FEDERAL UNIVERSITY OF CEARÁ

CENTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

POST-GRADUATION PROGRAM IN COMPUTER SCIENCE

DOCTORAL DEGREE IN COMPUTER SCIENCE

SAULO ANDERSON FREITAS DE OLIVEIRA

ON MODEL COMPLEXITY REDUCTION IN INSTANCE-BASED LEARNERS

FORTALEZA

2021

SAULO ANDERSON FREITAS DE OLIVEIRA

ON MODEL COMPLEXITY REDUCTION IN INSTANCE-BASED LEARNERS

Thesis submitted to the Post-Graduation Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Computer Science. Concentration Area: Machine Learning.

Advisor: Prof. Dr. João Paulo Pordeus Gomes.

Co-advisor: Prof. Dr. Ajalmar Rêgo da Rocha Neto.

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

O51o Oliveira, Saulo Anderson Freitas de.
On model complexity reduction in instance-based learners / Saulo Anderson Freitas de Oliveira. – 2021.
106 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2021.
Orientação: Prof. Dr. João Paulo Pordeus Gomes.
Coorientação: Prof. Dr. Ajalmar Rêgo da Rocha Neto.

1. Instance selection. 2. Minimal learning machine. 3. Regularization. 4. Least-squares support vector machine. I. Título.

CDD 005

SAULO ANDERSON FREITAS DE OLIVEIRA

ON MODEL COMPLEXITY REDUCTION IN INSTANCE-BASED LEARNERS

Thesis submitted to the Post-Graduation Program in Computer Science of the Center of Science of the Federal University of Ceará, as a partial requirement for obtaining the title of Doctor in Computer Science. Concentration Area: Machine Learning.

Approved on: July 23th, 2021.

EXAMINATION BOARD

Prof. Dr. João Paulo Pordeus Gomes (Advisor)
Federal University of Ceará (UFC)

Prof. Dr. Ajalmar Rêgo da Rocha Neto (Co-advisor)
Federal Institute of Ceará (IFCE)

Prof. Dr. César Lincoln Cavalcante Mattos
Federal University of Ceará (UFC)

Prof. Dr. Leonardo Ramos Rodrigues
Institute of Aeronautics and Space (IAE)

Prof. Dr. Paulo César Cortez
Federal University of Ceará (UFC)

This work is dedicated to my parents, Antônio Chagas de Oliveira and Cleuma Janete Freitas Silva, for their life example, dedication to their children, and always supporting my choices. I also dedicate this thesis to my brother, Artur Jefferson Freitas de Oliveira, and my dear nephew, Bernardo Raulino Freitas.

ACKNOWLEDGEMENTS

First of all, to God, because without his permission, nothing would be possible.

To my parents for motivating me to follow the study path from early. For continually being with me, providing me with love, and for always being supportive.

To my advisor, Prof. Dr. João Paulo Pordeus Gomes, for his guidance, knowledge, and many other qualities, his perseverance encouraged me to execute and accomplish this Ph.D. degree through this research. He mostly listened to many of my unsound and clear ideas. Some of them have become part of this thesis. I have no other words to define Prof. João Paulo but supportive and empathic.

To my co-advisor, Prof. Dr. Ajalmar Rêgo da Rocha Neto, for the inspiring discussions and accompanying me in my career from the technical degree (Do you remember *Observatório EPCT*?) up to this very day, the end of a Ph.D.! What a journey!

Special thanks go to Lucas Silva de Sousa and José Alberth Florêncio Vasconcelos for the discussions that made many aspects clear of this thesis.

Thanks to my colleagues from the Grupo de Lógica e Inteligência Artificial (LoGIA) and Laboratório de Informática Aplicada (LIA), professors, and other employees from MDCC for collaborating during the thesis.

And finally, to all who have contributed to this work in some way.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001. Therefore, my gratitude to CAPES for supporting me with a scholarship for four years, which was essential to accomplish this journey.

"Everything should be made as simple as possible, but no simpler."

(Albert Einstein? Louis Zukofsky? Roger Sessions? William of Ockham? Who knows!)

ABSTRACT

Instance-based learners habitually adopt instance selection techniques to reduce complexity and avoiding overfitting. Such learners' most recent and well-known formulations seek to impose some sparsity in their training and prediction structure alongside regularization to meet such a result. Due to the variety of such instance-based learners, we will draw attention to the Least-Squares Support Vector Machines and Minimal Learning Machines because they embody additional information beyond the stored instances so they can perform predictions. Later, in this thesis, we formulate variants that constrain candidate solutions within a specific functional space where overfitting is avoided, and model complexity is reduced. In the Least-Squares Support Vector Machines context, this thesis follows the pruning fashion by adopting the Class-Corner Instance Selection. Such an approach focuses on describing the class-corner relationship among the samples on the dataset to penalize the ones close to the corners. As for the Minimal Learning Machine model, this thesis introduces a new proposal called the Lightweight Minimal Learning Machine. It adopts regularization in the complexity term to penalize each sample's learning, resulting in a direct method. Usually, this penalization goes in alongside the term error. This thesis describes strategies based on random and observed linearity conditions related to the data for regression tasks. And, as for classification tasks, this thesis employs the before-mentioned class-corner idea to regularize them. Thus, resulting in the ones close to the corners suffering more penalization. By adopting such a methodology, we reduced the number of computations inherent in the original proposal's multilateration process without requiring any instance selection criterion, yielding a faster model for out-of-sample prediction. Additionally, another remarkable feature is that it derives a unique solution when other formulations rely on overdetermined systems.

Keywords: instance selection; minimal learning machine; regularization; least-squares support vector machine.

RESUMO

Os algoritmos de aprendizagem com base em instâncias normalmente adotam técnicas de seleção de instâncias para reduzir a complexidade e evitar o sobreajuste. As formulações mais recentes e conhecidas de tais algoritmos buscam obter alguma esparsidade em sua estrutura de treinamento e predição junto com a regularização para atingir tal resultado. Devido à variedade de algoritmos com base em instâncias, nesta tese, daremos atenção para as Máquinas de Vetores de Suporte de Mínimos Quadrados e Máquinas de Aprendizagem Mínimas porque ambas incorporam informações adicionais além das instâncias armazenadas para que possam realizar predições. Posteriormente, nesta tese, formulamos variantes que restringem as soluções candidatas dentro de um espaço funcional específico onde o sobreajuste é evitado e a complexidade do modelo é reduzida. No contexto de Máquinas de Vetores de Suporte de Mínimos Quadrados, esta tese segue o método de poda, adotando um algoritmo de seleção de instâncias com base em canto de classe. Essa abordagem se concentra em descrever a relação de canto de classe entre as amostras no conjunto de dados para penalizar as demais amostras próximas aos cantos. Quanto às Máquinas de Aprendizagem Mínimas, esta tese apresenta uma nova proposta denominada Máquina de Aprendizagem Mínima de Pesos Leves. Essa nova proposta adota a regularização no termo de complexidade para penalizar o aprendizado de cada amostra, resultando em um método direto, já que normalmente essa penalização acompanha o termo erro. Esta tese descreve estratégias com base em condições aleatórias e características lineares da função alvo relacionadas aos dados para tarefas de regressão. E, quanto à tarefas de classificação, esta tese emprega a ideia de canto de classe mencionada anteriormente para regularizá-las. Assim, resultando em que as amostras próximas aos cantos sofram mais penalização. Ao adotar essa metodologia, reduzimos o número de cálculos inerentes ao processo de multilateração da proposta original sem exigir nenhum critério de seleção de instância, gerando um modelo mais rápido para predições. Além disso, outra característica notável é que é derivada uma solução única, enquanto que outras formulações dependem de sistemas sobredeterminados.

Palavras-chave: seleção de instâncias; máquina de aprendizagem mínima; regularização; máquina de vetor de suporte por mínimos quadrados.

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS AND ACRONYMS

ACC	Accuracy.
BAN	Banana data set.
CCIS	Class-Corner Instance Selection.
CC-LSSVM	Class-Corner LSSVM.
CCLW-MLM	Class-Corner LW-MLM.
CCP-LSSVM	Coupled Compressive Pruning LSSVM.
CD	Critical Difference.
FAST	Features from Accelerated Segment Test.
FCC-LSSVM	Fixed-size CC-LSSVM.
FSA-LSSVM	Fast Sparse Approximation scheme for LSSVM.
Full-MLM	Minimal Learning Machine using all RPs.
GAS-LSSVM	Single Genetic Algorithm for Sparse LSSVM.
IP-LSSVM	Two-step sparse LSSVM classifier.
K-NN	Nearest K-Neighbors.
KKT	Karush-Kuhn-Tucker Conditions.
KS2Sample	Kolmogorov-Smirnov hypothesis test.
LSSVM	Least-Squares Support Vector Machines.
LSTSVM	Least-Squares Twin Support Vector Machine.
LW-MLM	Lightweight Minimal Learning Machine.
MLM	Minimal Learning Machine.
MOGAS-LSSVM	Multi-Objective Genetic Algorithm for Sparse LSSVM.
NCD	Nearest Class-corner Distance.

NORM	Scaled norm value of matrix **B** in MLMs.
P-LSSVM	Pruning Least-Squares Support Vector Machine.
PCP-LSSVM	Pivoted Choleskian of Primal LSSVM.
QP-SVM	Quadratic Programming version of SVM.
R^2	Coefficient of determination.
Random-MLM	Minimal Learning Machine with Random selection for RPs.
Rank-MLM	Rank Minimal Learning Machine.
RBF	Radial Basis Function.
RF	Random Forest.
RIP	Ripley data set.
RMSE	Root Mean Squared Error.
RP	Reference Points.
RVM	Relevance Vector Machine.
SPR	Sparseness/Sparsity.
SSD-LSSVM	Sparsified Subsampled Dual LSSVM.
SV	Support Vectors.
SVM	Support Vector Machine Machines.
SVR	Support Vector Regressor.
TMN	Two Moons data set.

LIST OF SYMBOLS

$\mathbf{x}$	a pattern (feature vector).
$\mathbf{y}$	a label pattern.
$\hat{\mathbf{y}}$	an estimated label.
$\mathcal{D}$	a dataset.
$\mathbf{x}_i$	the i-th pattern from $\mathcal{D}$
$\mathbf{y}_i$	the i-th pattern label from $\mathcal{D}$
$d(\cdot,\cdot)$	a distance (dissimilarity) function.
$f(\cdot)$	the true function.
$h(\cdot)$	an aproximation of $f(\cdot)$.
$h^\star(\cdot)$	the best aproximation of f in $\mathcal{H}$.
$\mathcal{H}$	the hypothesis space.
N	number of samples.
$\mathbb{R}^D$	the dimension of the input space.
$\mathbb{R}^S$	the dimension of the output space.
$\mathcal{J}(\cdot)$	the learning cost function.
$\mathbf{w}$	the linear model weights.
$\boldsymbol{\varepsilon}$	the error (slack variables).
$\phi(\cdot)$	a feature map to a high-dimensional.
$\boldsymbol{\alpha}$	the Lagrangian multipliers.
b	the bias in LSSVM.
$\mathcal{K}(\cdot,\cdot)$	a Mercer kernel function.
$\mathbf{I}$	the Identity matrix.
σ	the (Gaussian) RBF kernel window paramater.
$\mathbf{D}$	the pairwise Euclidean distance matrix from input samples.
$\boldsymbol{\Delta}$	the pairwise Euclidean distance matrix from output samples.
$\mathbf{P}$	the regularization matrix sample-based in LW-MLM.

$\mathbf{E}$	the Residual matrix.
$\Phi(\cdot)$	the distance mapping function for input samples.
$\Psi(\cdot)$	the d istance mapping function for output samples.
$\|\cdot\|_2$	the Euclidean norm.
$\|\cdot\|_{\mathcal{F}}$	the Frobernius norm.
$\mathbf{B}$	the mapping matrix for linear transformations in MLM.
$\hat{\mathbf{B}}$	an estimated mapping matrix for linear transformations in MLM.
λ	the regularization parameter.
$\mathbb{1}[\cdot]$	the indicatior function.
$\Gamma(\cdot)$	the relevance function for a given sample in CCIS.
ζ	the maximum distance of a query sample to the class-corners.
$\varsigma(\cdot)$	the cost by class-corner nearness function.
$\mathcal{PS}$	the set of class-corner candidate samples.
$\mathcal{SV}$	the set of class-corner samples.
NCD$(\cdot)$	the Nearest Corner Distance NCD of a given sample.
R^2	the coefficient of determination between predicted and actual labels.

