One
OVR	One-Vs-Rest
PCA	Principal Component Analysis
PSVM	Proximal Support Vector Machine

QPP	Quadratic Programming Problem
QPPs	Quadratic Programming Problems
RBF	Radial Basis Function
RF	Random Fourier
RS	Random Search
RSVM	Reduced Support Vector Machine
SA	Simulated Annealing
SGD	Stochastic Gradient Descent
SMO	Sequential Minimal Optimization
SMW	Sherman-Morrison-Woodbury
SOR	Successive Over-Relaxation
SV	Support Vector
SVM	Support Vector Machine
SVMs	Support Vector Machines
SVs	Support Vectors
TBSVM	Twin Bounded Support Vector Machine
TSMO	Two-Phase Sequential Minimal Optimization
T-SNE	t-Distributed Stochastic Neighbor Embedding
TWSVM	Twin Support Vector Machine
VNS	Variable Neighborhood Search
WSS1	First-Order Approximation
WSS3	Second-Order Approximation

SUMÁRIO

1 INTRODUCTION

1.1 MOTIVATION

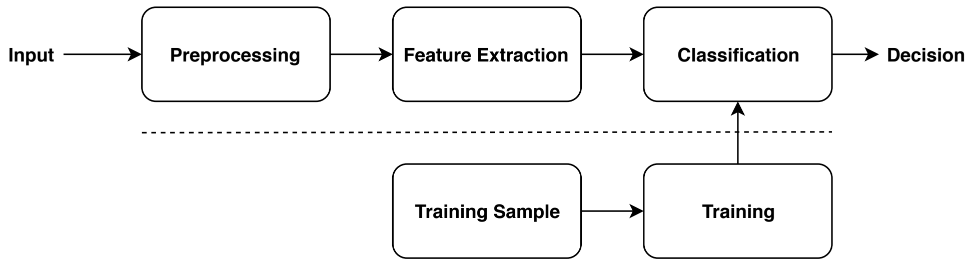
The problem of learning from data has been investigated by philosophers, scientists, and engineers throughout history, and there is a long history of studying this problem within the statistical framework. Researchers in the field of artificial intelligence started to consider the problem of learning from its beginning, as Alan Turing (TURING, 1950) proposed the idea of learning machines in 1950, and just a few years later the first examples of learning method were developed, as the Arthur Samuel's draughts player was an early example of reinforcement learning (SAMUEL, 1959), the Frank Rosenblatt's perceptron (ROSENBLATT, 1958), and the Solomonoff inductive learning in (SOLOMONOFF, 1964a) and (SOLOMONOFF, 1964b).

The field of machine learning is an important sub-field of artificial intelligence and is based on the idea of modeling learning systems as a problem of search in suitable hypothesis space (CRISTIANINI; SHAWE-TAYLOR, 2000). Over the past few decades, machine learning has become one of the mainstays of information technology, and with the increasing amounts of available data, data analysis is becoming indispensable to technological progress. A fundamental problem of machine learning is the searching for patterns in data, and the field of *pattern recognition* is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories (BISHOP, 2006). Pattern recognition is used in many areas of science and engineering, as the optical character recognition, speech recognition, computer vision, computer-aided diagnosis, character recognition, face and fingerprint recognition, galaxy classifying, temporal analysis, fault diagnosis, among others.

Pattern classification differs from classical statistical hypothesis testing, wherein the sensed data are used to decide whether or not to reject a null hypothesis in favor of some alternative hypothesis, i.e., pattern classification seeks to find the most probable hypothesis from a set of hypotheses (DUDA; HART; STORK, 2001). The general structure of a pattern classification system is given in Fig. 1, and we define its three main steps in the following (NIEMANN, 1990):

1. Data preprocessing: A pattern is transformed to some other pattern which is expected to be more suited for further processing and which should yield improved results of classification and analysis, i.e., it is the act of modifying the input data to simplify subsequent operations without losing relevant information. Some of the most common preprocessing procedures are the noise removal, spatial or temporal element segmentation, alignment or registration of the query to a canonical frame, fixed transformation of the data, and the transformation from a numerable

Figura 1 – The flowchart of a pattern classification system.



representation to a vector space.

2. Feature Extraction: The measurements which represent the data. The statistical model one uses is crucially dependent on the choice of features. Hence it is useful to consider alternative representations of the same measurements (i.e. different features). For example, different representations of the color values in an image. General techniques for finding new representations include discriminant analysis, Fourier analysis, principal component analysis, and clustering (RIPLEY, 1996).

3. Classification: Assigning a class to measurement, or equivalently, identifying the probabilistic source of a measurement. The only statistical model that is needed is the conditional model of the class variable given the measurement. This conditional model can be obtained from a joint model or it can be learned directly. The former approach is generative since it models the measurements in each class. It is more work, but it can exploit more prior knowledge, needs less data, is more modular, and can handle missing or corrupted data. Methods include mixture models and Hidden Markov Models. The latter approach is discriminative since it focuses only on discriminating one class from another. It can be more efficient once trained and requires fewer modeling assumptions. Methods include SVM, neural networks, logistic regression, generalized linear classifiers, and instance-based methods (RIPLEY, 1996).

This thesis concerns about the learning (classification), thus we are not focusing on the preprocessing and feature extraction steps. Learning refers to some form of algorithm for reducing the error on a set of training data, and comes in several general forms:

- Supervised Learning: The task is to estimate (or to predict, which is the same in this context) one or more outputs (or targets) based on one or more inputs using empirical evidence (a dataset) that is collected beforehand (RUSSELL; NORVIG; DAVIS, 2010).

- Unsupervised Learning (clustering): There is no explicit teacher, and the system forms clusters (or natural groupings) of the input patterns. It is always defined explicitly or implicitly in the clustering system itself and given a particular set of patterns or cost function, different clustering algorithms lead to different clusters (DUDA; HART; STORK, 2001).

- Semi-supervised Learning: Is the halfway between supervised and unsupervised learning, so the dataset contains labeled and unlabeled instances.
- Reinforcement Learning: It learns how to act optimally in a given environment, especially with delayed and nondeterministic rewards, and it is composed of two interleaved tasks: modeling the environment and making optimal decisions based on the model. The first task is a statistical modeling problem, and the second task is a decision theory problem: converting the expectation of delayed reward into immediate action (DUDA; HART; STORK, 2001).

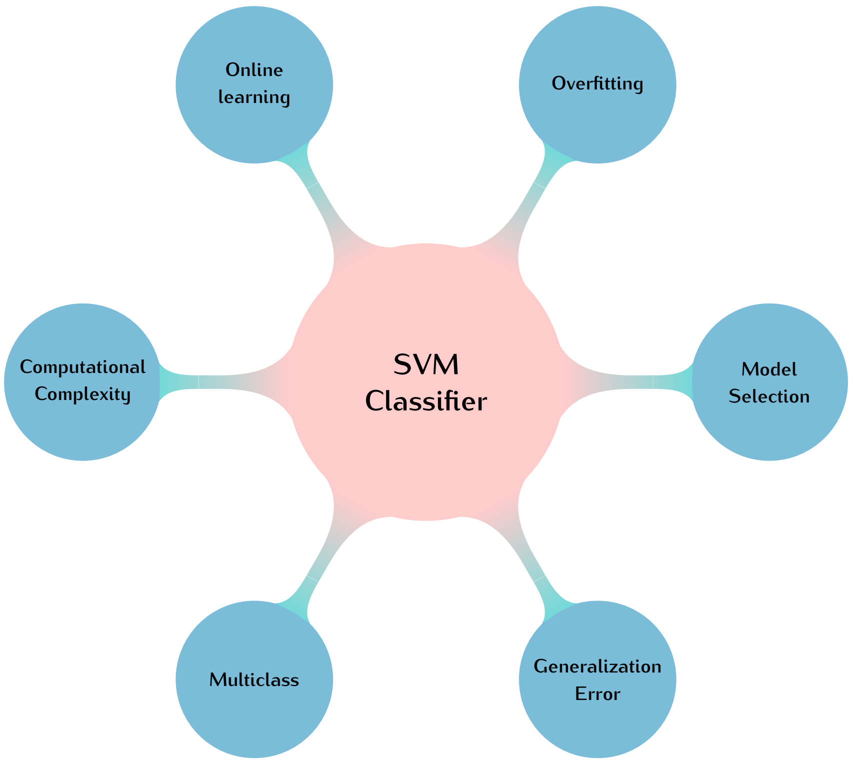
This thesis interest is in supervised learning, especifically in the SVM classifier, in this manner, we do not extend the theory and theorems to the other types of learning. To make viable the use (or development) of a supervised learning method there are several aspects that must be considered, as the computational complexity, that defines how an algorithm scales in the number of features dimensions, patterns, or categories (DUDA; HART; STORK, 2001). There are many characteristics from the SVM classifier that must be considered when applying it, or specially when designing a new one. Fig. 2 presents some important SVM's characteristics that are relevant to the development of this thesis.

The goal of a classifier is to accurately be able to predict outcomes values for previously unseen data, and the generalization error is the aspect that carries this property. Considering the input data $\boldsymbol{x}$, the unknown target function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is the input space (set of all possible input $\boldsymbol{x}$), and $\mathcal{Y}$ is the output space (in the classification problem case, the predicted label), there is a dataset $\mathcal{D}$ of input-output relation, i.e., for each input instance there is an associated label. The learning algorithm goal is, using the dataset D, to get a hypothesis set $\mathcal{H}$ (thus a single hypothesis $h \in \mathcal{H}$) through a formula $g : \mathcal{X} \to \mathcal{Y}$ that approximates f. The error rate within sample E_{in} is the fraction of $\mathcal{D}$ where f and h, whereas the out-of-sample error E_{out} measures how well the training data has generalized to a data that has not seen before (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012). In this manner, the generalization error is defined as the discrepancy between E_{in} and E_{out}[1].

The learning algorithms are evaluated over a set of finite samples, so the algorithm's evaluation may be sensitive to sampling error. As a result, the measurements of prediction error on the current data may not provide information about the generalization error, which characterizes the overfitting, that is the phenomenon where fitting the data well no longer indicates that we will get a decent out-of-sample error, and may

[1] The mathematical proof can be found in (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012).

Figura 2 – Aspects of a supervised learning classifier.



actually lead to the opposite effect (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012). Mathematically speaking, it is when the hypothesis has a lower E_{in} and results in a higher E_{out}. Thus, the generalization error can be minimized by avoiding the overfitting in the learning model.

The computational complexity in machine learning (KEARNS, 1990) is measured in terms of *information* and *computation*, where the first concerns about the generalization performance of learning (e.g., the required number of training instances), and the latter concerns the computation resources applied to the training data to extract from it learner's prediction. In this manner, there is a contradiction in improving the *information* and deteriorating the *computation*, and vice versa. The formal definition of the computational complexity of learning is: *Let $\mathcal{H}$ be a hypothesis class of functions defined over an instance space of size d, and let ε, δ be accuracy and confidence parameters. Let $m(\mathcal{H}, \varepsilon, \delta)$ be the sample complexity of learning $\mathcal{H}$ with accuracy ε and confidence $1 - \delta$. Then, the computational complexity of (ε, δ)-learning $\mathcal{H}$ is said to be bounded by $T(dm(\mathcal{H}, \varepsilon, \delta))$ if there exists a learning algorithm that (ε, δ)-learns $\mathcal{H}$ and whose runtime is bounded by $T(dm(\mathcal{H}, \varepsilon, \delta))$ and such that the runtime of the hypothesis it outputs on any new instance x is also bounded by $T(dm(\mathcal{H}, \varepsilon, \delta))$.* In this way, it is possible to solve the empirical risk minimization in time $O(|\mathcal{H}|dm(\mathcal{H}, \varepsilon, \delta))$ by performing an exhaustive search over $\mathcal{H}$ with a training set of size $m(\mathcal{H}, \varepsilon, \delta)$. Following this definition, it is possible to define an efficient learning as *a sequence of learning problems $(d_n, \mathcal{H}_n, \varepsilon_n, \delta_n)_{n=1}^{\infty}$ if there exists a polynomial p that for each n there is a learning* *algorithm that* $(\varepsilon_n, \delta_n)$*-learns* $\mathcal{H}_n$ *in time* $p(d_n m(\mathcal{H}_n, \varepsilon_n, \delta_n))$. The solutions proposed in this thesis must follow (and contribute) to the efficient learning principle, to turn viable the proposed methods use in a practical scenario.

The model selection plays an important role, and considering the SVM case, it refers to control the hyperparameters of the classification in order to achieve the lowest test error, i.e., the lowest probability of misclassification of unseen test instances (GOLD; SOLLICH, 2003), however, it falls in the No Free Lunch (NFL) theorem (WOLPERT, 1996) (WOLPERT; MACREADY, 1997) (WOLPERT, 2013), which implies that all learning algorithms perform equally well when averaged over all possible datasets. This nonintuitive idea meant that looking for a general, a predictive algorithm is not feasible, and it is well known in empirical research that some algorithms perform consistently much better than others (GÓMEZ; ROJAS, 2016). The NFL for search and optimization applies to finite spaces and algorithms that do not re-sample points. All algorithms that search for an extremum of a cost function perform exactly the same when averaged over all possible cost functions, so, for any search/optimization algorithm and in this case the control of hyperparameters tuning, any elevated performance over one class of problems is exactly paid for in performance over another class. For this reason, there is no single optimal set of hyperparameters to all situations, and to each different application, it is necessary to properly tune the hyperparameters, where depending on the application, can be time-consuming.

1.1.1 ONLINE LEARNING AND MULTICLASS

Online learning (or incremental learning) refers to the situation of continuous model adaptation based on a constantly arriving data stream, and it becomes necessary in interactive scenarios where training instances are provided over time (GEPPERTH; HAMMER, 2016). Therefore, online learning can continuously integrate new information into already constructed models, allowing to use new data as soon as it is available, which leads to all-time up to date models, and also reduces the costs for data storage and maintenance. The online learning algorithms challenges are: it must have limited memory resources, the model has to adapt gradually without complete retraining, it must preserve previously acquired knowledge without catastrophic forgetting, and only a limited number of training instances are allowed to be maintained (LOSING; HAMMER; WERSING, 2018). Due to the ability of continuous processing (in some cases the algorithms can deal with large-scale and real-time), online learning has gained the attention in the context of the Internet of Things, Big Data, and learning in nonstationary environments.

Supervised multiclass classification algorithms aim at assigning a class label for each input example when the problem has three or more classes. Several classification algorithms have been proposed to solve a two class problem (as the original SVM itself), and while some cases can be naturally extended to the multiclass case, there are methods that need formulation adaptation. To extend a binary classifier with multiclass problems we can either combine several binary classifiers or propose a larger optimization problem (HSU; LIN, 2002), and the choice of method must be done considering the other aspects of the supervised learning classifier, as the computational complexity, generalization error, overfitting, and online learning capability.

1.2 RESEARCH OBJECTIVES

The SVM is a supervised learning classifier with a computational complexity of $O(n^3)$ for n instances, thus it has some limitations to deal with large datasets, and considering the existence of hyperparameters, it requires a proper model selection to get the best generalization error as possible. In this manner, considering the aspects of the training dataset size and the model selection, larger datasets (consider here the number of instances and the dimension) requires more time to the SVM learn, thus it takes longer to properly tune the hyperparameters. So, there is a trade-off between the time available to tune the hyperparameters (because it is related to the dataset dimension) and the generalization error. In this manner, this thesis proposes solutions to ease the use of the SVM or a SVM-variant.

The research objective of this thesis is to develop methods to improve the usability of the SVM (or an SVM-variant) under real-world application scenarios, and its based on the following research questions:

1. How to make viable the use of an SVM classifier for large datasets, given that the computational complexity is $O(n^3)$, thus, the training procedure may be unpractical for datasets with a large number of instances and attributes?

   To make viable the usage of a regular SVM for large datasets, (i.e., cases that the dataset does not fit on available memory) we propose to preselect instance candidates that are more suitable to enhance the generalization error performance. This process increases the training and hyperparameter tuning procedures degrading as less as possible the generalization error performance.

2. How to tune the SVM hyperparameters in an efficient manner?

   The hyperparameter tuning may be a time-consuming task, so we propose to use a method with mathematical convergence properties that use a predefined local minima as stopping criteria and have a mechanism to escape from undesired local minima.

3. How to reduce the computational cost of the SVM adapting the optimization formula to deal with large and continuous data?

We propose an SVM-variant that has a smaller computational complexity compared to the original SVM, and it is able to deal with incremental and decremental procedures without the need for full retraining and being memory efficient.

1.3 METHODOLOGY AND VALIDATION

This thesis is inspired by other academic works with similar thematics related to SVM, and we narrow the related work to selecting training sets ((NALEPA; KAWULOK, 2016); (NALEPA; KAWULOK, 2018); (KAWULOK; NALEPA, 2012); (SHEN *et al.*, 2016); (GOTO; ISHIDA; UCHIDA, 2015); (LOPEZ CHAU *et al.*, 2010)), SVM hyperparameters tuning strategies ((CHEN; FLORERO-SALINAS; LI, 2017), (CHUNG *et al.*, 2003), (GOLD; SOLLICH, 2003)), and SVM-variations ((SUYKENS; VANDEWALLE, 1999), (MANGASARIAN, 1998), (FUNG, G.; MANGASARIAN, O. L., 2001), (JAYADEVA; KHEMCHANDANI, R.; CHANDRA, 2007), (CAUWENBERGHS; POGGIO, 2000), (TIAN; JU *et al.*, 2014), (GAO; WANG *et al.*, 2015)).

Considering the thesis' goals, we perform exploratory research to understand the SVM limitations, so we can formulate our hypothesis. We made an empirical research to further analyze and mitigate potential threats to the proposal validation, in our case, the proposed methods, through a bibliography research over a systematic review, selecting works considering the theme proximity to this thesis, and the presence of a solid methodological fundamentals and a suitability to be applied in real-world scenario.

Considering the many aspects of the SVM (as the optimization problem, solver, application, among others), we set the thesis scope as an SVM-variant suitable to continuous and large datasets with an effective hyperparameters tuning solution. It is important to highlight that the proposed methods do not depend on the values used to exemplify and illustrate this document. We conduct a quantitative research with a proposal validation that directly compares the proposed method with related work, using benchmark or controlled generated datasets and analyzing the accuracy, computational complexity, processing time, and RAM consumption (when applicable to the respective context).

We question the validation in two topics: the experimental protocols results and the applicability in real-world situations. We develop multiple experimental procedures to mitigate the uncertainty of the internal validation, to cover the maximum types of scenarios as possible. We design the experimental procedures to evaluate the variables respective to each contribution behavior under real-world and controlled datasets, with the goal of validating the proposed methods to generalization. For each experimental protocol, we define numerical parameters to compare the proposed methods with related others, as the accuracy, considering different scenarios that include diverse situations. All further detail about the used algorithms, datasets, and experimental protocols for validation are detailed in the respective chapters.

1.4 SUMMARY OF CONTRIBUTIONS

This thesis brings contributions to different aspects of the Support Vector Machines that can be used combined or separate in different application scenarios.

The contributions of Chapter 3 is about preselecting SV candidates as a form of passive sampling. The proposed technique preselects SV candidates that are in the convex-hull of its class and its neighbors from the Delaunay graph, and the SV candidates size is related to the Delaunay graph connections density from points that belong to the convex-hull of each class from the dataset, consequently, the SVM training time is proportionally faster. The method's restriction is that, if the dataset instances have more than 3 features, it is necessary to apply a dimensional reduction technique. The main contribution is that the proposed method works in a reduced space, which means that preselecting SV coincides well with original SV, as the accuracy of tested datasets does not decrease significantly. Also, the dimensional reduction techniques smooth eventual noises present in the dataset. Additionally, using a graph-based method to preselect SV works in datasets with the presence of imbalanced distribution.

Hyperparameters tuning for SVM is a Black-Box Optimization (BBO) problem that may be time-consuming (depending on the dataset dimension) and have a high sensitivity, i.e., small differences in the hyperparameters may lead to considerably different results. To improve the model selection performance, Chapter 4 presents the Ortho-MADS with the Nelder-Mead (NM) and the VNS for hyperparameter tuning. The contributions of the proposed method includes a flexibility of choosing parameters (from the method) to explore different strategies and situations, the method has mathematical convergence proof, a mechanism to escape from eventual undesired local minima, and by using the mesh size as a stopping criterion gives to the user the possibility of setting a specific local minimum region instead of using the number of evaluations, time, or fixed-size grid as a stopping criterion. Results show that the proposed method outperforms widely used and state-of-the-art methods for model selection.

Chapter 5 advances the SVM studies to a new incremental and decremental formulation. The FBTWSVM integrates ideas from several methods, and it can be configured (according to the application) to add new SV in the incremental step and the number of occurrences in the decremental step. The fuzzy concept enhances noise-resistance and generalization capability, while the use of a kernel approximation shows a good generalization performance with our linear model. The incremental solution follows the shrinking strategy and can run with different batch sizes, from a single individual to the number of data points that fits the available memory. The decremental procedure is fundamental to control the model complexity, keeping only the most critical SV in the model. According to the experimental results, the Directed Acyclic Graph (DAG) strategy showed a generalization capability and a fast training speed, but for further studies, the use of training data structural and statistical information in the training process may increase the generalization performance. The contributions of the FBTWSVM are the good generalization capability and a fast training speed compared to traditional incremental SVM, the incremental solution follows the shrinking strategy and can run with different batch sizes, from a single individual to the number of data points that fits the available memory, it can adapt current models using the window strategy without retraining, the fuzzy concept enhances noise-resistance and generalization capability, while the use of a kernel approximation shows an efficient generalization performance with our linear model, and the dual coordinate descent method with shrinking requires less memory storage than the TWSVM on the training procedure, as it discards points that are less likely to be SV.

1.5 ORGANIZATION

The remainder of this thesis is organized as follows.

- **Chapter 2** contains the common notations that are used through all the document, followed by the SVM introduction, formal definition, popular solvers, the multiclass problem, and important variations.
- **Chapter 3** presents the *Support Vector Candidates Selection via Delaunay Graph and Convex-Hull for Large and High-Dimensional Datasets*. The method preselect instances from the dataset that have a higher probability to be an SV, reducing the dataset size to enable faster training or hyperparameter tuning.
- **Chapter 4** proposes *A Novel Orthogonal Direction Mesh Adaptive Search Approach for SVM Hyperparameter Tuning*. To proper tune the SVM (or any SVM-variant) hyperparameters can be challenging, and the most common methods in the literature have fixed stopping criterion (as the number of iterations or time) and no mathematical convergence proof, in this way, we propose a method with a dynamic stopping criterion, mathematical convergence proof, and mechanisms to escape from eventual undesired local minima.
- **Chapter 5** proposes an *Incremental and Decremental Fuzzy Bounded Twin Suppor Vector Machine*, which is an SVM-variant that has lower computational complexity, compared to original SVM, an online learning capacity, and a decremental procedure to keep memory efficiency. The proposed method includes a fuzzy component to diminish the effect of outliers, and we propose the use of an approximate kernel operation to avoid the two-dimensional increase in the online learning process. We use the Error-Correcting Output Codes (ECOC) framework to expand the classifier to a multiclass situation.

- **Chapter 6** presents a conclusion for this thesis with a summary of the achievements and perspectives for future research.

2 SUPPORT VECTOR MACHINES

2.1 COMMON NOTATION

We use the following definitions and notations throughout the thesis. The problems are in a $n-$dimensional space $\mathcal{R}^n$. We denote the training data as $D{=}(\mathbf{x_i}, y_i)|i = 1, 2, \ldots, l$, where $\mathbf{x_i} \in \mathcal{R}^n$ is an instance (input data point), and l is the number of instances, with the corresponding label $y_i \in \{1, 2, \ldots, u\}$ where u is the number of classes.

We adopt the definition of incremental learning proposed by (LOSING; HAMMER; WERSING, 2018) as an algorithm that generates on a given stream of training data $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t$ a sequence of models $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_t$, where $(\boldsymbol{h}_i{:}\mathcal{R}^n|i = 1, 2, \ldots, l)$ is a model function solely depending on $\boldsymbol{h}_{i-1}$ and the recent p data points $\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{i-p}$ with p being strictly limited. The approach used to deal with multiclass problems is the DAG, where it is necessary to create $2u - 1$ binary problems. For each binary problem we assign either a positive or a negative label $y_i \in \{+1, -1\}$. Therefore, the training set D is divided into the $l_+ \times n$ dimensional matrix X_+ and $l_- \times n$ dimensional matrix X_- for positive and negative labels respectively, where l_+ and l_- denote the number of data points from each label. We define the aggregation per binary problem as $X{=}[X_+^\top X_-^\top]$, and it denotes all input data points from both classes.

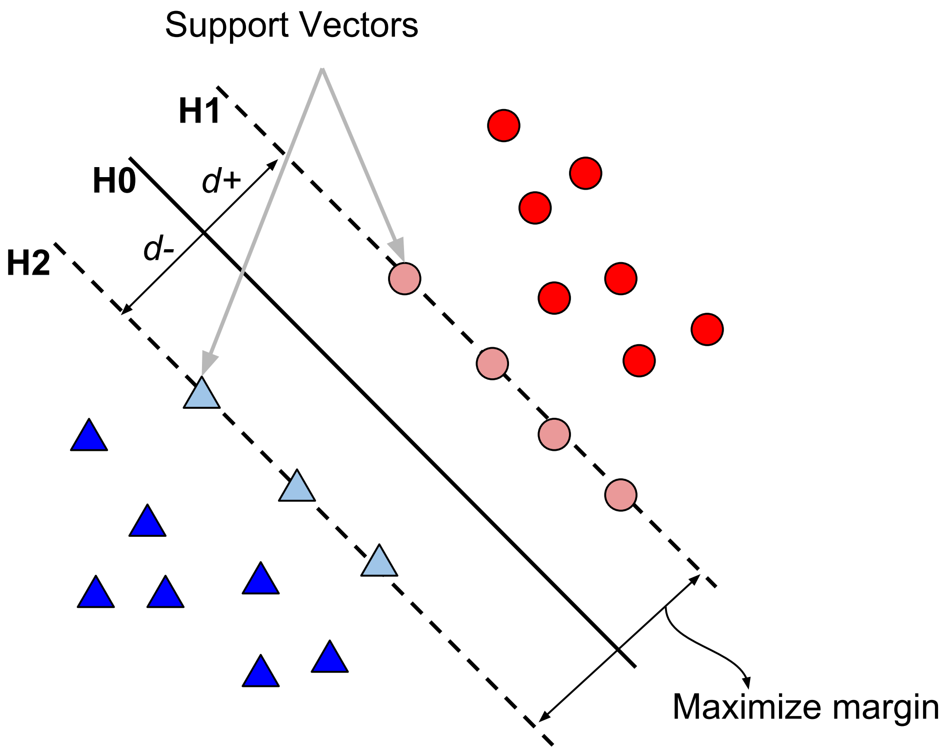
2.2 SVM INTRODUCTION

The SVM (BOSER; GUYON; VAPNIK, 1992; CORTES; CORTES; VAPNIK, 1995) is an algorithm used in machine learning for statistical pattern recognition tasks, originally designed for binary classification (HSU; LIN, 2002), and its first appearance, as so-called maximal margin classifier, optimizes the linear machines generalization error bounds by separating the data with a strategy to find the maximal margin hyperplane in an appropriately chosen kernel-introduced feature space (CRISTIANINI; SHAWE-TAYLOR, 2000). We can reduce the SVM problem to a convex optimization form, i.e., minimize a quadratic function under linear inequality constraints[1], and to solve this convex optimization, we fix the geometric margin through the functional margin to be equal to 1, minimizing the norm of the weight vector associated with the hyperplane $(\boldsymbol{w}, b)$ (it does not change if rescaled). Fig. 3 depicts the maximal margin hyperplane, where $H1$ and $H2$ are the geometric margin, $H0$ is the median between margins, the SV are on the margins $H1$ and $H2$, and the $d+$ is the shortest distance to the closest positive point, and $d-$ is the shortest distance to the closest negative point.

This way, a weight vector $\boldsymbol{w}$ realizes a functional margin of 1 on positive $\boldsymbol{x}^+$ and negative $\boldsymbol{x}^-$ instances, which its geometric margin is computed as:

[1] Aditional proves and deductions can be found in (CRISTIANINI; SHAWE-TAYLOR, 2000).

Figura 3 – A maximal margin hyperplane with its support vectors highlighted.



$$H1 : < \boldsymbol{w} \cdot \boldsymbol{x}^{+} > +b = 1$$

$$H2 : < \boldsymbol{w} \cdot \boldsymbol{x}^{-} > +b = -1$$

where b is the threshold (also known as bias term) that is re-computed after each step. To compute the geometric margin γ, $\boldsymbol{w}$ must be normalized, and is calculated by the functional margin of the following classifier:

$$\gamma = \frac{1}{||\boldsymbol{w}||_2}$$

The problems are in a $n-$dimensional space $\mathcal{R}^n$, so we denote the training data as D=$(\mathbf{x_i}, y_i)|i = 1, 2, \ldots, l$, where $\mathbf{x_i} \in \mathcal{R}^n$ is an instance (input data point), and l is the number of instances, with the corresponding label $y_i \in \{1, 2, \ldots, u\}$ where u is the number of classes. To solve a linear SVM for l linearly separable training samples $S = ((\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_l}, y_l))$ and its respective labels $y_l \in \{\pm 1\}$, we define the hyperplane $(\boldsymbol{w}, b)$ optimization problem as:

$$\begin{aligned} &minimize_{\boldsymbol{w},b} \quad < \boldsymbol{w} \cdot \boldsymbol{w} > \\ &\text{subject to} \quad y_i(< \boldsymbol{w} \cdot \boldsymbol{x_i} >) + b \geq 1 \\ &\qquad\qquad i = 1, ..., l \end{aligned}$$

where it realizes the maximal margin hyperplane with geometric margin $\gamma = 1/||\boldsymbol{w}||_2$. The optimization problem can now be rearranged to its corresponding dual through

Lagrange multipliers, $\boldsymbol{\alpha} \geq 0$, and it is found by differentiating with respect to $\boldsymbol{w}$ and b, imposing stationarity. The dual form is also necessary to introduce the use of kernels and the is presented as:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{l} \boldsymbol{\alpha_i} - 0.5 \sum_{i,j=1}^{l} y_i y_j \boldsymbol{\alpha_i} \boldsymbol{\alpha_j} < \boldsymbol{x_i} \cdot \boldsymbol{x_j} >$$

Considering the linearly separable data, and suppose the Lagrangians $\boldsymbol{\alpha}^*$, where $*$ stands for a generic solution, solve the quadratic optimization problem, the standard SVM dual is defined as:

$$\begin{aligned} &maximise \;\; \boldsymbol{W}(\boldsymbol{\alpha}) = \sum_{i=1}^{l} \boldsymbol{\alpha_i} - 0.5 \sum_{i,j=1}^{l} y_i y_j \boldsymbol{\alpha_i} \boldsymbol{\alpha_j} < \boldsymbol{x_i} \cdot \boldsymbol{x_j} > \\ &\text{subject to} \sum_{i=1}^{l} y_i \boldsymbol{\alpha_i} = 0, \\ &\qquad\quad \boldsymbol{\alpha} \geq 0, i = 1..., l \end{aligned} \tag{1}$$

The weight vector $\boldsymbol{w}^* = \sum_{i=1}^{l} y_i \boldsymbol{\alpha_i^*} \boldsymbol{x_i}$ realizes the maximal margin hyperplane with geometric margin. As b do not appear in the dual problem, it is found by the primal constraints:

$$b^* = -\frac{max_{y_i=-1}(< \boldsymbol{w}^* \cdot \boldsymbol{x_i} >) + min_{y_i=1}(< \boldsymbol{w}^* \cdot \boldsymbol{x_i} >)}{2}$$

The Karush–Kuhn–Tucker Complementary Conditions (KKT) provides informations about the solution structure, and the KKT states that the optimal solution $\boldsymbol{\alpha}^*$, $(\boldsymbol{w}^*, b)$ that must be satisfied is:

$$\boldsymbol{\alpha_i^*}[y_i(< \boldsymbol{w}^* \cdot \boldsymbol{x_i} > +b^*) - 1 = 0, \quad i = 1, ..., l$$

Thus, the $\boldsymbol{x_i}$ instances that the functional margin is 1 and lie closest to the hyperplane are called SV, and the corresponding $\boldsymbol{\alpha_i^*}$ is non-zero. The optimal hyperplane dual representation, in terms of the parameters subset, is expressed as:

$$f(\boldsymbol{x}, \boldsymbol{\alpha}^*, b^*) = \sum_{i \in SV} y_i \boldsymbol{\alpha_i^*} < \boldsymbol{x_i} \cdot \boldsymbol{x} > +b$$

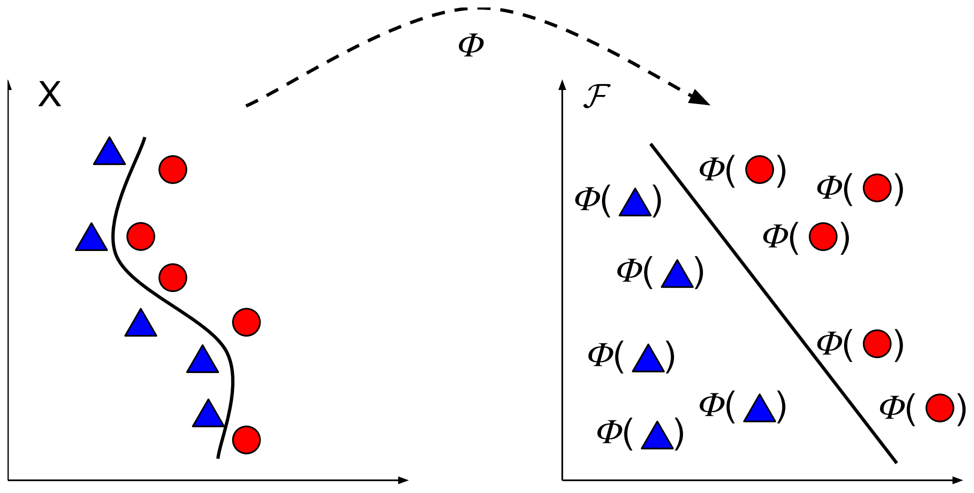
The Lagrange multipliers associated with each point become the dual variables, and points that are not SV have no influence in the final solution. A consequence of applying the KKT complementary conditions is that for $j \in SV$,

$$y_i f(\boldsymbol{x_j}, \boldsymbol{\alpha}^*, b^*) = y_j (\sum_{i \in SV} y_i \boldsymbol{\alpha_i^*} < \boldsymbol{x_i} \cdot \boldsymbol{x} > +b) = 1$$

therefore

$$< \boldsymbol{w}^* \cdot \boldsymbol{w}^* >= \sum_{i \in SV} \boldsymbol{\alpha_i^*}$$

Figura 4 – A feature map to simplify the classification task.



The geometric margin is now defined as:

$$\gamma = 1/||\boldsymbol{w}||_2 = (\sum_{i \in SV} \boldsymbol{\alpha}_i^*)^{1/2}$$

where in both dual objective and decision functions the instances only appear inside an inner product, which enables the kernel function use, i.e., the SVM nonlinearity is brought forth by the kernel function $\kappa(\boldsymbol{x}, \boldsymbol{x}^\top)$ that satisfies the distance relationship between transformed and original space, i.e., $\kappa(\boldsymbol{x}, \boldsymbol{x}^\top) = \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{x}^\top)$. The kernel function maps the training instances into some feature space $\mathcal{F} : \mathbb{R}^n \to \mathcal{F}$, and Fig. 4 illustrates the map transformation.