CONTENTS

1 INTRODUCTION

Instance-based learners are computational models that, instead of making explicit generalizations, compare instances of new problems with instances seen in the learning process (previously stored in memory) (AHA *et al.*, 1991). From such a class of models, we highlight the algorithm of the Nearest K-Neighbors (K-NN) (COVER; HART, 1967), the Support Vector Machine Machines (SVM) (CORTES; VAPNIK, 1995), the Relevance Vector Machine (RVM) (TIPPING, 2001) and, more recently, the Minimal Learning Machine (MLM) (SOUZA JÚNIOR *et al.*, 2015).

Such models build hypotheses directly from the training instances themselves, thus implying that the hypotheses' complexity can grow with the data. For complex enough models with a large number of free parameters, a perfect fit to the training data is possible (RUSSELL *et al.*, 2020). However, the generalization, as the complexity increases, tends to decrease.

In this case, reducing complexity means restricting the amount of data used in the learning process. By this reduction, we also restrict the space of hypotheses that the model can generalize. Following the principle of *Occam's razor* (MACKAY, 1992): balancing both the complexity of the induced model and the ability to generalize ends up being a challenging task, while it is also highly desired.

Also, the difficulties inherent in dealing with the instance selection problem increase, as some techniques still treat such a selection based on empirical assumptions about the instances' locations. Moreover, some techniques treat such a selection as an isolated task, not fully incorporating its effects into the induced models (GARCIA *et al.*, 2012).

Aditionally, from the instance selection perspective, analyzing the instance selection problem from a data set of size N individually, we observe two questions: (i) defining a subset from the data set, with K samples; and (ii) how to identify which specific combination of samples among the $\binom{N}{K}$ possible combinations.

By borrowing a definition from Least-Square Support Vector Machines (LSSVM) (SUYKENS; VANDEWALLE, 1999) literature, Mall and Suykens (2015) categorized efforts to reduce the number of support vectors in LSSVMs into two groups: (i) reduction methods and (ii) direct methods. The reduction methods focus on training a usual LSSVM and, then, apply some pruning strategy, identifying the new support vectors so that the newly trained model can be derived, while the direct ones enforce sparsity from the beginning. Additionally, in the most recent and adopted formulations of Minimal Learning Machine (MLM), named Random-

MLM (SOUZA JÚNIOR *et al.*, 2015), a lower-rank linear system is obtained by (randomly) selecting reference points (samples) as a manner to also impose some sparsity in both training and prediction alongside regularization.

In both cases, the sparsity criterion, often related to overfitting in neural network sparsity, is interpreted as an indicator of success in learning due to several empirical investigations (HUESMANN *et al.*, 2021). However, recent experiments suggest that our comprehension of sparsity is insufficient to fully understand its underlying relationship to generalization errors (HUESMANN *et al.*, 2021 apud LIU *et al.*, 2019) (HUESMANN *et al.*, 2021 apud ZHOU *et al.*, 2019). Interestingly, even though sparsity inevitably induces an underutilization of the network's capacity (AYINDE *et al.*, 2019), it has never been considered to be used to explain overfitting.

Nevertheless, sparsity in LSSVMs and MLMs is associated with restricting the hypothesis space since it acts by accounting for (and used for restricting) the number of parameters – the support vectors in LSSVM and the reference points in MLM – to later measure the model complexity. In short, one can indeed account for the degree of freedom via sparsity, but sparsity is not directly related to generalization.

Another aspect that is not widespread in the current literature, and the object of study and contribution of this thesis, is the use instance selection algorithm as regularization in the complexity term, resulting in a direct method. Usually, we see formulation respect to applying instance selection algorithms with a focus on the error term or simply acting as a reduction method. Such a path mentioned above does not thrive in this thesis.

1.1 Objectives

Provide an instance selection algorithm that later can be embedded into the estimation of model parameters in one solution. In this way, we will have an algorithm capable of dealing with two existing problems in instance-based models: (i) the lack of an instance selection mechanism; (ii) and control of the model's complexity. In a more specific sense, this thesis has the following objectives:

a) to provide an instance selection algorithm;
b) to assess such an instance selection algorithm in the LSSVM framework;
c) to provide a new MLM formulation able to integrate such an instance selection mechanism during the learning phase as regularization;
d) to evaluate aspects of the proposals mentioned above, namely, the prediction

error performance, the goodness-of-fit of estimated vs. measured values, and the model complexity.

1.2 Publications

The following articles were written and published during the period in which this thesis was under development:

a) **Saulo A. F. Oliveira**, João P. P. Gomes e Ajalmar R. Rocha Neto. "Sparse Least-Squares Support Vector Machines via Accelerated Segmented Test: a Dual Approach". Neurocomputing. 2018.

b) José A. V. Florêncio, **Saulo A. F. Oliveira**, João P. P. Gomes e Ajalmar R. Rocha Neto. "A new perspective for Minimal Learning Machines: A lightweight approach". Neurocomputing. 2020.

1.3 Document organization

This rest of the thesis is organized in six chapters, which are described below.

Chapter 2 presents the **Theoretical Background** and various aspects related to the foundation concepts of this thesis. We first present the Instance based-learners because we will treat the instance relevance later in the following chapters and contributions to this thesis. From this perspective, we present the Least-Squares Support Vector Machines and the Minimal Learning Machines, describing their modeling aspects related to when parameters pop in but driven by training examples. Furthermore, we present some foundations concepts related to the control of complexity (Regularization) associated with training examples.

In Chapter 3, we present the **first contribution** of this thesis, named Class Corner Instance Selection, as a manner to provide a control mechanism for Reducing Complexity by Instance Selection on Least-Squares Support Vector Machines. From that, we derived the **second contribution** of this thesis, a LSSVM model that incorporates the Instance Selection, that performs both reduce set and support vector selection.

In Chapter 4, we deal with reducing complexity in Minimal Learning Machines by Instance Selection. Here, we formulate the **third contribution** of this thesis: a "Lightweight" Minimal Leaning Machine model that employs regularization by Instance Selection. In this contribution, we highlight how the general MLM framework can take advantage of such a

strategy to learn in a restricted hypothesis space and generate a *faster* model for prediction in regression tasks.

In Chapter 5, we revisit the *Lightweight* Minimal Leaning Machine. This time, we combine both the **first contribution** and the **third contribution** of this thesis to deal with classification tasks in a straightforward manner. Such a combination results in **the fourth and last contribution** is the Class Corner Lightweight Minimal Learning Machine. It relies on regularizing the distance complement (as for a non-corner sample) to each class' corners to achieve the final parameters.

Finally, we present the concluding remarks and possible unfolding works from this thesis's four contributions in Chapter 6.

1.4 Statement of authorship

The work presented in this thesis comes from two previously published papers. Next, we now provide authorship information for such papers.

Chapter 3 is based on:

> Saulo A. F. Oliveira, João P. P. Gomes e Ajalmar R. Rocha Neto. "Sparse Least-Squares Support Vector Machines via Accelerated Segmented Test: a Dual Approach". In Neurocomputing, 2018.

The lead author – of the paper above and the thesis in hand - developed the problem formulation, developed the algorithm, developed the empirical tests, and wrote most of the paper. The second and third authors provided advisory feedback w.r.t. the design and development of the project. They contributed directly to the writing and editing of the paper.

Chapter 4 is based on:

> José A. V. Florêncio, Saulo A. F. Oliveira, João P. P. Gomes e Ajalmar R. Rocha Neto. "A new perspective for Minimal Learning Machines: A lightweight approach". In Neurocomputing, 2020.

The lead author developed the algorithm and empirical tests and wrote $\approx$ ½ of the paper. The second author developed the problem formulation, the theoretical analysis, provided feedback on the experiment design, and contributed significantly to the paper's general writing and editing. Both third and fourth authors provided advisory feedback w.r.t. the design and development of the project. They contributed directly to the writing and editing of the paper.

Finally, Chapter 5 comprises a novel proposal sent for peer review.

Saulo A. F. Oliveira, João P. P. Gomes e Ajalmar R. Rocha Neto. “Controlling complexity in Instance-based classifiers: a class-corner approach”.

The lead author developed the problem formulation, developed the algorithm, developed the empirical tests, and wrote most of the paper. The second and third authors provided advisory feedback w.r.t. the design and development of the project. They contributed directly to the writing and editing of the paper.

2 THEORETICAL BACKGROUND

A large class of machine learning problems ends up as being equivalent to a function estimation/approximation task. The function is "learned" during the learning/training phase by digging in the information that resides in the available training data set. This function relates the so-called input variables to the output variable(s). Once this functional relationship is established, one can in turn exploit it to predict the value(s) of the output(s), based on measurements from the respective input variables; these predictions can then be used to proceed to the decision making phase (THEODORIDIS, 2020).

In parametric modeling, the aforementioned functional dependence that relates the input to the output is defined via a set of parameters, whose number is fixed and a-priori known. The values of the parameters are unknown and have to be estimated based on the available input-output observations. In contrast to the parametric, there are the so-called nonparametric methods. In such methods, parameters may still be involved to establish the input–output relationship, yet their number is not fixed; it depends on the size of the data set and it grows with the number of observations (THEODORIDIS, 2020).

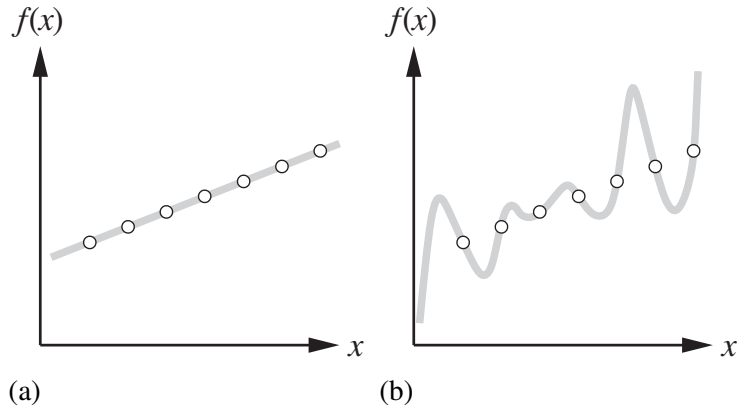
This approach is called **instance-based learning** or memory-based learning. The simplest instance-based learning method is **table lookup**: take all the training examples, put them in a lookup table, and then when asked for a given query $\mathbf{x}$, see if $\mathbf{x}$ is in the table; if it is, return the corresponding label $\mathbf{y}$. The problem with this table lookup method is that it does not generalize well: when $\mathbf{x}$ is not the table, all it can do is return some default value (RUSSELL *et al.*, 2020).

Fousing on the task of supervised learning which states the following: Given a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ of N example input–output sample pairs, where $\mathbf{x}_i \in \mathbb{R}^D$ and $\mathbf{y}_i \in \mathbb{R}^S$ where each $\mathbf{y}_i$ was generated by an unknown function $\mathbf{y} = f(\mathbf{x})$, discover a function $h(\cdot)$ that approximates the true function $f(\cdot)$ (RUSSELL *et al.*, 2020).

From the above, one can notice that learning is a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set. Also, since $f(\cdot)$ is unknown, we will approximate it with a function $h(\cdot)$ selected from a hypothesis space $\mathcal{H}$.

In Figure 1, one see a fundamental problem in inductive learning: *how do we choose from among multiple consistent hypotheses?* Defining simplicity is not easy, but it seems clear that a degree-1 polynomial is simpler than a degree-7 polynomial, and thus Figure 1(a) should

Figure 1 – Possible hypotheses for some data set. In (a) there is a consistent linear hypothesis, while in (b) there is a consistent degree-7 polynomial hypothesis for the same data set.



Source – Adapted from Russell *et al.* (2020).

be preferred to Figure 1(b).

2.1 Instance-based learning

The earliest instance-based algorithms were synonymous with nearest neighbor for pattern classification, though the field has now progressed well beyond the use of such algorithms. At that time, the principle of instance-based methods was often understood in the earliest literature as follows: *"... similar instances have similar classification"* (AHA *et al.*, 1991). However, according to a broader and more powerful principle to characterize such methods would be: *" Similar instances are easier to model with a learning algorithm, because of the simplification of the class distribution within the locality of a test instance"* (AGGARWAL, 2014).

The primary output of an instance-based algorithm is a concept description. As in the case of a classification model, this is a function that maps instances to category values. However, unlike traditional classifiers, which use extensional concept descriptions, instance-based concept descriptions may typically contain a set of stored instances and some information about how the stored instances may have performed in the past during classification (AGGARWAL, 2014). According to AHA *et al.* (1991), there are three primary components in all instance-based learning algorithms:

a) **Similarity or Distance Function**. It computes the similarities between the training instances, or between the test instance and the training instances. This is

used to identify a locality around the test instance;

b) **Classification Function**. It yields a classification for a particular test instance with the use of the locality identified with the use of the (similarity) distance function. In the earliest descriptions of instance-based learning, a nearest neighbor classifier was assumed, though this was later expanded to the use of any kind of locally optimized model;

c) **Concept Description Updater**. It typically tracks the classification performance, and makes decisions on the choice of instances to include in the concept description.

Traditional classification algorithms eagerly construct explicit abstractions and generalizations (e.g., decision trees or rules) in a pre-processing phase, while instance-based learners use them along with the training data to construct the concept descriptions. Thus, the approach is *lazy* in the sense that knowledge of the test sample is required before model construction.

Clearly, the tradeoffs are different in the sense that "eager" algorithms avoid too much work at prediction but are myopic in their ability to create a specific model for a test instance in the most accurate way. Additionally, traditional modeling techniques such as decision trees, regression modeling, Bayes, or rule-based methods are commonly used to create an optimized classification model around the test instance. The optimization inherent in the test sample localization provides the most significant advantages of instance-based learning (AHA *et al.*, 1991).

In this thesis, we will draw attention to such instance based-learners in the context of Nearest Neighbors, Least-Squares Support Vector Machines, and Minimal Learning Machines. Such learners embody, beyond the stored instances, additional information so they can perform predictions. Later, in this thesis, we formulate variants that find them constrained candidate solutions that lie within a specific functional space. We describe such learners in the following subsections.

2.1.1 The Nearest Neighbor Classifier

One can improve on table lookup with a slight variation: given a query $\mathbf{x}_q$, find the k examples that are nearest to $\mathbf{x}_q$. This is called k-nearest neighbors lookup. We will use the notation $\mathcal{NN}_k(\mathbf{x}_q)$ to denote the set of k nearest neighbors. To do classification, first find $\mathcal{NN}_k(\mathbf{x}_q)$, then take the plurality vote of the neighbors (which is the majority vote in the case of

binary classification). To avoid ties, k is always chosen to be an odd number. To do regression, we can take the mean or median of the k neighbors, or we can solve a linear regression problem on the neighbors (RUSSELL *et al.*, 2020).

As usual, cross-validation can be used to select the best value of k. The very word "nearest" implies a distance metric (RUSSELL *et al.*, 2020). Typically, distances are measured with a Minkowski distance or L^p norm to measure the distance from a query point $\mathbf{x}_q$ to an example point $\mathbf{x}_j$, defined as

$$d(\mathbf{x}_j, \mathbf{x}_q) = \left(\sum_i |x_{j,i} - x_{q,i}|^p \right)^{\frac{1}{p}}. \tag{2.1}$$

The $\mathcal{NN}_k(\mathbf{x}_q)$ function is conceptually trivial: given a set of N examples and a query $\mathbf{x}_q$, iterate through the examples, measure the distance to $\mathbf{x}_q$ from each one, and keep the best k (RUSSELL *et al.*, 2020). From this, the output prediction is according to:

$$h(\mathbf{x}) = \arg\max_{\mathbf{v}} \sum_{(\mathbf{x}_i, \mathbf{y}_i)} \mathbb{1}[\mathbf{y}_i = \mathbf{v}] \text{ such that } (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{NN}_k(\mathbf{x}). \tag{2.2}$$

See both Algorithm 1 and Algorithm 2 for the training and prediction procedure in k-KNN, respectively.

Algorithm 1 Training procedure in k-NN.

KNN-TRAINING$(\mathcal{X}, \mathcal{Y})$

1 Store samples into memory (perhaps build a data structure with indexed data).

Algorithm 2 Prediction procedure in KNN.

KNN-PREDICT($\mathbf{x}^\star$,k)

▷ Rescue the k nearest neighbors from memory.

1 Compute $\mathcal{S} \leftarrow \mathcal{NN}_k(\mathbf{x})$.

▷ Yield the majority label from nearest neighbors.

2 **return** $\arg\max_{\mathbf{v}} \sum_{(\mathbf{x}_i, \mathbf{y}_i)} \mathbb{1}[\mathbf{y}_i = \mathbf{v}]$ so that $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{NN}_k(\mathbf{x})$.

If we are satisfied with an implementation that takes $\mathcal{O}(N)$ execution time, then that is the end of the story. But instance-based methods are designed for large data sets, so we would like an algorithm with sublinear run time. Elementary analysis of algorithms tells us that exact

table lookup is $\mathcal{O}(N)$ with a sequential table, $\mathcal{O}(\log N)$ with a binary tree, and $\mathcal{O}(1)$ with a hash table (RUSSELL *et al.*, 2020).

An interesting property of the nearest-neighbor ($k = 1$) classifier is that, in the limit $N \to \infty$, the error rate is never more than twice the minimum achievable error rate of an optimal classifier, i.e., one that uses the true class distributions (BISHOP, 2016 apud COVER; HART, 1967).

2.1.2 Least-Squares Support Vector Machine Classifiers

The Least-Squares Support Vector Machine (LSSVM) (SUYKENS; VANDEWALLE, 1999) is a supervised method whose training step consists of solving a linear system resulting from the optimally conditions that appear from minimizing the primal optimization problem of SVM (CORTES; VAPNIK, 1995) in a least-squares sense. Output prediction for new incoming inputs is achieved by using the resulting solution (Lagrange multipliers and bias) alongside the input data into a high-dimensional feature space.

The LSSVM primal problem is defined as:

$$\min_{\mathbf{w},\boldsymbol{\xi}} \mathcal{J}(\mathbf{w},\boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^\intercal\mathbf{w} + \frac{\gamma}{2}\sum_{i=1}^{N}\xi_i^2$$

$$\text{such that } \mathbf{w}^\intercal\boldsymbol{\phi}(\mathbf{x}_i) + b = y_i - \xi_i, \;\; i = 1,\ldots,N, \tag{2.3}$$

where $\boldsymbol{\phi} : \mathbb{R}^D \to \mathbb{R}^S$ is a feature map to a high-dimensional feature space (which can be infinite dimensional, i.e., $S = \infty$), $\boldsymbol{\xi} = \{\xi_i\}_{i=1}^N$ are the errors ($\xi_i \in \mathbb{R}$) and $\gamma \in \mathbb{R}^+$ is the cost parameter that controls the trade-off between allowing training errors and forcing rigid margins. The solution for Equation 2.3 is the saddle point of the following Lagrangian function:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \mathcal{J}(\mathbf{w},\boldsymbol{\xi}) + \sum_{i=1}^{N}\alpha_i\left(y_i - \mathbf{w}^\intercal\boldsymbol{\phi}(\mathbf{x}_i) - b - \xi_i\right), \tag{2.4}$$

where $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^N$ are the Lagrangian multipliers, with $\alpha_i \in \mathbb{R}$. After eliminating $\mathbf{w}$ and $\boldsymbol{\xi}$, one obtains the Karush-Kuhn-Tucker (KKT) system $\mathbf{Au} = \mathbf{v}$ from the conditions for optimality:

$$\underbrace{\left[\begin{array}{c|c} 0 & \mathbf{1}^\intercal \\ \hline \mathbf{1} & \boldsymbol{\Omega} + \gamma^{-1}\mathbf{I} \end{array}\right]}_{\mathbf{A}} \underbrace{\left[\begin{array}{c} b \\ \boldsymbol{\alpha} \end{array}\right]}_{\mathbf{u}} = \underbrace{\left[\begin{array}{c} 0 \\ \mathbf{Y} \end{array}\right]}_{\mathbf{v}}, \tag{2.5}$$

where $\boldsymbol{\Omega}$ is the *kernel* matrix in which $\Omega_{i,j} = \langle\boldsymbol{\phi}(\mathbf{x}_i)^\intercal\boldsymbol{\phi}(\mathbf{x}_j)\rangle = \mathcal{K}(\mathbf{x}_i,\mathbf{x}_j)$ for $i,j = 1,2,\ldots,N$ with $\mathcal{K}(\cdot,\cdot)$ a Mercer kernel function, $\mathbf{I}$ is the identity matrix of size $N \times N$, $\mathbf{Y} = [y_1, y_2, \ldots, y_N]^\intercal$,

and $\mathbf{1}$ is a matrix of ones with a proper dimension. From the KKT conditions, we get that

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i), \tag{2.6}$$

$$\alpha_i = y_i \xi_i. \tag{2.7}$$

From Equation 2.7, the main problem with respect to LSSVM arises: the lack of sparseness. In real-world problems ξ_i usually is nonzero and $y_i \in \{\pm 1\}$, thus, resulting in $\alpha_i \neq 0$. The pratical effect of such construction is that most input samples will be employed as support vectors.

The learning algorithm of LSSVM simple requires the solution of the system of linear equations described in Equation 2.5 where $\mathbf{A}$ is a non-singular symmetric dense matrix and $\mathbf{v}$ is a dense vector. The unique dense solution $\mathbf{u} = [b,\ \boldsymbol{\alpha}^\intercal]^\intercal$ can be obtained as follows $\mathbf{u} = \mathbf{A}^{-1}\mathbf{v}$. In possession of the Lagrange multipliers $\boldsymbol{\alpha}$ and the bias b from $\mathbf{u}$, one can now predict the out-of-sample inputs.

Predicting the outputs for new input data mainly refers to employ the resulting Lagrange multipliers $\boldsymbol{\alpha}$ and bias b alongside the input data into a high-dimensional feature space, that is

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i) + b\right). \tag{2.8}$$

Among all the kernel functions, the (Gaussian) RBF kernel:

$$\mathcal{K}(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{\| \mathbf{x} - \mathbf{z} \|_2^2}{2\sigma}\right), \text{ for } \sigma > 0,$$

is the most popular choice. For a more details with respect to the LSSVM model the reader is referred to the work by Suykens and Vandewalle (1999).

Furthermore, we present the algorithms for training and out-of-sample prediction in Algorithm 3 and Algorithm 4, respectively.

2.1.3 Minimal Learning Machine

The Minimal Learning Machine (MLM) (SOUZA JÚNIOR *et al.*, 2015) is a supervised method used for pattern recognition and regression tasks. From the general framework for supervised learning, MLM estimates $h(\cdot)$ for the target function $f(\cdot)$ from the data $\mathcal{D}$ through the distance domain. For that, the problem is stated by employing pairwise distance matrices of each point of $\mathcal{D}$, namely, $\mathbf{D}$ and $\boldsymbol{\Delta}$, both representing the Euclidean distance – in the notation of $\mathrm{d}(\cdot,\cdot)$

Algorithm 3 Training procedure in LSSVM.

LSSVM-TRAINING$(\mathcal{X}, \mathcal{Y}, \gamma)$

▷ Construct the linear system $\mathbf{Au} = \mathbf{v}$.

1 $\mathbf{A} = \left[\begin{array}{c|c} 0 & \mathbf{1}^\intercal \\ \hline \mathbf{1} & \mathbf{\Omega} + \gamma^{-1}\mathbf{I} \end{array}\right]$.

2 $\mathbf{v} = \begin{bmatrix} 0 \\ \mathbf{Y} \end{bmatrix}$.

▷ Solve it.

3 $\mathbf{u} = \mathbf{A}^{-1}\mathbf{v}$.