One popular kernel function is the Radial Basis Function (RBF), defined as:

$$\begin{aligned} \kappa(\boldsymbol{x}, \boldsymbol{x}^\top) &:= \exp(\frac{-||\boldsymbol{x} - \boldsymbol{x}^\top||^2}{2\sigma^2}) \\ &= \exp(\gamma||\boldsymbol{x} - \boldsymbol{x}^\top||^2), \qquad \forall \boldsymbol{h}, \boldsymbol{h'} \in \mathbb{R}^n \end{aligned} \tag{2}$$

where $||\cdot||$ represents the l_2 norm and (γ, σ) are fixed constants with the geometric margin $\gamma = \frac{1}{2\sigma^2}$. The decision rule is given by $sgn(f(\boldsymbol{x}))$, that is equivalent to the maximum margin separating hyperplane in the feature space is defined as $\boldsymbol{\omega}^\top \Phi(\boldsymbol{x}) + b = 0$. The kernel must satisfy Mercer's conditions, which is equivalent to requirement that the matrix with entries $(K(x_i, x_j))_{i,j=1}^l$ be positive definite for all training sets. Hence, the optimization problem is convex.

For the cases that there is no method to make the instances separable in the feature space, the soft margin optimization is introduced by the use of *slack variables* ξ, allowing the margin constraints to be violated:

$$\text{subject to } y_i(<\boldsymbol{w} \cdot \boldsymbol{x}> + b) \geq 1 - \xi_i, i = 1, ..., l$$
$$\xi_i \geq 0, \; i = 1, ..., l$$

The 2-norm soft margin contains the ξ_i scaled by the weight vector $\boldsymbol{w}$, suggesting that an optimal choice for C in the resulting optimization problem objective function should be R^2. Thus, assuming the restrictions introduced with the slack variable, the objective function is updated to:

$$minimize_{\xi,\boldsymbol{w},b} < \boldsymbol{w} \cdot \boldsymbol{w} > +C\sum_{i=1}^{l}\xi^2$$

Eq. 3 describes the primal form of quadratic programming problem that represents the nonlinear soft-margin SVM, and its minimization results are the maximum margin separating hyperplane.

$$\begin{aligned} \min_{\boldsymbol{\omega},b,\xi} \quad & \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to } & y_i(\boldsymbol{\omega}\cdot\Phi(\boldsymbol{x_i}) + b) \geq 1-\xi_i \\ & \xi_i \geq 0, \forall i \end{aligned} \tag{3}$$

where ξ is the slack variable that introduces the soft margin optimization (allowing the margin constraints to be violated), and the constant $C > 0$ is a trade-off hyperparameter. If $\xi < 0$, the first constraint will hold if $\xi_i = 0$, reducing the value of the objective function. Hence, the optimal solution can be obtained by removing the positivity constraint on ξ_i. The parameter C varies through a wide range of values, thus the norm $||\boldsymbol{w}||_2$ varies smoothly though a corresponding range. Thus, choosing a particular C corresponds to selecting a value for $||\boldsymbol{w}||_2$ (methods like cross-validation are usually applied), then minimizing $||\xi||_2$ for size $\boldsymbol{w}$. There are two dual form of the margin slack vector, that result in the soft margin algorithms.The 2-Norm Soft Margin (weighting the diagonal) is a dual form found by differentiating with respect to $\boldsymbol{w}$, ξ and b, imposing stationarity. The objective function is maximized over $\boldsymbol{\alpha}$, and KKT complementary conditions are applied. The feature space are implicitly defined by the kernel $\kappa(x,z)$, and parameters α^* solve the following quadratic optimization problem:

$$\begin{aligned} maximize \quad & W(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\boldsymbol{\alpha_i} - 0.5\sum_{i,j=1}^{l} y_i y_j \boldsymbol{\alpha_i}\boldsymbol{\alpha_j}(\kappa(\boldsymbol{x_i},\boldsymbol{x_j}) + \frac{1}{C}\delta_{ij} \\ \text{subject to } & \sum_{i=1}^{l} y_i\boldsymbol{\alpha_i} = 0 \\ & \boldsymbol{\alpha_i} > 0, \quad i = 1,...,l \end{aligned} \tag{4}$$

where δ_{ij} is the Kronecker δ defined to be 1 if $i = j$ and 0. The decision rule given by $sgn(f(x))$ is equivalent to the hyperplane in the feature space which solves the optimization problem, and the slack variables are defined relative to the geometric margin:

$$\gamma = (\sum_{i\in SV}\boldsymbol{\alpha}_i^* - \frac{1}{C} < \boldsymbol{\alpha}^* \cdot \boldsymbol{\alpha}^* >)^{-1/2}$$

The 1-Norm Soft Margin (box constraint) follow the same idea of the 2-Norm Soft Margin, with the difference that the constraint $C - \boldsymbol{\alpha_i} - r_i = 0$, with $r_i \geq 0$, enforces $\boldsymbol{\alpha_i} \leq C$, while $\xi_i \neq 0$ only if $r_i = 0$, and therefore $\boldsymbol{\alpha_i} = C$. The KKT conditions implies that a non-zero slack variables only occur when $\boldsymbol{\alpha_i} = C$. Using the same training sample and kernel already defined, and suppose α^* solve the following quadratic optimization problem, the 1-Norm Soft Margin SVM is defined as:

$$
\begin{aligned}
& minimize \quad W(\boldsymbol{\alpha}) = -\sum_{i=1}^{l} \boldsymbol{\alpha_i} + 0.5 \sum_{i,j=1}^{l} y_i y_j \boldsymbol{\alpha_i} \boldsymbol{\alpha_j} (\kappa(\boldsymbol{x_i}, \boldsymbol{x_j}) \\
& \text{subject to } \sum_{i=1}^{l} y_i \boldsymbol{\alpha_i} = 0 \\
& \forall i : \quad 0 \leq \boldsymbol{\alpha_i} \leq C, \quad i = 1, ..., l
\end{aligned} \tag{5}
$$

Using decision rule function, b^* is chosen so that $y_i f(\boldsymbol{x_i}) = 1$ for any i with $C > \boldsymbol{\alpha}^* > 0$. The decision rule is also given by $sgn(f(x))$, and the slack variables are defined relative to the following geometric margin:

$$
\gamma = (\sum_{i \in SV} y_i y_j \boldsymbol{\alpha_i^*} \boldsymbol{\alpha_j^*} \kappa(\boldsymbol{x_i}, \boldsymbol{x_j}))^{-1/2}
$$

The maximum margin separating hyperplane in the feature space is defined as $\boldsymbol{\omega}^\top \Phi(\boldsymbol{x}) + b = 0$. The Eq. 6 calculates the classification score regarding the distance from an instance $\boldsymbol{x}$ to the decision boundary.

$$
f(h) = \sum_{i=1}^{d} \boldsymbol{\alpha}_i y_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{6}
$$

To solve the SVM it requires a large memory and a high CPU power since the computational complexity of the SVM for n data points is $O(n^3)$. A solution to solve the iterative quadratic programming optimization inherent to SVM is the Sequential Minimal Optimization (SMO) algorithm (PLATT, 1998). It is derived by taking the idea of decomposition method to its extreme and optimizing a minimal subset of just two points at each iteration (CRISTIANINI; SHAWE-TAYLOR, 2000). The SMO jointly optimize analytically two elements, chosen by a heuristic, α_i and α_j at each step, since the condition from standard SVM implies that the smallest number of multipliers to be optimized at each step is two. The algorithm fixes all other hyperparameters, and the α vector is updated accordingly. For theoretical properties, extensions and experimental details, refer to (FAN; CHEN; LIN, 2005) and (CHEN; BUJA, 2006) work. The work of (FAN; CHEN; LIN, 2005) concludes that the methods to choose the working set with either first or second order approximation leads to faster convergence (fewer iterations) than using a conventional SMO algorithm. Also, the second order approximation shows that it is better than the original SMO considering time and iterations (FAN; CHEN;

LIN, 2005), this way it is the standard training method of LIBSVM library (CHANG; LIN, 2011a). It is clear that the choice of α is essential to the SMO algorithm.

The SVM with a kernel (as the Gaussian or RBF) has two important hyperparameters that impact greatly in the performance of the learned model: the soft-margin C and the kernel hyperparameter γ. Therefore, the application of the SVM with Gaussian kernel to a classification problem requires an appropriate selection of hyperparameters, called hyperparameter tuning or model selection. Given these two hyperparameters and a training dataset, an SVM solver can find a unique solution of the constrained quadratic optimization problem and return a classifier model. Unfortunately, there is no standard procedure for hyperparameter tuning, and a common approach is to split the dataset into training and validation set, and for each C and γ from a suitable set, select the pair that results in an SVM model that when trained on the training set has the lowest error rate over the corresponding validation set (WAINER; CAWLEY, 2017).

For the multiclass problem cases we can extend the SVM by reducing the classification problem with three or more classes to a set of binary classifiers using methods as the One-Vs-One (OVO), the One-Vs-Rest (OVR) (HSU; LIN, 2002), the ECOC model (DIETTERICH; BAKIRI, 1994), or the DAG model (PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000). The multiclass method choice does not interfere in the proposed solution, however, different methods may results in different classification outcomes.

2.2.1 SVM VARIANTS

Based on the SVM, other variations were proposed, and we highlight the most cited in the literature. The Least Squares Support Vector Machine (LS-SVM), proposed by (SUYKENS; VANDEWALLE, 1999), change the formulation to a set of linear equations and changes the inequality constraint to an equality. This way, the LS-SVM includes the following properties (SUYKENS; VAN GESTEL *et al.*, 2002):

- Choice of kernel function: the kernel function still have to be positive definite and satisfy Mercer condition.
- Global and unique solution: the dual problem of LS-SVM corresponds to solve a linear KKT system which is a square system with a unique solution when the matrix has full rank.
- The KKT system as a core problem: solving SVM classifier can be considered as iteratively solving KKT systems where each take a similiar form as one single LS-SVM classifier (NAVIA-VAZQUEZ *et al.*, 2001).
- Lack of sparseness and interpretation of support vectors: every data point is a support vector, thus no α_i value will be exactly zero. In the LS-SVM case all training data point will contribute to the model, and certain data points are more

important than other, in this manner, points with large $|\alpha_i|$ are located close and far from decision boundary.

- Non-parametric/parametric issues: have the same primal-dual interpretations as the SVM. In primal weight space, the problem is parametric with fixed size vector, and in the dual space the problem is non-parametric as the size of the solution vector $\alpha \in \mathbb{R}^N$ grows with the number of training data N.

As highlighted by (JOSHI; KALE, 2014), the LS-SVM advantage is, due to the equality constraints, a set of linear equations has to be solved instead of a quadratic programming problem. On the other hand, this approach is more suitable for small datasets. A toolbox develop by the authors is available at `http://www.esat.kuleuven.be/sista/lssvmlab/toolbox.html`.

The Lagrangian Support Vector Machine (LSVM) was developed for classification tasks (MANGASARIAN; MUSICANT, 2000) by reformulating the original SVM dual to an implicit Lagrangian version. This modification leads to the minimization of an unconstrained differentiable convex function in a space of dimensionality equal to the number of classified points. The LSVM is linearly convergent, however it requires the inversion at the outset of a single matrix. The authors proposes to use the Sherman-Morrison-Woodbury (SMW) identity (GOLUB; LOAN, 1996) to calculate the approximate inverse matrix.

The Proximal Support Vector Machine (PSVM) was proposed by (FUNG, G.; MANGASARIAN, O. L., 2001), where instances are classified by assigning to the closest of two parallels planes (in input or feature space) that are pushed apart as far as possible. The 2-norm of the error vector is minimized, and the margin between the bounding hyperplanes are maximized with respect to orientation and relative location to the origin, realizing into a convexity in the objective function. The constraints are equalities, enabling to write an explicit exact solution in terms of the problem data. This way, the hyperplanes are not bounded but proximal, clustering and pushing the classes as far apart as possible.

The Generalized Support Vector Machine (GSVM) (MANGASARIAN, 1998) creates a nonlinear separating surface using a completely arbitrary kernel. This method is a strongly convex minimization problem without constraints that has a unique solution, however, its objective function is not twice differentiable.

The Reduced Support Vector Machine (RSVM) (LEE; MANGASARIAN, 2001) generates a non-linear separating surface for a large dataset requiring a small portion of the respective dataset for its characterization. The problem is interpreted as possible instance-based learning where the small samples are learning from much larger training set by performing a rectangular kernel relationship between original and reduced sets. The RSVM is useful problems with many support vectors, and it was designed for large scale nonlinear kernel (JOSHI; KALE, 2014).

The Linear Programming Support Vector Machine (LPSVM) (FUNG, G. M.; MANGASARIAN, O., 2004) is a linear programming SVM formulation solved with a fast Newton method. The SVM formulation is changed to a linear programming form, generating very sparse solutions (BRADLEY; MANGASARIAN, 1998), thus it is suitable for feature selection in classification problems. The algorithm only requires a linear equation solver.

2.3 SVM LIMITATIONS

The SVM have a high computational cost during training phase, which makes it unpractical for situations which the data is too large, i.e., do not fit on available memory. When using a kernel, the SVM requires a model selection, that comprehends choosing a kernel function and a manual tune of the hyperparameters, and it implies in an additional RAM requirement during training phase. This limitation is related to the first research question: *How to make viable the use of an SVM classifier for large datasets, given that the computational complexity is $O(N^3)$, thus, the training procedure may be unpractical for datasets with a large number of instances and attributes?*.

The hyperparameters tuning may be time consuming, and there is no standard procedure. The most used methods are naive based (exhaustive search), heuristic methods (without convergence proof), and direct search (do not have a mechanism to escape from local minima). We relate the hyperparameters tuning limitation with the second research question: *How to tune the SVM hyperparameters in an efficient manner?*

The training phase also requires that all data is simultaneously accessed, in this way, the SVM is not suitable for scenarios where the data is partially available at a time. Considering the open-end scenario, where the data is temporally available and there is no prior knowledge about the incoming data stream end, the SVM will always increase the model size, and eventually it will reach the computer RAM memory limitations. This problem becomes worse when using the kernel, as the RAM consumption scale much faster. We relate this problem to our last question: *How to reduce the computational cost of the SVM adapting the optimization formula to deal with large and continuous data?*

3 SUPPORT VECTOR CANDIDATES SELECTION VIA DELAUNAY GRAPH AND CONVEX-HULL FOR LARGE AND HIGH-DIMENSIONAL DATASETS

3.1 INTRODUCTION

The SVM have an generalization capability for classification tasks (CRISTIANINI; SHAWE-TAYLOR, 2000), however, two disadvantages are well known: firstly, the model requires manual tunes and hyperparameters choices, which comprehends the selection of the kernel function and its tuning, and the choice of tolerance values. Secondly, the high computational cost during the training phase, especially with tasks that involve large-scale datasets. Despite the large-scale dataset be effective for training classifiers (WANG; NESKOVIC; COOPER, 2005), both the training process and the hyperparameters tuning procedure (which the latter require multiple training repetitions) may be challenging and time demanding. Among various methods to reduce the computational complexity of SVM training there is the SV candidates pre-selection (SHIN; CHO, 2007), as shortening the training set implies in speeding-up the training time. In this chapter we introduce a method to pre-select SV candidates based on the relationship between inputs depicted by a network representation. The dataset dimensionality is reduced to 3 dimensions, if necessary, and we describe the training dataset with a Delaunay graph (DELAUNAY, 1934). We create convex-hulls to each class of training dataset and select all points that belong to a convex-hull as SV candidates, as well as their neighbors from Delaunay graph.

From this chapter we created the following paper:

Reeberg de Mello, A.; STEMMER, M. R.; Oliveira Barbosa, F. G. Support vector candidates selection via Delaunay graph and convex-hull for large and high-dimensional datasets. Pattern Recognition Letters, North-Holland, v. 116, p. 43–49, dec 2018. ISSN 0167-8655.

3.2 LITERATURE REVIEW

The work of (NALEPA; KAWULOK, 2018) presents a review on selecting training sets for SVM, where the techniques to reduce the cardinality of a training set are categorized based on the underpinning optimization strategy: data geometry analysis and its subbranches clustering or non-clustering based, neighborhood analysis, evolutionary, active learning, and random sampling. Due to the page limit, we only reference the works that are most related to this paper.

The work of (WU; PHAM; NGUYEN, 2017) proposes a two-phase Two-Phase Sequential Minimal Optimization (TSMO) approach to scale down the training cost of large-scale data, and a two-phased-in-differential-learning particle swarm optimization (tDPSO) to ensure the accuracy of under-sampled data through parameter selection. The first phase is a supporting-vector-based working set selection named fast working set selection (FWSS), which uses only first-order gradient information. The second phase is the modification of Lagrange multipliers using a Second-Order Approximation (WSS3) algorithm (FAN; CHEN; LIN, 2005). Tests comparing TSMO, First-Order Approximation (WSS1) and WSS3 (last two from LibSVM (CHANG; LIN, 2011a)) present similar accuracy, despite the different number of Support Vectors (SVs) used for training in each case. Also, TSMO training time is significantly smaller, since there are fewer points to consider as SVs during SMO training process.

The work of (NALEPA; KAWULOK, 2016) proposes a new adaptive memetic algorithm (PCA^2MA), enhanced with a Principal Component Analysis (PCA) based preprocessing, to select valuable SVM training data. It starts with a population of reduced training sets undergoes the evolution, which is complemented by refinement procedures, exploiting both training set (a priori) information and the knowledge attained dynamically during PCA^2MA execution to enhance the refined sets. The authors also introduce a new adaptation scheme to control the pivotal algorithm parameters on the fly, based on the current search state. PCA^2MA central features include: preprocess all dataset using PCA to exploit geometrical properties and to extract potentially valuables vectors before the evolutionary optimization, and create an initial population and guide the evolution effectively towards high-quality refined training sets. The adaptation scheme has no essential parameters, which contour the drawback present in other adaptation procedures. The reported experiments show efficacy and convergence capabilities, especially compared with others evolutionary algorithms approach.

The work of (SHEN *et al.*, 2016) proposes a clustering-based training set selection for SVM. The method explores clustered and scattered data points in a cluster, in which dense points that lie around the clustering centroid, i.e., are in the inner layer, are removed. Scattered data points are usually sparse and located in an outside layer, thus are reserved. The boundary between clustered and scattered data in each cluster is set using the Fisher Discriminant Ratio, which calculates the distance densities of data points to the cluster centroid. Lastly, redundant clustered data points are removed to speed up SVM training process. Experimental results, on balanced datasets, show a significant reducing on training time without degrading classification accuracy.

The work of (GUO; BOUKIRA, 2015) introduces a method to select support vector candidates based on ensemble learning, that consists in applying a random forest algorithm followed by an ensemble-method called *Small Votes Instances Selections* (SVIS). The method uses an unsupervised margin concept, introduced by (SCHAPIRE *et al.*, 1997), that combines the first and second most voted class labels under the model. The bottleneck is the memory cost, which is $O(N \times T)$, where N is the number of inputs and T is the number of trees in the ensemble. Tests show a training time 30 to 40 times faster using SVIS when comparing to traditional training using raw dataset, with a result degradation under 5%.

To preselect support vector candidates for large-scale character recognition, (GOTO; ISHIDA; UCHIDA, 2015) proposed the use of a relative neighborhood graph (RNG). The method analyzes data distribution via network representation to preselect SV candidates, where the RNG represents the entire training dataset. It selects boundary pattern nodes (called *bridge vectors*) to be SV candidates by looking for edge connections between different class patterns. The modification proposed to construct RNG scales the graph creation to $O(N^2)$. The SV preselected with RGN coincide well with original SV, so SVs candidates selected by RNG reduces the training data to 10% and accelerates the SVM training process from 5 to 15 times without degradation (GOTO; ISHIDA; UCHIDA, 2015).

The work of (LÓPEZ CHAU; LI; YU, 2013) proposes a convex-concave hull for SVM classification, firstly introduced in (LOPEZ-CHAU; LI; YU, 2012). The algorithm looks for points that are on the outer boundaries of the set. It starts using (JARVIS, 1973) march method to compute the convex hulls, and from the convex hulls' vertices, it creates concave hulls. The authors also introduce a grid method based on binary trees to preprocess the dataset, considering the leaves as versions of the original points, which makes possible to look down from a certain height of the tree and take all leaves as subsets. The grid process contains four parameters that lead to a trade-off: higher binary trees leads to a finer quantization, but increase the amount of memory required to store the grid. Another consideration is that the convex-concave hull searching algorithm calculates angles, so it works fine in a two-dimensional space, and in case of higher-dimension datasets, a dimension reduction technique is needed. To compute the convex hull in more than three dimensions, the problem search convex-concave hull's vertices in several two-dimensional spaces subsets, and in the end get back to original aspect concerning fixed center values of each subset. Results show that the training time is decreased considerably, while accuracy is similar to classic SVM.

Also based on convex hull vertices selection within each class, (DI WANG *et al.*, 2013) proposes the vertices selection online SVM (VS-OSVM), that is composed of two steps: i) the sample selection process which selects points from a small number of skeleton samples constituting an approximate convex hull in each class of the current training set, and ii) the online updating process, in which the classifier is updated with newly arriving and selected skeleton samples. The first $d+1$ (where d is the dimension of the input samples) selected samples are the vertices of the convex hull, and it updates the classifier if the newly arrived data with distances to current classifier are within a given threshold. The initial data is randomly chosen with a size of 50% of the dataset size. The authors remark that the method is not suitable to be directly applied to datasets with heavy noise since it is based on vertices selection of the convex hull of the samples in each class. Experimental presents that the method does not compromise classification accuracy, but considerably decrease SVM training time.

(KAWULOK; NALEPA, 2012) introduces a method for selecting valuable training dataset for SVM for large, noisy sets using a genetic algorithm. The technique relies on analyzing the geometric properties of the data or adapts a randomized selection. However, it has a high computational cost for large sets (more than 4000 samples).

The work of (LOPEZ CHAU *et al.*, 2010) proposes a non-convex hull set strategy that envelops elements with the same label to select SV candidates. Instead of using a convex-hull approach, which only considers points located on the boundaries, non-convex hull methods add more points, therefore are less probable to miss support vectors, as it includes points near borders. The proposed algorithm uses Jarvi's approach to find boundary points while K-Nearest Neighbor provides with local candidates (LOPEZ CHAU *et al.*, 2010), implemented in parallel to accelerate training time. The selected support vector candidates from the proposed method give a similar accuracy to the whole dataset, but with a fraction of training time.

The work of (CERVANTES *et al.*, 2008) proposed a four-step SVM classification for large datasets via minimum enclosing ball clustering (MEB). The first step consists in clustering the data using MEB with a $(1+\varepsilon)-$approximation with a random sampling method to generate l ball centers with a radius r. The distance between each point and its belonging ball center must be less than r. The last clustering step consists in to perform for all l, and if there is any point outside all balls, increase the radius and repeat the process. The clustering process results in three ball categories: positive, negative, and one mixed samples. The second stage of SVM classification is the training, that is, find the support vectors considering only the balls center. Next step is the de-clustering, which consist in retrieving all points that belong to centers that are support vectors from the first SVM training. The last step is to retrain a new SVM considering all data retrieved from the third step. The proposed method has shown to be as fast as possible depending on the accuracy requirement, resulting training data size is considerably smaller than original. Memory space, algorithm complexity and training time are smaller when compared to a regular SVM, without decreasing classification accuracy.

The work of (WANG; SHI, 2008) presents a sample reduction technique based on data structure analysis (SR-DSA) to improve SVM scalability. The algorithm has three steps: i) find the data structures for positive and negative class independently, where it defines a data structure as units in which the data points are considered to share the same dispersion. A hierarchical clustering (Ward-linkage) is used to detect the clusters in each class, and it can be performed in either linear or nonlinear SVM. The clustering output is a dendrogram, which topology also represents the clustering process, so the number of clusters is chosen by selecting dendrogram's point of maximum curvature. ii) remove interior samples in each cluster by selecting points that are smaller than a threshold, which is related to Mahalanobis distance between the points and its respective cluster center. iii) for each cluster, remove exterior samples that are distant from opposite class. The remaining data are used to train the SVM. SR-DSA can also be used in multiclass classifiers using the DAG combination. Results show no degradation on classification accuracy on binary datasets.

The work of (ZENG *et al.*, 2008) proposes a method to select convex hull samples in the feature space (SebSVM) for SVM training with linear time complexity. It applies a kernel function to map training set into a higher space, followed by creating two balls of minimum radius. Samples that lie on or close to the balls' boundary are selected to train the SVM. Experimental on low dimensional datasets (smaller than 54 dimensions) using a Gaussian kernel shows a significant decrease in training time, preserving the generalization performance.

A neighborhood property-based pattern selection (NPPS) algorithm is proposed by (SHIN; CHO, 2007) to select patterns located near decision boundaries. The neighborhood property explored is that patterns situated near the decision boundary tend to have more heterogeneous neighbors in its class memberships. An entropy concept is utilized to measure the heterogeneity of class labels among k-nearest neighbors, which leads to estimate the proximity. Tests comparing performance over NPPS, random sample set (RAN) and all dataset show that the accuracy of NPPS preselected SVs is better than RAN, and similar to all dataset, despite the number of patterns selected from NPPS be around 5% of all data, which reduces training time proportionally.

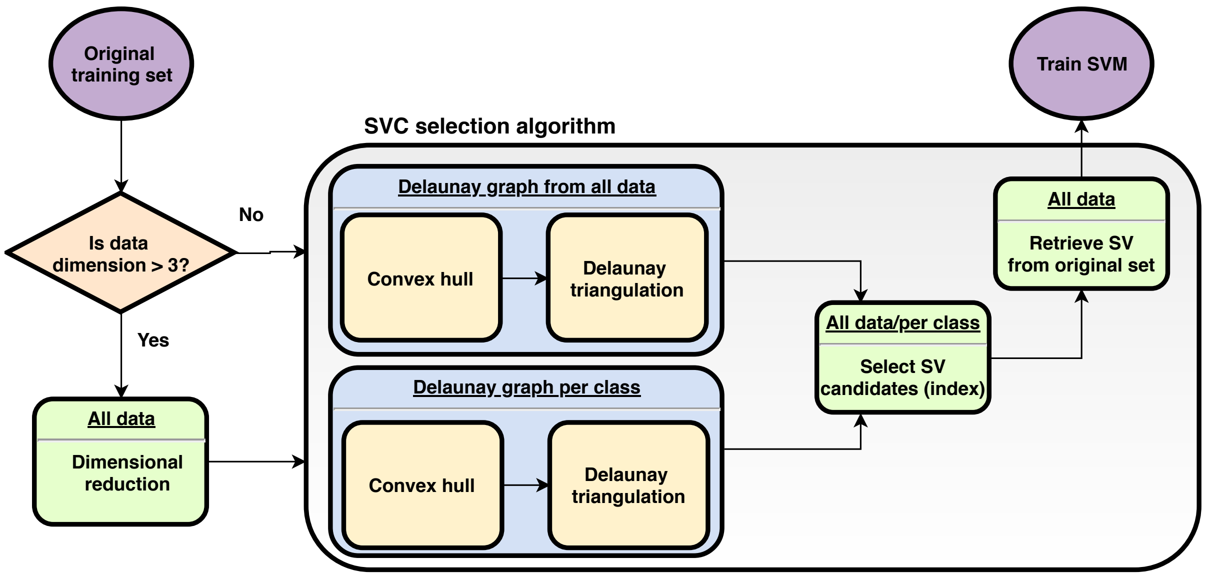
The work of (SCHOLKOPF; BURGES; SMOLA, 1999) introduces three new concepts; *shrinking* the SVM by selecting a working set among training data, *caching* to improve computational efficiency, and *incremental updates* of the gradient. To select a feasible working set, the first-order approximation to the target function is used, finding the steepest feasible direction of descent which has only non-zero elements (which will compose the working set), and (SCHOLKOPF; BURGES; SMOLA, 1999) was one of the first works to make reasonable the use of SVM on large-scale datasets.

3.3 PROPOSED SOLUTION

The proposed solution to determine SV candidates consists in finding the relationship between Delaunay graph neighborhood and vertices located in the convex-hull of each class. There is a relationship between SV points and different class neighbor points in the Gabriel Graph (WAN ZHANG; IRWIN KING, 2002), and as Delaunay graph is a sub-graph of Gabriel graph (URQUHART, 1980), points that belong to the convex-hull of each class are likely to be SVs. Figure 5 presents the flowchart of the subsequent processing steps for the proposed method.

The first step is to check the data dimension, and in cases of bigger than three, it requires a dimensional reduction technique. Formally, for any convex set P with n points in p dimension, it is possible to create a lower dimension dataset P' with n points in q dimension, where $q \leq 3 \leq p$. The dimensionality reduction technique influences the

Figura 5 – Flowchart of subsequent processing steps.



proposed algorithm outcome because each method results in a different distribution in space $\mathbb{R}^3$. The dimensional reduction techniques were chosen based on computational cost, which implies in a shorter processing time overall. Among all existing techniques, we recommend the use of linear methods as the PCA introduced by (PEARSON, 1901), and the Linear Discriminant Analysis (LDA) proposed by (FISHER, 1936). For comparison purpose we use two versions of the same nonlinear dimensionality reduction technique, based on the dataset size: for datasets smaller then 5000 inputs we apply the classical t-Distributed Stochastic Neighbor Embedding (T-SNE) (MAATEN; HINTON, 2008), and for more massive datasets we use the accelerated T-SNE (MAATEN, 2014). All implementations were adopted by the same author (MAATEN; MAATEN *et al.*, 2009),(MAATEN; HINTON, 2008), and (MAATEN, 2014)[1]. For the PCA, the worst complexity case is $O(D^3 + n^{3/2+1})$, and when points are uniformly distributed, is $O(D^3 + n^{1+1/3})$; while for the LDA, the worst case is $O(nD + nt + Dt + n^{3/2+1})$, and for uniformly distributed points $O(nD + nt + Dt + n^{1+1/3})$ (FISHER, 1936), (MAATEN; MAATEN *et al.*, 2009). The computational and memory complexities of regular T-SNE are $O(n^2)$, while the accelerated T-SNE run in $O(nlogn)$ and require $O(n)$ memory ((MAATEN, 2014)).

The proposed method selects SV candidates based on the Delaunay Graph (DG) neighborhood in 2 or 3 dimensions (thus the method runs in reduced space for datasets with dimension bigger than three), so we first need to create those Delaunay Graphs (DGs). There are two approaches to construct a DG: extract it from other graphs, because DG is a subgraph of *Relative neighborhood graph* $\subseteq Urquhart \subseteq Gabriel \subseteq Delaunay$ (URQUHART, 1980); or create a DG through the Delaunay triangulation, which is our choice due to computational and memory complexity. The convex hull of a

[1] available at `https://lvdmaaten.github.io/software/#dimensionality-reduction`

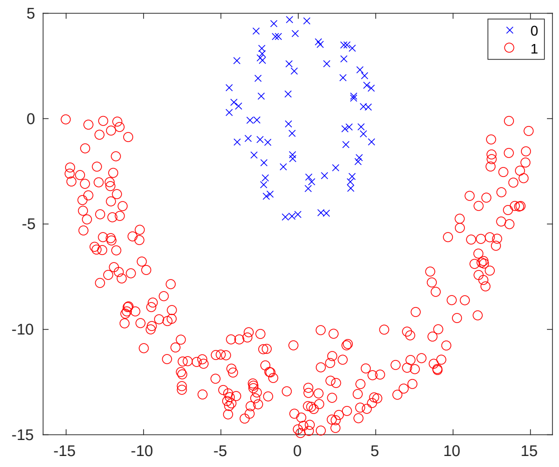
set P is a convex polytope, which is used to solve the Delaunay triangulation (BARBER; DOBKIN; HUHDANPAA, 1996), (DEVILLERS; HORNUS; JAMIN, 2017), which in turn a graph is a Delaunay graph if it is the Delaunay triangulation of some set of points in the plane. In this manner, to create a DG, we primarily extract the convex hull of the desired set, which in case it could be all data or by class. Convex-hull is defined as: given a finite set $A \subseteq \mathbb{R}^n$, where n is the dimension, for l points $p_1, ..., p_l$ of a subset $P \subseteq A$, and to each point p_l is assigned a non-negative coefficient λ_l which sums to one, equation 7 formally describe each subset.

$$Conv(P) = \left\{ \sum_{i=1}^{l} \lambda_i p_i | l \in \sum_{i=1}^{l} \lambda_i = 1 \wedge \forall i \in \{i, ..., l\} : \right. \\ \left. \lambda \geq 0 \wedge p_i \in P \right\} \tag{7}$$

We use the Quick hull (*Qhull*) algorithm ((BARBER; DOBKIN; HUHDANPAA, 1996)), that is currently natively supported by Matlab, where *Qhull's* conjecture is: let l be the number of input points in R^n, and v the number of output points. If the input points precision is $O(logl)$, the worst case complexity is $O(nlogv)$ for $n \leq 3$ and $O(nf_v/v)$ for $n \geq 4$, that leads the algorithm to work efficiently in dimensions from 2 to 8. (K. Q. BROWN, 1979) discovered an intimate relationship between convex hull and Delaunay triangulation. Given a set P of n points in plane $z = 0$, the points are projected onto the unit elliptic paraboloid $z = x^2 + y^2$ to yield a point set P' such that for each $p = (p_x, p_u) \in P$ a point $p' \in P$ is computed as $p' = (p_x, p_y, p_x^2 + p_y^2)$. The $CH(P')$ contains every $p' \in P'$. Projecting the downward-facing facets, which are normal vectors with a negative z-value, in $CH(P')$ onto $z = 0$ yields the Delaunay triangulation of P ($Dt(P)$) (FISHER *et al.*, 2004), (EDELSBRUNNER; SEIDEL, 1986).

The complex $Del(P)$ is the Delaunay triangulation of the convex hull of P ((GALLIER, 2008)), so Dt ($Del(P)$) is formally defined as: let $P = \{p_1, ..., p_l\}$ be a set of l points in $\mathbb{E}^m$, and consider the Voronoi diagram of P $Vor(P)$. The complex $Del(P)$ contains k-simplex $\{p_1, ..., p_{k+1}\}$ iff $V_1 \cap ... \cap V_{k+1} \neq \emptyset$, where $0 \leq k \leq m$. For any convex set $P \in \mathbb{R}^2$, Delaunay graphs are the dual of Voronoi diagram of a set of points concerning the convex distance function defined by P ((BONICHON *et al.*, 2010)). Expanding this concept for any convex set in any dimension $P \in \mathbb{R}^n$, the Delaunay graph can be created provided that the convex distance function is also in n dimensions, as stated by (DEVILLERS; HORNUS; JAMIN, 2017) in CGAL library, where the complexity is related to dimension. The Delaunay triangulation algorithm complexity for non-spatially sorted data is $O(l^{\frac{n}{2}+1})$ (BOISSONNAT; DEVILLERS; HORNUS, 2009), therefore, to create Delaunay graphs for large sets in a reasonable time complexity the dimensions is limited to 3 ($p' = 3$). The Delaunay triangulation implementation used is from the computational geometry algorithms library (CGAL) (DEVILLERS; HORNUS; JAMIN,

Figura 6 – 2D two-class toy dataset.



2017), using (BOISSONNAT; DEVILLERS; HORNUS, 2009) approach, which is also available in Matlab. For the worst case scenario (without spatial sort), the complexity is $O(l^{\frac{d}{2}+1})$, and when points are uniformly distributed, the localization complexity is $O(l^{\frac{1}{n}})$, and the size of triangulation is $O(n)$, resulting in a complexity of $O(l^{1+\frac{1}{n}})$. Thus, with spatial sort and random points, the expected complexity is $O(llogn)$.

To summarize DG creation, for any dataset P, considering all convex-hull subsets $P_t \subseteq P'$ and the Delaunay graph of $Dg(P')$, points p_i that are vertices of the Delaunay graph and belongs to any Convex hull subsets $p_i \in Dg(P') \cap P_t$, and its t neighbors, form the set of SV candidates $P_{svq} \subset P'$ (in q dimension). Two observations are relevant about the chosen method: (i) after creating the DG of a set we already have the convex-hull information, and (ii) by creating a DG from a set P and others subsets $P' \subseteq P$, we can map the interactions between those subsets.