▷ Rescue the Lagrange multipliers $\boldsymbol{\alpha}$ and bias b from $\mathbf{u}$, that is $\mathbf{u} = [b,\ \boldsymbol{\alpha}^\intercal]^\intercal$.

4 **return** $\boldsymbol{\alpha}$, b.

Algorithm 4 Prediction procedure in LSSVM.

LSSVM-PREDICT$(\mathbf{x}^\star)$

▷ Employ both $\boldsymbol{\alpha}$ and b alongside the input data into a high-dimensional feature space.

1 Compute $\hat{\mathbf{y}} = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i) + b\right)$

2 **return** $\hat{\mathbf{y}}$.

– of each point from $\mathcal{D}$ to the i-th reference point of $\mathcal{D}$, i.e., $D_{i,j} = \mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{\Delta}_{i,j} = \mathrm{d}(\mathbf{y}_i, \mathbf{y}_j)$, then they have $N \times N$ dimensions.

For the sake of simplicity, in the following description and notation of MLM, consider the two distance mapping functions $\Phi : \mathbb{R}^D \to \mathbb{R}^N$ and $\Psi : \mathbb{R}^S \to \mathbb{R}^N$. Here $\Phi(\mathbf{x}) = \big[\mathrm{d}(\mathbf{x}, \mathbf{x}_1), \mathrm{d}(\mathbf{x}, \mathbf{x}_2), \ldots, \mathrm{d}(\mathbf{x}, \mathbf{x}_N)\big]^\intercal$ while $\Psi(\mathbf{y}) = \big[\mathrm{d}(\mathbf{y}, \mathbf{y}_1), \mathrm{d}(\mathbf{y}, \mathbf{y}_2), \ldots, \mathrm{d}(\mathbf{y}, \mathbf{y}_N)\big]^\intercal$. Furthermore, we call $\mathbf{D}$ and $\mathbf{\Delta}$ the input and output spaces, respectively. Stated that, by assuming that the mapping between the distance matrices has a linear structure for each response, the MLM model can be rewritten in the form:

$$\mathbf{\Delta} = \mathbf{DB} + \mathbf{E}, \tag{2.9}$$

where $\mathbf{E}$ is a residuals matrix.

The learning process consists of finding the mapping between the distances in the input and output space. Since we assume such a mapping has a linear structure for each response, the regression model can be rewritten in the form:

$$\min_{\mathbf{B}} \ \mathcal{J}(\mathbf{B}) = \| \mathbf{DB} - \mathbf{\Delta} \|_{\mathcal{F}}^2, \tag{2.10}$$

it can be estimated by:

$$\hat{\mathbf{B}} = \mathbf{D}^{-1}\boldsymbol{\Delta}. \tag{2.11}$$

Predicting the outputs for new input data mainly refers to project the new data point through the mapping and estimate the image of such a projection. Therefore, it is necessary that the pattern $\mathbf{x}$ be also represented in the domain of distances so that we can represent it in the output space. Such a representation is achieved by $\Phi(\mathbf{x})\,\mathbf{B}$. From this, the problem is to estimate the image $\hat{\mathbf{y}} = h(\mathbf{x})$, from $\Phi(\mathbf{x})\,\mathbf{B}$ and the images of reference points. This problem can be treated as a multilateration (NIEWIADOMSKA-SZYNKIEWICZ; MARKS, 2009). In a geometric viewpoint, estimate $\hat{\mathbf{y}}$ belonging to the set $\mathbb{R}^S$ is equivalent to solving the determined set of N non-linear equations corresponding to the S-dimensional hyper-spheres centered on the images of the reference points, denoted by $\{\mathbf{y}_i\}_{i=1}^{N}$. The location of $\hat{\mathbf{y}}$ can be estimated by minimizing the objective function below:

$$\hat{\mathbf{y}} = h(\mathbf{x}) = \arg\min_{\mathbf{y}} \parallel \Psi(\mathbf{y}) - \Phi(\mathbf{x})\,\mathbf{B} \parallel_2 . \tag{2.12}$$

For better understanding, we present, in short, the general algorithms for training and prediction procedures in MLM. Keep in mind that according to the selected formulation (presented in the following subsection), one might require other hyperparameters. The MLM training procedure is sketched in Algorithm 5. The training procedure comprises two parts: (i) the pairwise distance matrices computation; and (ii) the multi-response linear regression solution on such matrices. Additionally, the prediction procedure is done by projecting the data to the output space and find its image, as one can see in Algorithm 6.

Algorithm 5 Training procedure in MLM.

MLM-TRAINING$(\mathcal{X}, \mathcal{Y})$
1 Compute both distance pairwise matrices $\mathbf{D}$ and $\boldsymbol{\Delta}$ from $\mathcal{X}$ and $\mathcal{Y}$, respectively.
2 Compute $\hat{\mathbf{B}} = \mathbf{D}^{-1}\boldsymbol{\Delta}$.
3 **return** $\hat{\mathbf{B}}$.

2.2 Model Complexity in Machine Learning

In describing the computational complexity of an algorithm, we are generally interested in the number of basic mathematical operations, such as additions, multiplications and

Algorithm 6 Prediction procedure in MLM.

MLM-PREDICT($\mathbf{x}^\star, \hat{\mathbf{B}}$)

▷ Project $\mathbf{x}^\star$ to the output space by computing $\Phi(\mathbf{x}^\star)\,\hat{\mathbf{B}}$.

1 Compute $\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} \parallel \Psi(\mathbf{y}) - \Phi(\mathbf{x}^\star)\,\hat{\mathbf{B}} \parallel_2$.

2 **return** $\hat{\mathbf{y}}$.

divisions it requires, or in the time and memory needed on a computer. Although in the current context, the complexity issue emerges in a somewhat disguised form is usually defined in terms of the number of free parameters the model can learn, e.g., hidden units and layers and their connectivity, the form of activation functions, and free parameters of the learning algorithm itself.

From this perspective, one can choose different functions and norms as terms in the model formulation to account such a complexity.

2.2.1 Model performance vs. Model complexity

The central challenge in machine learning is that our algorithm must perform well on new, previously unseen inputs–not just those on which the model was trained. The ability to perform well on previously unobserved inputs is called generalization.

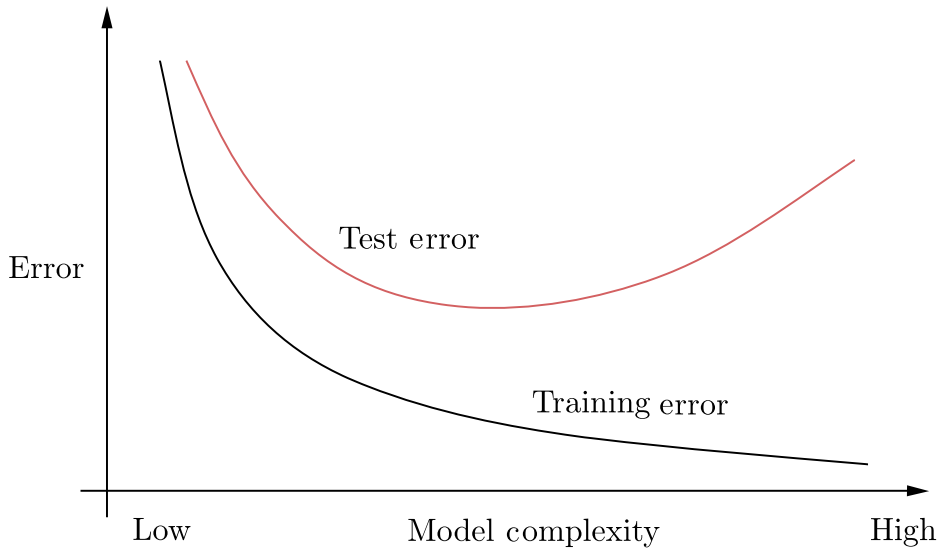
How can we affect performance on the test set when we can observe only the training set? The field of statistical learning theory provides some answers (GOODFELLOW *et al.*, 2016). Under this process, the expected test error is greater than or equal to the expected value of training error. The factors determining how well a machine learning algorithm will perform its ability to:

a) make the training error small;

b) make the gap between training and test error small.

In Figure 2, we see a classical expected result of having model complexity variation and the training error. The training error tends to zero as the model complexity increases; for complex enough models with a large number of free parameters, a perfect fit to the training data is possible. However, the test error initially decreases, because more complex models "learn" the data better, up to a certain point. After that point of complexity, the test error increases (THEODORIDIS, 2020).

Figure 2 – Model complexity vs. Error (Model performance).

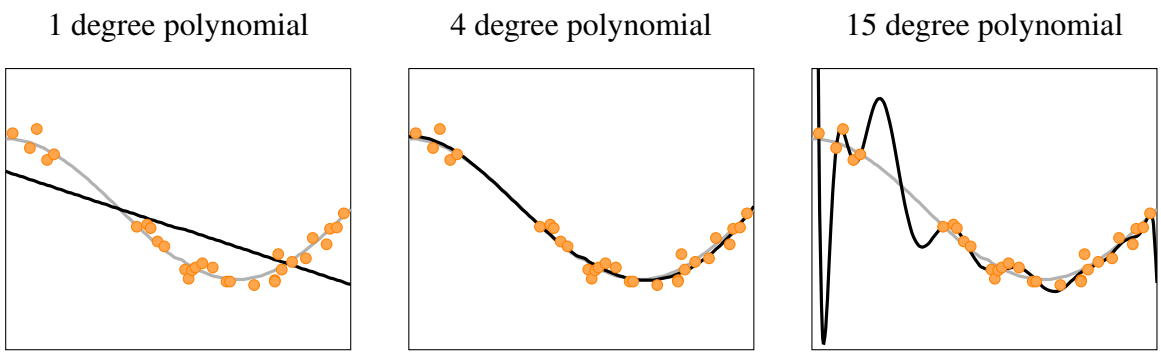


Source – Adapted from Theodoridis (2020).

Some efforts advocate for the idea of having a model complexity that corresponds to the minimum of the respective curve depicted in Figure 2. Such a task is indeed difficult because it is not possible to choose the model that fits the data best: more complex models can always fit the data better, leading us inevitably to implausible overparameterized models that generalize poorly.

From that, two central challenges in machine learning arise: **underfitting** and **overfitting**. Underfitting occurs when the model cannot obtain a sufficiently low error value on the training set. Overfitting occurs when the gap between the training error and test error is too large (GOODFELLOW *et al.*, 2016).

Figure 3 – Appropriate Fitting vs. Underfitting vs. Overfitting.



(a) Underfitting. (b) Appropriate capacity. (c) Overfitting.

Source – Own authorship. Inspired from Goodfellow *et al.* (2016).

One way to control the capacity of a learning algorithm is by choosing its hypothesis space, the set of functions that the learning algorithm is allowed to select as being the solution (GOODFELLOW *et al.*, 2016). In Figure 3, one can see how capacity is related to complexity. Models with insufficient capacity are unable to solve complex tasks. Models with high capacity can solve complex tasks, but when their capacity is higher than needed to solve the present task, they may overfit.

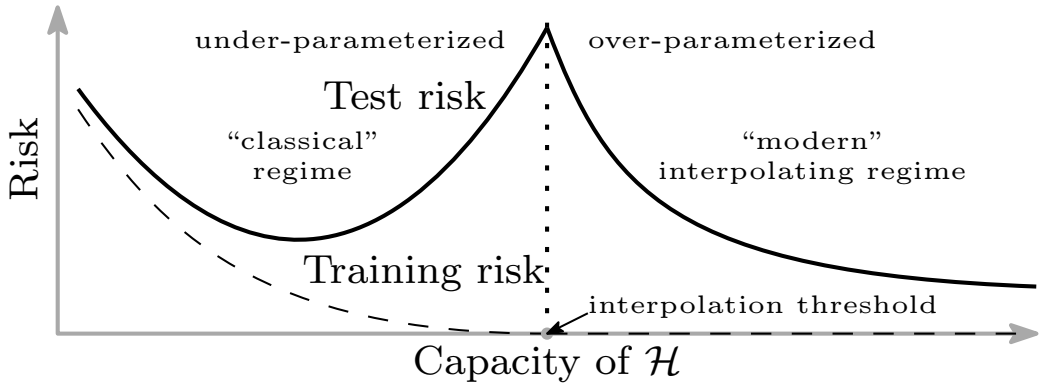
In Figure 3, one can see such a principle in action when comparing linear, degree-4, and degree-15 predictors attempting to fit a problem where the actual underlying function is from the cos family. The linear function is unable to capture the curvatures in the actual underlying problem, so it underfits. The degree-15 predictor can represent the correct function. However, it can also represent infinitely many other functions that pass exactly through the training points because there are more parameters than training examples. We have little chance of choosing a solution that generalizes well when so many wildly different solutions exist. In such an example, the degree-4 model perfectly matched the task's actual structure, so it generalizes well to new data.