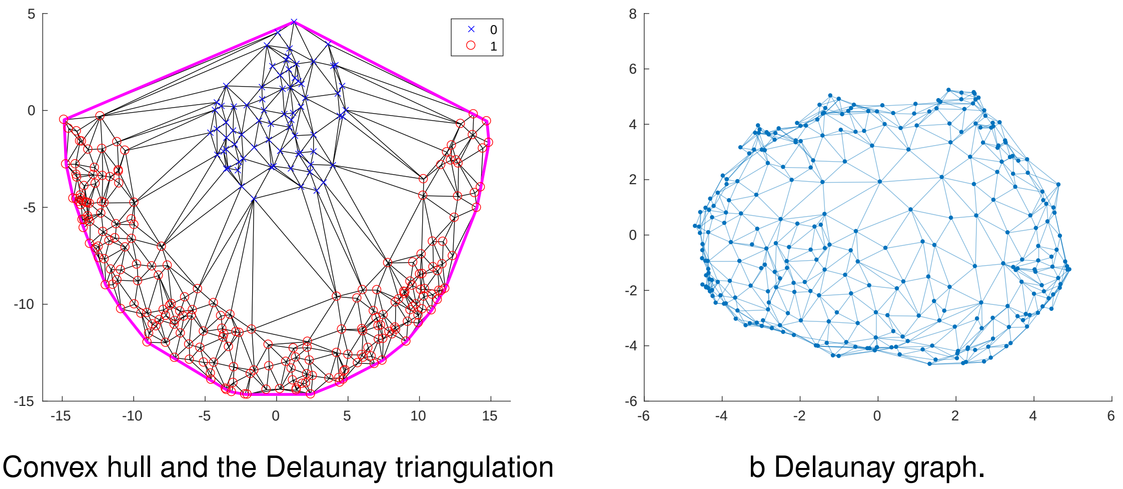
Figure 5 shows that SV candidates selection algorithm has two initial parallel branches, where one acts in all data, and the other in each class. Therefore, we create a DG using all data to map the interaction between all nodes and classes, and a DG to each class for mapping the interaction between points inner class. To help visualize the process, we present a 2D two-class toy dataset (figure 6).

Figure 7a presents the convex hull (magenta line) and the Delaunay triangulation (black lines) considering all data, and figure 7b shows resultant DG.

To create a DG for each class, we repeat the process, however considering each one as a set. Figure 8 shows the convex-hull (dashed lines) and the Delaunay triangulation (thinner and lighter lines) by class.

To find the nodes that are in the boundary between classes we use a convex hull in each class 8, and all nodes that are in the convex hull of a class and are at least one 1-hop neighbor from a different class are considered main nodes. All main nodes and its t-hops neighbors are considered SV candidates, so larger t implies in more candidates. In figure 9a we identify both classes in all data DG, and the black dots are

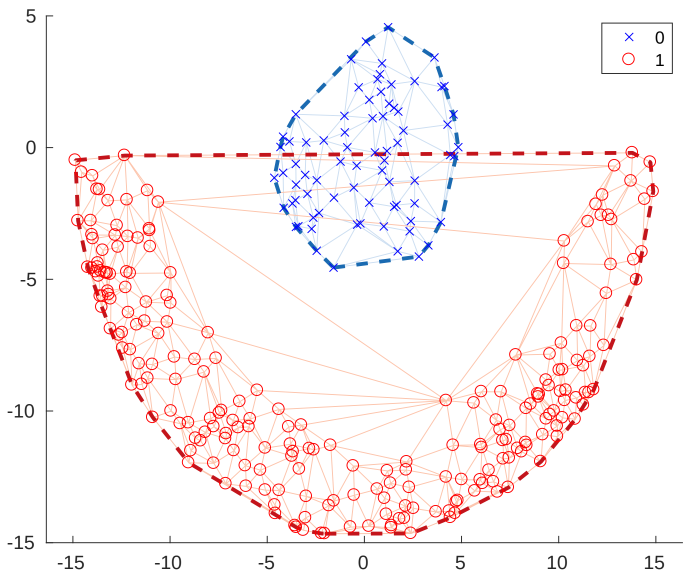
Figura 7 – The steps to create a DG considering all data as one set.



a Convex hull and the Delaunay triangulation

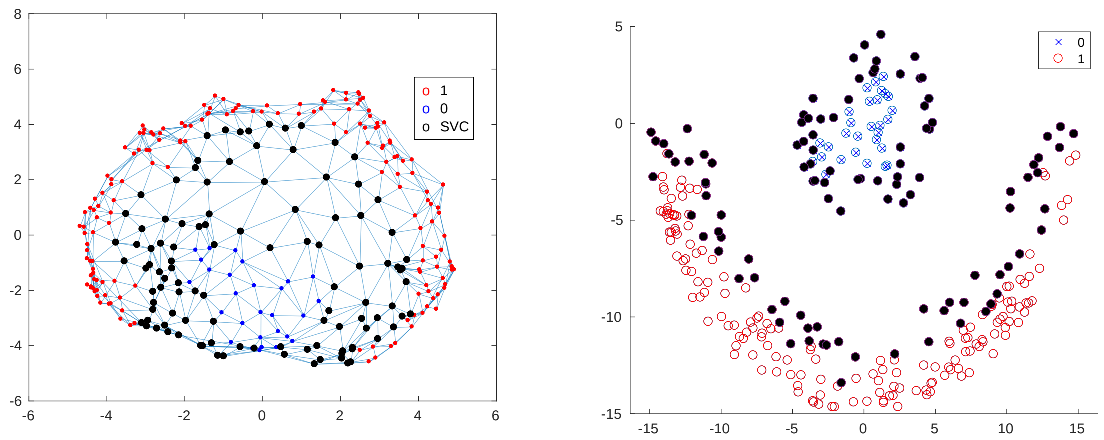
b Delaunay graph.

Figura 8 – Each class convex hull and Delaunay triangulation.



the SVC for $t = 2$, that is, we consider the convex-hull points and their neighbors. The last step is to retrieve the SVC index from DG to original space, as presented by figure 9b, where the black dots are the SVC in original space. The computational complexity of the proposed method is the sum of dimensionality reduction, Qhull and Delaunay triangulation.

Figura 9 – Support vector candidates in black.



a Nodes with different class neighbors and its 1hop neighbors in black.

b Delaunay graph.

3.4 EXPERIMENTAL METHOD AND RESULTS

The experimental method is composed of: dataset selection, dimensionality reduction, SV candidates set creation, SVM training, and validation. The computer used is an Intel Core i7-7700HQ, with dual-channel memory 16384 MB, 1300 MHz, and datasets loading time was not considered.

We used the following datasets in the experiments[2], summarized by table 2:

- MNIST dataset (LECUN *et al.*, 1998): it is composed of 28×28 images (with 784 attributes each) that represent handwritten digits from 0 to 9 in a binary form; therefore it is a multivariate class dataset. The training set has 60000 examples, and the test set has 10000 examples.

- Letter Recognition dataset (FREY; SLATE, 1991): it is made of black-and-white rectangular pixel that displays the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20000 unique stimuli. Each stimulus was converted into 16 numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. Training is performed on the first 16000 items, while the testing on the remaining 4000.

- Covertype dataset (BLACKARD; DEAN, 1999): the dataset was created to predict the forest cover type from cartographic variables. The actual forest cover type for a given observation ($30x30$ meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables

[2] available at UCI Machine Learning Repository (LICHMAN, 2013)

were derived from data originally obtained from US Geological Survey (USGS) and USFS data. Data is in raw form (not scaled) and contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types). The dataset is imbalanced and has 54 attributes and 581012 instances, divided (in a stratified manner) in 389279 for training and 191733 for testing.

- Skin and non-skin dataset (BHATT *et al.*, 2009) (Skin): randomly sampled RGB values from faces of various age groups (young, middle, and old), race groups (white, black, and asian), and genders, obtained from FERET database (PHILLIPS *et al.*, 1998) and PAL database (MINEAR; PARK, 2004). The dataset is univariate (skin or non-skin), and has 245057 RGB-dimensional training examples.

Tabela 2 – Training and test sets size (*No. of samples* × *No. of attributes*)

Dataset	**Training**	**Testing**
MNIST	60000×768	10000×768
Letter	16000×16	4000×16
Covertype	38927989×54	191733×54
Skin	164189×3	80868×3

For the datasets with dimension bigger than three (MNIST, Letter Recognition, and Covertype) the PCA, LDA and aTSNE dimensional reduction techniques are applied, reducing the dimension to 3, besides the label. Table 3 summarizes the processing time for each dataset.

Tabela 3 – Time (in seconds) to compute PCA, LDA, and accelerated TSNE

	PCA(s)	**LDA(s)**	**aTSNE**
MNIST	0.49	2.60	2398.62
Letter	0.19	0.03	488.92
Covertype	0.10	0.24	15894.83

We create SV candidates sets for all datasets using the proposed method, and for those with dimension bigger than 3, we create an SV candidates set for each dimensionality reduction technique described. The SV candidates set for the Skin dataset contains 16075 candidates, and the processing time took 13.55 seconds. Table 4 presents the number of elements per SV candidates set for the remaining datasets.

Tabela 4 – Number of elements per SV candidates set in 3D and its processing time

	PCA	**PCA(s)**	**LDA**	**LDA(s)**	**aTSNE**	**aTSNE(s)**
MNIST	49806	18.63	41607	19.30	28325	18.60
Letter	6998	5.00	7165	5.07	11106	5.44
Convertype	324547	126.37	351266	119.05	305076	126.13

We train all SVM models using SMO algorithm with second-order approximation and radial basis function (RBF) kernel. We use Matlab's built-in functions to perform the experiments: *fitsvm* to create a model for binary classification with a Matlab automated function to optimize the hyper-parameters from SVM and RBF ($exp(-\gamma * |u - v|^2)$) on a single thread; and *fitecoc* to create K binary models for multiclass classification using an one-versus-all (OVA) strategy with error-correcting output codes (ECOC) (FÜRNKRANZ, 2002) (ESCALERA; PUJOL; RADEVA, 2009) (also using an automated function from Matlab to optimize the hyper-parameters), and parallel computing).

To evaluate the performance over the datasets, table 5 show the results of the hold-out validation method using the training and test sets described in table 2.

Tabela 5 – Training and testing time, and accuracy percentage

	Training(s)	**Testing(s)**	**Accuracy(%)**
MNIST-original	1850.70	111.25	96.84
MNIST-PCA	1207.44	101.59	96.46
MNIST-LDA	675.54	87.71	96.54
MNIST-aTSNE	370.21	108.55	96.50
Letter-original	36.14	1.15	96.30
Letter-PCA	18.22	0.63	93.38
Letter-LDA	18.43	0.57	93.28
Letter-aTSNE	28.66	1.41	96.25
Covertype-original	12917.73	1263.17	86.05
Covertype-PCA	9597.93	1137.27	85.84
Covertype-LDA	10231.07	1170.40	85.28
Covertype-aTSNE	9881.28	1179.17	86.33
Skin-original	11040.48	11044.94	81.10
Skin-SVC	8906.69	8906.80	92.56

The RNG based method of (GOTO; ISHIDA; UCHIDA, 2015) applied on MNIST resulted in an SV candidates set of 5313 inputs, which is less than 10% of the original dataset, with a processing time of 7.61×10^2 seconds. Compared to our method, we have a bigger SV candidates set, however the processing time to generate it is much smaller when considering linear dimensionality reduction methods. (GUO; BOUKIRA, 2015) method SVIS (Small Votes Instance Selection) showed a reduction of 91% in the letter dataset with a training set of 10000 samples, achieving the same accuracy as our method. SVIS have an execution time much smaller ($0.44s$) compared to our proposed solution, however, it was only tested in small datasets (Letter dataset was the largest) with low dimension (up to 64). (SHEN *et al.*, 2016) applied their method (redundant data reduction) in a binary classification version of Cover type, and despite not presenting the SV candidates generation time, the training size was reduced from 570000 to 334891 samples, which is more or less same as ours. The PCA^2MA of (NALEPA; KAWULOK, 2016) shrunk the Skin dataset in a range from 61.1 to 89.3 %

(with a mean of 70.8%), hence, our method reduces consistently 89.7%, when using 1-hop neighbors.

3.5 SUMMARY AND SYNTHESIS OF CONTRIBUTION

In this chapter, we propose a method for preselecting SV candidates by convex-hull and Delaunay graph. The technique preselects SV candidates that are in the convex-hull of its class and its neighbors from the Delaunay graph, and the SV candidates size is related to the Delaunay graph connections density from points that belong to the convex-hull of each class from the dataset. The dimensional reduction is necessary when dimensions are bigger than 3, and more sophisticated dimensionality reduction methods may improve the results. One of our main contributions is the fact that the proposed method works in a reduced space. Preselecting SV coincides well with original SVs, as the accuracy of tested datasets does not decrease significantly. The developed system is efficient, with a time complexity of $O(n^{3/2+1})$ to $O(n^{1+1/3})$ plus time complexity of dimensionality reduction technique. It is possible to extend the proposed method by substituting the Delaunay graph by its sub-graphs, resulting in a higher time complexity and smaller SV candidates set. The dimensional reduction techniques smooth eventual noises present in the dataset, and distinct dimensionality reduction techniques produce different SV candidates set. The most apparent finding to emerge from this study is that the SVM training time is accelerated proportionally to the size difference of the original dataset and SV candidates set. Finally, we validate the contributions through experiments over four datasets.

The contributions of this chapter are:

- The pre-selection of SV candidates results in a reduction of the training dataset, that is related to the density of connections in the Delaunay graph from points that are in the convex-hull of each class; consequently, the SVM training time is proportionally faster.
- It can be performed in a reduced space.
- The proposed method works with imbalanced dataset distribution.
- We got consistent results for different size and dimension datasets.

4 A NOVEL ORTHOGONAL DIRECTION MESH ADAPTIVE DIRECT SEARCH APPROACH FOR SVM HYPERPARAMETER TUNING

4.1 INTRODUCTION

The application of the SVM with Gaussian kernel to a classification problem requires an appropriate selection of hyperparameters, called hyperparameter tuning or model selection. Given these two hyperparameters and a training dataset, an SVM solver can find a unique solution of the constrained quadratic optimization problem and return a classifier model. Unfortunately, there is no standard procedure for hyperparameter tuning, and a common approach is to split the dataset into training and validation set, and for each C and γ from a suitable set, select the pair that results in an SVM model that when trained on the training set has the lowest error rate over the corresponding validation set (WAINER; CAWLEY, 2017). The BBO is the study of the design and analysis of algorithms that assume that the objective and/or constraint functions are given by a black-box (**audet2017**). Some of the most used methods to solve BBO problems are: (i) the naive methods such as the exhaustive search, grid search, and coordinate search; (ii) the heuristic methods, such as the genetic algorithm (and its variations) and the NM search (NELDER; MEAD, 1965); and (iii) the direct search algorithms, such as the Generalized Pattern Search (GPS) and the MADS (**audet2006**). The naive and the heuristic methods do not guarantee convergence, while the direct search methods combine a flexible framework with proof of convergence (**audet2017**).

4.2 LITERATURE REVIEW

Considering the relationship between the data geometric structure in the feature space and the kernel function (that is not relevant to C), (JIANCHENG *et al.*, 2010) determine C and γ separately by two stages. (CHEN; FLORERO-SALINAS; LI, 2017) proposed another two-step procedure for efficient parameter selection by exploiting the geometry of the training data to select γ directly via nearest neighborhood and choosing the C value with an elbow method that finds the smallest C leading to the highest possible validation accuracy. This approach reduces the number of candidate points to be checked, while maintaining comparable accuracy to other classical methods. (CHUNG *et al.*, 2003) and (GOLD; HOLUB; SOLLICH, 2005) introduced the Bayesian Optimization (BO) approach for tuning the kernel hyperparameters, which constructs a probabilistic model to find the minimum function $f(x)$ on some bounded set $\mathcal{X}$, that exploits the model to make decisions about where in $\mathcal{X}$ to next evaluate the function while integrating out uncertainty (**snoek2012**). (ACERBI; JI, 2017) proposed the Bayesian Adaptive Direct Search (BADS) which combines a Bayesian optimization with the MADS framework via a local Gaussian surrogate process, implemented with a number of heuristics. The BADS' goal is fitting moderately expensive computational models, and it achieved state-of-the-art accuracy on computational neuroscience models (such as convolution neural networks). (CHANG; CHOU, 2015) proposed a two-stage method for hyperparameter tuning. The first step consists of tuning the γ using the generalization error of a k-NN classifier with the aim of maximizing the margins and extend the class separation. The second stage defines the value of C by an analytic function obtained with the jackknife technique.

Although many efforts to properly tuning the hyperparameters of a SVM with Gaussian kernel have been made, most of the BBO-based methods lack convergence proof. The BBO problems may present a limited precision or may be corrupted by numerical noise, which leads to an invalid output. Considering a fixed starting point, a BBO algorithm may provide different outputs, and there are unreliable properties frequently encountered in real problems (AUDET, 2014). Furthermore, most of BBO-based methods use the time or number of function evaluations as a stopping criterion, which creates an uncertainty on the achieved local minimum.

The Nonlinear Optimization by Mesh Adaptive Direct Search (NOMAD) software (LE DIGABEL; SÉBASTIEN, 2011) (COUTURE *et al.*, 2018) is a *C++* implementation of the MADS algorithm which can efficiently explore a design space in search of better solutions for a large spectrum of BBO problems as described by (AUDET, 2018), and inspired by cases of success as (ACERBI; JI, 2017) and (AUDET, 2018). We use the Ortho-MADS (**Abramson2009**) (that is a MADS improvement), with two different search strategies NM (NELDER; MEAD, 1965) and the VNS (AUDET; BÉCHARD; DIGABEL, 2008), to tune hyperparameters of a SVM with Gaussian kernel considering a dynamic stopping criterion. The proposed method relies on the MADS convergence properties and it combines different search strategies to reach the desired local minimum (that is set by the mesh size) and escape from undesired local minimum. The experimental results on benchmark datasets have shown that the proposed approach achieves state-of-the-art accuracy, besides presenting several interesting properties such as the guarantee of convergence and a dynamic stopping criterion. Furthermore, it also provides many other tools that aid to adjust the hyperparameters regarding the particularities of each application.

4.3 BASIC CONCEPTS

A practical difficulty in the nonlinear SVM is how to properly tune the hyperparameters γ and C from Eqs. 2 and 3 respectively. For the BBO-based methods that are not based on internal metrics, the tuning process attempts to maximize the accuracy on a training data usually using a cross-validation procedure, which is equivalent to minimizing a loss function ($\mathcal{L}$), as described by Eq. 8.

$$\max_{\gamma>0,\ C>0} \text{accuracy}(\gamma, C) \quad \equiv \quad \min\ \mathcal{L}(\gamma, C) \tag{8}$$

Almost all loss functions commonly used in the literature met convexity assumption, however, different loss functions lead to different theoretical behaviors, i.e., different convergence rates (ROSASCO *et al.*, 2004). It is possible to use any loss function that meets convexity with the proposed approach, but based on (ROSASCO *et al.*, 2004) and considering the loss function options for classification tasks, we have chosen the Hinge loss (HL). The HL leads to a convergence rate practically equal to the convergence rate of a logistic loss and better than the convergence rate of a square loss. However, considering a hypothesis space sufficiently rich, the thresholding stage has little impact on the obtained bounds. We define the weighted average classification hinge loss to tune the hyperparameters of the SVM as Eq. 9.

$$\mathcal{L} = \sum_{i=1}^{d} w_i \max\{0.1 - m_i\} \tag{9}$$

where w_i is the normalized weight for an observation i ($\sum_{i=1}^{d} w_i = 1$) and $m_i = y_i f(\boldsymbol{h_i})$ is the scalar classification score when the model predicts true for the instance $\boldsymbol{h_i}$.

We can formulate the problem of tuning the hyperparameters of the SVM with Gaussian as a BBO function $f(x) : \mathbb{R}^n \to \mathbb{R}$ in the sense that we obtain a function value from a given $x \in \mathbb{R}$ (in this case x corresponds to the hyperparameters C and γ) for which the analytic form of f is unknown. The objective function f and the different functions defining the set Ω are provided as a bounded optimization problem of the form:

$$\min_{z\ \in\ \Omega \subseteq \mathbb{R}^n} f(x) \tag{10}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\},\ \Omega = \{x \in X : c_j(x) \leq 0, j = 1, 2, \cdots, m\}$, and $X \in \mathbb{R}^n$ represents closed constraints which are bounded by $L \leq x \leq U$, with the lower (L) and the upper (U) bounds in $(\mathbb{R} \cup \{\pm\infty\})^n$, and the functions c_j represent the other m open constraints (AUDET; BÉCHARD; DIGABEL, 2008; ZEGAL; ESSADDAM; BRIMBERG, 2012).

Following we briefly introduce separately the methods that we propose for tuning the hyperparameters of a SVM with Gaussian kernel, which are the MADS and the *Ortho* variation, as well as the two search methods used with the Ortho-MADS: the NM search and the VNS.

4.3.1 Ortho Mesh Adaptive Direct Dual Search (Ortho-MAD2S)

In this section, we introduce the Ortho Mesh Adaptive Direct Dual Search (Ortho-MAD2S), which is a method for tuning the hyperparameters of an SVM with Gaussian kernel. We briefly introduce the methods that compose the Ortho Mesh Adaptive Direct Dual Search (Ortho-MAD2S)‘, which are: the MADS and the Ortho variation, the Nelder-Mead (NM) search, and Variable Neighborhood Search (VNS).

4.3.1.1 MESH ADAPTIVE DIRECT SEARCH (MADS)

The Mesh Adaptive Direct Search (MADS) (**audet2006**) is a direct search method to solve problems of the form Eq. 10, in which the goal at each iteration is to replace the current best feasible points (called incumbent solution, or the incumbent) by a better one. The algorithm starts from a finite collection of initial points $V_0 \in \mathbb{R}^n$, and the incumbent x_k at iteration k is defined as $f(x) : x \in V^k \cap \Omega$, where V^k is the set of trial points where the black-box was previously evaluated. To achieve a better solution at each iteration it uses a mesh M^k to generate trial points on a discretized space, defined by Eq. 11.

$$M^k := \{x^k + \delta^k D\, y : y \in \mathbb{N}^p\} \tag{11}$$

where δ^k defines the mesh size, D is the set of directions, and p is the number of directions. At iteration k, the algorithm attempts to improve the incumbent solution by executing the *search* or *poll* stage, allowing an attempt of improving the current solution or escaping from an undesired local minimum. The *search* stage attempts to improve the incumbent by evaluating points that are generated from a finite trial points of the subset M^k, based on the D set of directions, and that are not necessarily close to the current incumbent. When the *search* stage does not succeed, the *poll* stage is executed to evaluate points inside a frame defined by:

$$F^k := \{x \in M^k :\parallel x - x^k \parallel_\infty < \Delta^k b\} \tag{12}$$

where x is the point being evaluated, Δ^k is the frame size that must satisfy $0 < \delta^k \leq \Delta^k$, and b is defined by:

$$b = \max\{\parallel d' \parallel_\infty : d' \in D\} \tag{13}$$

where d' is a direction.

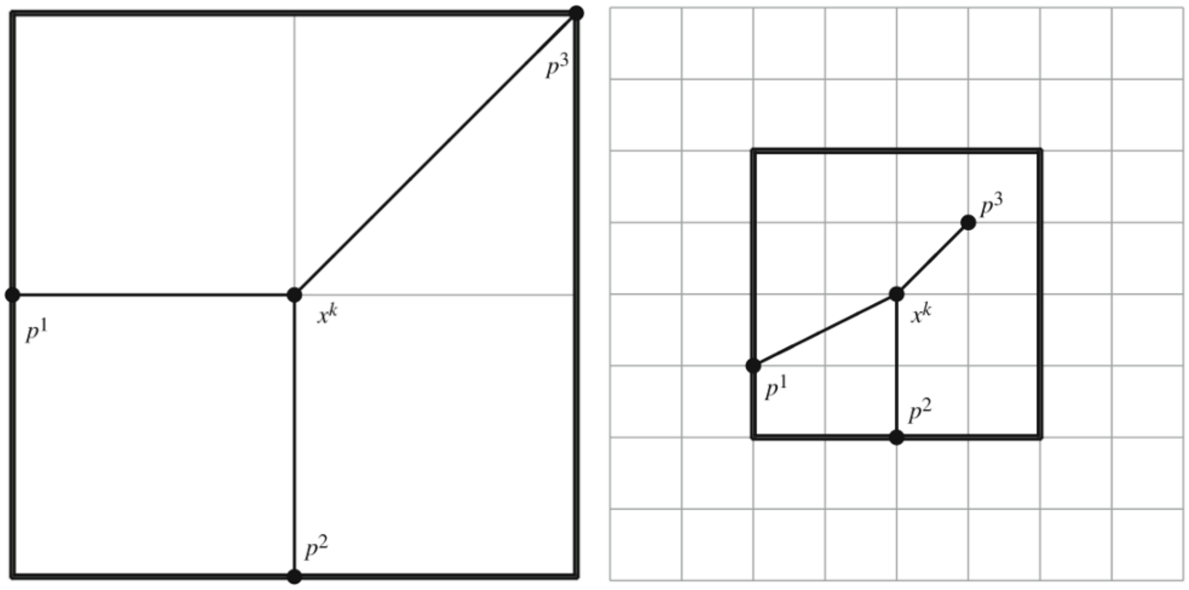
The frame is always larger than the mesh and provides a broader sampling. If the *poll* stage obtains a new incumbent, the frame moves to this point and increases its size, otherwise, it decreases its size and maintains the current incumbent. We define the initial poll size as $\Delta_j^0 = \frac{U_j - L_j}{10}$ according to (AUDET; LE DIGABEL; TRIBES, 2016), and the initial mesh size as $\delta^0 = \Delta^0$. The *poll* stage selects points based on a subset

D^k of directions from D, δ and Δ, and at each iteration, the mesh size δ is updated as $\delta = min\{\Delta^k, \Delta^{k^2}\}$, changing the frame size by $\tau^{-1}\Delta$ when increased and $\tau^1\Delta$ when decreased, where $\tau \in [0, 1]$. This process allows a continuous refinement of the mesh around the incumbent, and consequently reducing the objective function value. A further complexity analysis on direct search based methods can be found at (DODANGEH; VICENTE, 2016).

The Ortho-MADS (**Abramson2009**) is an improvement in the *poll* stage of the MADS algorithm, where the choice of the polling directions is deterministic and orthogonal to each other. This leads, at each iteration, to convex cones of missed directions that are minimal in a reasonable measure (**Abramson2009**). The Ortho-MADS changes the evaluation point selection method during the *poll* stage by creating D^k set of orthogonal directions, which define a dense sphere that increases with the number of iterations k[1].

Fig. 10a depicts a mesh and frame of size one, where the *poll* stage points p^1, p^2, and p^3 are limited by the frame and the mesh size, which in this case, the *search* and *poll* stage would find both the same points, performing similarly. In Fig. 10b, the mesh size is half of the frame size, and the *poll* stage can use 24 points defined by the mesh intersections (excluding x^k), allowing a broader exploration of the search space than the searching step.

Figura 10 – Two mesh and frame representations with different sizes.



a mesh size $\delta^k = 1$ and frame size $\Delta^k = 1$.   b Mesh size $\delta^k = \frac{1}{4}$ and frame size $\Delta^k = \frac{1}{2}$.

Among the traditional stopping criterion, such as time and number of function evaluations, the MADS can terminate the optimization process when it achieves a desired minimum mesh size value, that corresponds to achieve a desired local minimum.

[1] for further information refers to (**Abramson2009**).

During the MADS optimization process, it continuously updates the mesh (δ) and the frame (Δ) sizes. Both δ and Δ become smaller as the algorithm moves towards the minimum of $f(x)$. This mesh size verification is the last step in the MADS iterations. For the SVM with Gaussian kernel, we have two hyperparameters (thus two variables) C and γ that may have different scales, e.g. $C \in [0, 100]$ and $\gamma \in [1e^{-5}, 1]$. The NOMAD can assume different minimum mesh sizes for each dimension, pre-defined at the beginning of the BBO optimization process.

4.3.2 SEARCH METHODS

The *search* stage is not necessary for the convergence analysis and can be done using different strategies such as the VNS (AUDET; BÉCHARD; DIGABEL, 2008) or the NM search (NELDER; MEAD, 1965). The MADS algorithm does not dictate the selection of points in the *search* stage.

The NM (NELDER; MEAD, 1965) is a method for function minimization proposed by (AUDET; TRIBES, 2018) to be used in the *search* stage of the MADS algorithm. The NM continuously replaces the worst point x^n from a set of n points defined as $\mathbb{X} = \{x^0, x^1, \cdots, x^n\}$, where $\mathbb{X}$ is a set of n vertices from a simplex problem for the minimization function $f(x)$. A point x_{new} is considered better, or is said to dominate another point x, if $f(x_{new}) < f(x)$. The domination defines the function Best(x, x_{new}) as in Eq. 14, where:

$$\text{Best}(x_{new}, x) = \begin{cases} x_{new}, & \text{if } f(x_{new}) < f(x) \\ x, & \text{otherwise} \end{cases} \tag{14}$$

At each iteration, the NM evaluates $f(x)$ at the points given by the simplex and replaces x^n according to the following criteria:

$$x^n = \begin{cases} shrink(\mathbb{X}) & \text{if } x^{ic} \in \text{inside contraction zone} \\ x^{ic} & \text{if } x^{ic} \notin \text{inside contraction zone} \\ \text{Best}(x^r, x^e) & \text{if } x^r \in \text{expansion zone} \\ x^r & \text{If } x^r \in \text{reflexion zone} \\ \text{Best}(x^r, x^{oc}) & \text{If } x^r \in \text{outside contraction zone} \end{cases} \tag{15}$$

where x^r, x^e, x^{oc}, and x^{ic} are defined by Eqs. 17 to 20 as follows:

$$x^c = \frac{1}{n} \sum_{i=0}^{n-1} x^i \tag{16}$$

$$x^r = x^c + (x^c - x^n) \tag{17}$$

$$x^e = x^c Q^e (x^c - x^n) \tag{18}$$

$$x^{oc} = x^c Q^{oc} (x^c - x^n) \tag{19}$$

$$x^{ic} = x^c Q^{ic} (x^c - x^n) \tag{20}$$

$$\begin{aligned} \text{shrink}(\mathbb{X}) = & x^0, x^0 + \zeta(x^1 - x^0), x^0 + \zeta(x^2 - x^0), \\ & \cdots, x^0 + \zeta(x^n - x^0) \end{aligned} \tag{21}$$

where Q^e, Q^{oc}, and Q^{ic} are expansion, outside contraction and inside contraction respectively, being defined as $Q^e = 2$, $Q^{oc} = -\frac{1}{2}$, and $Q^{ic} = \frac{1}{2}$. ζ is the shrinking parameter usually defined as $\zeta^{ic} = \frac{1}{2}$.

The zone definition of a new point x is given as follows:

$$x \in \begin{cases} \text{inside contraction zone} & \text{if } x^n \text{ dominates } x \\ \text{expansion zone} & \text{if } x \text{ dominates } x^0 \\ \text{reflection zone} & \text{if } x \text{ dominates at least two } \mathbb{X}\ points \\ \text{outside contraction zone} & \text{otherwise} \end{cases}$$

and in the last case, x dominates none or one point of $\mathbb{X}$.

The NM method improves the search stage by selecting new points in a more controlled way than e.g., a random selection. The MADS algorithm can also use other methods in the search stage, such as the Bayesian Optimization method, which is the basis of the BADS algorithm. The convergence rate of the MADS algorithm depends on the quality of the search method.

In order to include a far-reaching search step to escape from an undesired local minimum, (AUDET; BÉCHARD; DIGABEL, 2008) also incorporates the VNS (MLADENOVIĆ; HANSEN, 1997; HANSEN; MLADENOVIĆ, 2001) as a *search* stage in the MADS algorithm. The VNS algorithm complements the MADS *poll* stage, i.e., when current iteration results in no success, which means that it could not find points with a smaller $f(x)$, the next *poll* stage generates trial points closer to the poll center, while the VNS explores a more distant region with a larger perturbation amplitude. The VNS uses a random perturbation method to attempt to escape from a local optimum solution so that a new descent method from the perturbed point leads to an improved local optimum. The VNS requires a neighborhood structure that defines all possible trial points reachable from the current solution, and a descent method that acts in the structure. The VNS amplitude of iteration k is parameterized by a non-negative scalar $\Upsilon_k \in \mathbb{N}$ that gives the order of the perturbation.

The MADS mesh provides the required neighborhood structure to the VNS, and by adding a VNS exploration in the search step, it introduces two new parameters: one is related to the VNS shaking method $\Delta_v > 0$, and the other defines a stopping criterion for the descent $\rho > 0$ (AUDET; BÉCHARD; DIGABEL, 2008). The shaking of iteration k generates a point x' belonging to the current mesh $M(k, \Delta_k)$, and the amplitude of the perturbation is relative to a coarser mesh, which is independent of Δ_k. The VNS mesh size parameter defined as $\Delta_v > 0$ and the VNS mesh M(k, Δ_v) are constant and independent on the iteration number k not to be influenced by a specific MADS behavior (the perturbation amplitude Υ_k is updated outside the VNS search step). The shaking function is defined as:

$$\begin{aligned} \text{shaking} : (M(k,\Delta_k), \mathbb{N}) &\to M(k,\Delta_v) \subseteq M(k,\Delta_k) \\ (x, \Upsilon_k) &\longmapsto x' = \text{shaking}(x, \Upsilon_k) \end{aligned} \tag{22}$$

The VNS descent function generates a finite number of mesh points and it is defined as:

$$\begin{aligned} \text{descent} : M(k,\Delta_v) &\to M(k,\Delta_k) \\ x' &\longmapsto x'' = \text{descent}(x') \end{aligned} \tag{23}$$

where x' is the point resultant from the previous shaking and x'' is a point based on x' but with improved $f(x)$ value. The improvement is important because x' has low probability to generate good optimization results due to its random choice by the shaking.

The descent step must lead towards a local optimum, and in the MADS context the local optimality is defined with respect to the mesh, so the descent step acts with respect to the current step size Δ_k and the directions used. To reduce the number of function evaluations and to avoid exploring a previously visited region, the descent step stopping criterion is defined as $||x - x_{new}||_\infty \leq \rho$, where x_{new} is a trial point close to another point x considered previously[2]. The VNS generally brings about a higher number of black-box evaluations, but these additional evaluations lead to better results (AUDET; BÉCHARD; DIGABEL, 2008).

We choose to use the mesh size parameter δ_k as stopping criterion because it corresponds to a situation where new refinements could not find a better solution, meaning a local minimum, and according to (**audet2017**), the mesh size parameter goes to zero faster than the poll size parameter Δ_k. The NM improves the quality of the solutions in the *search* stage (AUDET; TRIBES, 2018), while the VNS allows the far-reaching exploration from current incumbent (AUDET; BÉCHARD; DIGABEL, 2008). The pseudo-code in Alg. 1 describes the high-level procedure of the MADS technique

[2] Further information of the VNS and the MADS integration can be found at (AUDET; BÉCHARD; DIGABEL, 2008)

considering the Ortho-MADS algorithm with the NM and the VNS search strategies and the minimum mesh size as stopping criterion implemented by NOMAD.

To run the Ortho-MAD2S with the NOMAD software we need to set the lower bound $L = [l_1, l_2]$, the upper bound $U = [u_1, u_2]$, and the initial poll and mesh size are defined as $\Delta_j^0 = \delta_j^0 = \frac{u_j - l_j}{10}$, where the mesh size parameter is associated with the variable $j \in \{1, 2, \ldots, n\}$. The algorithm (Alg. 1) starts evaluating $f(x)$ within the initial point. The first iteration executes a search step, and in case of a failure, that is, not finding a $f(x)$ value smaller than $f(x_k)$, it executes a poll step with Ortho-MADS direction. In case of a successful iteration of the NM-Search and the poll step, the next iteration runs the VNS-search to try escaping from an eventual and undesired local minimum.

Algorithm 1 NOMAD high-level procedure

```
Input: Initial point $x_0 = \{C_0, \gamma_0\}$, VNS amplitude parameter $\Upsilon$, Minimum mesh size $\{\delta_{\min C}, \delta_{\min \gamma}\}$
Output: Best point $x_{\text{best}} = \{C_{\text{best}}, \gamma_{\text{best}}\}$
Initialization: $k \leftarrow 0$
 1: while $\delta_C^k > \delta_{\min C}$ and $\delta_\gamma^k > \delta_{\min \gamma}$ do
 2:     $x' \leftarrow shaking(x_k, \delta_k)$
 3:     $x'' \leftarrow descent(x')$
    Search stage
 4:     $S_k \leftarrow$ finite number of points of $M(k, \Delta_k)$        ▷ $M(k, \Delta_k)$ is the current mesh size
 5:     Evaluates $f(t)$ on $S_k \cup x''$        ▷ $f(t)$ is the sub-step evaluation inside the search stage
 6:     if $f(t) < f(x_k)$ for some $t$ in a finite subset of $S_k \subset M_k$ using NM-SEARCH then
 7:         $x_{k+1} \leftarrow t$
 8:         goto 20
 9:     end if
    Poll stage
10:     Compute $p$ MADS directions $D_k \in \mathbb{R}^{nxp}$        ▷ $p$ is the number of Poll stage points and $D_k$ is the
    directions set at $k$
11:     Construct the frame $P_k \subseteq M(k, \Delta_k)$
12:     Evaluates $f(p)$ on $p$ points of $P_k$
13:     if $f(p) < f(x_k)$ then
14:         $x_{k+1} \leftarrow t$
15:         $\Delta_{k+1} \leftarrow \tau^{-1}\Delta_k$
16:     else
17:         $x_{k+1} \leftarrow x_k$
18:         $\Delta_{k+1} \leftarrow \tau^{1}\Delta_k$
19:     end if
20:     Update VNS amplitude ($\Upsilon_{k+1} \leftarrow \Upsilon_0$ or $\Upsilon_{k+1} \leftarrow \Upsilon_k + \delta$)
21:     Updates of solution and mesh
22:     $k \leftarrow k + 1$
23: end while
24: return $x_k$
```

4.4 EXPERIMENTAL PROTOCOL AND RESULTS

We use the NOMAD black-box optimization software (COUTURE *et al.*, 2018) (LE DIGABEL; SÉBASTIEN, 2011) (version 3.9.1), which provides several interfaces to run the Ortho-MADS and its variations, including MATLAB. The other approaches used for comparison are also implemented in MATLAB, and we developed an experimental protocol to compare the black-box optimization methods in a machine learning classification context based on (**audet2017**) and (MORÉ; WILD, 2009). The experimental protocol consists of three steps:

1. Select datasets;
2. Algorithm comparison using a common configuration;
3. Evaluation of the proposed strategy.

4.4.1 DATASETS

We have selected thirteen benchmark datasets with different numbers of instances and dimensions to evaluate the proposed approach, and to compare it with other strategies available in the literature. These datasets are publicly available at the LIBSVM website[3]. Tab. 6 summarizes the main characteristics of each dataset.

Dataset	#Class	#Features	#Train	#Test	#Valid
Astroparticle (HSU; CHANG; LIN, 2003)	2	4	3,089	4,000	NA
Car (HSU; CHANG; LIN, 2003)	2	21	1,243	41	NA
DNA (HSU; LIN, 2002)	3	180	1,400	1,186	600
Letter (DHEERU; KARRA TANISKIDOU, 2017)	26	16	10,500	5,000	4,500
Madelon (**guyon2005**)	2	500	2,000	600	NA
Pendigits (DHEERU; KARRA TANISKIDOU, 2017)	10	16	7,494	3,498	NA
Protein (WANG, 2002)	3	357	14,895	6,621	2871
Satimage (HSU; LIN, 2002)	6	36	3,104	2,000	1,331
Shuttle (HSU; LIN, 2002)	7	9	30,450	14,500	13,050
Splice (DHEERU; KARRA TANISKIDOU, 2017)	2	60	1,000	2,175	NA
SVMguide4 (HSU; CHANG; LIN, 2003)	4	10	300	312	NA
USPS (**hull1994**)	10	256	7,291	2,007	NA
Vowels (DHEERU; KARRA TANISKIDOU, 2017)	11	9	598	462	NA

NA: Not available.

Tabela 6 – Benchmark datasets used in the experiments.

4.4.2 OTHER BBO METHODS

We have selected five widely used BBO-based methods of hyperparameter tuning for comparison purposes: BO, BADS, Simulated Annealing (SA), Grid Search (GS), and Random Search (RS). These methods are briefly described as follows:

- The BO (**snoek2012**) uses a probabilistic function $f(x)$ as a model for the problem. It benefits from previous information in contrast with other methods that use gradients or Hessians. Bayesian optimization uses prior, which is the probabilistic model of the objective function, and the acquisition function, which defines the

[3] www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

next points to evaluate. In this comparison, we used the Gaussian process as prior and the expected improvement as acquisition function. We use the *bayesopt* function from MATLAB.

- The BADS (ACERBI; JI, 2017) uses a Mesh Adaptive Direct Search (MADS) (**audet2006**) with a Bayesian optimization in the search step. The search step performs the discovering of the points with the intention of inserting domain-specific information and improving the quality of the points. When the search stage does not find suitable points, the poll stage of MADS broadly evaluates new points. The poll step is a computational expensive process and explores the objective function's shape for new points. We use the MATLAB implementation provided by (ACERBI; JI, 2017)[4].

- The Simulated Annealing SA mimics the process of annealing in metal. The temperature value dictates the probability function of the distance to a new random point based on the current point, while the distance to new points reduces as the temperature decreases with time. This procedure does not limit the new points to minimal points, this means that new points can have higher objective function value, helping to avoid the local minimum. We use the *simulannealbnd* function from MATLAB.

- The Grid Search GS algorithm consists of testing a combination of values for all the hyperparameters. This is a naive method that needs N^M evaluations where M is the number of hyperparameters and N is the number of values for each hyperparameter. The advantage of this method is that it can be easily parallelized, however, the method itself does not define a maximum number of evaluations. Therefore, we have split the search space based on the lower and upper bounds of C and γ to obtain the same number of evaluations allowed for other methods. We implemented the method in MATLAB.

- The Random Search RS algorithm starts by generating a set of points in the pre-defined search space. Subsequently, it gets the minimum among them and refines the search around it. It repeats this process until achieving the stop criterion. We implemented the method in MATLAB.

As one may see we do not compare the proposed method with derivative or evolutionary methods. In the framework of BBO, which includes the hyperparameter optimization problem, the derivative is unavailable, making the former unsuitable, while the later consists of global search heuristic methods (e.g. genetic algorithm and particle

[4] Available at `https://github.com/lacerbi/bads`

swarm optimization) in which the emphasis is on finding a decent global solution instead of finding an accurate local solution that provides a stopping criterion with some assurance of optimality.

We have evaluated the hinge-loss output value and the classification accuracy in the test set after 100 evaluations using a common configuration for all methods and the Ortho-MADS onsidering only the Nelder-Mead as search step. We used the functions *fitcsvm* and *fitcecoc* from the *MATLAB Statistics and Machine Learning Toolbox* to train the SVM for the binary or multiclass cases respectively, and the function *predict* to evaluate the learned model in the test sets. We defined the lower bound of the search space as $L = [0.01, 0.01]$ and its upper bound as $U = [100, 100]$. Considering that the starting point plays an important role in BBO optimization, we have compared the methods using six different initialization points $x_0 = (C, \gamma) = \{(0.5, 0.5), (10, 10), (50, 50), (90, 90), (1, 90), (90, 1)\}$. We have used the pre-defined validation set when it was available on the hinge-loss function, and for the cases where there was not a pre-defined validation set, we have used the training set with a stratified 3-fold cross validation strategy. For the RS and the GS, there is no pre-defined initialization point, and we have set a linear search of 100 iterations, i.e., $C \in \{1, 10 \times 1, 10 \times 2, \ldots, 10 \times 9\}$ and $\gamma \in \{1, 10 \times 1, 10 \times 2, \ldots, 10 \times 9\}$.

Tab. 7 shows the mean accuracy, standard deviation and maximum accuracy for all hyperparameter tuning methods. Both Ortho-MADS and BADS have shown to be more consistent to achieve a competitive mean accuracy for all datasets. The BO, SA, and RS present competitive results in ten datasets, and the GS shows competitive results in six datasets. Tab. 8 presents the mean loss $\mathcal{L}$, its standard deviation and the minimum loss $\mathcal{L}_{min}$. For most of the datasets, the behavior is similar to that presented in Tab. 7. The Bayesian, SA, RS, and GS provided the lowest $\mathcal{L}_{min}$ for the Splice dataset. However this result does not necessarily translate into high accuracy because in this case, the function might be overfitting.

	Ortho-MADS	Bayesian	SA	RS	GS	BADS
Astro	0.970$\pm$0.001 | 0.971	0.970$\pm$0.000 | 0.970	0.969$\pm$0.000 | 0.970	0.955$\pm$0.004 | 0.959	0.967$\pm$0.001 | 0.967	0.970$\pm$0.001 | 0.972
Car	0.715$\pm$0.013 | 0.732	0.695$\pm$0.034 | 0.732	0.687$\pm$0.052 | 0.732	0.407$\pm$0.230 | 0.707	0.715$\pm$0.040 | 0.732	0.724$\pm$0.030 | 0.780
DNA	0.942$\pm$0.000 | 0.942	0.942$\pm$0.000 | 0.942	0.616$\pm$0.005 | 0.624	0.943$\pm$0.002 | 0.945	0.943$\pm$0.001 | 0.945	0.945$\pm$0.003 | 0.949
Letter	0.943$\pm$0.030 | 0.957	0.954$\pm$0.000 | 0.955	0.935$\pm$0.045 | 0.959	0.838$\pm$0.022 | 0.859	0.943$\pm$0.000 | 0.943	0.955$\pm$0.002 | 0.958
Madelon	0.587$\pm$0.016 | 0.607	0.573$\pm$0.003 | 0.577	0.547$\pm$0.024 | 0.565	0.589$\pm$0.011 | 0.607	0.573$\pm$0.001 | 0.575	0.578$\pm$0.012 | 0.603
Pendigits	0.973$\pm$0.001 | 0.973	0.968$\pm$0.001 | 0.970	0.971$\pm$0.002 | 0.975	0.931$\pm$0.012 | 0.956	0.859$\pm$0.000 | 0.859	0.973$\pm$0.002 | 0.976
Protein	0.690$\pm$0.000 | 0.691	0.690$\pm$0.001 | 0.691	0.692$\pm$0.000 | 0.693	0.678$\pm$0.008 | 0.694	0.691$\pm$0.001 | 0.692	0.685$\pm$0.004 | 0.689
Satimage	0.912$\pm$0.001 | 0.912	0.915$\pm$0.001 | 0.917	0.915$\pm$0.004 | 0.918	0.876$\pm$0.016 | 0.908	0.705$\pm$0.020 | 0.718	0.910$\pm$0.003 | 0.915
Shuttle	0.905$\pm$0.000 | 0.905	0.913$\pm$0.005 | 0.917	0.918$\pm$0.001 | 0.918	0.885$\pm$0.018 | 0.910	0.718$\pm$0.000 | 0.718	0.907$\pm$0.007 | 0.917
Splice	0.897$\pm$0.001 | 0.899	0.607$\pm$0.005 | 0.611	0.614$\pm$0.010 | 0.626	0.870$\pm$0.020 | 0.899	0.582$\pm$0.000 | 0.582	0.897$\pm$0.002 | 0.901
Svmguide4	0.846$\pm$0.010 | 0.859	0.774$\pm$0.004 | 0.780	0.774$\pm$0.057 | 0.824	0.540$\pm$0.117 | 0.696	0.712$\pm$0.007 | 0.720	0.846$\pm$0.006 | 0.856
USPS	0.943$\pm$0.001 | 0.944	0.943$\pm$0.001 | 0.944	0.943$\pm$0.001 | 0.944	0.942$\pm$0.003 | 0.945	0.943$\pm$0.001 | 0.945	0.943$\pm$0.001 | 0.944
Vowels	0.622$\pm$0.013 | 0.641	0.630$\pm$0.011 | 0.641	0.587$\pm$0.016 | 0.608	0.498$\pm$0.052 | 0.589	0.591$\pm$0.003 | 0.593	0.625$\pm$0.011 | 0.641

Tabela 7 – Mean accuracy, standard deviation and maximum accuracy for all hyperparameter optimization methods in 13 datasets. The best results are underlined.

	Ortho-MADS	Bayesian	SA	RS	GS	BADS
Astro	$1.042\pm0.001 \mid 1.041$	$\underline{1.041\pm0.001 \mid 1.039}$	$1.042\pm0.001 \mid 1.039$	$1.082\pm0.015 \mid 1.067$	$1.045\pm0.000 \mid 1.044$	$1.043\pm0.002 \mid 1.041$
Car	$1.190\pm0.003 \mid 1.186$	$1.194\pm0.005 \mid 1.186$	$1.194\pm0.004 \mid 1.190$	$1.206\pm0.012 \mid 1.186$	$\underline{1.188\pm0.001 \mid 1.186}$	$1.191\pm0.004 \mid 1.184$
DNA	$\underline{1.033\pm0.000 \mid 1.033}$	$\underline{1.033\pm0.000 \mid 1.033}$	$1.046\pm0.002 \mid 1.044$	$1.034\pm0.001 \mid 1.034$	$1.044\pm0.001 \mid 1.042$	$1.035\pm0.002 \mid 1.033$
Letter	$\underline{1.001\pm0.000 \mid 1.001}$	$\underline{1.001\pm0.000 \mid 1.001}$	$1.006\pm0.007 \mid 1.003$	$1.003\pm0.001 \mid 1.002$	$\underline{1.001\pm0.000 \mid 1.001}$	$\underline{1.001\pm0.000 \mid 1.001}$
Madelon	$1.459\pm0.006 \mid 1.453$	$1.450\pm0.004 \mid 1.445$	$1.455\pm0.005 \mid 1.446$	$1.498\pm0.024 \mid 1.462$	$\underline{1.449\pm0.003 \mid 1.444}$	$1.452\pm0.009 \mid 1.443$
Pendigits	$\underline{1.001\pm0.000 \mid 1.001}$	$1.041\pm0.001 \mid 1.040$	$1.042\pm0.002 \mid 1.040$	$1.090\pm0.012 \mid 1.068$	$1.043\pm0.001 \mid 1.042$	$\underline{1.001\pm0.000 \mid 1.001}$
Protein	$\underline{1.157\pm0.000 \mid 1.157}$	$\underline{1.157\pm0.000 \mid 1.157}$	$1.163\pm0.000 \mid 1.163$	$1.168\pm0.011 \mid 1.158$	$1.163\pm0.000 \mid 1.162$	$1.158\pm0.000 \mid 1.157$
Satimage	$\underline{1.011\pm0.000 \mid 1.011}$	$1.041\pm0.001 \mid 1.039$	$1.042\pm0.001 \mid 1.040$	$1.090\pm0.020 \mid 1.056$	$1.045\pm0.001 \mid 1.043$	$\underline{1.011\pm0.000 \mid 1.011}$
Shuttle	$\underline{1.000\pm0.000 \mid 1.000}$	$1.042\pm0.001 \mid 1.041$	$1.042\pm0.000 \mid 1.041$	$1.080\pm0.025 \mid 1.047$	$1.044\pm0.001 \mid 1.043$	$\underline{1.000\pm0.000 \mid 1.000}$
Splice	$1.179\pm0.005 \mid 1.175$	$\underline{1.041\pm0.001 \mid 1.040}$	$1.042\pm0.001 \mid 1.041$	$1.076\pm0.012 \mid 1.061$	$1.044\pm0.001 \mid 1.043$	$1.185\pm0.004 \mid 1.178$
SVMguide4	$1.044\pm0.002 \mid 1.041$	$\underline{1.041\pm0.001 \mid 1.040}$	$1.045\pm0.004 \mid 1.041$	$1.089\pm0.013 \mid 1.072$	$1.045\pm0.001 \mid 1.044$	$1.044\pm0.001 \mid 1.042$
USPS	$\underline{1.002\pm0.000 \mid 1.002}$	$\underline{1.002\pm0.000 \mid 1.002}$	$\underline{1.002\pm0.000 \mid 1.002}$	$1.003\pm0.001 \mid 1.002$	$\underline{1.002\pm0.000 \mid 1.002}$	$\underline{1.002\pm0.000 \mid 1.002}$
Vowels	$\underline{1.003\pm0.000 \mid 1.003}$	$\underline{1.003\pm0.000 \mid 1.003}$	$1.194\pm0.004 \mid 1.188$	$1.023\pm0.010 \mid 1.006$	$1.004\pm0.000 \mid 1.004$	$1.004\pm0.000 \mid 1.003$

Tabela 8 – Mean loss ($\mathcal{L}$), its standard deviation and minimum loss ($\mathcal{L}_{min}$) for all hyperparameter optimization methods in 13 datasets. The best results are underlined.

Another aspect to analyze is the convergence rate and the trajectory of the algorithms. For all datasets, the Ortho-MADS presents a competitive convergence rate, and in many cases (as exemplified in Figs. 11a and 11b), both the Ortho-MADS and the BADS (that also uses the MADS algorithm) have the fastest convergence rate to reach a minimum. In some cases, the Ortho-MADS may not have the fastest convergence, as depicted in Figs. 12a and 12b. However, it is still competitive with other methods, and may pass over its convergence rate to reach a lower local minimum, as shown in Fig. 12b. The BADS achieved the best results overall, with better accuracy in eight out of thirteen datasets (Astro, Car, DNA, Letter, Pendigits, Splice, Svmguide4, and USPS), and competitive accuracy for all other datasets with a low standard deviation. The Ortho-MADS has the second-best results, with best results in five out of thirteen datasets (Astro, Pendigits, Splice, Svmguide4, and USPS), and competitive accuracy for all other datasets with a low standard deviation.

The Bayesian and the SA methods presented the best results in four out of thirteen datasets, but sometimes the results are not competitive, as observed for the Splice and Svmguide4 datasets when applying the Bayesian method, and for the DNA, Splice, and Svmguide4 datasets when using the SA method. In addition, both methods have presented standard deviations higher than Ortho-MADS and BADS. The RS achieved the best accuracy in the Madelon dataset, however, RS depends on the randomness that leads to more iterations to achieve a good result, and GS depends on the grid choice, which creates unreachable spaces.

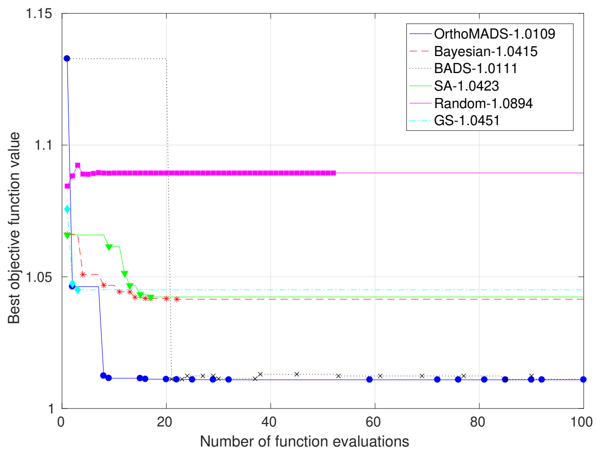
A close look at the Ortho-MADS standard deviation (this extends to other methods as well) from Tab. 7 indicates that in some cases we do not reach the best point, and this fact could be related to the choice of the starting point, as it has an important influence on the result or the method randomness that falls into a local minimum. Figs. 12a and 12b exemplify the influence of the starting point on the effectiveness of the algorithms. From the starting point $x_0 = \{0.5, 0.5\}$, depicted in Fig. 12a, the Ortho-MADS, Bayesian, BADS, SA, and GS achieved worse objective function value when compared to the starting point $x_0 = \{90, 1\}$ from Fig. 12b.

We generate an ordering of the methods based on the mean, maximum and worst accuracy reported in Tab. 7. Tab. 9 summarizes the comparison between all methods through an average ranking (BRAZDIL; SOARES, 2000) according to the measured accuracy mean, worst case, and best case. The BADS has the best rank among all considered methods, followed by the Ortho-MADS, Bayesian, SA, GS, and RS. The most consistent methods are the BADS and the Ortho-MADS, ranking 1 and 2 for the best mean and the best maximum accuracy, and 6 and 5 for the worst mean accuracy respectively.

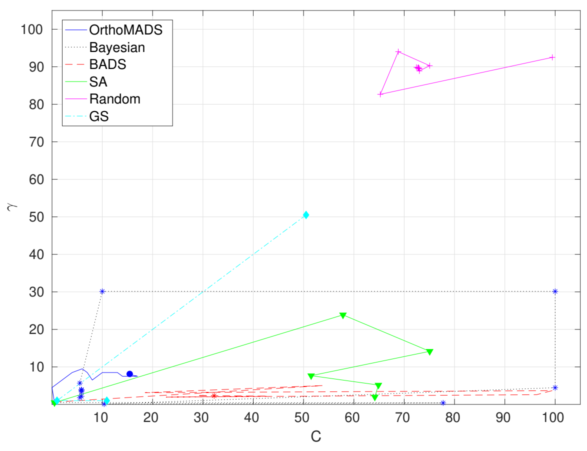
Algorithm	Best Mean Accuracy		Worst Mean Accuracy		Best Maximum Accuracy	
	$\bar{r}$	Rank	$\bar{r}$	Rank	$\bar{r}$	Rank
BADS	1.77	1	4.38	6	2.08	1
Ortho-MADS	2.07	2	3.77	5	2.69	2
Bayesian	2.31	3	3.46	4	3.23	4
SA	2.78	4	3.08	3	2.92	3
GS	3.54	5	2.38	2	4.38	6
RS	3.77	6	2.08	1	3.92	5

Tabela 9 – Average ranking (AR) considering the best mean accuracy, the worst mean accuracy and the maximum accuracy for the 13 datasets (BRAZDIL; SOARES, 2000).

Figura 11 – Satimage dataset with $x_0 = \{0.5, 0.5\}$.

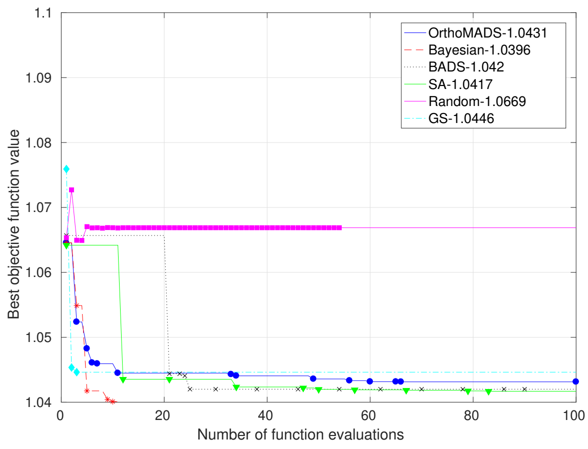


a Convergence plot comparison.

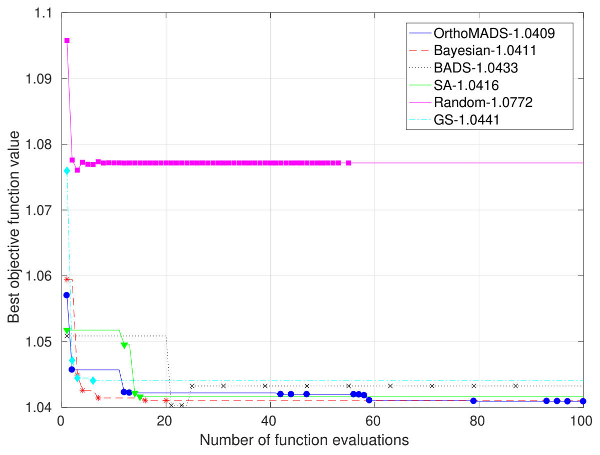


b Trajectory plot comparison.

Figura 12 – Astroparticle dataset comparison convergence plot with different starting points.



a at $x_0 = \{0.5, 0.5\}$.



b at $x_0 = \{90, 1\}$.

4.5 THE ORTHO-MADS WITH NELDER-MEAD AND VNS

The Ortho-MADS has shown to be competitive with other state-of-the-art methods (Tab. 7) to tune the hyperparameters of the SVM with Gaussian kernel, however, we propose to use the Ortho-MAD2S, that combines the Ortho-MADS convergence properties with two different search algorithms to enhance the stability and reachability, and to use the mesh size as stopping criterion. The NM search strategy leads to a faster convergence when compared to regular Ortho-MADS search strategy, i.e., it requires fewer function evaluations to reach the pre-defined minimum mesh size. The VNS explores regions far from the incumbent (increasing the number of function evaluations), helping to escape from an eventual undesired local minimum. The VNS counterbalance the NM fast convergence; however, it explores more regions from the search space and it mitigates the initial point influence. The Ortho-MADS attempts to find an accurate local solution and using the mesh-size as stopping criterion translates into stopping the algorithm when achieving a local solution (mesh-size) that satisfies the user needs.

We evaluate the accuracy and the number of function evaluations Ortho-MAD2S considering a non-opportunistic strategy for the search algorithms[5] using the minimum mesh size as stopping criterion. Because the MADS direction is randomly chosen, for each dataset we run 50 times using the following default configuration (empirically defined): the lower bound (L) as $[0.01, 0.01]$, the upper bound (U) as $[100.01, 100.01]$, the starting point (x_0) as $\{50, 50\}$, the minimum mesh size (δ_{min}) as 0.009 (that corresponds to three shrinking executions from the initial mesh size), and the perturbation amplitude is $\Upsilon = 0.25$. From the initial configuration, we further analyze the impact of changing the starting point x_0, the minimum mesh size δ_{min} and the perturbation amplitude Υ. Tab. 10 presents the mean accuracy, the standard deviation, and the maximum and median accuracy. For each measure we also present the corresponding number of function evaluations. As stated before, the incorporation of VNS in the search step aids the Ortho-MADS to escape from local minimum, which may lead to better results, and the NM counterbalance the number of function evaluations needed.

Tab. 10 presents the results of the proposed approach, and comparing with Tab. 7 we observe several improvements. Using the minimum mesh-size as stopping criterion may avoid unnecessary function evaluations. In our previous experiment (Tab. 7), the stopping criterion was 100 function evaluations, and using the new stopping criterion we reach the same or better accuracy with fewer function evaluations (as reported in Tab. 10) in all runs for five datasets (DNA, Madelon, Protein, Shuttle, and USPS). In addition, it may reduce the number of function evaluations in another five datasets (Astro, Pendigits, Shuttle, Splice, and Vowels), i.e., sometimes it achieves the minimum mesh-

[5] For each search iteration, the algorithm does not finish when a better incumbent is found. It only finishes when all points are evaluated.

	Mean	Std	Max	Median
Astro	0.968 | 167	0.001 | 45	0.971 | 81	0.969 | 162
Car⋆	0.720 | 141	0.027 | 41	0.829 | 105	0.707 | 137
DNA	0.942 | 51	0.000 | 0	0.942 | 51	0.942 | 51
Letter⋆	0.954 | 111	0.000 | 0	0.954 | 111	0.954 | 111
Madelon⋆	0.598 | 85	0.007 | 43	0.607 | 70	0.602 | 75
Pendigits	0.972 | 117	0.000 | 27	0.973 | 76	0.972 | 113
Protein	0.691 | 73	0.000 | 0	0.691 | 73	0.691 | 73
Satimage	0.913 | 108	0.000 | 0	0.913 | 108	0.913 | 108
Shuttle⋆	0.999 | 73	0.000 | 0	0.999 | 73	0.999 | 73
Splice	0.897 | 120	0.001 | 27	0.901 | 78	0.897 | 118
Svmguide4⋆	0.847 | 118	0.008 | 23	0.865 | 135	0.848 | 112
USPS	0.943 | 95	0.001 | 23	0.945 | 78	0.943 | 97
Vowels	0.624 | 128	0.011 | 32	0.643 | 91	0.621 | 123

Tabela 10 – Accuracy | No. of function evaluations for the proposed approach. Mean, standard deviation, maximum (and the number of function evaluations regarding the best result), and median for all 13 datasets. ⋆ indicates higher mean accuracy.

size with fewer than 100 function evaluations. Regarding the stability, the DNA, Letter, Protein, Satimage, and Shuttle datasets present a standard deviation of approximately zero. We achieved a better accuracy for the Car dataset (from 0.780 to 0.829), but it increases the number of function evaluations to reach the minimum mesh-size. For the Madelon dataset we reached the best accuracy reported in Tab. 7 by the RS algorithm, and increased the mean accuracy with fewer function evaluations (85 of mean, and best value achieved with 70). We achieved the best accuracy overall for the Shuttle dataset, with a lower number of function evaluations. In the Vowels dataset, the best accuracy improved from 0.641 to 0.643, however, with an increase in the number of function evaluations (from 100 to 128 of mean).

Here again, we generate an ordering of the methods but now replacing the Ortho-MADS by the proposed approach. Tab. 11 summarizes the comparison between all methods through an average ranking (BRAZDIL; SOARES, 2000) according to the measured accuracy mean, worst case, and best case. The proposed approach has the best rank among all considered methods, followed by the BADS, Bayesian, SA, GS, and RS. The most consistent methods are the proposed approach and the BADS, ranking 1 and 2 for the best mean and the best maximum accuracy, and 5 and 6 for the worst mean accuracy respectively.

The Friedman rank sum test shows a p-value of 0.00024, and Tab. 12 presents the Nemenyi test using Tabs. 7 and 10. The results indicate that the Ortho-MAD2S presents similar results to BADS, and the concordance of results between Ortho-MADS and BADS are smaller than the proposed approach. We can conclude that both BADS and Ortho-MAD2S present superior performance compared to other methods. Further-

Algorithm	Best Mean Accuracy		Worst Mean Accuracy		Best Maximum Accuracy	
	$\bar{r}$	Rank	$\bar{r}$	Rank	$\bar{r}$	Rank
Ortho-MAD2S	1.85	1	4.31	5	2.15	1
BADS	1.92	2	4.46	6	2.38	2
Bayesian	2.61	3	3.38	4	3.46	4
SA	3.08	4	3.08	3	3.15	3
GS	3.85	5	2.31	2	4.46	6
RS	4.15	6	2.00	1	4.08	5

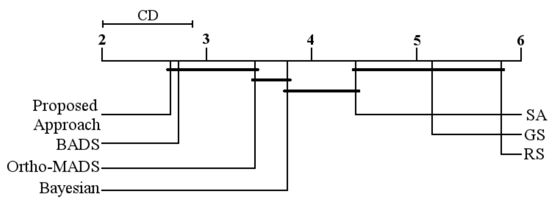
Tabela 11 – Average ranking (AR) considering the best mean accuracy, the worst mean accuracy and the maximum accuracy for the 13 datasets (BRAZDIL; SOARES, 2000).

more, the advantage of the proposed approach when compared to BADS is the false minimum avoidance and the stopping criterion. Fig. 13 depicts the critical difference graph comparing all methods, illustrating the results of Tab. 12. The confidence interval is 95% for the null hypotheses of $H_0 : \Theta_i = \Theta_j$ and the alternative hypotheses of $H_1 : \Theta_i \neq \Theta_j$. Considering the used confidence interval the p-values close to one do not reject the null hypotheses, while the p-values close to zero reject the H_0 and do not reject H_1.

	Proposed Approach	Ortho-MAD2S	Bayesian	SA	RS	GS
Ortho-MAD2S	0.9637	-	-	-	-	-
Bayesian	0.8446	0.9998	-	-	-	-
SA	0.3597	0.9175	0.9876	-	-	-
RS	0.0037	0.0821	0.1959	0.6601	-	-
GS	0.0497	0.4163	0.6601	0.9780	0.9876	-
BADS	1.0000	0.9780	0.8844	0.4163	0.0052	0.0642

Tabela 12 – Nemenyi test between all methods using the 13 datasets.

Figura 13 – Nemenyi test critical difference.



We choose the Car dataset (as it has the largest standard deviation among all datasets reported in Tab. 10) to analyze the impact of changing the VNS, the starting point and the minimum mesh size. We start considering different perturbation amplitudes $\Upsilon = \{0.25, 0.5, 0.75, 0.9\}$, and Tab. 13 shows the results after 50 runs for each Υ. By increasing Υ, the algorithm can reach regions more distant from the current incumbent, however, it requires more function evaluations to achieve a high mean accuracy. A large Υ does not translate into a better accuracy, as increasing Υ it creates sparser points to evaluate which may skip good regions, as the results for $\Upsilon = 0.75$ and $\Upsilon = 0.9$ indicate.

Car	$\Upsilon = 0.25$	$\Upsilon = 0.5$	$\Upsilon = 0.75$	$\Upsilon = 0.9$
Mean	0.720 | 141	0.720 | 172	0.710 | 275	0.710 | 362
Std	0.027 | 41.3	0.042 | 60.5	0.042 | 79.4	0.029 | 86.9
Max	0.829 | 105	0.829 | 107	0.805 | 200	0.780 | 205
Median	0.707 | 137	0.707 | 161	0.707 | 264	0.707 | 361

Tabela 13 – Mean accuracy, standard deviation, maximum accuracy and median accuracy and the corresponding No. of function evaluations by changing the VNS for the Car dataset for different perturbation amplitudes.

Tab. 14 shows that by decreasing δ_{min}, the proposed approach needs more function evaluations to reach the desired local minimum, as the mesh size reduction occurs in sequence (it is not possible to execute two mesh reduction operations in the same Ortho-MAD2S iteration). For $\delta_{min(C,\gamma)} = 9e-1$ we need fewer function evaluations to reach the stopping criterion, however, the maximum accuracy achieved was lower than the results from $\delta_{min(C,\gamma)} = 9e-3$. A smaller minimum mesh size, $\delta_{min(C,\gamma)} = 9e-7$, does not guarantee a good performance, but for sure it increases the number of function evaluations needed to reach the stopping criterion. In this case the model can discard good points or over-fit the model.

Car	$\delta_{min(C,\gamma)} = 9e-1$	$\delta_{min(C,\gamma)} = 9e-3$	$\delta_{min(C,\gamma)} = 9e-7$
Mean	0.738 | 34.9	0.729 | 184.4	0.716 | 236.7
Std	0.036 | 21.3	0.035 | 43.9	0.029 | 68.2
Max	0.804 | 60	0.829 | 133	0.804 | 212
Median	0.756 | 24	0.732 | 169	0.707 | 218

Tabela 14 – Mean accuracy, standard deviation, maximum accuracy and median accuracy and the corresponding No. of function evaluations for the Car dataset for different minimum mesh sizes (δ_{min}).

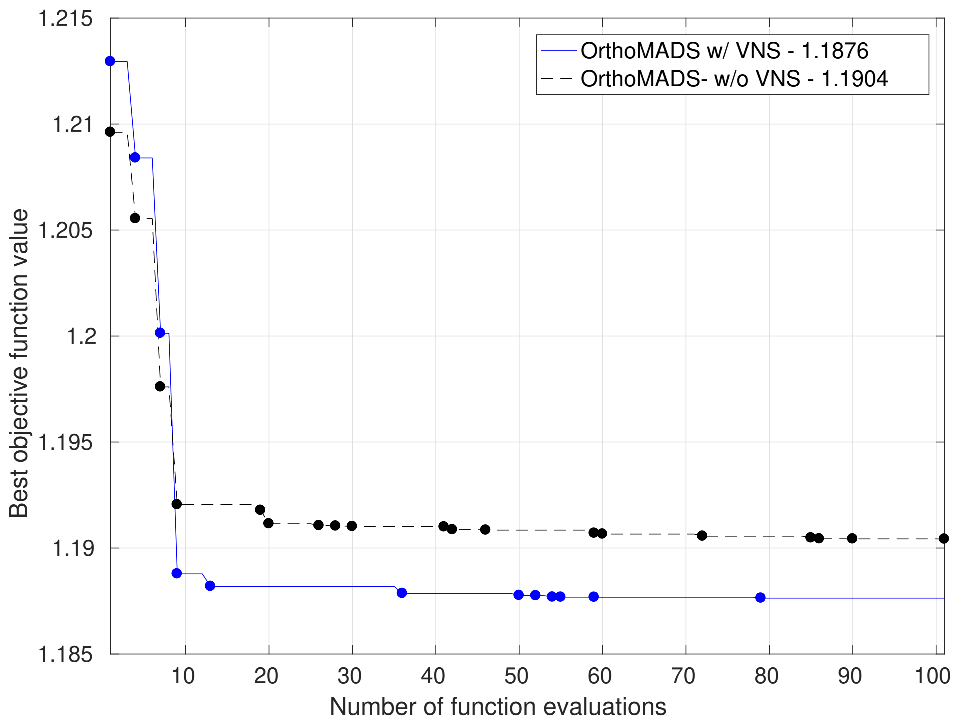
We have also evaluated the influence of the starting point for the Car dataset using five pre-defined and five random starting points $x_0 = \{(0.5, 0.5),(50, 50), (90, 90), (1, 90), (90, 1), (70.93, 75.21), (50.60, 64.29), (25.21, 79.05), (89.59, 13.49), (2.37, 57.91)\}$.