In defiance of such an idea that higher complexity (over-parametrized) models have lower bias but higher variance, deep learning practitioners have not experienced such an aspect in modern neural networks (NAKKIRAN *et al.*, 2019). Indeed, this behavior has guided a best practice in deep learning for choosing neural network architectures, specifically that the network should be large enough to permit effortless zero loss training (called interpolation) of the training data[1]. Such behavior is portrayed as "double descent" risk curve, see Figure 4, in which it subsumes the classical U-shaped (Figure 2) by extending it beyond the point of interpolation (BELKIN *et al.*, 2019).

All of the learned predictors to the right of the interpolation threshold fit the training data perfectly and have zero empirical risk. The capacity of the function class does not necessarily reflect how well the predictor matches the inductive bias appropriate for the problem at hand. Choosing the smoothest function that perfectly fits observed data is a form of Occam's razor. However, when considering larger function classes that contain more candidate predictors compatible with the data, we can find interpolating functions with smaller norms and are thus "simpler". Thus, increasing function class capacity improves the performance of classifiers (BELKIN *et al.*, 2019).

[1] Ruslan Salakhutdinov. Deep learning tutorial at the Simons Institute, Berkeley. Available at: https://simons.berkeley.edu/talks/ruslan-salakhutdinov-01-26-2017-1, 2017. It was accessed on September 05th, 2021.

Figure 4 – The double descent risk curve.



Source – Adapted from Belkin *et al.* (2019).

Finally, if there is one thing both classical statisticians and deep learning practitioners agree on is "more data is always better" (NAKKIRAN *et al.*, 2019).

2.2.2 Controlling complexity

We can control whether a model is more likely to overfit or underfit by altering its capacity. Informally, a model's capacity is its ability to fit a wide variety of functions. Models with low capacity may struggle to fit the training set. Models with high capacity can overfit by memorizing properties of the training set that do not serve them well on the test set (GOODFELLOW *et al.*, 2016).

From this perspective, it raises the *Occam's razor* principle (MACKAY, 1992) that states that unnecessarily complex models should not be preferred to simpler ones. Such a principle reinforces that employing mechanisms to penalize over-complex models is beneficial for model generalization.

One can employ Occam's razor principle in many different aspects during the model learning process. It goes from the feature selection/feature engineering up to model selection, through parameter optimization, among others, all focused on refining the resulting models. Such a principle advocates the same in each of these stages: "the simple is the better". A simple model that fits a data set well is likely to capture that data's essential features without absorbing too much noise.

It is also worthy to note that despite the wide popularity and acceptance of such a principle that states that simple explanations are generally more likely to be trustworthy than complex ones, there is little empirical evidence that demonstrates that the world is simple (SOBER,

2015). **Such opposite viewpoints often meet each other when the genuine risk of reducing complexity is at the expense of accuracy**. On such occasions, it is recommended to only apply Occam's razor when the available models' predictive performances are equally good.

Before we address how to control model complexity, considering both underfit and overfit, we need to discuss the relationship between the parameters and complexity. The models are categorized into two categories: the parametric and nonparametric ones.

2.2.2.1 Model complexity in parametric models

Linear regression and neural networks use the training data to estimate a fixed set of parameters $\mathbf{w}$. That defines our hypothesis $h_{\mathbf{w}}(\cdot)$, and at that point we can throw away the training data, because they are all summarized by $\mathbf{w}$. A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a **parametric model**. No matter how much data one throws at a parametric model, it will not change its mind about how many parameters it needs (RUSSELL *et al.*, 2020).

We can control the model performance by choosing what kind of functions we allow them to draw solutions from and controlling the representation of such functions. We can also give a learning algorithm a preference for one solution over another in its hypothesis space. Thus, when some functions are eligible, one is preferred over all others due to some criterion related to complexity (GOODFELLOW *et al.*, 2016).

Next, we present some strategies for accounting and dealing with model complexity that can be employed when preferring one solution face others:

a) **Decay strategies**. They encourage the parameter values to decay towards zero, unless supported by the data. In statistics, it provides an example of a parameter shrinkage method because it shrinks parameter values towards zero. It has the advantage that the error function remains a quadratic function of the parameters, and so its exact minimizer can be found in closed form – L^1 and L^2 regularization;

b) **Elimination strategies**. Differently from the Decay ones, they allow some parameters to have some relatively higher values while others parameters with values towards zero. Therefore, weights with small values can be interpreted as being of little relevance to the model's output, and thus can be eliminated. It is also commonly referred to as the ℓ_0-norm, although it is not an actual norm. Thus, contrasting with sparsity, which counts the number of strictly equal elements to

zero – FOcal Underdetermined System Solver (GORODNITSKY; RAO, 1997) and Ortogonal Matching Pursuit (TROPP; GILBERT, 2007);

c) **Bayesian strategies**. As the name implies, they result from the application of Bayesian inference techniques to verify the parameter relevance through the learning phase – Sparse Bayesian Learning (TIPPING, 2001);

d) **(Meta)Heuristics-based strategies** . Instead of performing a full search through the hipotesis space $\mathcal{H}$, one uses a (meta)heuristic to perform a model selection – Genetic algorithms (SIVANANDAM; DEEPA, 2007), Simulatead annealing, etc;

e) **Prunning**. They reduce the number of learned parameters, aiming to obtain models with the best possible generalization capacity. In general, such techniques use an overly complex model and to eliminate the less relevant parameters, thus, generating models with more generalization capabilities – Optimal Brain Damage (LECUN *et al.*, 1990) and Optimal Brain Surgeon (HASSIBI *et al.*, 1993).

2.2.2.2 Model complexity in nonparametric models

Since parametric models will not change its mind about how many parameters they need. Sometimes, when there are thousands or millions or billions of examples to learn from, it sounds better to let the data speak for themselves rather than forcing them to speak through a tiny set of parameters.

A **nonparametric model** is one that cannot be characterized by a bounded set of parameters. For example, suppose that each hypothesis we generate simply retains within itself all of the training examples and uses all of them to predict the next example. Such a hypothesis family would be nonparametric because the effective number of parameters is unbounded - it grows with the number of examples (RUSSELL *et al.*, 2020).

Since there are multiple degrees of freedom for constructing such models, including some freedom in penalizing them for finding consistent hypotheses is beneficial, they are still subject to underfitting and overfitting, just like parametric methods (RUSSELL *et al.*, 2020).

Regularization[2] is an elegant and efficient tool to cope with the complexity of the model; that is, to make it less complex, more smooth. There are different ways to achieve

[2] Regularization was first suggested by the great Russian mathematician Andrey Nikolayevich Tychonoff (sometimes spelled Tikhonov) for the solution of integral equations (THEODORIDIS, 2020).

this. One way is by constraining the norm of the unknown vector, as ridge regression does. When dealing with more complex, compared to linear, models, one can use constraints on the smoothness of the involved nonlinear function; for example, by involving derivatives of the model function in the regularization term. Also, regularization can help when the adopted model and the number of training points are such that no solution is possible (THEODORIDIS, 2020). Also, it allows complex models to be trained on data sets of limited size without severe over-fitting, essentially by limiting the effective model complexity (BISHOP, 2016).

It works by driving the searching for a hypothesis that directly minimizes loss and complexity into one metric, allowing us to find the best hypothesis all at once:

$$h^\star = \underset{h \in \mathcal{H}}{\arg\min} \ \texttt{loss}(h) + \lambda \ \texttt{complexity}(h), \tag{2.13}$$

where λ is a parameter, a positive number that serves as a conversion rate between loss and hypothesis complexity. Such a formulation requires us to make two choices: the loss function and the complexity measure, which is called a regularization function. The choice of regularization function depends on the hypothesis space. Usually, one employs one of the abovementioned strategies for accounting such a complexity measure (e.g. vector norms).

From a different viewpoint, reducing the hypothesis complexity (via norm) can be considered as an attempt to "simplify" the structure of the estimator, because a smaller number of components of the regressor now have an important say (THEODORIDIS, 2020). This viewpoint becomes more clear if one considers nonlinear models where the existence of the norm of the respective parameter vector in ridge regression forces the model to get rid of the less important terms in the nonlinear expansion (THEODORIDIS, 2020).

Since nonparametric models have their complexity associated with the number of samples (both in training and out-of-sample prediction), reducing the amount and their influence by performing instance selection or set size reduction can decrease the model complexity. A good example (without a penalty function) is tuning the number of neighbors involved in the k-NN classifier, which clearly constrains overfitting (VEENMAN; REINDERS, 2005).

There are a lot of techniques that are able to obtain a representative training set with a lower size compared to the original one and with a similar or even higher classification accuracy for new incoming data. Depending on the strategy followed by these techniques, they can remove noisy, redundant, and both kinds of samples. The main advantage indicated in these methods is the capacity to choose relevant samples without generating new artificial data (GARCIA *et al.*, 2012).

In the literature, they are referred as Reduction Techniques (WILSON; MARTINEZ, 2000), Instance Selection (JANKOWSKI; GROCHOWSKI, 2004), or Prototype Selection (PEKALSKA *et al.*, 2006). Such techniques are widely categorized into three types: condensation, edition, and hybrid techniques (JANKOWSKI; GROCHOWSKI, 2004; ANGIULLI, 2007). Condensation techniques aim at retaining the points which are closer to the class boundaries since the internal points do not affect the decision boundaries as much as border points. Edition techniques instead of keeping such border points, they aim at removing them. They remove points that are noisy or do not agree with their neighborhood, achieving smoother decision boundaries. Finally, the hybrid techniques mix both of the previous techniques as an attempt to maintain or even increase the generalization accuracy in test data (GARCIA *et al.*, 2012).

2.2.2.3 Other model complexity control

Usually, one can control complexity during optimization (training) by imposing a constraint, often under a regularization penalty, on the norm of the weights, as all the notions of complexity listed above depend on it. However, apart from the relationship between the parameters and complexity, the root cause for poor generalization capabilities can arise from many sources (e.g., too many feature dimensions, model assumptions, parameters, too much noise, and very little training data).

In the following, we present some strategies for increasing generalization without including any explicit regularization, neither as an explicit penalty, thus, controlling the model complexity implicitly:

a) **Data augmentation**. A larger training set has a smaller overfitting probability, thus expanding it is a time-saving method, but it varies in different fields. The key point here is reducing data bias to improve model generalization. Common strategies involve rotating and scaling images in the object recognition field, adding random noise to the input data in speech recognition, or replacing words with their synonyms in natural language processing applications;

b) **Early stopping**. It is very convenient to sample our model every few iterations and check how it works with our validation set. Every model that performs better than all the previous models is saved. From time to time (or other criteria), we evaluate the no progress, and when the performance decreases, one stops the learning;

c) **Implicit regularization via gradient descent**. A practical regularization effect of gradient descent can be achieved through the normalized weights, at least for exponential-type loss functions (POGGIO *et al.*, 2020);

d) **Dropout**. Simply put, Dropout randomly discards some inputs during the training process. In this case, the parameters corresponding to the discarded inputs are not updated. As an integration method, Dropout combines all sub-network results and obtains sub-networks by randomly dropping inputs (SRIVASTAVA *et al.*, 2014);

e) **Transfer learning**. Based on overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve related ones, transfer learning tries to discover a common structure of the hypothesis spaces shared by the multiple tasks to transfer knowledge from supervised learning to unsupervised learning (ANDO; ZHANG, 2005).