Tab. 15 shows that the mean value may be similar to the solution without VNS (as shown in Tab. 7), however, we could reach at least 0.805 of accuracy for all starting points, which is higher than the best solution from Tab. 7 (0.780 using BADS). Thus, the VNS mitigates the starting point effect and the MADS randomness. Fig. 14 exemplifies a comparison example between the Ortho-MADS with and without the VNS, both using the same starting point at $x_0 = (50, 50)$ and the number of function evaluations. The Ortho-MADS without VNS reached $\mathcal{L}_{min} = 1.1904$ and the accuracy of 0.7073 on the test set, while the Ortho-MAD2S reached $\mathcal{L}_{min} = 1.1876$ and the accuracy of 0.8048.

$x_0 = (C, \gamma)$	Mean	Max	Min
$(0.5, 0.5)$	0.717 | 1.191	0.805 | 1.187	0.780 | 1.185
$(50, 50)$	0.735 | 1.189	0.829 | 1.187	0.805 | 1.183
$(90, 90)$	0.706 | 1.190	0.805 | 1.188	0.732 | 1.183
$(1, 90)$	0.690 | 1.192	0.805 | 1.194	0.780 | 1.185
$(90, 1)$	0.721 | 1.192	0.805 | 1.196	0.707 | 1.118
$(70.93, 75.21)$	0.698 | 1.192	0.829 | 1.188	0.707 | 1.186
$(50.60, 64.29)$	0.728 | 1.191	0.805 | 1.196	0.780 | 1.184
$(25.21, 79.05)$	0.726 | 1.190	0.805 | 1.189	0.707 | 1.187
$(89.59, 13.49)$	0.727 | 1.191	0.805 | 1.193	0.732 | 1.185
$(2.37, 57.91)$	0.727 | 1.190	0.805 | 1.189	0.780 | 1.186

Tabela 15 – Mean accuracy, maximum accuracy and minimum accuracy and the corresponding $\mathcal{L}$ value for the Car dataset for different starting points x_0.

Figura 14 – Convergence plot comparison between Ortho-MADS with and without VNS for the Car dataset.



Tab. 16 compares the number of function evaluations that each method takes to achieve its best accuracy in the Madelon dataset. In the case of a good starting point (as we previously knew), all methods need the same or fewer function evaluations than the proposed approach to achieve their best accuracy. However, none can reach the accuracy achieved by the proposed method. In the case of a "bad"starting point, the BADS, Bayesian, and SA have a high probability of achieving an undesired local minimum, not reaching the highest accuracy. The GS and the RS depend on the grid designed by the user and on the dataset randomness to find a point that results in the best accuracy. Tab. 17 shows the accuracy of each method using the accuracy of 0.8 or one thousand function evaluations as stopping criteria and previously known "good"starting points. We choose the Car dataset because it has a significant difference in the best accuracy between the proposed approach and the other methods, and no other method achieved accuracy above 0.732.

We noticed in experiments that all methods were susceptible to fall at a minimum point that is not their best. Even the Ortho-MADS method without the VNS approach can reach this condition. The VNS approach is important to leave false minimum points. Ortho-MAD2S is less sensitive to the impact of initialization points and the false minimum problem. During the experiments we observed one situation where the RS achieved its minimum at five iterations with a random initialization point, but in the great majority of the situations, this method could not reproduce this result with less than 1,000 iterations.

Approach	Accuracy	# Evaluations
Proposed Approach	0.607	70
Bayesian	0.568	30
BADS	0.602	106
GS	0.571	57
RS	0.603	30
SA	0.601	19

Tabela 16 – Best maximum accuracy and No. of function evaluations for the Madelon dataset.

Approach	Accuracy	# Evaluations
Proposed Approach	0.829	105
Bayesian	0.707	1,000
BADS	0.707	1,000
GS	0.732	1,000
RS	0.707	1,000
SA	0.732	1,000

Tabela 17 – Best maximum accuracy and No. of function evaluations for the Car dataset with stopping condition of either 0.8 of accuracy or 1,000 function evaluations.

4.6 SUMMARY AND SYNTHESIS OF CONTRIBUTION

We proposed an approach for tuning the hyperparameters of SVM with Gaussian kernel based on black-box optimization algorithms which is an extension of Ortho-MADS. The proposed approach employs two different search strategies (Nelder-Mead and Variable Neighborhood) to escape from local minima as well as the mesh size as a stopping criteria to avoid unnecessary function evaluations. We have shown on benchmark datasets that the proposed approach outperforms other state-of-the-art methods for hyperparameter tuning which are widely used in machine learning. The alternation between the search and pool stage provides a robust performance, and the use of two different search methods attenuate the randomness of the MADS and the starting point choice. Besides that, the proposed approach has convergence proof, and using the mesh size as stopping criterion gives to the user the possibility of setting a specific local minimum region instead of using the number of evaluations, time, or fixed-size grid as a stopping criterion. In the cases where a test set is available, the evolution of the mesh size during the BBO iterations can help identify under and over fitting behaviors.

The proposed approach gives the user the flexibility of choosing parameters to explore different strategies and situations. From our experiments, we recommend starting with the proposed default configuration, and from there the user can customize the Ortho-MADS parameters if necessary, which may improve the quality of the results. Therefore, the Ortho-MAD2S can be used to replace current methods, as grid search, for tuning the hyperparameters of a SVM with Gaussian kernel. For future work, we expect to analyze the NOMAD with SVM variations, which includes incremental formulations and different kernels.

The contributions of this chapter are:

- We propose a novel approach to tune hyperparameters of an SVM with Gaussian kernel (thus we have 2 hyperparameters to be tuned) based on the Ortho-MADS (**Abramson2009**) and the combination of two different search strategies, the Nelder-Mead (NELDER; MEAD, 1965) and the Variable Neighborhood Search (VNS) (AUDET; BÉCHARD; DIGABEL, 2008). Combining such search strategies leads to a faster convergence (NM) and the possibility to escape from undesired local minimum (VNS).

- We propose a dynamic stopping criteria, i.e., we define the stopping criteria regarding the desired local minimum size which is set by the mesh size.
- The integration of the proposed approach into a black-box optimization framework that provides many other tools that aid to adjust the hyperparameters regarding the particularities of each application, as the mesh size, poll size, convergence step, and other stopping criteria such as the number of function evaluations and processing time. The experimental results on several benchmark datasets have shown that the proposed approach converges fast to appropriate hyperparameter values while achieving state-of-the-art accuracy. Besides that, the proposed approach presents several interesting properties such as the guarantee of convergence and a dynamic stopping criterion.

5 INCREMENTAL AND DECREMENTAL FUZZY BOUNDED TWIN SUPPORT VECTOR MACHINE

5.1 INTRODUCTION

Classical machine learning approaches, in which all data is simultaneously accessed, do not meet the requirements to deal with the scenario in which training data is partially available at a time or where the amount of data is so large that it does not fit into the memory or into the storage of a single machine. Incremental or on-line learning is an approach to tackle problems in which only a subset of the data is considered at each step of the learning process, or when the dataset is too large to be processed at once (KHEMCHANDANI; JAYADEVA; CHANDRA, 2009). From the computational point of view, incremental learning has three goals (HE, 2011): transform previously learned knowledge to current received data to facilitate learning from new data; accumulate experience over time to support the decision-making process; and achieve global generalization through learning to accomplish goals. Incremental learning often also refers to on-line learning strategies with limited memory resources, relying on creating a compact memory model that represents the already observed data but providing accurate results for all relevant settings.

5.2 LITERATURE REVIEW

The work of (LOSING; HAMMER; WERSING, 2018) evaluated the most common algorithms of incremental learning on diverse datasets, and the conclusion is that the Support Vector Machines (SVMs) are usually the highest accurate models. However, such an accuracy is at the expense of the most complex model besides many other shortcomings. The SVMs were developed to tackle two-class classification problems by solving a complex Quadratic Programming Problem (QPP) that determines a unique global hyperplane in the input space that maximizes the separation between the classes (CORTES; CORTES; VAPNIK, 1995). However, it requires a large memory and a high CPU power since the computational complexity of the SVM for l data points is $O(l^3)$, which makes it impractical for large datasets. To circumvent this problem, one may use the incremental version of SVM or its variants, that learns from new data by discarding past data points excepting the SVs, i.e., the new data is used to retrain the model together with the current SVs (CAUWENBERGHS; POGGIO, 2000; DOMENICONI; GUNOPULOS, 2001; KHEMCHANDANI; JAYADEVA; CHANDRA, 2009).

The Incremental Support Vector Machine (ISVM) proposed by Cauwenberghs and Poggio (CAUWENBERGHS; POGGIO, 2000) is an exact solution to the problem of on-line SVM that updates the optimal solution of the SVM by adding or removing one training data point. The bottleneck of the ISVM is that the computational complexity of a minor iteration of the algorithm is quadratic in the number of training data points learned so far. Therefore, the actual runtime depends on the balance between memory access and arithmetic operations in a minor iteration (LASKOV *et al.*, 2006). The LASVM (BORDES *et al.*, 2005) is an on-line kernel classifier that relies on the soft-margin SVM formulation to handle noisy data. The iterations are similar to the SMO algorithm but with a different search strategy. Furthermore, it introduces a SV removal step, where it removes the vectors collected in the current kernel expansion during the on-line process. The iterations run in epochs, where each epoch sequentially visits all the randomly shuffled training data points, and the stopping criteria is a predefined number of epochs. Multiple number of epochs can be used as a stochastic optimization algorithm in the off-line training, and a single epoch in the on-line step. The computational cost of the LASVM is $O(p \times lSV \times i)$, where (*nSV*) is the number of SVs, *i* is the number of on-line iterations, and p scales no more than linearly to the amount of training data points, which makes the training process faster than the ISVM. Empirical results suggest that using a single epoch yields to misclassification rates comparable with the SVM. Despite the effectiveness of the ISVM and the LASVM, both methods still need to deal with one large QPP, requiring large memory storage and CPU processing time on training and update steps.

The work of (MANGASARIAN; WILD, 2006) introduced the Generalized Eigenvalue Proximal Support Vector Machine (GEPSVM) that generates two non-parallel hyperplanes for a two-class problem. Thus, it solves two smaller Quadratic Programming Problems (QPPs) instead of a single complex QPP, laying each class data point in the proximity of a hyperplane, which reduces the complexity compared to the SVM. Jayadeva et al. (JAYADEVA; KHEMCHANDANI, R.; CHANDRA, 2007) proposed the TWSVM, which also solves a pair of QPPs where the data points of one class provide constraints to the other QPP and vice versa (TOMAR; AGARWAL, 2015; DING; YU *et al.*, 2014). The TWSVM classifies the data points of two classes using two non-parallel hyperplanes with a complexity of $O(2 \times (l/2)^3)$, which is four times lower than a SVM. Twin-based models are mathematically smaller than the SVM and they require low memory storage and CPU processing time.

Based on the TWSVM, several variants and solvers have been proposed (DING; ZHANG *et al.*, 2017; TOMAR; AGARWAL, 2015; TIAN; QI, 2014; JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, 2017a). Yuan-HaiShao et al. (SHAO; ZHANG *et al.*, 2011) suggested the Twin Bounded Support Vector Machine (TBSVM) that includes adherence to the structural risk minimization principle, so the dual formulation (whose inverse is guaranteed) can be solved by Successive Over-Relaxation (SOR) methodology. The Improved Twin Support Vector Machine (ITWSVM) (TIAN; JU *et al.*, 2014) uses a different representation from the TBSVM that leads to a different Lagrangian function for the primal problem and different dual formulations. The ITWSVM does not need to compute the inverse of large matrices before training and can be solved by the

SOR or the SMO. However, the matrices in the dual form must involve all the data points from both classes, which makes the dual QPPs larger than the TWSVM. Khemchandani et al. (KHEMCHANDANI; JAYADEVA; CHANDRA, 2008) proposed a novel fuzzy TWSVM that assigns a fuzzy weight to each data point to mitigate the effect of outliers and improve accuracy. Gao et al. (GAO; WANG *et al.*, 2015) proposed a coordinate descent fuzzy TWSVM, assigning a fuzzy membership function to mitigate the effect of noisy data points, and solving the QPPs with a coordinate descent with shrinking by active set. Other variants or extensions are the Least Square Twin Support Vector Machine (LS-TWSVM) (ARUN KUMAR; GOPAL, 2009) that solves the primal problems of the TWSVM, and the ν-TWSVM (PENG; XINJUN, 2010) where the ν parameter controls the bounds of the fractions of the SVs and the error margin.

Considering the TWSVM and its variations, Khemchandani et al. (KHEMCHANDANI; JAYADEVA; CHANDRA, 2009) introduced the Incremental Twin Support Vector Machine (I-TWSVM), which uses the concept of margin vectors and error vectors to select new data points to update the classifier. It learns from new data by retraining the model while discarding past data points except for the previous SVs and erroneous classified data points from the training dataset. However, for each new data point, both models need to be completely re-built. Hao et al. (HAO; ZHANG, 2014) proposed a fast incremental TWSVM that uses a distance-based strategy to determine if a new data point is above a predefined threshold. It selects the most important data points that are nearby the proximal hyperplane from the current training set, and keep data points that are not near the proximal hyperplane from the new training set. In each iteration, it retrains the model considering the previous SVs and the new data points (there is no decremental step). The On-line Twin Independent Vector Machine (OTWISVM) (ALAMDAR; GHANE; AMIRI, 2016) uses a modified Newton method to build a decision function via a subset of data points seen so far for each class separately (called basis). The basis vectors are found (or added during the on-line procedure) during iterative minimization by checking if a new data point is linearly independent in the feature space from the current basis. The basis size is limited, so it does not grow linearly with the training set. The OTWISVM does not have a decremental step, and as it utilizes a modified Newton solver, it needs to calculate the inverse of the Hessian on every update, making the method unfeasible to deal with high-dimensional datasets.

Besides improving the model with new data, it is also important to have a decremental procedure to prevent the model from growing indefinitely. Despite the update strategy be closely related to the model formulation, there are many alternatives on choosing the SVs to be removed, such as the time-window proposed by Fung et al. (FUNG, G.; MANGASARIAN, O. L., 2002), the concept of informative margin vectors and error vectors (CAUWENBERGHS; POGGIO, 2000), or decay coefficients (TVEIT; HETLAND; ENGUM, 2003).

Finally, to unveil the full power of the incremental SVM, it is necessary to adapt it to deal with non-linear problems using the kernel trick. However, conventional kernel approaches struggle to deal with large datasets due to the storage and computational issues in handling large kernel matrices. A feasible solution is the use of kernel approximations such as: exploiting low-rank approximation of the kernel matrix; reducing the kernel space definition; or exploiting a randomized kernel space definition (IOSIFIDIS *et al.*, 2016). Random Fourier (RF) approximations provide an efficient and elegant methodology (RAHIMI; RECHT, 2008), where the Fourier expansion generates features based on a finite set of random basis projections with inner products that are the kernel Monte Carlo approximations (LI; IONESCU; SMINCHISESCU, 2010). Fourier features are applicable to translation-invariant kernels, so it can be used to approximate the Gaussian kernel. Rahimi et al. (RAHIMI; RECHT, 2008) use RF to map the input data to a randomized low-dimensional feature space providing convergence bounds to approximate various radial basis kernel. Le et al. (LE; SARLOS; SMOLA, 2014) proposed an RF-based approximation called *Fastfood*, which requires a smaller computation and memory storage than Random Kitchen Sinks (RAHIMI; RECHT, 2009) to obtain an explicit function space expansion.

Although many efforts have been made, the incremental SVM approaches still have several shortcomings such as the impossibility of endless learning, high model complexity, high training time, high complexity of hyperparameter optimization, adaptability to concept drift, among others. In this paper we propose a novel incremental and decremental variant of the TWSVM called FBTWSVM that overcomes many of the shortcomings of the current approaches. The FBTWSVM combines a fast training and an incremental procedure (with the ability to handle noisy data) without weakening the accuracy when updated. The proposed approach can continuously integrate new information into already-built models and it is adherent to the structural risk minimization principle (as in (SHAO; ZHANG *et al.*, 2011)), and it uses the Dual Coordinate Descent (DCD) algorithm with active shrinking (TIAN; JU *et al.*, 2014; GAO; WANG *et al.*, 2015; GAO; WANG, 2017; KHEMCHANDANI; JAYADEVA; CHANDRA, 2008; GAO; WANG *et al.*, 2015) to create the off-line classifier. The incremental and decremental strategies are based on the DCD with shrinking, exploiting the relevance of each support vector. Furthermore, we propose the use of our linear formulation with a kernel approximation to speed up training and classification while maintaining the non-linearity. Finally, the FBTWSVM is extended to multiclass problems using a strategy based on the DAG. The experimental results on benchmarking datasets have shown that the proposed approach achieves accuracy comparable to the exact solution besides being faster to integrate new information and to discard outdated information into the already-built models.

5.3 BASIC CONCEPTS

5.3.1 THE FUZZY SVM

The Fuzzy SVM introduced by Lin et al. (CHUN-FU LIN; SHENG-DE WANG, 2002) uses the fuzzy theory to reduce the effect of outliers by applying a fuzzy membership to each data point. Fuzzy numbers, denoted as s_i, are assigned to each input data point to add information that reflects the noise contamination level, which is $0 \leq s_i \leq 1, i = 1, 2, ..., l$. Therefore, the training dataset D becomes a triple $D'=(x_i, y_i, s_i)$ to accommodate the fuzzy number and to reduce the influence of the contaminated data points in generating the decision functions. The fuzzy SVM is formulated as:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}||\boldsymbol{w}||^2 + C\boldsymbol{s}^\top\boldsymbol{\xi} \\ & \text{s.t. } y_i(\omega^\top \boldsymbol{x_i} + b) + \xi_i \geq 1 \\ & \xi_i \leq 0, i = 1, 2, ..., l \end{aligned} \tag{24}$$

where C is the trade-off scalar and ξ_i is the slack variable that represents the error associated with the i-th input data point. An important remark about this formulation is that a small s_i can reduce the effect of the slack variable ξ_i in Eq. 24, so reducing the importance of the corresponding data point $\boldsymbol{x_i}$. The classification of an input $\boldsymbol{x}$ is given by the sign of $\boldsymbol{\omega}^{*\top}\boldsymbol{x} + b^*$, where $\boldsymbol{\omega}^*$ and b^* are the solution of Eq. 24.

The construction of the membership functions follows the strategy used by Gao et al. (GAO; WANG *et al.*, 2015; GAO; WANG, 2017), which is inspired in (TANG, 2011). The method considers reducing the noise carried by outliers while keeping the importance of the SVs. We integrate the fuzzy SVM into the TWSVM formulation by selecting two different classes and assigning a positive label to the first class and a negative label to the second one. The class centers x_{c+} and x_{c-} are the mean points considering the input space of these two classes, defined by:

$$x_{c+} = \frac{1}{l+}\sum_{y_i=+1} \boldsymbol{x_i}, \;\; x_{c-} = \frac{1}{l-}\sum_{y_i=-1} \boldsymbol{x_i} \tag{25}$$

The hyperspheres radii $r+$ and $r-$ are constructed by measuring the distance of the farthest scattering data point of each class:

$$\begin{aligned} r_+ &= \max_i ||\boldsymbol{x_i} - x_{c+}|| \quad \text{if} \quad y_i = +1 \\ r_- &= \max_i ||\boldsymbol{x_i} - x_{c-}|| \quad \text{if} \quad y_i = -1 \end{aligned} \tag{26}$$

The membership of s_i is assigned according to the distance relationship between $||\boldsymbol{x_i} - x_{c+}||$ and $||\boldsymbol{x_i} - x_{c-}||$ when x_{c+}, x_{c-}, r_+, and r_- are known. Formally, s_i of a positive data point is given as:

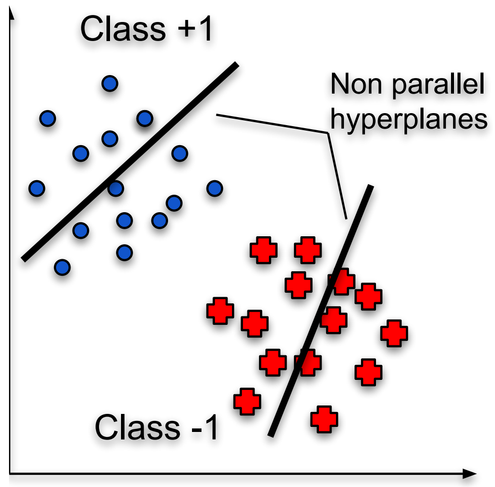
$$s_i = \begin{cases} \mu\big(1 - ||\boldsymbol{x_i} - x_{c+}||/(r_+ + \delta)\big) \\ \quad \text{if } ||\boldsymbol{x_i} - x_{c+}|| \geq ||\boldsymbol{x_i} - x_{c-}|| \,\wedge\, y_i = +1 \\ (1-\mu)\big(1 - ||\boldsymbol{x_i} - x_{c+}||/(r_+ + \delta)\big) \\ \quad \text{if } ||\boldsymbol{x_i} - x_{c+}|| < ||\boldsymbol{x_i} - x_{c-}|| \,\wedge\, y_i = +1 \end{cases} \tag{27}$$

where $\mu \in [0,1]$ is used to balance the effect of normal and noisy data points, and $\delta > 0$ is used to avoid fuzzy numbers equal 0. A data point is usually assigned by a proportional decreasing value s_i when it drifts farther from its native class center, which increases the uncertainty (GAO; WANG, 2017). A small positive real number μ is assigned to decrease the effect of outliers towards the hyperplane. The fuzzy numbers for the negative data points are calculated in an analogous manner.

5.3.2 THE TWIN SVM (TWSVM)

The TWSVM (JAYADEVA; KHEMCHANDANI, R.; CHANDRA, 2007) generates two non-parallel hyperplanes such that each hyperplane is closer to one class and is as far as possible from the other class (TOMAR; AGARWAL, 2015; JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, 2017b) as shown in Fig. 15. The two non-parallel decision planes are defined as:

Figura 15 – Binary classification using the TWSVM, inspired by (TOMAR; AGARWAL, 2015)



$$\boldsymbol{\omega}_+^\top \mathbf{x} + b_+ = 0 \text{ and } \boldsymbol{\omega}_-^\top \mathbf{x} + b_- = 0 \tag{28}$$

where $\boldsymbol{\omega}_+, \boldsymbol{\omega}_- \in \mathcal{R}^n$ indicate normal vectors to the hyperplane, and $b_+, b_- \in \mathcal{R}^n$ are the bias terms.

Considering a soft margin hyperplane to handle non-linearly separable data, the following pair of primal optimization problems is the set up to build the decision planes:

$$\begin{aligned} \min_{\boldsymbol{\omega}_+, b_+, \xi_-} \quad & \frac{1}{2}||X_+ \, \boldsymbol{\omega}_+ + \boldsymbol{e}_+ \, b_+||^2 + C_1 \, \boldsymbol{e}_-^\top \, \xi_- \\ \text{s.t. } & y_-(X_- \, \boldsymbol{\omega}_+ + \boldsymbol{e}_- \, b_+) + \xi_- \leq \boldsymbol{e}_-, \;\; \xi_- \geq 0 \end{aligned} \tag{29}$$

and

$$\begin{aligned} \min_{\boldsymbol{\omega}_-, b_-, \xi_+} \quad & \frac{1}{2}||X_- \, \boldsymbol{\omega}_- + \boldsymbol{e}_- \, b_-||^2 + C_2 \, \boldsymbol{e}_+^\top \, \xi_+ \\ \text{s.t. } & y_+(X_+ \, \boldsymbol{\omega}_- + \boldsymbol{e}_+ \, b_-) + \xi_+ \leq \boldsymbol{e}_+, \;\; \xi_+ \geq 0 \end{aligned} \tag{30}$$

where $C_1{>}0$ and $C_2{>}0$ are the penalty factors that trade-off the complexity and data misfit between the minimization of the two terms in the objective function, ξ_+ and ξ_- denote the slack variable vectors (the deviation from the margin that allows subsets of misclassification error for positive and negative classes respectively), $\boldsymbol{e}_-$, $\boldsymbol{e}_-$ correspond to unit row vectors with their dimensions exact to data point size in each class used for mathematical purpose only, y_+ and y_- are $+1$ and -1 respectively. In each QPP (Eqs. 29 and 30) the objective function corresponds to a particular class and the constraints are set by the data points of the opposite class. Assuming that the TWSVM is split into two QPPs of size $n/2$, and that the complexity of the original SVM is less or equal to n^3, the TWSVM is approximately four times faster than the original SVM ($2 \times (n/2)^3 = n^3/4$)(JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, 2017b).

After solving the Eqs. 29 and 30 for $(\boldsymbol{w}_+^*, b_+^*)$ and $(\boldsymbol{w}_-^*, b_-^*)$, respectively, we can classify a new data point $\boldsymbol{x}$ by:

$$f(x) = argmin_{\pm} \frac{|\omega_{\pm}^{*\top} + b_{\pm}^*|}{||\omega_{\pm}^*||} \tag{31}$$

and choose either $+1$ or -1 according to the lowest value of Eq. 31.

We can write Eqs. 29 and 30 as an unconstrained problem using Lagrangian multipliers. The dual formulation of the linear TWSVM for Eq. 29 is:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \boldsymbol{e}_-^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top H_-(H_+^\top H_+)^{-1} H_-^\top \boldsymbol{\alpha} \\ \text{s.t. } & 0 \leq \boldsymbol{\alpha} \leq C_1 \end{aligned} \tag{32}$$

where $H_+{=}[X_+, e_+]$, $H_-{=}[X_-, e_-]$, and $\boldsymbol{\alpha}{=}(\alpha_1, \ldots, \alpha_m)^\top$ is the Lagrangian vector.

In a similar manner we can write the dual formulation for Eq 30 as:

$$\begin{aligned} \max_{\boldsymbol{\nu}} \quad & \boldsymbol{e}_+^\top \, \boldsymbol{\nu} - \frac{1}{2}\boldsymbol{\nu}^\top H_+(H_-^\top H_-)^{-1} H_+^\top \boldsymbol{\nu} \\ \text{s.t. } & 0 \leq \boldsymbol{\nu} \leq C_2 \end{aligned} \tag{33}$$

where $\boldsymbol{\nu}$=$(\nu_1, \nu_2, \ldots, \nu_m)$ is the Lagrangian vector. For more detail on the dual formulation one may refer to (JAYADEVA; KHEMCHANDANI, R.; CHANDRA, 2007; JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, 2017b). Once we solve dual problems for $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$, we can get the vectors $[\boldsymbol{\omega}_{+}, b_{+}]^{\top}$ and $[\boldsymbol{\omega}_{-}, b_{-}]^{\top}$. Thus, the separating hyperplanes are given by:

$$\boldsymbol{x}^{\top}\boldsymbol{\omega}_{+} + b_{+} = 0 \text{ and } \boldsymbol{x}^{\top}\boldsymbol{\omega}_{-} + b_{-} = 0 \tag{34}$$

During testing, a new data point is assigned to the closest hyperplane regarding the two classes by:

$$class(\boldsymbol{x}) = argmin_{i=\{-1,+1\}}(d_r(\boldsymbol{x})) \tag{35}$$

where

$$d_r(x) = \frac{|\boldsymbol{x}^{\top}\boldsymbol{\omega_r} + b_r|}{||\boldsymbol{\omega^r}||} \tag{36}$$

5.3.3 KERNEL APPROXIMATION

Kernel machines that operate on the data kernel matrix (Gram matrix) scale more than quadratically in the data dimension (RAHIMI; RECHT, 2008; LI; IONESCU; SMINCHISESCU, 2010). This makes methods as the ISVM or the LASVM impractical to deal with large datasets or incremental data that requires sequential learning. Approximating non-linear kernels by linear kernels in the transformed space is a way to make possible the use of efficient linear methods that depend linearly on the size of the training set, allowing to solve large-scale and incremental learning problems efficiently (RAHIMI; RECHT, 2008; LI; IONESCU; SMINCHISESCU, 2010). Instead of relying on the kernel trick implicit lifting, the Random Fourier Features (RAHIMI; RECHT, 2008) explicitly map the data to a low-dimensional Euclidean inner product using a randomized feature map $z\colon \mathcal{R}^n \rightarrow \mathcal{R}^N$, described as:

$$\kappa(\boldsymbol{x_1}, \boldsymbol{x_2}) = \langle \varphi(\boldsymbol{x_1}), \varphi(\boldsymbol{x_2}) \rangle \approx z(\boldsymbol{x_1})^{\top} z(\boldsymbol{x_2}) \tag{37}$$

where z is a low-dimensional space. The feature space approximates shift-invariant kernels $\kappa(\boldsymbol{x_1} - \boldsymbol{x_2})$ to within an error err with $N = O(err^{-2} n \log \frac{1}{err^2})$ dimensions. Rahimi and Recth (RAHIMI; RECHT, 2008) show empirically that a similar classification performance can be obtained for dimensions smaller than N.

The first set of transformed features are the Random Fourier bases $\cos(\tau^{\top}\boldsymbol{x} + b)$, where $\tau \in \mathcal{R}^n$ and $b \in \mathcal{R}$ are random variables. It maps projected data on a randomly chosen line, followed by passing the resulting scalar through a sinusoidal function. The direction of these lines, in an appropriate distribution, guarantees that the product of two transformed points approximates a desired shift-invariant kernel (RAHIMI; RECHT, 2008). The transformation follows Bochner's theorem: *A continuous kernel $\kappa(x, y) = \kappa(x - y)$ on $\mathcal{R}^n$ is positive definite if and only if $\kappa(\delta)$ is the Fourier transform of a non-negative measure.* For a properly scaled shift-invariant kernel $\kappa(\delta)$ Bochener's guarantees that its Fourier transform $p(\tau)$ is a proper probability distribution:

$$\kappa(x - y) = \int_{\mathcal{R}^n} p(\tau)e^{j\tau^\top(x-y)}d\tau = E_\tau[\zeta_\tau(x)\zeta_\tau(y)^*] \tag{38}$$

where $\zeta_\tau(x) = e^{j\tau^\top x}$. $\zeta_\tau(x)\zeta_\tau(y)^*$ is an unbiased estimate of $k(x, y)$ when τ is drawn for p (note that here $*$ is the complex conjugate). The integral of Eq. 38 converges when the complex exponentials are replaced by cosines, $z_\tau(x)$=$\sqrt{2}cos(\tau^\top x + b)$, obtaining a real-valued mapping that satisfies the condition $E[z_\tau(x)z_\tau(y)]$, where τ is drawn from $p(\tau)$ and b is uniformly distributed from $[0, 2\pi]$. The variance of the estimate of the kernel can be reduced by concatenating N randomly chosen z_τ into one N-dimensional normalized vector, .i.e. the inner product $z(x)^\top z(y) = \frac{1}{N}\sum_{j=1}^{N} z_\tau(x)z_\tau(y)$ is a lower variance approximation to the expectation of Eq. 38[1]

To summarize, the Random Fourier Feature algorithm starts by getting a randomized feature map $z(x)$:$\mathcal{R}^n \rightarrow \mathcal{R}^N$, so that $z(x)^\top z(y) \approx k(x - y)$. The second step is to compute the Fourier transform or p of the kernel k as,

$$p(\tau) = \frac{1}{2\pi}\int e^{j\tau^\top\delta}k(\delta)d\Delta \tag{39}$$

The third step is to draw N independent and identically distributed (iid) data points $\tau_1, ..., \tau_N \in \mathcal{R}^n$ from p and N iid data points $b_1, ..., b_N \in \mathcal{R}$ from the uniform distribution on $[0, 2\pi]$. Finally, $z(x)$ is computed as:

$$z(\boldsymbol{x}) \equiv \sqrt{\frac{2}{N}}[cos(\tau_1^\top \boldsymbol{x} + b_1), \ldots, cos(\tau_N^\top \boldsymbol{x} + b_N)]^\top \tag{40}$$

The scalar σ_p^2 is equal to the trace of Hessian of k at 0, that quantifies the curvature of the kernel at the origin. For a Gaussian kernel denoted as $k(\boldsymbol{x_1}, \boldsymbol{x_2})$=$\exp(-\gamma||\boldsymbol{x_1} - \boldsymbol{x_2}||^2)$, we have σ_p^2=$2n\gamma$, that approximates the kernel to:

$$p(\boldsymbol{\tau}) = 2\pi^{-\frac{N}{2}}e^{-\frac{||\tau||_2^2}{2}} \tag{41}$$

The important implications of using this kernel approximation in our incremental approach are: (i) we approximate the non-linear model accuracy with a linear model; (ii) it is faster to calculate the approximate kernel than the regular kernel; (iii) and mainly, we increment the model only in one dimension, so we do not need to recalculate the kernel approximation for the previous data.

[1] The proof can be found in (RAHIMI; RECHT, 2008).

5.4 THE FUZZY BOUNDED TWIN SVM (FBTWSVM)

We present a formulation based on the original TWSVM (JAYADEVA; KHEMCHANDANI, R.; CHANDRA, 2007) and inspired by the FRTSVM (GAO; WANG *et al.*, 2015; GAO; WANG, 2017) and by the TBSVM (SHAO; ZHANG *et al.*, 2011) to include the fuzzy formulation (Eq. 24) in the TWSVM (Eqs. 29 and 30) and write the duals. We also incorporate the TBSVM (SHAO; ZHANG *et al.*, 2011) solution to maintain the structural risk minimization principle by automatically getting the dual formulation inverse matrix guarantee to circumvent the drawback of the standard TWSVM formulation that only adheres to the empirical risk minimization principle in the dual problem. The FBTWSVM primal formulation is defined as:

$$\begin{aligned} \min_{\boldsymbol{\omega}_+, b_+, \boldsymbol{\xi}_-} \quad & \frac{1}{2}C_1(||\boldsymbol{\omega}_+||^2 + b_+^2) + \frac{1}{2}||X_+\ \boldsymbol{\omega}_+ + \boldsymbol{e}_+\ b_+||^2 \\ & + C_3 \boldsymbol{s}_-^\top\ \boldsymbol{\xi}_- \\ \text{s.t. } & y_-(X_-\ \boldsymbol{\omega}_+ + \boldsymbol{e}_-\ b_+) + \boldsymbol{\xi}_- \geq \boldsymbol{e}_-, \ \ \boldsymbol{\xi}_- \geq 0 \end{aligned} \tag{42}$$

$$\begin{aligned} \min_{\boldsymbol{\omega}_-, b_-, \boldsymbol{\xi}_+} \quad & \frac{1}{2}C_2(||\boldsymbol{\omega}_-||^2 + b_-^2) + \frac{1}{2}||X_-\ \boldsymbol{\omega}_- + \boldsymbol{e}_-\ b_-||^2 \\ & + C_4 \boldsymbol{s}_+^\top\ \boldsymbol{\xi}_+ \\ \text{s.t. } & y_+(X_+\ \boldsymbol{\omega}_- + \boldsymbol{e}_+\ b_-) + \boldsymbol{\xi}_+ \geq \boldsymbol{e}_+, \ \ \boldsymbol{\xi}_+ \geq 0 \end{aligned} \tag{43}$$

where C_1, C_2, C_3, and C_4 are the trade-off parameters between the margin and the complexity for weighting the regularization, $\boldsymbol{s}_+ \in R^{l_+}$ and $\boldsymbol{s}_- \in R^{l_-}$ are the fuzzy number vectors sequentially associated with the positive and negative input data points, which introduce the desired robustness in the weighted regularized model (GAO; WANG *et al.*, 2015; GAO; WANG, 2017). The additional b_+ and b_- in Eqs. 42 and 43 minimize the structural risk.

The two hyperplanes in $\mathcal{R}^n$ are defined as $\boldsymbol{\omega}_\pm^\top + b_\pm = 0$, and since the TWSVM has two proximal decision functions, two margin terms $1/||\boldsymbol{\omega}_\pm||$ are defined for the proximal decision function (GAO; WANG *et al.*, 2015). The margin between two classes can be measured by the distance between the proximal hyperplane $\boldsymbol{x}^\top \boldsymbol{\omega}_+ + b_+ = 0$ and the bounding hyperplane $\boldsymbol{x}^\top \boldsymbol{\omega}_+ + b_+ = -1$. The distance is $1/||\boldsymbol{\omega}_+||^2$, and it is the one-sided margin between the two classes with respect to the hyperplane $\boldsymbol{x}^\top \boldsymbol{\omega}_+ + b_+ = 0$ (SHAO; ZHANG *et al.*, 2011; JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, 2017b). The process is analogous to the other hyperplane.

We need to derive the dual problems to obtain the solutions of Eqs. 42 and 43.

We start by taking the Lagrangian of Eq. 42 to obtain the Wolfe dual:

$$\begin{aligned} L(\boldsymbol{\omega}_+, b_+, \boldsymbol{\xi}_-) = {} & \frac{1}{2}C_1(||\boldsymbol{\omega}_+||^2 + b_+^2) + \frac{1}{2}||X_+\boldsymbol{\omega}_+ + \boldsymbol{e}_+ b_+||^2 \\ & - \boldsymbol{\alpha}^\top(-(X_-\ \boldsymbol{\omega}_+ + \boldsymbol{e}_-\ b_+) + \xi_-\ -\ \boldsymbol{e}_-) \\ & + C_3\ \boldsymbol{s}_-^\top\ \boldsymbol{\xi}_-\ -\ \eta^\top \boldsymbol{\xi}_- \end{aligned} \tag{44}$$

where $\boldsymbol{\alpha}$=$(\alpha_1, \ldots, \alpha_{X_+})^\top$, and $\boldsymbol{\eta}$=$(\eta_1, \ldots, \eta_{X_+})^\top$ are the Lagrange multiplier vectors. Considering that Eq. 42 is a convex optimization problem, the Karush-Kuhn-Tucker (KKT) optimality conditions are both necessary and sufficient, and they are written as:

$$\begin{aligned} \nabla\boldsymbol{\omega}_+\ L = C_1\ \boldsymbol{\omega}_+ + X_+^\top(X_+\ \boldsymbol{\omega}_+ + \boldsymbol{e}_+\ b_+) \\ + X_-^\top\ \boldsymbol{\alpha} = 0 \end{aligned} \tag{45a}$$

$$\begin{aligned} \nabla b_+\ L = C_1\ b_+ + \boldsymbol{e}_+^\top(X_+\ \boldsymbol{\omega}_+ + \boldsymbol{e}_+\ b_+) \\ + \boldsymbol{e}_-^\top\ \boldsymbol{\alpha} = 0 \end{aligned} \tag{45b}$$

$$\nabla\boldsymbol{\xi}_-\ L = -\boldsymbol{\alpha}^\top - \boldsymbol{\eta}^\top + C_3\ \boldsymbol{s}_- = 0 \tag{45c}$$

$$-(X_-\ \boldsymbol{\omega}_+ + \boldsymbol{e}_-\ b_+) + \boldsymbol{\xi}_-\ \geq\ \boldsymbol{e}_-\boldsymbol{\xi}_-\ \geq 0 \tag{45d}$$

$$\boldsymbol{\alpha}^\top(\boldsymbol{\omega}_-\ X_+ + \boldsymbol{e}_-\ b_+ - \boldsymbol{\xi}_- + \boldsymbol{e}_-) = 0; \quad \boldsymbol{\eta}^\top\boldsymbol{\xi}_- = 0 \tag{45e}$$

$$\boldsymbol{\alpha} \geq 0, \quad \boldsymbol{\eta} \geq 0, \quad \boldsymbol{\xi}_- \geq 0 \tag{45f}$$

Considering that $\eta \geq 0$ and $\alpha \geq 0$ from Eq. 45f, and using Eq. 45c, we know that α is bounded as $0 \leq \alpha \leq C_3 s_-$. Summing Eqs. 45a and 45b, and using Eqs. 45c to 45f for simplification, we obtain:

$$([X_+, \boldsymbol{e}_+]^\top[X_+, \boldsymbol{e}_+] + C_1 I)[\boldsymbol{\omega}_+, b_+] + [X_-, \boldsymbol{e}_-]^\top\boldsymbol{\alpha} = 0 \tag{46}$$

Defining H_+=$[X_+, \boldsymbol{e}_+]$, H_-=$[X_-, \boldsymbol{e}_-]$, $\boldsymbol{u}_+$=$[\boldsymbol{\omega}_+, b_+]$ and $\boldsymbol{u}_-$=$[\omega_-, b_-]$ (one to each class), we can rewrite Eq. 46 as:

$$\begin{aligned} (H_+^\top H_+ + C_1 I)\boldsymbol{u}_+^\top + H_-^\top\ \boldsymbol{\alpha} = 0 \quad \text{or} \\ \boldsymbol{u}_+^\top = -(H_+^\top H_+ + C_1 I)^{-1} H_-^\top\ \boldsymbol{\alpha} \end{aligned} \tag{47}$$

Using our notation, the Wolfe dual is defined as:

$$\begin{aligned} \max\ \ & L(\boldsymbol{\omega}_+, b_+, \boldsymbol{\xi}_-, \boldsymbol{\alpha}, \boldsymbol{\eta}) \\ \text{s.t}\ \ & \nabla_{\omega_+} L(\boldsymbol{\omega}_+, b_+, \boldsymbol{\xi}_-, \boldsymbol{\alpha}, \boldsymbol{\eta}) \\ & \frac{\partial L}{\partial b_+} = 0 \\ & \frac{\partial L}{\partial \boldsymbol{\xi}_-} = 0 \\ & \boldsymbol{\alpha} \geq 0, \quad \boldsymbol{\eta} \geq 0 \end{aligned} \tag{48}$$

Using the KKT conditions (from Eqs. 45a to 45f) and Eq. 47, the Wolfe dual of Eqs. 42 and 43 can be written as:

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \boldsymbol{e}_-^\top \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top H_-(H_+^\top H_+ + C_1 I_1)^{-1} H_-^\top \boldsymbol{\alpha} \\ \text{s.t. } & 0 \leq \boldsymbol{\alpha} \leq C_3 \boldsymbol{s}_-\end{aligned} \tag{49}$$

$$\begin{aligned}\max_{\boldsymbol{\nu}} \quad & \boldsymbol{e}_+^\top \boldsymbol{\nu} - \frac{1}{2}\boldsymbol{\nu}^\top H_+(H_-^\top H_- + C_2 I_2)^{-1} H_+^\top \boldsymbol{\nu} \\ \text{s.t. } & 0 \leq \boldsymbol{\nu} \leq C_4 \boldsymbol{s}_+\end{aligned} \tag{50}$$

where I_1 and I_2 are identity matrices. The matrices $(H_+^\top H_+ + C_1 I_1)$ and $(H_-^\top H_- + C_2 I_2)$ from Eqs. 49 and 50 are non-singular naturally, therefore their inverses are guaranteed to exist, which adds the adherence to the structural risk minimization principle (SHAO; ZHANG *et al.*, 2011; JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, 2017b). Notice that the dual for Eq. 43 can be obtained is an analogous way.

By solving the duals (Eqs. 49 and 50), we obtain the optimal solutions for $\boldsymbol{\alpha}^*$ and $\boldsymbol{\nu}^*$, and furthermore, the corresponding classes $\boldsymbol{u}_\pm^*$ (as defined in Eq. 47) and the non-parallel hyperplanes. The dual of Eq. 49 and 50 relates to the primal problems (Eqs. 42 and 43) as:

$$\begin{aligned}\boldsymbol{u}_+^* &= -(H_+^\top H_+ + C_1 I_1)^{-1} H_-^\top \boldsymbol{\alpha}^* \\ \boldsymbol{u}_-^* &= (H_-^\top H_- + C_2 I_2)^{-1} H_+^\top \boldsymbol{\nu}^*\end{aligned} \tag{51}$$

Finally, for a test data point $\boldsymbol{x} \in \mathcal{R}^n$, the classification decision function is given by Eq. 31.

5.4.1 THE NON-LINEAR FBTWSVM

In the non-linear FBTWSVM, the input data points $x \in \mathcal{R}^n$ are mapped to a high-dimensional space $\mathcal{H}$ through $\varphi(x)$. The kernel function $\kappa(\cdot,\cdot)$ calculates implicitly the dot product of a pair of transformations, which is applied as $\kappa(x_1, x_2) = \langle\varphi(x_1), \varphi(x_2)\rangle$. The non-linear dual proximal hyperplanes are:

$$\begin{aligned}\kappa(x, x^\top)\boldsymbol{\omega}_+ + b_+ = 0 \\ \kappa(x, x^\top)\boldsymbol{\omega}_- + b_- = 0\end{aligned} \tag{52}$$

and the primal problems used to obtain the dual proximal hyperplanes are:

$$\begin{aligned}\min_{\boldsymbol{\omega}_+, b_+, \boldsymbol{\xi}_-} \quad & \frac{1}{2}C_1(||\boldsymbol{\omega}_+||^2 + b_+^2) + C_3\, \boldsymbol{s}_-^\top\, \boldsymbol{\xi}_- \\ & + \frac{1}{2}||\kappa(X_+, X^\top)\boldsymbol{\omega}_+ + \boldsymbol{e}_+\, b_+||^2 \\ \text{s.t. } & y_-(\kappa(X_-, X^\top)\boldsymbol{\omega}_+ + \boldsymbol{e}_-\, b_+) + \boldsymbol{\xi}_- \geq \boldsymbol{e}_-, \;\; \boldsymbol{\xi}_- \geq 0\end{aligned} \tag{53}$$

$$\begin{aligned}\min_{\boldsymbol{\omega}_-, b_-, \boldsymbol{\xi}_+} \quad & \frac{1}{2}C_2(||\boldsymbol{\omega}_-||^2 + b_-^2) + C_4\ \boldsymbol{s}_+^\top\ \boldsymbol{\xi}_+ \\ & + \frac{1}{2}||\kappa(X_+, X^\top)\boldsymbol{\omega}_- + \boldsymbol{e}_-\ b_-||^2 \\ \text{s.t. } & y_+(\kappa(X_+, X^\top)\boldsymbol{\omega}_- + \boldsymbol{e}_+\ b_-) + \boldsymbol{\xi}_+ \geq \boldsymbol{e}_+,\ \ \boldsymbol{\xi}_+ \geq 0\end{aligned} \tag{54}$$

The dual forms of Eq. 53 and 54 are:

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \boldsymbol{e}_-^\top\ \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^\top S_-(S_+^\top S_+ + C_1 I_1)^{-1} S_-^\top\ \boldsymbol{\alpha} \\ \text{s.t. } & 0 \leq \boldsymbol{\alpha} \leq C_3 \boldsymbol{s}_-\end{aligned} \tag{55}$$

$$\begin{aligned}\max_{\boldsymbol{\nu}} \quad & \boldsymbol{e}_+^\top\ \boldsymbol{\nu} - \frac{1}{2}\boldsymbol{\nu}^\top S_+(S_-^\top S_- + C_2 I_2)^{-1} S_+^\top\ \boldsymbol{\nu} \\ \text{s.t. } & 0 \leq \boldsymbol{\nu} \leq C_4 \boldsymbol{s}_+\end{aligned} \tag{56}$$

where $S_+ = [\kappa(X_+, X^\top), \boldsymbol{e}_+]$ and $S_- = [\kappa(X_-, X^\top), \boldsymbol{e}_-]$.

The solutions of the primal problems of Eqs. 53 and 54 are $\boldsymbol{v}_\pm^* = [\boldsymbol{\omega}_\pm^{*\top}, b_\pm^*]^\top$, which are the parametric relationships between the optimal $\boldsymbol{v}_\pm^*$ and the optimal solutions $\boldsymbol{\alpha}^*$ and $\boldsymbol{\nu}^*$ of the dual forms of Eqs. 55 and 56:

$$\begin{aligned}v_+^* &= -(S_+^\top S_+ + C_1 I_1)^{-1} S_-^\top\ \alpha^* \\ v_-^* &= (S_-^\top S_- + C_2 I_2)^{-1} S_+^\top\ \nu^*\end{aligned} \tag{57}$$

Once Eqs. 55 and 56 are solved to obtain the surfaces (Eq. 52), a new data point $\boldsymbol{x} \in \mathcal{R}^n$ can be classified in a similar manner to the linear case by Eq. 31.

5.4.2 SOLVING THE FBTWSVM

The DCD method (CHANG; HSIEH; LIN, 2008), that was used by Shao and Deng (SHAO; DENG, 2012) to solve the TWSVM, is used to solve the dual problem of the FBTWSVM. The DCD leads to fast training by updating one variable at a time through a single-variable sub-problem minimization. Such a fast training allows the processing of large and incremental datasets (SHAO; DENG, 2012).

The dual problems of Eqs. 49 and 50 and Eqs. 55 and 56 are solved in the same way. However, for convenience, we only present the solution of Eq. 49. We start by considering $Q{=}H_-(H_+^\top H_+ + C_1 I_1)^{-1} H_-^\top$ and $Q'{=}(H_+^\top H_+ + C_1 I_1)^{-1} H_-^\top$. Consequently, $Q{=}H_- Q'$, where q_{ii} and $\overline{Q}$ can be pre-computed and stored if necessary. The matrix inversion is calculated with the Sherman-Morison-Woodbury formula. Assuming $\boldsymbol{\alpha}^{\boldsymbol{k},\boldsymbol{i}}{=}[\alpha_1^{k+1,i}, \ldots, \alpha_{i-1}^{k+1,i}, \alpha_i^{k,i}, \ldots, \alpha_{X_-+1}^{k,1}]$, where $i{=}(1, \ldots, X_-{+}1)$ is the index for the data points and $k = (-1, +1)$ is the data label. We use the following problem updating from $\boldsymbol{\alpha}^{\boldsymbol{k,i}}$ to $\boldsymbol{\alpha}^{\boldsymbol{k,i+1}}$,

$$\begin{aligned} &\min_{d} \quad f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\ \boldsymbol{e_i}) \\ &\text{s.t. } 0 \leq \boldsymbol{\alpha}_{\boldsymbol{i}}^{\boldsymbol{k}} + d \leq C_3 \boldsymbol{s}_- \end{aligned} \tag{58}$$

where $\boldsymbol{e_i}$=$[0, \ldots, 0, 1, 0, \ldots, 0]^\top$ (the $i-$th position is 1), and $\boldsymbol{d_i}$ is an optimum solution to the problem of minimizing $f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\boldsymbol{e_i})$ subject to $\boldsymbol{d_i} \in \mathcal{R}^n$, i.e., $f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\boldsymbol{e_i})$ achieves a minimum at $\boldsymbol{d_i}$ only if $\nabla f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\boldsymbol{e_i})^\top \boldsymbol{e_i} = \nabla f(\boldsymbol{\alpha}^{\boldsymbol{k,i+1}})^\top \boldsymbol{e_i} = 0$[2]. The objective function of Eq. 58 is a quadratic function of d:

$$f(\boldsymbol{\alpha}^{\boldsymbol{k,i}} + d\boldsymbol{e_i}) = \frac{1}{2} Q_{ii} \boldsymbol{d^2} + \nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}}) d + \text{constant} \tag{59}$$

where $\nabla_i f$ is the i-th component of the gradient ∇f. Eq. 58 has an optimum at d=0 iff:

$$\nabla_i^P f(\boldsymbol{\alpha}^{\boldsymbol{k,i}}) = 0 \tag{60}$$

where $\nabla_i^P f(\alpha)$ is the projected gradient which is defined as:

$$\nabla_i^P f(\alpha) = \begin{cases} \min(0, \nabla_i f(\alpha)), & \alpha_i = 0, \\ \nabla_i f(\alpha), & 0 \leq \alpha_i \leq C_3 s_- \\ \max(0, \nabla_i f(\alpha)), & \alpha_i = C_3 s_- \end{cases} \tag{61}$$

If Eq. 60 is satisfied, we can move to the next iteration (i+1) without updating $\boldsymbol{\alpha}_{\boldsymbol{i}}^{\boldsymbol{k,i}}$ in X_-, .i.e., we only update $\boldsymbol{\alpha}_{\boldsymbol{i}}^{\boldsymbol{k,i}}$ to temporally meet the optimal solution of Eq. 58. The optimum of Eq. 59 is reached by introducing the Lipschitz continuity:

$$\boldsymbol{\alpha}_{\boldsymbol{i}}^{\boldsymbol{k,i+1}} = \min(\max(\boldsymbol{\alpha}_{\boldsymbol{i}}^{\boldsymbol{k,i}} - \nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}})/Q'_{ii}, 0), C_3 s_{i_-}) \tag{62}$$

In the update of Eq. 62, $Q'_{i,i}$ can be pre-calculated by $Q'_{ii} = H_{-i} Q_i$, and $\nabla_i f(\boldsymbol{\alpha}^{\boldsymbol{k,i}})$ can is obtained by:

$$\nabla_i f(\alpha) = (Q'\alpha)_i - 1 = \sum_{j=1}^{X_-} Q'_{ij} \alpha_j - 1 \tag{63}$$

[2] The proof can be found in (CHANG; HSIEH; LIN, 2008)

The computation of Eq. 63 is approximated as $O(X_{-}\bar{l})$, where $\bar{l}$ is the average count of non-zero elements in Q' per data point. To reduce the number of operations, we can alternatively compute Eq. 63 as:

$$\nabla_i f(\alpha) = -H_{-i}u_{+} - 1 \tag{64}$$

with a predefined $\boldsymbol{u}_{+} = -Q\alpha$ and i is the row of the matrix H_{-}, so the number of operations is $O(\overline{n})$. To maintain $\boldsymbol{u}_{+}$ throughout coordinate descent procedure, we use:

$$\boldsymbol{u}_{+i} \leftarrow \boldsymbol{u}_{+i} - Q_i(\alpha_i - \overline{\alpha}_i) \tag{65}$$

The complexity to maintain $\boldsymbol{u}_{+}$ iteratively is $O(\bar{l})$. Starting with $\alpha^0 = 0$, the optimal solution of $\boldsymbol{u}_{+}$ is obtained by iterative updating Eq. 65, and furthermore, the optimal solution of Eq. 49. The cost per iteration for the whole process is $O(X_2\overline{n})$, and the memory requirement is the size of H_{-} and Q'.

5.4.3 IMPLEMENTATION

The dual problem of Eq. 49 has the constraint $0 \leq \alpha_i \leq C_3\boldsymbol{s}_{-}$, and if α_i is either 0 or $C_3\boldsymbol{s}_{-}$, it may achieve a steady state. Considering that our formulation produces many bounded Lagrange multipliers, we apply the proposed shrinking technique to reduce the size of the optimization problem without considering some bounded variables (JOACHIMS, 1999).

Considering Z as a subset of X after removing all data points that have non-bounded Lagrange multipliers, and $\overline{Z}=\{1,\ldots,X_{-}\}/Z$ its complement subset, the dual of Eq. 49 can be represented by a smaller problem that consumes less time and memory:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}_Z} \frac{1}{2}\boldsymbol{\alpha}_Z^\top Q'_{ZZ}\boldsymbol{\alpha}_Z + (Q'_{Z\overline{Z}}\boldsymbol{\alpha}_{\overline{Z}} - \boldsymbol{e}_Z)^\top \boldsymbol{\alpha}_Z \\ &\text{s.t. } 0 \leq \boldsymbol{\alpha}_Z \leq C_3\boldsymbol{s}_{-Z} \end{aligned} \tag{66}$$

where Q_{ZZ} and $Q_{Z\overline{Z}}$ are sub-matrices of Z and $\boldsymbol{\alpha}_{\overline{Z}}$ is a vector of Lagrangian multipliers. To solve Eq. 66, we compute $\nabla_i f(\boldsymbol{\alpha})$ as:

$$\nabla_i f(\boldsymbol{\alpha}) = Q_{i,Z}\boldsymbol{\alpha}_Z + Q_{i,\overline{Z}}\boldsymbol{\alpha}_{\overline{Z}} - 1 \tag{67}$$

If $i \in Z$, and defining $\boldsymbol{u_1}$ as:

$$\boldsymbol{u_1} = -(Q'_{i\in Z}\boldsymbol{\alpha}_{i\in Z} + Q'_{i\in\overline{Z}}\boldsymbol{\alpha}_{i\in\overline{Z}}) \tag{68}$$

We have $\nabla_i f(\boldsymbol{\alpha})$=$H_{-}\boldsymbol{u_1}-1$, which turns $\nabla_i f(\alpha)$ easy to obtain. For a linear kernel we only need to update $(\overline{Q}_{i\in Z}\boldsymbol{\alpha}_{i\in Z})$, and we do not need to reconstruct all $\nabla f(\boldsymbol{\alpha})$ to implement the shrink step [3].

Considering the projected gradient $\nabla^P f(\boldsymbol{\alpha})$ defined in Eq. 61, and following the optimality condition of bound-constrained problems, α is optimal iff $\nabla^P f(\boldsymbol{\alpha}) = 0$. During the iteration procedure, the inequality $\nabla^P f(\boldsymbol{\alpha}) \neq 0$ means either $\max_j \nabla^P f(\boldsymbol{\alpha}) > 0$ or $\min_j \nabla^P f(\boldsymbol{\alpha}) < 0$, and at the $k-1$ step, we obtain $m^{k-1}_{\max} \equiv \max_j \nabla^P f(\boldsymbol{\alpha})$ and $m^{k-1}_{\min} \equiv \min_j \nabla^P f(\boldsymbol{\alpha})$. In this way, at each inner step of the $k-$th iteration, and before updating $\boldsymbol{\alpha}$, the element is shrunken if one of the two conditions holds:

$$\begin{gathered} \alpha_i^{k,i} = 0 \quad \text{and} \quad \nabla^P f(\boldsymbol{\alpha^{k,i}}) > {m'}^{k-1}_{\max} \\ \text{or} \\ \alpha_i^{k,i} = C_3\boldsymbol{s}_{-} \quad \text{and} \quad \nabla^P f(\boldsymbol{\alpha^{k,i}}) < {m'}^{k-1}_{\min} \end{gathered} \tag{69}$$

where ${m'}^{k-1}_{\max}$ must be strictly positive and ${m'}^{k-1}_{\min}$ must be strictly negative, and they are defined as:

$${m'}^{k-1}_{\max} = \begin{cases} m^{k-1}_{\max}, & \text{if } m^{k-1}_{\max} > 0, \\ \infty, & otherwise \end{cases} \tag{70}$$

$${m'}^{k-1}_{\min} = \begin{cases} m^{k-1}_{\max}, & \text{if } m^{k-1}_{\min} < 0, \\ -\infty, & otherwise \end{cases} \tag{71}$$

Next, we multiply both ${m'}^{k}_{\max}-1$ and ${m'}^{k}_{\min}-1$ by a shrinking rate smaller than one. A tolerance ϵ indicates if the optimal value is satisfied after a finite number of iterations, thus it is used as a valid stop criteria:

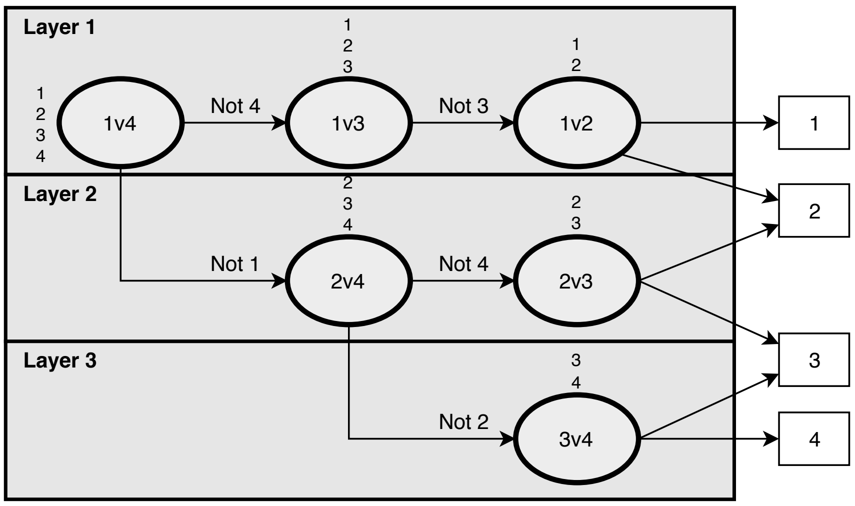
$${m'}^{k}_{\max} - {m'}^{k}_{\min} < \epsilon \tag{72}$$

If in the $k-$th iteration the condition stated in Eq. 72 is satisfied for Eq. 66, we can enlarge the active set Z to $\{1,\ldots,X_{-}+1\}$, and set ${m'}^{k}_{\max}$ $=\infty$ and ${m'}^{k}_{\min}$ $=-\infty$, and continue with the regular iterations. We store the previous values of ${m'}_{\max}$ and ${m'}_{\min}$ during the DCD process to avoid recalculation them during the incremental step. Therefore, the shrinking technique is a key step to avoid calculating and storing all training data during the training phase.

Our method process one class at each time, however, the inner processing can be done in parallel, where one input is assigned to an available processor to calculate

[3] The proof can be found in (HSIEH *et al.*, 2008).

Figura 16 – A 4-class classification problem based one the DAG topology.



the fuzzy membership followed by the Lagrangian multiplier. We present the pseudo-code of the FBTWSVM training algorithm (Alg. 2) for the positive class.

5.4.4 THE MULTICLASS FBTWSVM

The FBTWSVM is inspired by the TWSVM which considers only binary problems. However, we can extend the FBTWSVM to multiple classes by building and combining several binary classifiers instead of considering all data in one optimization formula (HSU; LIN, 2002). The multiclass FBTWSVM is based on the Decision DAG, which achieves better accuracy while requiring less training time than other multiclass approaches (TOMAR; AGARWAL, 2015; PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000). The DAG-based multiclass classifier was originally proposed by Platt et al. (PLATT; CRISTIANINI; SHAWE-TAYLOR, 2000) for the multiclass SVM approach, and further introduced by Chen and Ji (CHEN; JI, 2010) into the Twin approach as the Optimal Directed Acyclic Graph to the Least Squares Twin Support Vector Machine (ODAG-LSTSVM).

The multiclass approach is based on the DAG topology, hence for an $u-$class classification problem, there are $u(u-1)/2$ sub-classifier nodes divided into $u-1$ layers. During the classification process, there is no need for combining all sub-classifiers, so to assign a class to a test data point it takes $u-1$ decisions. The classification process starts at the root node, located in the first layer and includes all possible classification labels (node 1v4 in Fig. 16). The decision-making step eliminates the most excluded category at each sub-classifier decision, i.e., considering a 4-class problem with a test data point with label y_i $=4$ and the topology presented in Fig. 16. The root node sub-classifier eliminates the possibility of $y_i = 1$, following the *Not* 1 line. The next sub-classifier eliminates the possibility of y_i $=2$ following the *Not* 2, and the last sub-classifier eliminates the possibility of y_i $=3$, assigning class 4 to the test data point.

Algorithm 2 Training procedure of FBTWSVM

Input: X, **y**, C_1,C_2,C_3,C_4

Output: $\boldsymbol{u}_+$

```
 1: Compute $Q = (H_+^\top H_+ + C_1 I)^{-1} H_-^\top$ and $Q'_{ii} = H_{-i} Q_i$
 2: Let $Z = \{1, \ldots, X_-\}$
 3: Given $\epsilon$, $\boldsymbol{\alpha} = 0$ and $\boldsymbol{u}_+ = 0$
 4: $m'_{\max} = \infty$ and $m'_{\min} = -\infty$
 5: while  do
 6:     Let $m_{\max} = -\infty$, $m_{\min} = \infty$
 7:     for $\forall i \in Z$, (a randomly and exclusively selected) do        ▷ this thread runs in parallel
 8:         $\nabla_i f(\boldsymbol{\alpha}) = -H_{-i} \boldsymbol{u}_+ - 1$
 9:         $\nabla_i^P f(\boldsymbol{\alpha}) = 0$
10:         if $\alpha_i = 0$ then
11:             if $\nabla_i^P f(\boldsymbol{\alpha}) > m'_{\max}$ then $X = X/\{i\}$
12:             end if
13:             if $\nabla_i^P f(\boldsymbol{\alpha}) < 0$ then $\nabla_i^P f(\boldsymbol{\alpha}) = \nabla_i f(\boldsymbol{\alpha})$
14:             end if
15:             if $\alpha_i = C_3 s_-$ then
16:                 if $\nabla_i^P f(\boldsymbol{\alpha}) < m'_{\min}$ then $X = X/\{i\}$
17:                 end if
18:                 if $\nabla_i^P f(\boldsymbol{\alpha}) > 0$ then $\nabla_i^P f(\boldsymbol{\alpha}) = \nabla_i f(\boldsymbol{\alpha})$
19:                 end if
20:             end if
21:         else$\nabla_i^P f(\boldsymbol{\alpha}) = \nabla_i f(\boldsymbol{\alpha})$
22:         end if
23:         $m_{\max} = \max(m_{\max}, \nabla_i^P f(\boldsymbol{\alpha}))$
24:         $m_{\min} = \min(m_{\min}, \nabla_i^P f(\boldsymbol{\alpha}))$
25:         if $\nabla_i^P f(\boldsymbol{\alpha}) \neq 0$ then
26:             $\overline{\alpha} = \alpha_i$
27:             $\alpha_i = \min(\max(\alpha_i - \nabla_i f(\boldsymbol{\alpha})/Q'_{ii}, 0) C_3 s_{i-})$
28:             $u_{+i} = u_{+i} - Q_i(\alpha_i - \overline{\alpha_i})$
29:         end if
30:     end for
31:     if $m_{\max} - m_{\min} < \epsilon$ then
32:         if $X = \{1, \ldots, l_-\}$, then break
33:         end if
34:     else
35:         $X = \{1, \ldots, l_-\}, m'_{\max} = \infty, m'_{\min} = -\infty$
36:     end if
37:     if $m_{\max} \leq 0$ then $m'_{\max} = \infty$
38:     else $m'_{\max} = m_{\max}$
39:     end if
40:     if $m_{\max} \geq 0$ then $m'_{\min} = -\infty$
41:     else $m'_{\min} = m_{\min}$
42:     end if
43: end while
44: return $\boldsymbol{u}_+$
```

5.5 INCREMENTAL AND DECREMENTAL FBTWSVM

There are some properties of the FBTWSVM that suits the incremental learning, as the generalization capability similar to SVM, the fast training (due to the formulation and the solver choice), and mainly, it can continuously integrate new data points into the already constructed model without reconstructing the model or degrading its accuracy. The incremental FBTWSVM is based on the shrinking heuristic that can increment the current model considering the fuzzy information of new values. We update the model by selecting only new data points that extrapolate the minimum ($m'_{\min}$) and maximum ($m'_{\max}$) values of the projected gradient from the previous training step. Therefore, we do not need to process all the new incoming data points.

Considering a set of new data points as X_{new}, and the subsets X'_{new+} and X'_{new-} denoting the positive and negative labeled data respectively. Since not necessarily both subsets may exist, here we consider that $X_{new} = X'_{new+}$ to maintain the notation. We evaluate the projected gradient of the new set of data points as:

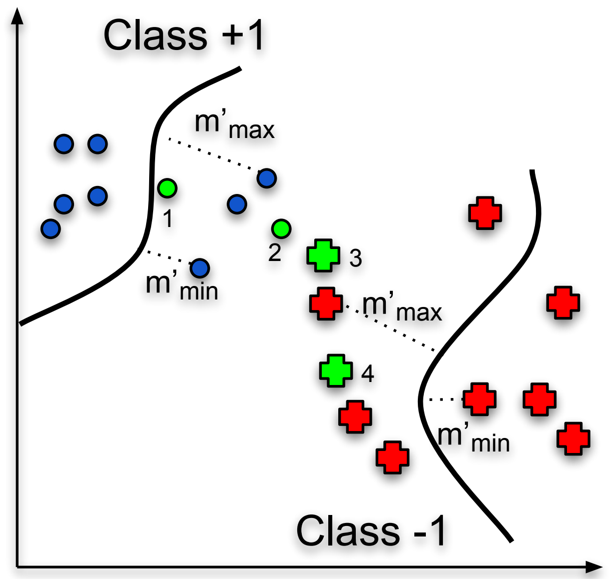
$$\nabla_i f(\boldsymbol{\alpha_{new}}) = X'_{new+} \pm \boldsymbol{u}_+ - 1 \tag{73}$$

This operation is used to maintain u_+ in the coordinate descent procedure in Eq. 64. We set a new heuristic based on the Eq. 69 rule to select only new data points that are more likely to become SVs. We consider the new data points as SVs if the projected gradient values are bounded by $(m'_{\min} < \nabla_i f(\boldsymbol{\alpha_{new}}) < m'_{\max})$. As our method adheres structural and risk minimization principle, all Lagrangian multipliers can be interpreted as SVs, and to let the evaluation of Eq. 73 be more permissive, we can replace the $\max$ and $\min$ operators by the median, mean, or superior and inferior quartiles.

Fig. 17 depicts four new data points (in green and numbered), two of each class. The new data points must have projected gradient out of the bounds from the respective model to be considered in the incremental procedure. For instance, the circle 1 has a projected gradient lower than the $(m'_{\min})$ of class +1 model, and the circle 2 has a projected gradient greater than $(m'_{\max})$ of class +1 model. The cross 3 has a projected gradient greater than $(m'_{\max})$ of class -1, so it is not discarded, but the cross 4 is bounded by the $(m'_{\min})$ and $(m'_{\min})$ of class -1 model, so it is discarded. The new data points that have projected gradient lower than $(m'_{\min})$ should interfere in the model shape and placement regarding only its own class, while the new data points that have projected gradient greater than $(m'_{\max})$ interfere in the hyperplane placement regarding the opposite class.

We calculate the fuzzy membership to each new data point from X_{new} that extrapolates the projected gradient bounds, where X_{over} is the data matrix that extrapolates the bounds. Then, we start a new training iteration $k \to k+1$ to update the model by

Figura 17 – Blue circles and red crosses represent the classes respectively, and the green circles and crosses represent new data points from each class.



enlarging the active set with X_{over}. The algorithm 3 presents the pseudo-code for the incremental procedure.

Algorithm 3 Incremental procedure

Input: X_{new}, $\boldsymbol{y}_{new}$, C_1,C_2,C_3,C_4, and the previous model
Output: updated model

```
 1: Let $Z_{new-} = \{1, \dots, X_-\}$
 2: Given $\epsilon$, $\boldsymbol{\alpha_{new}} = 0$ and $\boldsymbol{u_{+new}} = 0$
 3: $\nabla_i f(\boldsymbol{\alpha}_{new}) = -H_{-i}\boldsymbol{u}_+ - 1$
 4: if $\nabla_i f(\boldsymbol{\alpha_{new}}) > \max m_{\max}$ or $\nabla_i f(\boldsymbol{\alpha}_{new}) < \min m_{\min}$ then ▷ We check previous $m_{\max}$ and $m_{\min}$'s
 5:     Compute the fuzzy membership $\boldsymbol{s}$
 6:     Compute $Q = (H_+^\top H_+ + C_1 I)^{-1} H_-^\top$ and $Q'_{ii} = H_{-i} Q_i$
 7: end if
 8: while do
 9:     for $\forall i \in Z_{new}$, (a randomly and exclusively selected in the case of batch) do ▷ this thread runs in parallel
10:         Run algorithm 2 from line 8 to 31
11:     end for
12: end while
13: return updated model $\leftarrow \boldsymbol{u}_+$, $\boldsymbol{\alpha}$, $m_{\max}$, $m_{\min}$
```

The incremental procedure adds X_{over} data to the model at each iteration, remembering that we need to calculate beforehand the fuzzy membership value to each data point in X_{over}, which increases the processing time. At the worst case we have $X_{over} = X_{new}$, so the model dimension grows linearly with the number of new data points, as well as the processing time increases at each new training iteration. To avoid the continuous growth of the model dimension caused by the incremental procedure, we introduce a decremental procedure to control the model dimension by removing data that has low or no interference in the model accuracy. The proposed decremental procedure is also based on the shrinking technique, where the SVs that have both Lagrangian multipliers smaller than a threshold (ϕ) after (d) occurrences are removed. The decremental procedure is executed before each incremental training (except for the first training). We use the vector $Z_{re} = [0_1, \dots, 0_q]$ (initially all points are assigned to zero) to keep track of the number of occurrences per input data point.