2.2.3 Controlling complexity in LSSVMs

With respect to the LSSVM model, one can simply highlight that the model complexity term is with respect to the $\mathbf{w}$ in the *primal form* and with respect to the Lagrangian multipliers $\boldsymbol{\alpha}$ in the *dual form* since $\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i)$. Furthermore, we highlight that by such a manipulation, enforcing $\boldsymbol{\alpha} \approx \mathbf{0}$, not only reduces the complexity during the learning phase, but also in the prediction since $\boldsymbol{\alpha}$ are employed there too (see Equation 2.8).

As highlighted in Mall and Suykens (2015), choosing an appropriate structure of a learning machine is one of the perplexing problems in the field of machine learning. As the LSSVM model lacks from sparseness, several works address such a drawback as an attempt to reduce the model complexity in several aspects (e.g. memory usage to build the linear system, solving the system –perform training– and the out-of-sample prediction). According to Mall and Suykens (2015) they can be separated into two groups: (*i*) **reduction methods** and (*ii*) **direct methods**. The reduction methods focus on training an usual LSSVM and, then, apply some pruning strategy, identifying the new support vectors so that a newly trained model can be derived. The direct method paradigm enforces on the sparsity from the beginning.

2.2.3.1 Reduction methods for LSSVMs

For reduction methods the reader is referred to following variations: the Pruning LSSVM (P-LSSVM) (SUYKENS *et al.*, 2000), the Least-Squares Twin Support Vector Machine (LSTSVM) (KUMAR; GOPAL, 2009), the Two-step sparse LSSVM classifier (IP-LSSVM) (CARVALHO; BRAGA, 2009), the Coupled Compressive Pruning (CCP-LSSVM, (YANG *et al.*, 2014)) and the Single and Multi-Objective Genetic Algorithm for Sparse LSSVM (GAS-LSSVM and MOGAS-LSSVM), (SILVA *et al.*, 2015), among others (GEEBELEN *et al.*, 2012; NETO; BARRETO, 2013; VIEIRA *et al.*, 2016; DING *et al.*, 2017a; EBUCHI; KITAMURA, 2017; DING *et al.*, 2017b). In P-LSSVM, after training the usual LSSVM, some support vectors are eliminated according to the absolute value of their Lagrange multipliers $|\alpha_i|$ iteratively complying with some quality criterion (e.g. validation test error, number of support vectors, etc). Such elimination sets $\alpha_i = 0$ and the new resulting linear system is solved at each iteration.

In a similar pruning fashion, but based on the Quadratic Programming version of SVM (QP-SVM) (CORTES; VAPNIK, 1995), the IP-LSSVM is a two-step approach that only solves two linear systems: the usual LSSVM and a new one with less variables according to correspondence between the Lagrangian values in QP-SVM and LSSVM, keeping those support vectors with $\alpha_i \gg 0$. Also based on the QP-SVM, the Twin Support Vector Machine (TSVM) (JAYADEVA *et al.*, 2007) computes a plane for each class so that each plane is as close as possible to the samples belonging to its own class and meantime as far as possible from the samples belonging to the other class (DING *et al.*, 2017a). In possession of these two nonparallel (possible) planes, new samples are assigned to one of the classes according to its proximity to the two nonparallel hyperplanes. The original QP-SVM problem turns into two smaller ones in such a formulation, thus, resulting in a reducing method.

Later, the least-squares version of TSVM (named LSTSVM) is formulated based on the solution of two dual quadratic programing problems of smaller size rather than solving a single dual one. Based on a filtering approach, the Fast Sparse LSSVM (FS-LSSVM) (EBUCHI; KITAMURA, 2017) and Improved Sparse LSSVM classifiers adopt selection strategies on the empirical feature space spanned by the mapped samples. By selecting the linearly independent samples through phase angles (a dissimilarity measure) or block addition (that employs Cholesky factorization into small subsets), they achieve sparse solutions. Adopting an evolutionary approach to impose sparseness, the GAS-LSSVM and MOGAS-LSSVM variations employed a new *a priori* multi-objective fitness function which incorporates a cost of pruning in both

formulations. Thus, they are both able to balance between sparsity (in a fixed-size paradigm) and accuracy, being the Genetic Algorithm set the one that define the model complexity.

2.2.3.2 Direct methods for LSSVMs

As for the direct methods the reader is referred to following variations: the Fast Sparse Approximation scheme for LSSVM (FSA-LSSVM) (JIAO *et al.*, 2007), the Sparsified Subsampled Dual LSSVM (SSD-LSSVM) (MALL; SUYKENS, 2015), and the Pivoted Choleskian of Primal LSSVM (PCP-LSSVM) (ZHOU, 2016), among others (LI *et al.*, 2006; KARSMAKERS *et al.*, 2011; SANTOS; BARRETO, 2017; SANTOS; BARRETO, 2018). In FSA-LSSVM, the decision function is built by iteratively adding one basis function from a kernel-based dictionary until some termination criterion (ε-based) is achieved. Inspired on the subsampled dual LSSVM (SD-LSSVM) (MALL; SUYKENS, 2015), the SSD-LSSVM employs a fast initialization by embedding the active subset selection method described in Brabanter *et al.* (2010) to the LSSVM model solved in the dual formulation. As result, a sparse and low computational cost solution is achieved. Recently, the PCP-LSSVM was proposed based on the assumption that some low-rank kernel matrices-based solutions can be equivalent to the primal ones. The authors explored the Cholesky factorization and its theoretical and experimental results to attest the PCP-LSSVM is able to achieve sparse solutions.

2.2.3.3 LSSVM formulations summary

Next, we summarize in Table 1 some LSSVM formulations over the Literature alongside their complexity control strategy.

Table 1 – LSSVM formulations summary and their complexity control strategy.

FORMULATION	**COMPLEXITY CONTROL STRATEGY**
P-LSSVM (SUYKENS *et al.*, 2000)	Pruning SVs with Lagragians multipliers close to 0.
FSA-LSSVM (JIAO *et al.*, 2007)	Basis function addition at a time with a ε-based stop criterion.
IP-LSSVM (CARVALHO; BRAGA, 2009)	Keeping SVs with higher absolute Lagrange multipliers.
CCP-LSSVM (YANG *et al.*, 2014)	Subset selection via Compressive sampling.
(MO)GAS-LSSVM (SILVA *et al.*, 2015)	Subset selection via Genetic Algorithms.
SSD-LSSVM (MALL; SUYKENS, 2015)	Active subset selection.
PCP-LSSVM (ZHOU, 2016)	Low-rank kernel matrices via Cholesky factorization.
FS-LSSVM (EBUCHI; KITAMURA, 2017)	Subset selection via Cholesky factorization.

Source – Own authorship

2.2.4 Controlling complexity in MLMs

Concerning the MLM model, as a linear model, the complexity term is related to **B**. Even though some variants employ the reference point selection (data discard) as the principal aspect of the MLM learning algorithm, they pretty much embody a lower rank linear system through such a selection as a manner to also impose some sparsity in both training and speed-up prediction. Increasing sparsity is a manner to reduce complexity.

However, employing both reduced data set and regularization in MLM is likely to lead to underfitting since if **B** acts as a poor transformation, i.e., **B** can not correctly map both input and output spaces, the prediction is strongly impaired. Also, note that by employing both strategies as they are presented, they simply restrict the hypothesis that the learning phase can find.

With that said, we focus on this hypothesis that the model complexity is mainly based on the mapping produced by **B**. Thus, we adopt as the complexity measurement in MLM the Frobenius norm of **B**, denoted by $\| \mathbf{B} \|_{\mathcal{F}}$. In the following, we present some literature review with respect to other MLM variants that employ regularization.

2.2.4.1 Random MLM

The Random MLM (SOUZA JÚNIOR *et al.*, 2015) model is a formulation from the original MLM model for both regression and classification tasks. Analogously to the RBF network, such a formulation uses a randomly chosen subset of the learning points as centers, here called reference points (RPs). In this formulation, the number of reference points, or centers, K corresponds to a hyperparameter to be optimized based on a specific data set. Since the RPs are to be a subset of the original data set, the training and predict procedures have to rely on the approximate solution provided by the ordinary least-squares estimate of **B** because both **D** and **Δ** have dimensions $N \times K$, resulting in:

$$\min_{\mathbf{B}} \ \mathcal{J}_{\text{Rand}}(\mathbf{B}) = \| \mathbf{DB} - \mathbf{\Delta} \|_{\mathcal{F}}^2, \tag{2.14}$$

which yields the following solution:

$$\hat{\mathbf{B}}_{\text{Rand}} = (\mathbf{D}^{\intercal}\mathbf{D})^{-1}\mathbf{D}^{\intercal}\mathbf{\Delta}. \tag{2.15}$$

2.2.4.2 *Rank-MLM*

The Rank-MLM (ALENCAR *et al.*, 2015) model is also a formulation of the original MLM model for Ranking tasks with a regularization parameter ($\lambda \in \mathbb{R}^+$) included in the cost function. Thus, its main prominent feature is a training procedure which contains a regularized cost function described as follows:

$$\min_{\mathbf{B}} \mathcal{J}_{\text{Rank}}(\mathbf{B}, \lambda) = \| \mathbf{DB} - \mathbf{\Delta} \|_{\mathcal{F}}^2 + \lambda \| \mathbf{B} \|_{\mathcal{F}}^2 \tag{2.16}$$

which yields the following solution:

$$\hat{\mathbf{B}}_{\text{Rank}} = (\mathbf{D}^\intercal \mathbf{D} + \lambda \mathbf{I})^{-1} \mathbf{D}^\intercal \mathbf{\Delta} \tag{2.17}$$

where $\mathbf{I}$ is a $N \times N$ identity matrix.

2.2.4.3 *Weighted Minimal Learning Machine (wMLM)*

The Weighted Minimal Learning Machine (*w*MLM) (GOMES *et al.*, 2015) is a generalized least-squares fit between distances in the input and output spaces bringing in a differential weighting of residuals in the regression step of the MLM a way to account for errors that are not independently and identically distributed with zero mean and constant variance. In such a model, the loss function is defined as follows:

$$\min_{\mathbf{B}} \mathcal{J}_{\text{Weighted}}(\mathbf{B}, \mathbf{W}) = \| \mathbf{W}(\mathbf{DB} - \mathbf{\Delta}) \|_{\mathcal{F}}^2 \tag{2.18}$$

where $\mathbf{W}$ is a symmetric positive definite diagonal matrix with each element $W_{i,i}$ of its diagonal representing the weight of each training sample $\mathbf{x}_i$. The above formulation yields the following solution:

$$\hat{\mathbf{B}}_{\text{Weighted}} = (\mathbf{D}^\intercal \mathbf{W} \mathbf{D})^{-1} \mathbf{D}^\intercal \mathbf{W}^\intercal \mathbf{\Delta}. \tag{2.19}$$

2.2.4.4 $\ell_{1/2}$*-MLM*

Deriving out of pruning techniques, the $\ell_{1/2}$-MLM (DIAS *et al.*, 2018) is based on regularization methods, which have been successfully used as feasible approaches to prune neural networks. The model seeks the solution for the following optimization problem:

$$\min_{\mathbf{B}} \mathcal{J}_{\ell_{1/2}}(\mathbf{B}, \lambda, \alpha, K) = \| \mathbf{DB} - \mathbf{\Delta} \|_{\mathcal{F}}^2 + \lambda \| \mathbf{B} \|_{1/2}^{1/2} \tag{2.20}$$

where $\lambda \in \mathbb{R}^+$ is a trade-off parameter balancing estimation quality with diversity, α and K control the prunning process, being α the pruning factor and K the number of reference points.