Considering the current active set Z (without non-bounded Lagrangian multipliers), for each training data point there are two sets of Lagrangians $Z_\alpha = \{\alpha_1, \dots, \alpha_q\}$ and $Z_\nu = \{\nu_1, \dots, \nu_q\}$, where q is the number of Lagrangian multipliers. After each training iteration $k \to k+1$, the inputs that result in $(\alpha_m \wedge \beta_m) < \phi$ are updated (in its corresponding position) in the vector Z_{re}. When the number of occurrences reaches d, the input and all related data are removed, so it will not be used in the next incremental training. The algorithm 4 presents the pseudo-code for the decremental procedure.

Algorithm 4 Decremental procedure

```
Input: current model → $X$, $\boldsymbol{y}$, $\boldsymbol{\alpha}$
Output: updated model ← $X$, $\boldsymbol{y}$, $\boldsymbol{\alpha}$
1: Let $Z = \{1, \dots, X\}$                          ▷ We assume that there is an existing classifier
2: Given $d$, $\phi$ , $Z_{re}$, $Z_\alpha$, and $Z_\nu$                        ▷ $Z_{re}$ is initially zero
3: for do$\forall i \in Z$
4:     if $Z_{\alpha i} < \phi \wedge Z_{\nu i} < \phi$ then Increment $Z_{re\,i}$
5:     end if
6:     if $Z_{re\,i} = d$ then Remove $Z_{re\,i},\ Z_{\alpha\,i}, Z_{\nu\,i}$
7:     end if
8: end for
9: return updated model ← $X$, $\boldsymbol{y}$, $\boldsymbol{\alpha}$
```

5.6 EXPERIMENTAL RESULTS

In this section we present the experimental protocol[4] used to evaluate the FBTWSVM on benchmarking datasets. The focus of our evaluation is in incremental online learning, although we can use the FBTWSVM in offline mode. For comparison purposes, we have used an experimental protocol similar to Losing et al. (LOSING; HAMMER; WERSING, 2018), who compares a broad range of state-of-the-art on-line classification algorithms, namely: ISVM with RBF kernel, LASVM with RBF kernel, Online Random Forest (ORF) (SAFFARI *et al.*, 2009), Incremental Learning Vector Quantization (ILVQ) (SATO; YAMADA, 1996), Learn++ (POLIKAR *et al.*, 2001), Incremental Extreme Learning Machine (IELM) (LIANG *et al.*, 2006), Naive Bayes (ZHANG; LU, 2004), and Stochastic Gradient Descent (SGD). However, we have restricted to the

[4] All tests were performed in a machine running Ubuntu 16.04 LTS with an Intel Core i7-7700HQ CPU @ 2.80GHz and 16,144MB of RAM memory.

evaluation of the methods that led to the best accuracy in the on-line learning experiments for at least one of the datasets, which are the ISVM, LASVM, ORF, and ILVQ (LOSING; HAMMER; WERSING, 2018). The ORF (SAFFARI *et al.*, 2009) is an incremental Random Forest algorithm that grows continuously from a pre-defined number of trees by adding splits whenever enough data points are gathered within one leaf. It uses Extreme Random Trees (GEURTS; ERNST; WEHENKEL, 2006) to optimize the split, using a pre-defined number of random values. The ILVQ is a dynamic growth model derived from the static Generalized Learning Vector Quantization (SATO; YAMADA, 1996), where the insertion rate is guided by the number of misclassified data points. The ISVM and the LASVM were already described in Section 5.4.

The implementation used for comparison is from (LOSING; HAMMER; WERSING, 2015a), that introduces a prototype placement strategy to minimize the loss of a sliding window of recent data points. The experimental procedure of Losing et al. (LOSING; HAMMER; WERSING, 2018) for on-line methods uses a window/chunk size from 500 to 2,000, and set all hyper-parameters using the Hyperopt library (BERGSTRA, J. *et al.*, 2015) with the Tree-of-Parzen-Estimators (BERGSTRA, J. S. *et al.*, 2011) search algorithm, in which each parameter is individually adjusted within 250 iterations of a 3-fold Cross Validation (CV) using only the training data. We have carried out all our experiments with FBTWSVM using the approximated RBF kernel described in Section 5.3.3, which enables the use of our linear formulation (Eqs. 42 and 43). We optimized the model hyper-parameters using grid-search with a 3-fold cross validation on the training set, and we set the Kernel approximation size following the strategy proposed by Rahimi and Recht (RAHIMI; RECHT, 2008).

Considering that we do not need to process all data points to obtain a model, the use of batches accelerates the training phase. In this way, we use different batch sizes but with the constraint that it must encompass at least 5% of the data points of the fold and the batch must contain at least one element from each class in the first training. We evaluated the FBTWSVM with six different forgetting window sizes empirically defined as $\varphi=\{1, 2, 4, 10\}$ and without the decremental procedure. We used publicly available datasets without any preprocessing, although all attributes are numerical, either integer or real values (LICHMAN, 2013) (CHANG; LIN, 2011b). The pre-defined train-tests-splits were used when available. Otherwise, we adopted a stratified train-test-split of 70-30%.Besides that, we have also created 15 synthetic datasets (HALL *et al.*, 2009)(WITTEN *et al.*, 2016) to evaluate the scalability of the proposed method as well as a very large dataset of 23M samples (SCHMIDT *et al.*, 2018). However, for such datasets, we have compared the FBTSVM just with other SVM-based methods. We have used the following streaming generators (GOMES *et al.*, 2017) with 10% of noise added: (i) The LED generator (BREIMAN *et al.*, s.d.) yields instances with 24 Boolean features that correspond to the segments of a seven-segment LED display and another 17 irrelevant features; (ii) The SEA generator (STREET; KIM, 2001) generates streams from two relevant continuous attributes f_1, f_2 and an irrelevant f_3, with a range of values within 0 and 10; (iii) The Random Tree Generator (RTG) (DOMINGOS; HULTEN, 2000) builds a decision tree by randomly selecting attributes as split nodes and assigning random classes to each leaf.The number of values per nominal is set to 5, the max three depth is 3, the first leaf value is 3, and the leaf fraction is 0.15; (iv) The Radial Basis Function (RBF) generator creates 50 centroids at random positions and associates them with a standard deviation value, a weight, and a class label. In this way, new instances are set according to the random direction chosen to offset the centroid, which forms a Gaussian distribution according to the standard deviation associated with the given centroid; (v) The HYPER (DOMINGOS; HULTEN, 2000) generates instances that are separable by a hyperplane. We consider 10% sigma percentage, and there is no magnitude change or drift attributes. We have created three datasets from each streaming generator with 10,000, 100,000, and 1,000,000 training instances and 3,000, 30,000, and 300,000 testing instances respectively. The datasets encompass generated, artificial and real-world problems with different number of classes (from 2 to 100), data points (from 2,586 to 23M) and attributes (from 2 to 5,000), as shown in Tab. 18[5].

Tab. 19 shows the parameter setting used for each dataset which was defined in a 3-fold CV, where the $\#Points$ stands for the initial training set size. For all datasets we used a fixed fuzzy parameter $\mu = 0.1$. Using the 4-D case (C_1, C_2, C_3, C_4 are independent variables) for the hyperparameter tuning may result in a model with a better generalization performance, i.e., the loss function may achieve a lower value during the model selection compared to the 2-D case (we assume $C_1 = C_3$ and $C_2 = C_4$), however, performing the hyperparameter tuning in a 2-D space may decrease substantially the number of function evaluations needed, especially given that the grid search is essentially a brute force search strategy that takes. Other model selection strategies are able to speed-up the hyperparameter tuning, however, this is not the scope of this paper. In many cases, using the 2-D space instead of the 4-D is a valid heuristic estimation to decrease the number of function evaluations needed, and using the Overlap dataset as an example, Fig. 18 depicts that using the 2-D space it requires 34 function evaluations to achieve the accuracy loss value of 0.1732, while the 4-D space requires 5,143 function evaluations to achieve 0.1703.

Tab. 20 shows the accuracy of the incremental and decremental FBTWSVM against the best on-line algorithms reported in (LOSING; HAMMER; WERSING, 2018). The FBTWSVM achieved equal or better results in 9 out of 11 datasets (from Border to Gisette, excluding the generated datasets) relative to the best on-line algorithms. The SUSY dataset contains a significant amount of data and to train the FBTWSVM we had limited the size of the kernel approximation based on the memory available, and

[5] All datasets and algorithms are available at `https://github.com/areeberg/FBTSVM`

Tabela 18 – The datasets and their characteristics

Dataset	**#Train**	**#Test**	**#Attrib.**	**#Class**
Border	4,000	1,000	2	3
Overlap	3,960	990	2	4
Letter	16,000	4,000	16	26
SUSY	4,500,000	500,000	18	2
Outdoor	2,600	1,400	21	40
COIL	1,800	5,400	21	100
DNA	1,400	1,186	180	3
USPS	7,291	2,007	256	10
Isolet	6,238	1,559	617	26
Mnist	60,000	10,000	784	10
Gisette	6,000	1,000	5,000	2
WESAD	21,668,504	1,537,900	8	3
LED	10k|100k|1M	3k|30k|300k	24	10
SEA	10k|100k|1M	3k|30k|300k	3	2
RTG	10k|100k|1M	3k|30k|300k	10	2
RBF	10k|100k|1M	3k|30k|300k	10	5
HYPER	10k|100k|1M	3k|30k|300k	10	2

Tabela 19 – Experimental settings

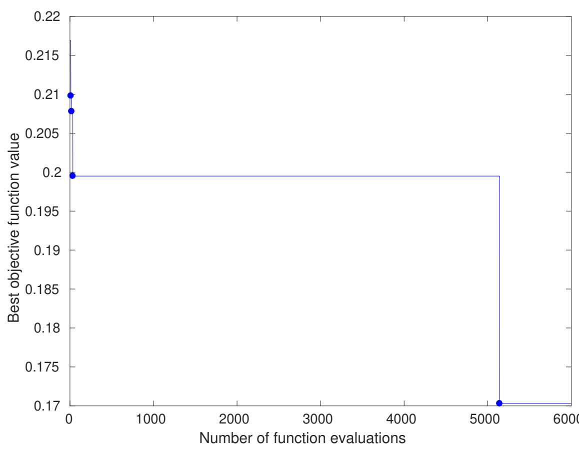
Dataset	**#Kernel**	γ	$C_1 = C_3$	$C_2 = C_4$	**#Points**
Border	150	0.4	8	2	100
Overlap	150	0.4	8	2	100
Letter	350	0.01	8	2	1000
SUSY	300	0.2	10	2	100,000
Outdoor	500	0.001	10	1	300
COIL	400	20	4	4	500
DNA	500	0.003	4	4	50
USPS	1,000	0.007	8	2	1,000
Isolet	1,000	0.002	10	10	500
Mnist	2,400	0.0002	10	10	10,000
Gisette	linear	linear	8	2	500
WESAD	linear	linear	8	2	1,537,900
LED	linear	linear	8	2	5,000
SEA	linear	linear	10	1	5,000
RTG	1,400	0.6	2.5	2	5,000
RBF	300	0.45	8	2	5,000
HYPER	linear	linear	5	4	5,000

this also reduces the accuracy. For instance, both the ISVM and the LASVM with RBF kernel could not be trained with this dataset due to the uncontrolled growth of the kernel matrix. We run an experiment using the SUSY full training set considering a kernel approximation size of 600, resulting in 77.67%. The Outdoor is a visual dataset that consists of objects recorded outdoors under lighting conditions (LOSING; HAMMER; WERSING, 2015b). The dataset creation method has caused a difference between training and test data (LOSING; HAMMER; WERSING, 2018), which reflects on the performance of the learning algorithms. On-line algorithms with an adaptive learning mechanism presented the accuracy of about 20% better than off-line methods (the best result found was the off-line ISVM with 71.9% (LOSING; HAMMER; WERSING, 2018)).

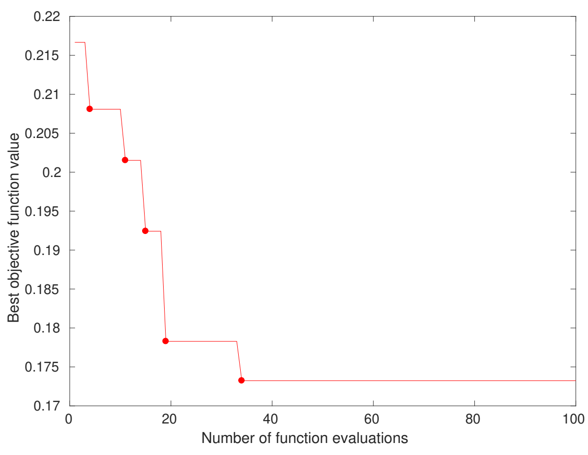
Tabela 20 – On-line accuracy of the incremental and decremental FBTWSVM compared to other incremental algorithms on several benchmark datasets. Statistically significant differences are marked with $\star$.

	Accuracy (%)			
Dataset	**FBTWSVM**		**Best Method**	
	Best	**Mean/StdDev**		**Best**
Border	**98.70**	97.60/1.10	ISVM	98.50
Overlap	**84.14***	82.58/1.49	ISVM	81.70
Letter	**96.75***	96.68/0.07	LASVM	92.70
SUSY	77.67	76.00/1.20	ORF	**79.30**
Outdoor	74.44	73.72/0.42	ISVM	**86.40**
COIL	**95.11***	94.99/0.14	ILVQ	79.10
DNA	**93.59***	92.90/0.30	ISVM/LASVM	89.50
USPS	95.47	94.91/0.30	ISVM	**96.70**
Isolet	**96.28***	95.88/0.37	ISVM	93.60
Mnist	**97.80**	97.00/0.12	LASVM	97.50
Gisette	**96.50**	96.40/0.01	LASVM	96.40

Figura 18 – Overlap dataset convergence plot considering the 4-D and 2-D cases.



a Convergence plot comparison.



b Convergence plot for the 2-D case.

Tab. 21 shows the relation between accuracy and the number of SVs resulting from different forgetting scores (d). The decremental procedure discards points that are less likely to be SVs. Smaller d leads to classifiers with lower generalization performance, and for most of the datasets, the best performance was achieved without forgetting or with large forgetting scores. On the other hand, the number of SVs using the decremental procedure is considerably smaller, so the forgetting score must be chosen according to the application. Tab. 21 also presents the comparison between the online and offline approaches, in which the online had better accuracy to all datasets with a smaller number of SVs compared to offline. A smaller d also implies in a faster training and classification time, and Tab. 22 shows that the difference in training time can be substantial (check the SUSY dataset values for example).

Tabela 21 – On-line accuracy with different forgetting scores (d) and the corresponding number of support vectors (nSVs).

Dataset	d**=1 / nSVs**	d**=2 / nSVs**	d**=4 / nSVs**	d**=10 / nSVs**	d**=∞ / nSVs**	**Off-line / nSVs**
Border	91.90 / 538	93.30 / 784	98.20 / 1.3k	97.20 / 2.8k	**98.70 / 7k**	98.50 / 8k
Overlap	76.97 / 1.9k	79.70 / 2.3k	82.22 / 3.5k	82.83 / 7k	**84.14 / 11.8k**	83.30 / 11k
Letter	93.38 / 83k	94.83 / 122k	96.10 / 204k	96.10 / 338k	**96.63 / 361k**	96.90 / 384k
SUSY	45.84 / 754k	45.85 / 1.5M	**77.67 / 2.5M**	73.78 / 3.4M	76.45 / 3.8M	-
Outdoor	72.00 / 24k	72.25 / 39k	73.69 / 68k	**74.44 / 83k**	73.88 / 92k	74.00 / 93k
COIL	93.21 / 98k	94.01 / 100k	94.57 / 153k	**95.11 / 156k**	94.93 / 166k	95.00 / 178k
DNA	91.82 / 770	92.16 / 848	92.50 / 1.2k	**93.78 / 1.9k**	93.59 / 2.6k	93.50 / 2.7k
USPS	93.92 / 9k	94.32 / 21k	94.87 / 42k	95.36 / 60k	**95.47 / 60k**	95.30 / 65.5k
Isolet	95.19 / 61k	95.51 / 85k	95.32 / 128k	95.89 / 143k	**96.28 / 143k**	95.60 / 155k
Mnist	97.17 / 59k	97.48 / 169k	97.66 / 342k	**97.91 / 455k**	97.80 / 455k	97.80 / 540k
Gisette	96.50 / 1.3k	**97.00 / 1.9k**	96.90 / 2.9k	96.20 / 5.3k	96.50 / 6k	96.50 / 6k

Tabela 22 – Training and testing processing time with different forgetting scores (d) in seconds.

	Training | Testing Time (sec)		
Dataset	d**=1**	d**=10**	d**=∞**
Border	12.39 | 0.01	34.12 | 0.02	43.22 | 0.02
Overlap	22.96 | 0.02	82.58 | 0.02	94.37 | 0.02
Letter	88.14 | 0.68	179.18 | 0.72	192.36 | 0.70
SUSY	940.60 | 1.88	5264.41 | 2.62	-
Outdoor	111.00 | 0.73	153.09 | 0.80	153.55 | 0.81
COIL	154.63 | 5.00	169.10 | 5.29	179.12 | 4.51
DNA	6.89 | 0.03	8.63 | 0.03	8.53 | 0.03
USPS	21.59 | 0.25	41.59 | 0.24	41.67 | 0.23
Isolet	117.35 | 0.52	181.14 | 0.49	160.10 | 0.51
Mnist	349.47 | 1.21	419.49 | 1.19	435.48 | 1.26
Gisette	25.26 | 0.01	50.52 | 0.01	49.41 | 0.01

The accuracy of the COIL dataset with a forgetting score d=10 have similar accuracy (95.11%) when compared to the offline implementation of the IFBTSVM (95%), the ISVM (96.5%), the LASVM (93.2%) (LOSING; HAMMER; WERSING, 2018), and the multiclass SVM implemented with the Error-Correcting Output Codes (ECOC) from MATLAB (96.52%). In this manner, the forgetting strategy does not discard crucial support vectors, keeping the accuracy score near the offline approach. Tab. 22 presents the accuracy performance evolution when increasing the forgetting score, which corroborates with the forgetting strategy, i.e., lower forgetting scores tend to have smaller accuracy, however, by keeping the important SV the accuracy does not fall substantially (the accuracy difference between d=1 and d=10 is 1.9%).

Tab. 23 compares the training time (in seconds), the real RAM consumed of the current process and its children (in Gigabytes), and the accuracy of the FBTSVM with the other SVM based methods (ISVM and LASVM). For these experiments we split the dataset into the largest batches that we can (that fits on the available memory, initially 15.4Gb), to reduce the reloading procedure of the dataset during the execution (more loading implies in a larger training time). Both the IFBTSVM and the ISVM (the ISVM multiclass adopt one-versus-one strategy) are implemented in MATLAB, thus it requires more real RAM than the LASVM, that is for binary cases only and it is a C++ implementation. We do not consider the LASVM in the multiclass cases (LED and RBF), and we discard the situations that the training time took over 12 hours. All methods present competitive accuracy, however, the FBTSVM is the only method (compared to ISVM and LASVM) able to train all dataset sizes in an acceptable time, having the smallest training time for almost all situations (the only exception is the LASVM for the RTG10K). The FBTSVM forgetting strategy is one of the factors (the kernel approximation also plays an important role) that makes the training into large datasets possible, as Tab. 23 shows that the real RAM consumed difference between the 100K and 1M datasets is not very expressive. The LED dataset has a bigger memory difference between the datasets for the IFBTSVM, and this is caused by the use of the multi-thread instead of the single processor version. In this way, the scalability of the IFBTSVM is superior to other online SVM-based methods, as it requires a smaller training time to process large datasets and can handle the memory consumption in an efficient manner. To further explore the FBTWSVM potential for large datasets, we have also evaluated the accuracy, training time, and memory consumption on the WESAD dataset (SCHMIDT *et al.*, 2018) considering three classes (baseline, stress, and amusement), eight attributes acquired from a sensor attached to the chest, and using the leave-one-subject-out cross-validation (in total we have 17 subjects). The best result reported by (SCHMIDT *et al.*, 2018) is 76.5% using a Linear Discriminant Analysis, however, this is an offline approach and the authors do not present the training time or memory consumption. Our method achieved the accuracy of 75.5% (23, with training time of 6,789 seconds and peak memory consumption of 9.8GB.

Tabela 23 – Comparison of the training time (TT) in seconds (s), real memory usage (RMU) in Gigabytes (GB) and percentual accuracy (Acc) (%) for five synthetic datasets and three different amount of data.

Dataset	IFBTSVM 10k|100k|1M			ISVM 10k|100k|1M			LASVM 10k|100k|1M		
	TT (s)	RMU (GB)	Acc (%)	TT (s)	RMU (GB)	Acc (%)	TT (s)	RMU (GB)	Acc (%)
LED	6|19|380	3.2|3.8|7.3	74.1|74.1|74.2	41|-|-	0.9|-|-	74.2|-|-	-	-	-
SEA	0.7|2|63	0.8|0.9|1.3	89.0|87.1|89.1	8.8|4.5k|-	0.9|1|-	89.9|89.3|-	2.4|328|-	0.06|0.4|-	84.0|87.0|-
RTG	9.6|119|3.4k	1.8|10|10.6	95.8|95.5|96.0	63.9|-|-	0.9|-|-	90.0|-|-	2.4|546|-	0.034|0.4|-	88.0|95.7|-
RBF	11.4|164|5.3k	1.1|2.1|7.1	88.3|89.4|89.0	31.8|-|-	0.9|-|-	87.6|-|-	-	-	-
HYPER	0.7|2.6|28	0.8|0.9|1.4	89.1|94.7|94.0	14.4|4.9k|-	0.8|0.9|-	94.1|93.0|-	2.2|519|-	0.05|0.374|-	91.7|93.8|-
WESAD	6.8k	9.8	75.5	-	-	-	-	-	-

- "denotes the non-available results due to training times greater than 12 hours.

5.7 SUMMARY AND SYNTHESIS OF CONTRIBUTION

In this chapter we introduced a novel SVM approach suitable for incremental and decremental on-line learning. The incremental and decremental FBTWSVM integrates ideas coming from different SVM approaches such as the Twin SVM (JAYADEVA; KHEMCHANDANI, R.; CHANDRA, 2007), the Fuzzy SVM (CHUN-FU LIN; SHENG-DE WANG, 2002), the Bounded TWSVM (SHAO; ZHANG *et al.*, 2011), the Fast and Robust TWSVM (GAO; WANG *et al.*, 2015; GAO; WANG, 2017), the Optimal DAG TWSVM (CHEN; JI, 2010), and the dual coordinate descent method (CHANG; HSIEH; LIN, 2008). The FBTWSVM is flexible and both incremental and decremental procedures can be configured according to the application, changing the threshold of adding new SVs in the incremental step and the number of occurrences in the decremental step. The FBTWSVM calculates a pair of nonparallel hyperplanes using two smaller QPPs, rather than one large QPP as in the original SVM, but with adherence to structural risk minimization principle. The dual form of the FBTWSVM leads to a pair of convex quadratic programming problems with a unique solution and singularity avoidance. The dual coordinate descent method with shrinking requires less memory storage than the TWSVM, as it discards points that are less likely to be SVs. The fuzzy concept enhances noise-resistance and generalization capability, while the use of a kernel approximation shows a good generalization performance with our linear model. The incremental solution follows the shrinking strategy and can run with different batch sizes, from a single individual to the number of data points that fits the available memory. The decremental procedure is fundamental to control the model complexity, keeping only the most critical SVs in the model. According to the experimental results, the DAG strategy showed a good generalization capability and a fast training speed, but for further studies the use of training data structural and statistical information in the training process may increase the generalization performance. A practical difficulty in the FBTWSVM is the optimization of the six hyperparameters $C_1, C_2, C_3, C_4, \mu, \gamma$ and the kernel approximation size, however, this problem will be addressed in the future. Therefore, as a future work, we will evaluate the FBTWSVM use in the context of concept drift, novelty detection, and big data.

The contributions of this chapter are:

- It have a good generalization capability and a fast training speed compared to traditional incremental SVM.
- The incremental solution follows the shrinking strategy and can run with different batch sizes, from a single individual to the number of data points that fits the available memory.
- It can adapt current models using the window strategy, or even add new models (e.g. in case of new classes) without retraining.
- The fuzzy concept enhances noise-resistance and generalization capability, while the use of a kernel approximation shows a good generalization performance with our linear model.
- The dual coordinate descent method with shrinking requires less memory storage than the TWSVM on the training procedure, as it discards points that are less likely to be SVs.

6 CONCLUDING REMARKS

The objective of machine learning is to extract information from data based on past experience and build systems that can make a decision based on that information. This capacity of turning raw data into information is a complex task that has attracted interest from various domains of application, and many fields of science, business, and engineering have used machine learning to improve its performance. It is well known the advantages of using machine learning techniques to improve performance, however, there are some data characteristics that must be considered before using it. Nowadays, we are producing data faster than ever, so we need efficient algorithms that can process a large amount of data. These algorithms must reach out the most informative instances in the training data in a cost-efficient way, be able to train models in a reasonable time, and build models that use less memory in both training and classification phases. Frequently, the data may contain noise that degrades the data quality, thus, it may decrease the computational efficiency of learning algorithms. The data distribution is also an aspect that must be considered, as a significantly uneven number of instance per class may lead to difficulties during the recognition phase.

This thesis presents methodologies that address the aforementioned issues with the goal of improving scalability, computational and data efficiency, and generalization performance of machine learning algorithms with a particular focus on Support Vector Machines. One way to deal with large datasets in learning problems is to use the passive sampling, that presents the learning algorithm a smaller view of the entire dataset that can be handled within time an computational resource constraints, maintaining its generalization capability. Online learning algorithms are usually associated with problems where the complete training set is not available beforehand, but large scale problems can also benefit from the computational properties of online learning, thus, online learning can be a solution to both problems. Despite the SVM formulation, hyperparameters tuning is necessary and may be time-consuming depending on the situation. The most common methods for hyperparameters tuning does not provide mathematical convergence properties and dynamic stopping criteria, which may lead to sub-optimal outcomes. In this way, an efficient model selection technique leads to an outcome less susceptible to sub-optimal results and requires inferior processing time.

6.1 PUBLICATIONS

The following paper were published based on the research work whose outcomes are presented in this thesis:

Reeberg de Mello, A.; STEMMER, M. R.; Oliveira Barbosa, F. G. Support vector candidates selection via Delaunay graph and convex-hull for large and high-dimensional *datasets. Pattern Recognition Letters, North-Holland, v. 116, p. 43–49, dec 2018. ISSN 0167-8655.*

6.2 FUTURE WORK

A list of suggestions for future work follows:

- Further explore the SV candidates selection for Delaunay sub-graphs or other types of graphs, as the Relative Neighborhood graph, Urquhart graph, and Gabriel graph. Using Delaunay sub-graphs may result in a smaller set of SV which reduces the training time, whilst other types of graphs may result in different set sizes of SV candidates that may be a better representative of the original set.
- Analyze the feasibility of Ortho-MADS with NM and VNS for SVM variations (that may contain more than one hyperparameter) with different kernels. We expect to improve the results even more when using the Ortho-MADS with NM and VNS for hyperparameters tuning with more than three hyperparameters compared to traditional methods.
- Expand the use of Ortho-MADS with NM and VNS to different machine learning algorithms. The SVM is a sensitive algorithm to hyperparameters, however, other methods may also benefit from the proposed model selection approach, as the ensemble learning methods.
- Develop a new FBTWSVM formulation to make viable the inverse-free update for online cases (inspired by (TIAN; JU *et al.*, 2014)). This upgrade may result in a less computational expensive model update.
- Extend the FBTWSVM to unsupervised and semi-supervised cases. Labeling data is an expensive task, so using a smoothness, cluster, or manifold algorithm to viable the unsupervised and semi-supervised increases the applicability of the FBTWSVM to many other cases.
- Explore new approximate kernel types. Different types of approximation kernel have distinct characteristics (as each approximation is related to a specific function), in this manner, to each application some kernel approximations are more suitable than others.
- Propose a new multiclass architecture for the FBTWSVM to improve the performance on imbalanced datasets.
- Expand the FBTWSVM to endless learning. Endless learning is tightly related to memory efficiency, and to viable the use of FBTWSVM in real situations the

endless learning is a must have. One possible direction is to implement the code of the FBTWSVM to split the data when the available batch is larger than the available memory, and to prune old SVs when the models get near available memory size.

- Analyze the viability of using the FBTWSVM for cases with concept drift. A initial direction is to use a drift-detector algorithm (as the Drift Detection Method by (GAMA; SEBASTIÃO; RODRIGUES, 2013)) using the FBTWSVM as classifier.
- Propose a dynamic windowing method for the FBTWSVM. The windowing method is closely related to training time, and have a high impact on the concept-drift detection method. A possible solution is to explore the fuzzy values of FBTWSVM to determine the window size.

REFERÊNCIAS

ABU-MOSTAFA, Yaser S.; MAGDON-ISMAIL, Malik.; LIN, Hsuan-Tien. **Learning from data : a short course**. [*S.l.*]: AMLBook.com, 2012. p. 201. ISBN 1600490069.

ACERBI, Luigi; JI, Wei. Practical Bayesian Optimization for Model Fitting with Bayesian Adaptive Direct Search. *In:* ADVANCES in Neural Information Processing Systems. [*S.l.*]: Curran Associates, Inc., 2017. p. 1836–1846.

ALAMDAR, Fatemeh; GHANE, Sara; AMIRI, Ali. On-line twin independent support vector machines. **Neurocomputing**, Elsevier, v. 186, p. 8–21, abr. 2016. ISSN 0925-2312. DOI: 10.1016/J.NEUCOM.2015.12.062.

ARUN KUMAR, M.; GOPAL, M. Least squares twin support vector machines for pattern classification. **Expert Systems with Applications**, Pergamon, v. 36, n. 4, p. 7535–7543, mai. 2009. ISSN 0957-4174. DOI: 10.1016/J.ESWA.2008.09.066.

AUDET, Charles. A Survey on Direct Search Methods for Blackbox Optimization and Their Applications. *In:* MATHEMATICS Without Boundaries. New York, NY: Springer New York, 2014. p. 31–56. DOI: 10.1007/978-1-4939-1124-0_2.

AUDET, Charles. Tuning Runge-Kutta parameters on a family of ordinary differential equations. **International Journal of Mathematical Modelling and Numerical Optimisation**, v. 8, n. 3, p. 277, 2018. ISSN 2040-3607. DOI: 10.1504/IJMMNO.2018.088992.

AUDET, Charles; BÉCHARD, Vincent; DIGABEL, Sébastien Le. Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search. **Journal of Global Optimization**, Springer US, v. 41, n. 2, p. 299–318, jun. 2008. ISSN 0925-5001. DOI: 10.1007/s10898-007-9234-1.

AUDET, Charles; LE DIGABEL, Sébastien; TRIBES, Christophe. Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization. **Optimization and Engineering**, Springer US, v. 17, n. 2, p. 333–358, jun. 2016. ISSN 1389-4420. DOI: 10.1007/s11081-015-9283-0.

AUDET, Charles; TRIBES, Christophe. Mesh-based Nelder–Mead algorithm for inequality constrained optimization. **Computational Optimization and Applications**, Springer US, v. 71, n. 2, p. 331–352, nov. 2018. ISSN 0926-6003. DOI: 10.1007/s10589-018-0016-0.

BARBER, C. Bradford; DOBKIN, David P.; HUHDANPAA, Hannu. The quickhull algorithm for convex hulls. **ACM Transactions on Mathematical Software**, ACM, v. 22, n. 4, p. 469–483, dez. 1996. ISSN 00983500. DOI: 10.1145/235815.235821.

BERGSTRA, James S. *et al.* Algorithms for Hyper-Parameter Optimization. *In:* NEURAL Information Processing Systems (NIPS). [*S.l.*: *s.n.*], 2011. p. 2546–2554.

BERGSTRA, James *et al.* Hyperopt: a Python library for model selection and hyperparameter optimization. **Computational Science & Discovery**, IOP Publishing, v. 8, n. 1, p. 014008, jul. 2015. ISSN 1749-4699. DOI: 10.1088/1749-4699/8/1/014008.

BHATT, Rajen B. *et al.* Efficient Skin Region Segmentation Using Low Complexity Fuzzy Decision Tree Model. *In:* 2009 Annual IEEE India Conference. [*S.l.*]: IEEE, 2009. p. 1–4. DOI: 10.1109/INDCON.2009.5409447.

BISHOP, Christopher M. **Pattern recognition and machine learning**. [*S.l.*]: Springer, 2006. p. 738. ISBN 9780387310732.

BLACKARD, Jock A.; DEAN, Denis J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. **Computers and Electronics in Agriculture**, Elsevier, v. 24, n. 3, p. 131–151, dez. 1999. ISSN 0168-1699. DOI: 10.1016/S0168-1699(99)00046-0.

BOISSONNAT, Jean-Daniel; DEVILLERS, Olivier; HORNUS, Samuel. Incremental construction of the Delaunay graph in medium dimension. **Proceedings of the twenty-fifth annual symposium on Computational geometry**, p. 208–216, jun. 2009.

BONICHON, Nicolas *et al.* Connections between Theta-Graphs, Delaunay Triangulations, and Orthogonal Surfaces. *In:* INTERNATIONAL Workshop on Graph-Theoretic Concepts in Computer Science. [*S.l.*]: Springer, Berlin, Heidelberg, 2010. p. 266–278. DOI: 10.1007/978-3-642-16926-7-25.

BORDES, Antoine *et al.* Fast Kernel Classifiers with Online and Active Learning. **Journal of Machine Learning Research**, v. 6, Sep, p. 1579–1619, 2005. ISSN ISSN 1533-7928.

BOSER, Bernhard E.; GUYON, Isabelle M.; VAPNIK, Vladimir N. A training algorithm for optimal margin classifiers. *In:* PROCEEDINGS of the fifth annual workshop on Computational learning theory - COLT ’92. New York, New York, USA: ACM Press, 1992. p. 144–152. DOI: 10.1145/130385.130401.

BRADLEY, Paul S.; MANGASARIAN, O. L. Feature Selection via Concave Minimization and Support Vector Machines. *In:* PROCEEDINGS of the Fifteenth International Conference on Machine Learning. [*S.l.*]: Morgan Kaufmann Publishers, 1998. p. 580.

BRAZDIL, Pavel B.; SOARES, Carlos. A Comparison of Ranking Methods for Classification Algorithm Selection. *In:* MACHINE Learning: ECML 2000. [*S.l.*]: Springer, Berlin, Heidelberg, 2000. p. 63–75. DOI: 10.1007/3-540-45164-1_8.

BREIMAN, Leo *et al.* **Classification and regression trees**. [*S.l.*: *s.n.*]. ISBN 9781138469525.

CAUWENBERGHS, Gert; POGGIO, Tomaso. **Incremental and decremental support vector machine learning**. [*S.l.*]: MIT Press, 2000. p. 388–394.

CERVANTES, Jair *et al.* Support vector machine classification for large data sets via minimum enclosing ball clustering. **Neurocomputing**, Elsevier, v. 71, n. 4-6, p. 611–619, jan. 2008. ISSN 0925-2312. DOI: 10.1016/J.NEUCOM.2007.07.028.

CHANG, Chih-Chung; LIN, Chih-Jen. LIBSVM: A library for support vector machines. **ACM Transactions on Intelligent Systems and Technology**, ACM, v. 2, n. 3, p. 1–27, abr. 2011. DOI: 10.1145/1961189.1961199.