Since there is no closed-form expression to solve Equation 2.20, to perform the minimization, one must apply the gradient descent algorithm for the following cost function:

$$J(\mathbf{B}) = \mathrm{Tr}\{(\mathbf{DB}-\mathbf{\Delta})(\mathbf{DB}-\mathbf{\Delta})^\intercal\} + \lambda \sum_{i=1}^{K}\sum_{j=1}^{K} |B_{i,j}|^{1/2} \tag{2.21}$$

which produces the following update rule:

$$\mathbf{B}^{(t+1)} = \mathbf{B}^{(t)} - \eta \frac{\nabla J(\mathbf{B}^{(t)})}{\| \nabla J(\mathbf{B}^{(t)}) \|_{\mathcal{F}}} \tag{2.22}$$

where the resulting Jacobian of the loss function $\nabla J(\mathbf{B}^{(t)}) = 2\mathbf{D}^\intercal(\mathbf{\Delta}-\mathbf{DB}) + \lambda \nabla \| \mathbf{B} \|_{1/2}^{1/2}$, $\eta \in (0,1]$ and t is the training epoch. Also, the Jacobian associated with the $\ell_{1/2}$-regularizer is:

$$\nabla \| \mathbf{B} \|_{1/2}^{1/2} = \frac{1}{2} \begin{bmatrix} \frac{\mathrm{sign}(B_{1,1})}{\sqrt{|B_{1,1}|}} & \cdots & \frac{\mathrm{sign}(B_{1,K})}{\sqrt{|B_{1,K}|}} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{sign}(B_{K,1})}{\sqrt{|B_{K,1}|}} & \cdots & \frac{\mathrm{sign}(B_{K,K})}{\sqrt{|B_{K,K}|}} \end{bmatrix}. \tag{2.23}$$

At the end of every training epoch, the authors evaluate the norm of each row of $\mathbf{B}$ to identify which reference points should be removed. The authors were inspired by (FAN *et al.*, 2014) to adopt an adaptative threshold of γ to remove such rows as follows:

$$\gamma^{(t)} = \frac{\alpha}{K} \sum_{k=1}^{K} \| \mathbf{B}_{k,*}^{(t)} \|_2, \tag{2.24}$$

where $\mathbf{B}_{k,*}^{(t)}$ stands for the k-th row of $\mathbf{B}$ and α is the pruning factor.

2.2.4.5 RFM-MLM

Again from adopting the same pruning fashion, it arises the Regularized M-FOCUSS MLM (RMF-MLM) (DIAS *et al.*, 2019). The RMF-MLM relies on a simultaneous sparse approximation method named regularized M-FOCUSS (COTTER *et al.*, 2005) for the task of identifying which reference points are not relevant to the MLM's performance by the following optimization problem:

$$\min_{\mathbf{B}} \ \mathcal{J}_{\mathrm{RFM}}(\mathbf{B}, \lambda, p) = \| \mathbf{DB}-\mathbf{\Delta} \|_{\mathcal{F}}^2 + \lambda \sum_{m=1}^{M} \| \mathbf{B}_{m,*} \|_2^p \tag{2.25}$$

where $\lambda \in \mathbb{R}^+$ is a trade-off parameter balancing estimation quality with diversity measure minimization, $p \in [0,2]$ is for defining an ℓ_p-norm-like diversity measure, and $\mathbf{B}_{m,*}$ denotes the m-th row of $\mathbf{B}$. Since RMF-MLM employs the factored-gradient approach of (RAO; KREUTZ-DELGADO, 1999) to minimize Equation 2.25, the algorithm iteratively updates $\mathbf{B}$ using the following steps:

- $\mathbf{W}^{(t+1)} = \text{diag}\left\{ \left(c_m^{(t)}\right)^{1-\frac{p}{2}} \right\}$, with $c_m^{(t)} = \left(\sum_{k=1}^{K} \left(b_{k,m}^{(t)}\right)^2 \right)^{\frac{1}{2}}$.
- $\mathbf{Q}^{(t+1)} = \left(\mathbf{D}^{(t+1)}\right)^{\mathsf{T}}$.
- $\mathbf{B}^{(t+1)} = \mathbf{W}^{(t+1)}\mathbf{Q}^{(t+1)}$.

2.2.4.6 MLM formulations summary

Next, similarly to LSSVMs, we summarize in Table 2 some MLM formulations over the Literature alongside their hyperparameters and cost function.

Table 2 – Some MLM variants' summarization. Here, we describe their complexity control strategy.

MLM VARIANT	**COMPLEXITY CONTROL STRATEGY**
FULL MLM	None.
Random-MLM	Subset selection.
Rank-MLM (ALENCAR *et al.*, 2015)	Weight decay.
*w*MLM (GOMES *et al.*, 2015)	Weight decay.
R-MLM (GOMES *et al.*, 2017)	Weight decay.
NN-MLM (MESQUITA *et al.*, 2017)	Subset selection.
C-MLM (MESQUITA *et al.*, 2017)	Subset selection.
Co-MLM (CALDAS *et al.*, 2018)	Subset selection.
$\ell_{1/2}$-MLM (DIAS *et al.*, 2018)	Elimination and subset selection.
EMLM (KARKKAINEN, 2019)	Subset selection.
RMF-MLM (DIAS *et al.*, 2019)	Weight decay and subset selection.
LW-MLM (FLORENCIO V *et al.*, 2020)	Weight decay without subset selection during training.
Clustering + MLM (HAMALAINEN *et al.*, 2020)	Subset selection.

Source – Adapted from Florencio V *et al.* (2020).

3 REDUCING COMPLEXITY BY INSTANCE SELECTION

Our first attempt to deal with the complexity of learning models was to explore the instance selection perspective. To do so, we developed a novel Instance Selection algorithm based on FAST, an image corner detector, and applied it to LSSVMs in a pruning fashion. In the following, we first present the Instance Selection algorithm, named Class Corner Instance Selection, and then embed it into the learning phase in LSSVMs.

3.1 CCIS: Class Corner Instance Selection

Our Instance Selection algorithm is mainly based on FAST (ROSTEN; DRUMMOND, 2006), an image corner detector. The main idea is to establish a corner in FAST and then apply the same reasoning as the Instance Selection algorithm. From the risen selection result, we build the learning algorithm model. However, it turns out that FAST formulation only deals with image data, i.e., two-dimensional samples uniformly spaced in a grid. To overcome such a limitation, we extended FAST so that we can apply it to high-dimensional inputs in a straightforward way.

3.1.1 Features from Accelerated Segment Test

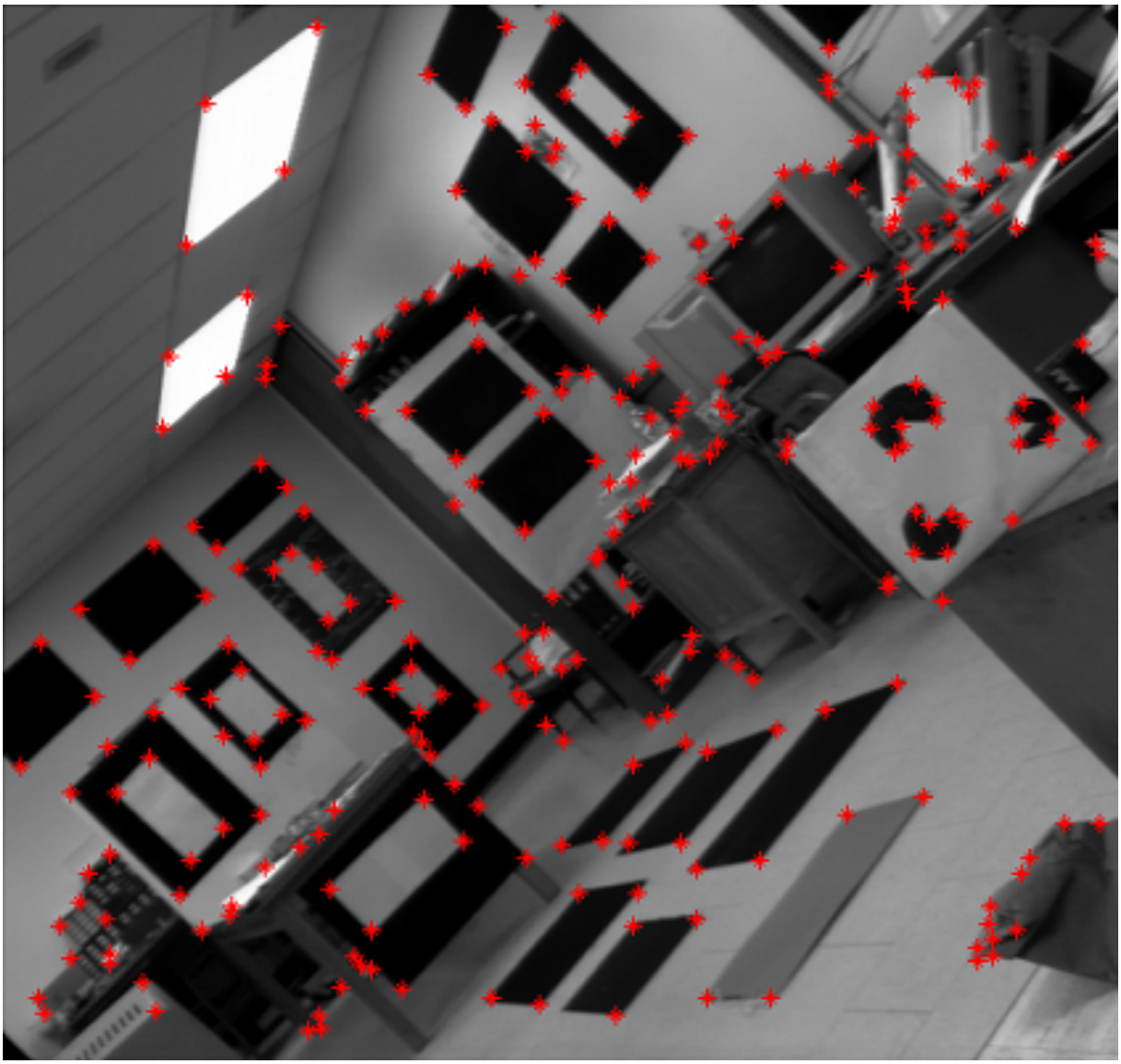
For the sake of simplicity, consider the following description and notation to describe some operations regarding pixels. In the context of a gray-level image setting, we define an image $\boldsymbol{I} \in \mathbb{R}^{W \times H}$ with width W and height H. Then, we use $\boldsymbol{I}_{x,y}$ to rescue the intensity of a pixel as the result of indexing a pair of coordinates (x,y). Finally, we especify the following notation $(x,y) \in \boldsymbol{I}$ to define a valid pair of coordinates into the image, i.e., (x,y) such that $1 \leq x \leq W \wedge 1 \leq y \leq W$.

FAST is a corner detector (Figure 5) in which a Bresenham's circle of diameter 3.4 pixels is used as test mask for each pixel. The resulting mask set

$$\mathcal{B}(x,y) = \left\{ (i,j) \in \boldsymbol{I} \;\middle|\; \sqrt{(x-i)^2 + (y-j)^2} \approx \frac{3.4}{2\pi} \right\}, \tag{3.1}$$

with 16 pixels (i.e., $|\mathcal{B}(I_{x,y})| = 16$) on the discretized circle describing the segment (see Figure 6) is compared to the value of the nucleus pixel $I_{x,y}$. The criterion to classify a nucleus pixel $I_{x,y}$ as a corner candidate in FAST is that there must be at least P contiguous pixels (usually $P = 12$) on $\mathcal{B}(I_{x,y})$ which are brighter or darker than a given threshold T. The parameter T controls the sensitivity of the corner response. Intuitively, this approach discards pixels that are located at

Figure 5 – Corner detection results in a gray scale image. The detected corners are highlighted in red asterisks (∗).