CHANG, Chih-Chung; LIN, Chih-Jen. LIBSVM: A library for support vector machines. **ACM Transactions on Intelligent Systems and Technology**, ACM, v. 2, n. 3, p. 1–27, abr. 2011. DOI: 10.1145/1961189.1961199.

CHANG, Chin-Chun; CHOU, Shen-Huan. Tuning of the hyperparameters for L2-loss SVMs with the RBF kernel by the maximum-margin principle and the jackknife technique. **Pattern Recognition**, Pergamon, v. 48, n. 12, p. 3983–3992, dez. 2015. ISSN 0031-3203. DOI: 10.1016/J.PATCOG.2015.06.017.

CHANG, Kai-Wei; HSIEH, Cho-Jui; LIN, Chih-Jen. Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines. **Journal of Machine Learning Research**, v. 9, Jul, p. 1369–1398, 2008. ISSN ISSN 1533-7928.

CHEN, Guangliang; FLORERO-SALINAS, Wilson; LI, Dan. Simple, fast and accurate hyper-parameter tuning in Gaussian-kernel SVM. *In:* 2017 International Joint Conference on Neural Networks (IJCNN). [*S.l.*]: IEEE, mai. 2017. p. 348–355. DOI: 10.1109/IJCNN.2017.7965875.

CHEN, Jing; JI, Guangrong. Multi-class LSTSVM classifier based on optimal directed acyclic graph. *In:* 2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE). [*S.l.*]: IEEE, fev. 2010. p. 100–104. DOI: 10.1109/ICCAE.2010.5452037.

CHEN, Lisha; BUJA, Andreas. Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Drawing, and Proximity Analysis. **Journal of the American Statistical Association**, v. 104, n. 485, p. 209–219, mar. 2006. ISSN 0162-1459. DOI: 10.1198/jasa.2009.0111.

CHUN-FU LIN; SHENG-DE WANG. Fuzzy support vector machines. **IEEE Transactions on Neural Networks**, v. 13, n. 2, p. 464–471, mar. 2002. ISSN 10459227. DOI: `10.1109/72.991432`.

CHUNG, Kai-Min *et al.* Radius Margin Bounds for Support Vector Machines with the RBF Kernel. **Neural Computation**, v. 15, n. 11, p. 2643–2681, nov. 2003. ISSN 0899-7667. DOI: `10.1162/089976603322385108`.

CORTES, Corinna; CORTES, Corinna; VAPNIK, Vladimir. Support-Vector Networks. **MACHINE LEARNING**, v. 20, p. 273–297, 1995.

COUTURE, M.A. Abramson *et al.* **The NOMAD project**. Montreal: Software available at \url{https://www.gerad.ca/nomad/}, 2018.

CRISTIANINI, Nello.; SHAWE-TAYLOR, John. **An introduction to support vector machines : and other kernel-based learning methods**. Cambridge: Cambridge University Press, 2000. p. 189. ISBN 0521780195.

DELAUNAY, Boris. Sur la sphère vide. A la mémoire de Georges Voronoï. **Bulletin de l'Académie des Sciences de l'URSS**, v. 1, n. 6, 1934.

DEVILLERS, Olivier; HORNUS, Samuel; JAMIN, Clément. dD Triangulations. *In:* CGAL User and Reference Manual. 4.11. ed. [*S.l.*]: CGAL Editorial Board, 2017. p. 1.

DHEERU, Dua; KARRA TANISKIDOU, Efi. **UCI Machine Learning Repository**. [*S.l.*]: University of California, Irvine, School of Information e Computer Sciences, 2017.

DI WANG *et al.* Online Support Vector Machine Based on Convex Hull Vertices Selection. **IEEE Transactions on Neural Networks and Learning Systems**, v. 24, n. 4, p. 593–609, abr. 2013. ISSN 2162-237X. DOI: `10.1109/TNNLS.2013.2238556`.

DIETTERICH, T. G.; BAKIRI, G. Solving Multiclass Learning Problems via Error-Correcting Output Codes. **Journal of Artificial Intelligence Research**, dez. 1994. arXiv: `9501101 [cs]`.

DING, Shifei; YU, Junzhao *et al.* An overview on twin support vector machines. **Artificial Intelligence Review**, Springer Netherlands, v. 42, n. 2, p. 245–252, ago. 2014. ISSN 0269-2821. DOI: `10.1007/s10462-012-9336-0`.

DING, Shifei; ZHANG, Nan *et al.* Twin support vector machine: theory, algorithm and applications. **Neural Computing and Applications**, Springer London, v. 28, n. 11, p. 3119–3130, nov. 2017. ISSN 0941-0643. DOI: `10.1007/s00521-016-2245-4`.

DODANGEH, M.; VICENTE, L. N. Worst case complexity of direct search under convexity. **Mathematical Programming**, v. 155, n. 1, p. 307–332, jan. 2016. ISSN

1436-4646. DOI: 10.1007/s10107-014-0847-0. Disponível em: https://doi.org/10.1007/s10107-014-0847-0.

DOMENICONI, C.; GUNOPULOS, D. Incremental support vector machine construction. *In:* PROCEEDINGS 2001 IEEE International Conference on Data Mining. [*S.l.*]: IEEE Comput. Soc, 2001. p. 589–592. DOI: 10.1109/ICDM.2001.989572.

DOMINGOS, Pedro; HULTEN, Geoff. Mining high-speed data streams. *In:* PROCEEDINGS of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00. New York, New York, USA: ACM Press, 2000. p. 71–80. DOI: 10.1145/347090.347107.

DUDA, Richard O.; HART, Peter E. (Peter Elliot); STORK, David G. **Pattern classification**. [*S.l.*]: Wiley, 2001. p. 654. ISBN 9780471056690.

EDELSBRUNNER, Herbert; SEIDEL, Raimund. Voronoi diagrams and arrangements. **Discrete & Computational Geometry**, Springer New York, v. 1, n. 1, p. 25–44, mar. 1986. ISSN 0179-5376. DOI: 10.1007/BF02187681.

ESCALERA, Sergio; PUJOL, Oriol; RADEVA, Petia. Separability of ternary codes for sparse designs of error-correcting output codes. **Pattern Recognition Letters**, North-Holland, v. 30, n. 3, p. 285–297, fev. 2009. ISSN 0167-8655. DOI: 10.1016/J.PATREC.2008.10.002.

FAN, Rong-En; CHEN, Pai-Hsuen; LIN, Chih-Jen. Working Set Selection Using Second Order Information for Training Support Vector Machines. **Journal of Machine Learning Research**, v. 6, p. 1889–1918, 2005.

FISHER, John *et al.* Curve and surface interpolation and approximation. *In:* PROCEEDINGS of the 9th annual SIGCSE conference on Innovation and technology in computer science education. New York, New York, USA: ACM Press, 2004. p. 146–150. DOI: 10.1145/1007996.1008036.

FISHER, R. A. THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. **Annals of Eugenics**, Blackwell Publishing Ltd, v. 7, n. 2, p. 179–188, set. 1936. ISSN 20501420. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

FREY, Peter W.; SLATE, David J. Letter recognition using Holland-style adaptive classifiers. **Machine Learning**, Kluwer Academic Publishers, v. 6, n. 2, p. 161–182, mar. 1991. ISSN 0885-6125. DOI: 10.1007/BF00114162.

FUNG, Glenn M.; MANGASARIAN, O.L. A Feature Selection Newton Method for Support Vector Machine Classification. **Computational Optimization and Applications**, Kluwer Academic Publishers, v. 28, n. 2, p. 185–202, jul. 2004. ISSN 0926-6003. DOI: 10.1023/B:COAP.0000026884.66338.df.

FUNG, Glenn; MANGASARIAN, Olvi L. Incremental Support Vector Machine Classification. *In:* PROCEEDINGS of the 2002 SIAM International Conference on Data Mining. Philadelphia, PA: Society for Industrial e Applied Mathematics, abr. 2002. p. 247–260. DOI: `10.1137/1.9781611972726.15`.

FUNG, Glenn; MANGASARIAN, Olvi L. Proximal support vector machine classifiers. *In:* PROCEEDINGS of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01. New York, New York, USA: ACM Press, 2001. p. 77–86. DOI: `10.1145/502512.502527`.

FÜRNKRANZ, Johannes. Round Robin Classification. **Journal of Machine Learning Research**, v. 2, Mar, p. 721–747, 2002. ISSN ISSN 1533-7928.

GALLIER, Jean. **Notes on Convex Sets, Polytopes, Polyhedra, Combinatorial Topology, Voronoi Diagrams and Delaunay Triangulations**. [*S.l.*], mai. 2008. p. 183. arXiv: `0805.0292`.

GAMA, João; SEBASTIÃO, Raquel; RODRIGUES, Pedro Pereira. On evaluating stream learning algorithms. **Machine Learning**, Springer US, v. 90, n. 3, p. 317–346, mar. 2013. ISSN 0885-6125. DOI: `10.1007/s10994-012-5320-9`.

GAO, Bin-Bin; WANG, Jian-Jun. **A Fast and Robust TSVM for Pattern Classification**. [*S.l.*: *s.n.*], nov. 2017. p. 14. arXiv: `1711.05406`.

GAO, Bin-Bin; WANG, Jian-Jun *et al.* Coordinate Descent Fuzzy Twin Support Vector Machine for Classification. *In:* 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA). [*S.l.*]: IEEE, dez. 2015. p. 7–12. DOI: `10.1109/ICMLA.2015.35`.

GEPPERTH, Alexander; HAMMER, Barbara. Incremental learning algorithms and applications. *In:* EUROPEAN Symposium on Artificial Neural Networks (ESANN). [*S.l.*: *s.n.*], 2016. p. 357–368.

GEURTS, Pierre; ERNST, Damien; WEHENKEL, Louis. Extremely randomized trees. **Machine Learning**, Kluwer Academic Publishers, v. 63, n. 1, p. 3–42, abr. 2006. ISSN 0885-6125. DOI: `10.1007/s10994-006-6226-1`.

GOLD, Carl; HOLUB, Alex; SOLLICH, Peter. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. **Neural Networks**, Pergamon, v. 18, n. 5-6, p. 693–701, jul. 2005. ISSN 0893-6080. DOI: `10.1016/J.NEUNET.2005.06.044`.

GOLD, Carl; SOLLICH, Peter. Model selection for support vector machine classification. **Neurocomputing**, Elsevier, v. 55, n. 1-2, p. 221–249, set. 2003. ISSN 0925-2312. DOI: `10.1016/S0925-2312(03)00375-8`.

GOLUB, Gene H.; LOAN, Charles F. Van. **Matrix computations**. [*S.l.*]: Johns Hopkins University Press, 1996. p. 694. ISBN 0801854148.

GOMES, Heitor M. *et al.* Adaptive random forests for evolving data stream classification. **Machine Learning**, Springer US, v. 106, n. 9-10, p. 1469–1495, out. 2017. ISSN 0885-6125. DOI: `10.1007/s10994-017-5642-8`.

GÓMEZ, David; ROJAS, Alfonso. An Empirical Overview of the No Free Lunch Theorem and Its Effect on Real-World Machine Learning Classification. **Neural Computation**, MIT Press One Rogers St., Cambridge, MA 02142-1209 USA journals-info@mit.edu, v. 28, n. 1, p. 216–228, jan. 2016. ISSN 0899-7667. DOI: `10.1162/NECO_a_00793`.

GOTO, Masanori; ISHIDA, Ryosuke; UCHIDA, Seiichi. Preselection of support vector candidates by relative neighborhood graph for large-scale character recognition. *In:* 2015 13th International Conference on Document Analysis and Recognition (ICDAR). [*S.l.*]: IEEE, ago. 2015. p. 306–310. DOI: `10.1109/ICDAR.2015.7333773`.

GUO, Li; BOUKIRA, Samia. Fast data selection for SVM training using ensemble margin. **Pattern Recognition Letters**, North-Holland, v. 51, p. 112–119, jan. 2015. ISSN 0167-8655. DOI: `10.1016/J.PATREC.2014.08.003`.

HALL, Mark *et al.* The WEKA data mining software. **ACM SIGKDD Explorations Newsletter**, ACM, v. 11, n. 1, p. 10, nov. 2009. ISSN 19310145. DOI: `10.1145/1656274.1656278`.

HANSEN, Pierre; MLADENOVIĆ, Nenad. Variable neighborhood search: Principles and applications. **European Journal of Operational Research**, North-Holland, v. 130, n. 3, p. 449–467, mai. 2001. ISSN 0377-2217. DOI: `10.1016/S0377-2217(00)00100-4`.

HAO, Yunhe; ZHANG, Haofeng. A Fast Incremental Learning Algorithm Based on Twin Support Vector Machine. *In:* 2014 Seventh International Symposium on Computational Intelligence and Design. [*S.l.*]: IEEE, dez. 2014. p. 92–95. DOI: `10.1109/ISCID.2014.38`.

HE, Haibo. Incremental Learning. *In:* SELF-ADAPTIVE Systems for Machine Intelligence. Hoboken, NJ, USA: John Wiley & Sons, Inc., set. 2011. p. 13–43. DOI: `10.1002/9781118025604.ch2`.

HSIEH, Cho-Jui *et al.* A dual coordinate descent method for large-scale linear SVM. *In:* PROCEEDINGS of the 25th international conference on Machine learning - ICML '08. New York, New York, USA: ACM Press, 2008. p. 408–415. DOI: `10.1145/1390156.1390208`.

HSU, Chih-Wei; CHANG, Chih-Chung; LIN, Chih-Jen. **A Practical Guide to Support Vector Classification**. [*S.l.*: *s.n.*], 2003.

HSU, Chih-Wei; LIN, Chih-Jen. A comparison of methods for multiclass support vector machines. **IEEE Transactions on Neural Networks**, v. 13, n. 2, p. 415–425, mar. 2002. ISSN 10459227. DOI: 10.1109/72.991427.

IOSIFIDIS, Alexandros *et al.* A review of approximate methods for kernel-based big media data analysis. *In:* 2016 24th European Signal Processing Conference (EUSIPCO). [*S.l.*]: IEEE, ago. 2016. p. 1113–1117. DOI: 10.1109/EUSIPCO.2016.7760421.

JARVIS, R.A. On the identification of the convex hull of a finite set of points in the plane. **Information Processing Letters**, Elsevier, v. 2, n. 1, p. 18–21, mar. 1973. ISSN 0020-0190. DOI: 10.1016/0020-0190(73)90020-3.

JAYADEVA; KHEMCHANDANI, R.; CHANDRA, Suresh. Twin Support Vector Machines for Pattern Classification. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 29, n. 5, p. 905–910, mai. 2007. ISSN 0162-8828. DOI: 10.1109/TPAMI.2007.1068.

JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, Suresh. **Twin Support Vector Machines**. Cham: Springer International Publishing, 2017. v. 1, p. 1–211. (Studies in Computational Intelligence). ISBN 978-3-319-46184-7. DOI: 10.1007/978-3-319-46186-1.

JAYADEVA; KHEMCHANDANI, Reshma; CHANDRA, Suresh. **Variants of Twin Support Vector Machines: Some More Formulations**. [*S.l.*]: Springer, Cham, 2017. p. 103–123. DOI: 10.1007/978-3-319-46186-1_5.

JIANCHENG, Sun *et al.* Analysis of the Distance Between Two Classes for Tuning SVM Hyperparameters. **IEEE Transactions on Neural Networks**, v. 21, n. 2, p. 305–318, fev. 2010. ISSN 1045-9227. DOI: 10.1109/TNN.2009.2036999.

JOACHIMS, T. Making large-scale SVM learning practical. *In:* ADVANCES in kernel methods: support vector learning. [*S.l.*]: MIT Press, 1999. p. 376. ISBN 0262194163.

JOSHI, Snehal S.; KALE, Navnath D. Survey: Support Vector Machine and Its Deviations in Classification Techniques. **International Journal of Advanced Research in Computer Science and Software Engineering**, v. 4, n. 12, p. 993–998, 2014.

K. Q. BROWN. **Geometric Transforms for Fast Geometric Algorithms**. 1979. Tese (Doutorado) – Carnegie Mellon University.

KAWULOK, Michal; NALEPA, Jakub. Support Vector Machines Training Data Selection Using a Genetic Algorithm. *In:* STRUCTURAL, Syntactic, and Statistical Pattern Recognition. [*S.l.*]: Springer, Berlin, Heidelberg, 2012. p. 557–565. DOI: 10.1007/978-3-642-34166-3_61.

KEARNS, Michael J. **The computational complexity of machine learning**. [*S.l.*]: MIT Press, 1990. p. 165. ISBN 9780262111522.

KHEMCHANDANI, Reshma; JAYADEVA; CHANDRA, Suresh. Fuzzy Twin Support Vector Machines for Pattern Classification. *In:* MATHEMATICAL Programming and Game Theory for Decision Making. [*S.l.*: *s.n.*], abr. 2008. p. 131–142. DOI: `10.1142/9789812813220_0009`.

KHEMCHANDANI, Reshma; JAYADEVA; CHANDRA, Suresh. Incremental Twin Support Vector Machines. *In:* S K NEOGY (INDIAN STATISTICAL INSTITUTE, India); A K DAS (INDIAN STATISTICAL INSTITUTE, India); R B BAPAT (INDIAN STATISTICAL INSTITUTE, India) (Ed.). **Modeling, Computation and Optimization**. [*S.l.*]: World Scientific Publishing Co. Pte. Ltd, abr. 2009. cap. 17, p. 263–272. DOI: `10.1142/9789814273510_0017`.

LASKOV, Pavel *et al.* Incremental Support Vector Learning: Analysis, Implementation and Applications. **Journal of Machine Learning Research**, v. 7, Sep, p. 1909–1936, 2006. ISSN ISSN 1533-7928.

LE DIGABEL, Sébastien; SÉBASTIEN. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS Algorithm. **ACM Transactions on Mathematical Software**, ACM, v. 37, n. 4, p. 1–15, fev. 2011. DOI: `10.1145/1916461.1916468`.

LE, Quoc Viet; SARLOS, Tamas; SMOLA, Alexander Johannes. **Fastfood: Approximate Kernel Expansions in Loglinear Time**. [*S.l.*: *s.n.*], ago. 2014. arXiv: `1408.3060`.

LECUN, Y. *et al.* Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998. ISSN 00189219. DOI: `10.1109/5.726791`.

LEE, Yuh-jye; MANGASARIAN, Olvi L. RSVM: Reduced support vector machines. **DATA MINING INSTITUTE, COMPUTER SCIENCES DEPARTMENT, UNIVERSITY OF WISCONSIN**, p. 00–07, 2001.

LI, Fuxin; IONESCU, Catalin; SMINCHISESCU, Cristian. Random Fourier Approximations for Skewed Multiplicative Histogram Kernels. *In:* LECTURE Notes in Computer Science - 6376. [*S.l.*]: Springer, Berlin, Heidelberg, 2010. p. 262–271. DOI: `10.1007/978-3-642-15986-2_27`.

LIANG, Nan-Ying *et al.* A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. **IEEE Transactions on Neural Networks**, v. 17, n. 6, p. 1411–1423, nov. 2006. ISSN 1045-9227. DOI: `10.1109/TNN.2006.880583`.

LICHMAN, M. **UCI Machine Learning Repository**. [*S.l.*]: University of California, Irvine, School of Information e Computer Sciences, 2013.

LÓPEZ CHAU, Asdrúbal; LI, Xiaoou; YU, Wen. Convex and concave hulls for classification with support vector machine. **Neurocomputing**, Elsevier, v. 122, p. 198–209, dez. 2013. ISSN 0925-2312. DOI: `10.1016/J.NEUCOM.2013.05.040`.

LOPEZ CHAU, Asdrubal *et al.* Support vector candidates pre selection strategy based on non convex hulls. *In:* 2010 7th International Conference on Electrical Engineering Computing Science and Automatic Control. [*S.l.*]: IEEE, set. 2010. p. 345–350. DOI: `10.1109/ICEEE.2010.5608592`.

LOPEZ-CHAU, Asdrubal; LI, Xiaoou; YU, Wen. Convex-Concave Hull for Classification with Support Vector Machine. *In:* 2012 IEEE 12th International Conference on Data Mining Workshops. [*S.l.*]: IEEE, dez. 2012. p. 431–438. DOI: `10.1109/ICDMW.2012.76`.

LOSING, Viktor; HAMMER, Barbara; WERSING, Heiko. Incremental on-line learning: A review and comparison of state of the art algorithms. **Neurocomputing**, v. 275, p. 1261–1274, jan. 2018. ISSN 09252312. DOI: `10.1016/j.neucom.2017.06.084`.

LOSING, Viktor; HAMMER, Barbara; WERSING, Heiko. Interactive online learning for obstacle classification on a mobile robot. *In:* 2015 International Joint Conference on Neural Networks (IJCNN). [*S.l.*]: IEEE, jul. 2015. p. 1–8. DOI: `10.1109/IJCNN.2015.7280610`.

LOSING, Viktor; HAMMER, Barbara; WERSING, Heiko. Interactive online learning for obstacle classification on a mobile robot. *In:* 2015 International Joint Conference on Neural Networks (IJCNN). [*S.l.*]: IEEE, jul. 2015. p. 1–8. DOI: `10.1109/IJCNN.2015.7280610`.

MAATEN, L.J.P. van der; MAATEN, L.J.P. van der *et al.* **Dimensionality Reduction: A Comparative Review**. [*S.l.*], 2009. p. 1–36.

MAATEN, Laurens van der. Accelerating t-SNE using Tree-Based Algorithms. **Journal of Machine Learning Research**, v. 15, p. 3221–3245, 2014.

MAATEN, Laurens van der; HINTON, Geoffrey. Visualizing Data using t-SNE. **Journal of Machine Learning Research**, v. 9, Nov, p. 2579–2605, 2008. ISSN ISSN 1533-7928.

MANGASARIAN, O. L. Generalized Support Vector Machines. **ADVANCES IN LARGE MARGIN CLASSIFIERS**, p. 135–146, 1998.

MANGASARIAN, O. L.; MUSICANT, David R. Lagrangian support vector machines. **CrossRef Listing of Deleted DOIs**, JMLR.org, v. 1, p. 161–177, 2000. DOI: `10.1162/15324430152748218`.

MANGASARIAN, O.L.; WILD, E.W. Multisurface proximal support vector machine classification via generalized eigenvalues. **IEEE Transactions on Pattern Analysis**

and Machine Intelligence, v. 28, n. 1, p. 69–74, jan. 2006. ISSN 0162-8828. DOI: `10.1109/TPAMI.2006.17`.

MINEAR, Meredith; PARK, Denise C. A lifespan database of adult facial stimuli. **Behavior research methods, instruments, & computers : a journal of the Psychonomic Society, Inc**, v. 36, n. 4, p. 630–3, nov. 2004. ISSN 0743-3808.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, Elsevier Science Ltd., v. 24, n. 11, p. 1097–1100, nov. 1997. ISSN 03050548. DOI: `10.1016/S0305-0548(97)00031-2`.

MORÉ, Jorge J.; WILD, Stefan M. Benchmarking Derivative-Free Optimization Algorithms. **SIAM Journal on Optimization**, Society for Industrial e Applied Mathematics, v. 20, n. 1, p. 172–191, jan. 2009. ISSN 1052-6234. DOI: `10.1137/080724083`.

NALEPA, Jakub; KAWULOK, Michal. Adaptive memetic algorithm enhanced with data geometry analysis to select training data for SVMs. **Neurocomputing**, Elsevier, v. 185, p. 113–132, abr. 2016. ISSN 0925-2312. DOI: `10.1016/J.NEUCOM.2015.12.046`.

NALEPA, Jakub; KAWULOK, Michal. Selecting training sets for support vector machines: a review. **Artificial Intelligence Review**, Springer Netherlands, p. 1–44, jan. 2018. ISSN 0269-2821. DOI: `10.1007/s10462-017-9611-1`.

NAVIA-VAZQUEZ, A. *et al.* Weighted least squares training of support vector classifiers leading to compact and adaptive schemes. **IEEE Transactions on Neural Networks**, v. 12, n. 5, p. 1047–1059, 2001. ISSN 10459227. DOI: `10.1109/72.950134`.

NELDER, J. A.; MEAD, R. A Simplex Method for Function Minimization. **The Computer Journal**, Oxford University Press, v. 7, n. 4, p. 308–313, jan. 1965. ISSN 0010-4620. DOI: `10.1093/comjnl/7.4.308`.

NIEMANN, Heinrich. **Pattern Analysis and Understanding**. [*S.l.*]: Springer Berlin Heidelberg, 1990. p. 371. ISBN 9783642748998.

PEARSON, Karl. On lines and planes of closest fit to systems of points in space. **Philosophical Magazine Series 6**, Taylor & Francis Group, v. 2, n. 11, p. 559–572, nov. 1901. DOI: `10.1080/14786440109462720`.

PENG, Xinjun; XINJUN. A ν-twin support vector machine (ν-TSVM) classifier and its geometric algorithms. **Information Sciences**, Elsevier Science Inc., v. 180, n. 20, p. 3863–3875, out. 2010. ISSN 00200255. DOI: `10.1016/j.ins.2010.06.039`.

PHILLIPS, P.Jonathon *et al.* The FERET database and evaluation procedure for face-recognition algorithms. **Image and Vision Computing**, v. 16, n. 5, p. 295–306, abr. 1998. ISSN 02628856. DOI: `10.1016/S0262-8856(97)00070-X`.

PLATT, John C. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. **ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING**, p. 185–208, 1998.

PLATT, John C.; CRISTIANINI, Nello; SHAWE-TAYLOR, John. Large Margin DAGs for Multiclass Classification. *In:* ADVANCES in Neural Information Processing Systems 12 (NIPS 1999). [*S.l.*: *s.n.*], 2000. p. 547–553.

POLIKAR, R. *et al.* Learn++: an incremental learning algorithm for supervised neural networks. **IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)**, v. 31, n. 4, p. 497–508, 2001. ISSN 10946977. DOI: `10.1109/5326.983933`.

RAHIMI, Ali; RECHT, Benjamin. Random Features for Large-Scale Kernel Machines. *In:* NEURAL Information Processing Systems (NIPS). [*S.l.*: *s.n.*], 2008. p. 1177–1184.

RAHIMI, Ali; RECHT, Benjamin. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. *In:* NEURAL Information Processing Systems (NIPS). [*S.l.*]: Neural Information Processing Systems Conference, 2009. p. 1313–1320.

RIPLEY, Brian D. **Pattern recognition and neural networks**. [*S.l.*]: Cambridge University Press, 1996. p. 403. ISBN 0521460867.

ROSASCO, Lorenzo *et al.* Are Loss Functions All the Same? **Neural Computation**, v. 16, n. 5, p. 1063–1076, mai. 2004. ISSN 0899-7667. DOI: `10.1162/089976604773135104`.

ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. **PSYCHOLOGICAL REVIEW**, p. 65–386, 1958.

RUSSELL, Stuart J. (Stuart Jonathan); NORVIG, Peter.; DAVIS, Ernest. **Artificial intelligence : a modern approach**. [*S.l.*]: Prentice Hall, 2010. p. 1132. ISBN 0136042597.

SAFFARI, Amir *et al.* On-line Random Forests. *In:* 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops. [*S.l.*]: IEEE, set. 2009. p. 1393–1400. DOI: `10.1109/ICCVW.2009.5457447`.

SAMUEL, A. L. Some Studies in Machine Learning Using the Game of Checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, jul. 1959. ISSN 0018-8646. DOI: `10.1147/rd.33.0210`.

SATO, Atsushi; YAMADA, Keiji. Generalized Learning Vector Quantization. *In:* NEURAL Information Processing Systems Conference (NIPS). [*S.l.*: *s.n.*], 1996. p. 423–429.

SCHAPIRE, Robert E. *et al.* **Boosting the margin: A new explanation for the effectiveness of voting methods**. [*S.l.*]: Morgan Kaufmann Publishers, 1997. p. 430. ISBN 1558604863.

SCHMIDT, Philip *et al.* Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection. *In:* PROCEEDINGS of the 2018 on International Conference on Multimodal Interaction - ICMI '18. New York, New York, USA: ACM Press, 2018. p. 400–408. DOI: `10.1145/3242969.3242985`.

SCHOLKOPF, Bernhard.; BURGES, Christopher J. C.; SMOLA, Alexander J. **Advances in kernel methods : support vector learning**. [*S.l.*]: MIT Press, 1999. p. 376. ISBN 0262194163.

SHAO, Yuan-Hai; DENG, Nai-Yang. A coordinate descent margin based-twin support vector machine for classification. **Neural Networks**, Pergamon, v. 25, p. 114–121, jan. 2012. ISSN 0893-6080. DOI: `10.1016/J.NEUNET.2011.08.003`.

SHAO, Yuan-Hai; ZHANG, Chun-Hua *et al.* Improvements on Twin Support Vector Machines. **IEEE Transactions on Neural Networks**, v. 22, n. 6, p. 962–968, jun. 2011. ISSN 1045-9227. DOI: `10.1109/TNN.2011.2130540`.

SHEN, Xiang-Jun *et al.* Large-scale support vector machine classification with redundant data reduction. **Neurocomputing**, Elsevier, v. 172, p. 189–197, jan. 2016. ISSN 0925-2312. DOI: `10.1016/J.NEUCOM.2014.10.102`.

SHIN, Hyunjung; CHO, Sungzoon. Neighborhood Property–Based Pattern Selection for Support Vector Machines. **Neural Computation**, v. 19, n. 3, p. 816–855, mar. 2007. ISSN 0899-7667. DOI: `10.1162/neco.2007.19.3.816`.

SOLOMONOFF, R.J. A formal theory of inductive inference. Part I. **Information and Control**, Academic Press, v. 7, n. 1, p. 1–22, mar. 1964. ISSN 0019-9958. DOI: `10.1016/S0019-9958(64)90223-2`.

SOLOMONOFF, R.J. A formal theory of inductive inference. Part II. **Information and Control**, Academic Press, v. 7, n. 2, p. 224–254, jun. 1964. ISSN 0019-9958. DOI: `10.1016/S0019-9958(64)90131-7`.

STREET, W. Nick; KIM, YongSeog. A streaming ensemble algorithm (SEA) for large-scale classification. *In:* PROCEEDINGS of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01. New York, New York, USA: ACM Press, 2001. p. 377–382. DOI: `10.1145/502512.502568`.

SUYKENS, J.A.K.; VANDEWALLE, J. Least Squares Support Vector Machine Classifiers. **Neural Processing Letters**, Kluwer Academic Publishers, v. 9, n. 3, p. 293–300, 1999. ISSN 13704621. DOI: `10.1023/A:1018628609742`.

SUYKENS, Johan A K; VAN GESTEL, Tony *et al.* **Least Squares Support Vector Machines**. [*S.l.*]: WORLD SCIENTIFIC, nov. 2002. p. 310. ISBN 978-981-238-151-4. DOI: `10.1142/5089`.

TANG, Wan Mei. Fuzzy SVM with a New Fuzzy Membership Function to Solve the Two-Class Problems. **Neural Processing Letters**, Springer US, v. 34, n. 3, p. 209–219, dez. 2011. ISSN 1370-4621. DOI: `10.1007/s11063-011-9192-y`.

TIAN, YingJie; JU, XuChan *et al.* Improved twin support vector machine. **Science China Mathematics**, Springer Berlin Heidelberg, v. 57, n. 2, p. 417–432, fev. 2014. ISSN 1674-7283. DOI: `10.1007/s11425-013-4718-6`.

TIAN, Yingjie; QI, Zhiquan. Review on: Twin Support Vector Machines. **Annals of Data Science**, Springer Berlin Heidelberg, v. 1, n. 2, p. 253–277, jun. 2014. ISSN 2198-5804. DOI: `10.1007/s40745-014-0018-4`.

TOMAR, Divya; AGARWAL, Sonali. Twin Support Vector Machine: A review from 2007 to 2014. **Egyptian Informatics Journal**, Elsevier, v. 16, n. 1, p. 55–69, mar. 2015. ISSN 1110-8665. DOI: `10.1016/J.EIJ.2014.12.003`.

TURING, A. M. COMPUTING MACHINERY AND INTELLIGENCE. **Mind**, v. LIX, n. 236, p. 433–460, 1950. DOI: `10.1093/mind/LIX.236.433`.

TVEIT, Amund; HETLAND, Magnus Lie; ENGUM, Håavard. Incremental and Decremental Proximal Support Vector Classification using Decay Coefficients. *In:* DATA Warehousing and Knowledge Discovery. [*S.l.*]: Springer, Berlin, Heidelberg, 2003. p. 422–429. DOI: `10.1007/978-3-540-45228-7_42`.

URQUHART, R.B. Reply: Algorithms for computing relative neighbourhood graph. **Electronics Letters**, v. 16, n. 22, p. 860, 1980. ISSN 00135194. DOI: `10.1049/el:19800612`.

WAINER, Jacques; CAWLEY, Gavin. Empirical Evaluation of Resampling Procedures for Optimising SVM Hyperparameters. **Journal of Machine Learning Research**, v. 18, n. 15, p. 1–35, 2017.

WAN ZHANG; IRWIN KING. A study of the relationship between support vector machine and Gabriel graph. *In:* PROCEEDINGS of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290). [*S.l.*]: IEEE, 2002. p. 239–244. DOI: `10.1109/IJCNN.2002.1005476`.

WANG, Defeng; SHI, Lin. Selecting valuable training samples for SVMs via data structure analysis. **Neurocomputing**, Elsevier, v. 71, n. 13-15, p. 2772–2781, ago. 2008. ISSN 0925-2312. DOI: `10.1016/J.NEUCOM.2007.09.008`.

WANG, J. Y. **Application of support vector machines in bioinformatics**. 2002. Master's thesis – National Taiwan University.

WANG, Jigang; NESKOVIC, Predrag; COOPER, Leon N. Training Data Selection for Support Vector Machines. *In:* ADVANCES in Natural Computation. [*S.l.*]: Springer, Berlin, Heidelberg, 2005. p. 554–564. DOI: `10.1007/11539087_71`.

WITTEN, I. H. (Ian H.) *et al.* **Data mining : practical machine learning tools and techniques**. [*S.l.*: *s.n.*], 2016. p. 621. ISBN 9780128043578.

WOLPERT, D.H.; MACREADY, W.G. No free lunch theorems for optimization. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 67–82, abr. 1997. ISSN 1089778X. DOI: `10.1109/4235.585893`.

WOLPERT, David H. The Lack of A Priori Distinctions Between Learning Algorithms. **Neural Computation**, v. 8, n. 7, p. 1341–1390, out. 1996. ISSN 0899-7667. DOI: `10.1162/neco.1996.8.7.1341`.

WOLPERT, David H. Ubiquity symposium: Evolutionary computation and the processes of life: what the no free lunch theorems really mean: how to improve search algorithms. **Ubiquity**, v. 2013, December, p. 1–15, dez. 2013. ISSN 15302180. DOI: `10.1145/2555235.2555237`.

WU, Shinq-Jen; PHAM, Van-Hung; NGUYEN, Thi-Nga. Two-phase optimization for support vectors and parameter selection of support vector machines: Two-class classification. **Applied Soft Computing**, v. 59, p. 129–142, out. 2017. ISSN 15684946. DOI: `10.1016/j.asoc.2017.05.021`.

ZEGAL, Walid; ESSADDAM, Naceur; BRIMBERG, Jack. A New VNS Metaheuristic Using MADS as a Local Optimizer. **Journal of Multi-Criteria Decision Analysis**, v. 19, n. 5-6, p. 257–262, set. 2012. ISSN 10579214. DOI: `10.1002/mcda.1475`.

ZENG, Zhi-Qiang *et al.* A geometric approach to train SVM on very large data sets. *In:* 2008 3rd International Conference on Intelligent System and Knowledge Engineering. [*S.l.*]: IEEE, nov. 2008. p. 991–996. DOI: `10.1109/ISKE.2008.4731074`.

ZHANG, Dengsheng; LU, Guojun. Review of shape representation and description techniques. **Pattern Recognition**, v. 37, n. 1, p. 1–19, jan. 2004. ISSN 00313203. DOI: `10.1016/j.patcog.2003.07.008`.