Source – Kahaki *et al.* (2014).

homogeneous regions and pixels at heterogeneous regions with low variance. We denote the set of elements of $\mathcal{B}(I_{x,y})$ which are T brighter or darker than $I_{x,y}$ by $\mathcal{G}_T(I_{x,y})$, that is

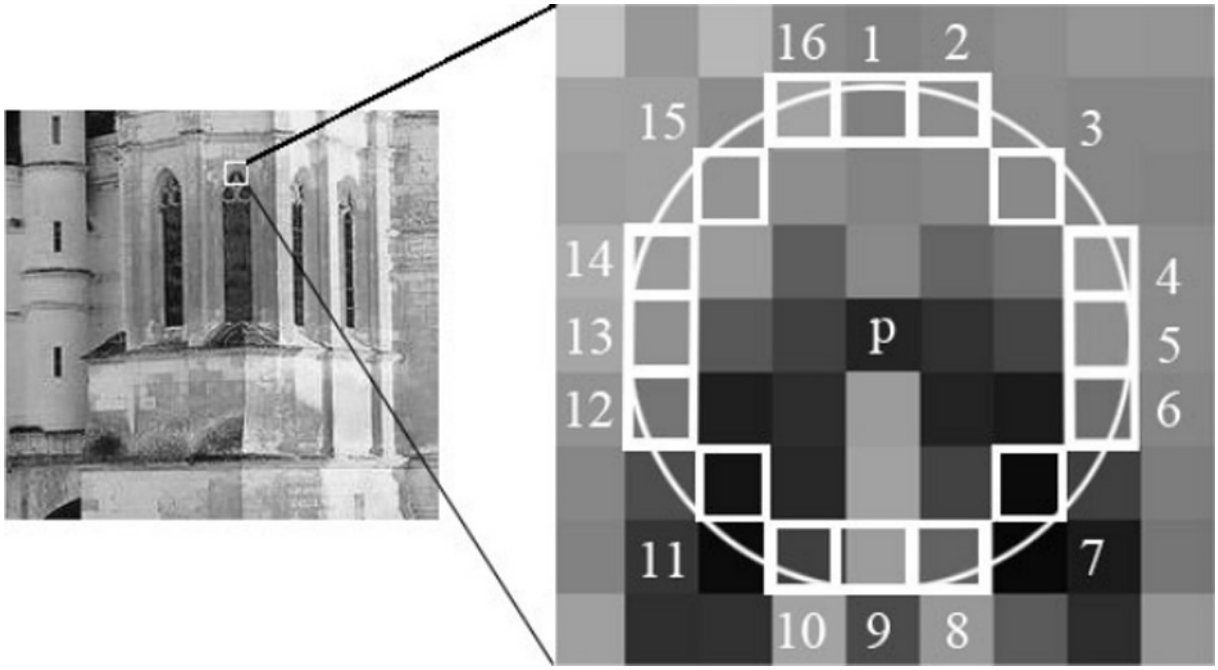
$$\mathcal{G}(x,y) = \left\{(i,j) \in \mathcal{B}(x,y) \;\middle|\; |(\boldsymbol{I}_{x,y} - \boldsymbol{I}_{i,j})| \geq T)\right\}. \tag{3.2}$$

By selecting those pixels that satisfy both criteria defined in Equation 3.2, one obtains a set of corner candidates denoted by

$$\mathcal{C}^{\star} = \left\{(x,y) \in \mathbf{I} \;\middle|\; (|\mathcal{G}(x,y)| \geq P) \wedge \text{contiguous}\left\{\mathcal{G}(x,y)\right\}\right\}, \tag{3.3}$$

where contiguous$\{\,\cdot\,\}$ is a predicate that is true if its argument contains only contiguous pixels and false otherwise. To check if a given set of pixels is contiguous or not, one should verify if

Figure 6 – Corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel $\mathrm{p} = (x, y)$ located at the center of the patch is the one being tested. The Bresenham's circle is indicated by the continuous line).



Source – Figure adapted from Rosten and Drummond (2006).

the pixels' positions with respect to the Bresenham's circle form an contiguous arc. Such test can be speeded up by first checking if the pixels located at positions $1, 5, 9$, and 13 are T brighter or darker than (x, y) so that the whole circle is evaluated.

In possession of $\mathcal{C}$, one obtains the set of corner candidate points. However, a lot of adjacent pixels are yielded and a filtering step must be employed to remove weak candidates, i.e., adjacent corner-candidates points with less relevance. For this purpose, a non-maximal suppression is employed to select the stronger candidates. The common procedure at this step is to check whenever a candidate to be a real corner $(x, y) \in \mathcal{C}$ has any other adjacent candidate more relevant than it. The relevance is computed by a relevance function $\delta(\cdot, \cdot)$ defined by the user. If such previous condition is true, i.e., $\exists (i, j) \in \mathcal{A}(x, y)$ so that $\delta(x, y) < \delta(i, j)$, then, this candidate is not part of the final corner set $\mathcal{C}^\star$, thus, we discard it. Using the sum of the absolute difference between the pixels in the segment circle and the nucleus pixel is reported to be a good choice for relevance function, that is

$$\delta(x, y) = \sum_{(i,j)} \left| I_{x,y} - I_{i,j} \right|, \forall (i, j) \in \mathcal{B}(x, y). \tag{3.4}$$

Let $\mathcal{A}(x, y)$ be the set of adjacent candidates from a given candidate (x, y), defined

by

$$\mathcal{A}(x,y) = \left\{ (i,j) \in \mathcal{C} \;\middle|\; \sqrt{(x-i)^2 + (y-j)^2} \leq \frac{3.4}{2\pi} \right\}, \tag{3.5}$$

being the set of weak candidates is given by

$$\mathcal{C}_{\text{WEAK}} = \left\{ (x,y) \in \mathcal{C} \;\middle|\; \exists (i,j) \in \mathcal{A}(x,y) : \delta(x,y) < \delta(i,j) \right\}, \tag{3.6}$$

resulting in the final corner set $\mathcal{C}^\star = \mathcal{C} \setminus \mathcal{C}_{\text{WEAK}}$.

The pixels in $\mathcal{C}^\star$ are those highlighted in red asterisks ($*$) in Figure 5. An algorithm summarizing FAST is presented in Algorithm 7.

Algorithm 7 Features from Accelerated Segmented Test

FAST-DETECTOR(Input image $\mathcal{I}$, the contiguous threshold P, corner response threshold T)

▷ *Step 1: Corner-candidate detection.*
1 Compute $\mathcal{C}$ according to Equation 3.3.
▷ *Step 2: Weak candidate detection.*
2 Compute $\mathcal{C}_{\text{WEAK}}$ according to Equation 3.6.
▷ *Step 3: Weak candidate removal.*
3 $\mathcal{C}^\star \leftarrow \mathcal{C} \setminus \mathcal{C}_{\text{WEAK}}$.
4 **return** the corner set $\mathcal{C}^\star$

3.1.2 CCIS formulation

In FAST, all pixels are evaluated as corners or not. However, before such a classification, two subsets are derived: the candidate and the actual corner set. Here in CCIS, we use a similar analogy but for general data points. First, we perform greedy filtering to identify the input samples that somehow lie in each class's class-corner regions. Then we select a *strong* subset from the filtering result to achieve the class-corner ones.

Since we cannot assume we have equally spaced samples in $\mathcal{D}$ as the pixels in an image grid, nor the samples share neighboring trivially, the idea of getting the Bresenham's circle mask does not sound feasible at a glance.

It is also worth mention that finding such a set of neighbors exactly in Bresenham's circle is not always possible for real-world data. To overcome such a drawback, we adopt a different strategy by handling the input samples that lie within a closed ball centered in the query point $\mathbf{x}$ with radius R as a way to emulate the Bresenham's circle mask. Let us define the R-ball

neighborhood of a query sample $\mathbf{x}$ by

$$\mathcal{N}_R(\mathbf{x}) = \Big\{ (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{NN}_K(\mathbf{x}) \;\Big|\; 0 < \|\mathbf{x}_i - \mathbf{x}\|_2 \leq R \Big\}, \tag{3.7}$$

where $R \in \mathbb{R}_+$ is the radius of the circle mask.

In possession of the R-ball neighborhood of a query sample $\mathbf{x}$, we perform a FAST-like test (see Equation 3.3) to select the class-corner samples by eliminating the contiguous constraint and only take into account the R-ball neighborhood variability. Such a neighborhood variability is accounted as follows,

$$\Gamma(\mathbf{x}) = \sum_{\mathbf{x}_i} \mathbb{1}[\mathbf{y} \neq \mathbf{y}_i],\ \mathbf{x}_i \in \mathcal{N}_R(\mathbf{x}), \tag{3.8}$$

where $\mathbb{1}[\cdot]$ is the indicator function that it is equal to 1 if its argument is true or 0 otherwise. Note that, $\Gamma(\cdot)$ is a simple counting function that yields the number of neighbors with different class labels than the query sample $\mathbf{x}$ inside the R-ball.

Now, one can classify a given query sample $\mathbf{x}$ as a class-corner by simply verifying its neighboring variability information. From such a classification, one can derive a subset with only strong class-corner samples as follows,

$$\mathcal{CCIS}(\mathcal{D}, P) = \Big\{ (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D} \;\Big|\; \Gamma(\mathbf{x}_i) > P \Big\}, \tag{3.9}$$

where $0 < P < K$ is a threshold that accounts for the variability in terms of class labels around $\mathbf{x}$. Such filtering approach works as the same as in FAST with respect to the corner sensitivity, see Equation 3.3 and Figure 6. Also, since P controls the test's corner sensitivity, fewer corners are yielded by increasing P.

3.1.3 Hyperparameter estimate in CCIS

CCIS has three hyperparameters to be tuned: the distance threshold R, the corner sensitivity threshold P, and the number of nearest neighbors K. We discuss in the following how to provide default values for them. However, users can tune them according to their taste.

3.1.3.1 Borrowing default values from FAST for K and P

A detailed experiment concerning FAST was carried out in Rosten *et al.* (2010), showing that the versions of FAST using $P = 12$ (75%) or $P = 9$ (55%) out of 16 pixels in the Bresenham's circle were enough. Although such experiments suggested that the performance

between such FAST versions is different on modern hardware, for memory requirements, we opt for the one with fewer points to evaluate, i.e., $P = 9$. Regarding K, we suggest $K = 16$ as a manner to emulate the number of pixels in the Bresenham's test mask. However, one might follow other strategies, such as the well-known thumb rule of K-NN defined as $K = \sqrt{N}$, and the one presented in (MALL; SUYKENS, 2015) which scales $K = \lfloor k \times \sqrt{N} \rfloor$ with k optimized by grid search.

3.1.3.2 Estimating the distance threshold R

Since we can not assume that all input samples have the same distance relationship as the pixels in an image grid, estimating a value for R that CCIS can apply in different domains is difficult. Thus, to overcome this condition, we propose an automatic manner to estimate a suitable value for the distance threshold denoted by $\hat{R}$ based on the input distance relationship in which is guaranteed at least one neighbor within such a threshold.

The automatic threshold R is given by the maximum value for the minimal distances of each input to its nearest neighbor:

$$\hat{R} = \max\{d_1^*, d_2^*, \ldots, d_N^*\}, d_i^* = \|\mathbf{x}_i - \mathbf{x}_j\|_2,\ \mathbf{x}_j \in \mathcal{NN}_1(\mathbf{x}_i). \tag{3.10}$$

The computation of the distance drives the complexity of estimating $\hat{R}$. By using a *kd*-tree to compute the minimal distances of each input and, then, performing a linear search in $\{d_1^*, d_2^*, \ldots, d_N^*\}$, it results in $\mathcal{O}(N\log(N))$. Also, we highlight that, again, any search approach can be employed with respect to find a feasible value for $\hat{R}$ as long as it comply with possible distances on the data set.

3.1.4 Computational complexity of CCIS

The instance selection complexity is mainly driven by computing $\mathcal{PS}$ where for each input sample, we compute the $\Gamma(\cdot)$ function. The cost of $\Gamma(\cdot)$ is related to finding the K nearest neighbors for each sample. By adopting a *kd*-tree indexing the whole data set $\mathcal{D}$, the cost of building such data structure once is $\mathcal{O}(N\log(N))$, while performing a single search is $\mathcal{O}(\log(N))$. As we mentioned, $\Gamma(\cdot)$ performs K searches; thus, its cost for a single call is $\mathcal{O}(K\log(N))$.

Stated that, the overall cost of finding the class-corner samples is given by building a *kd*-tree and performing N calls to $\Gamma(\cdot)$, resulting in $\mathcal{O}(N\log(N)) + \mathcal{O}(NK\log(N))$. Since K is fixed in all searches, we can conclude that $\mathcal{O}(N\log(N)) + \mathcal{O}(NK\log(N)) \sim \mathcal{O}(N\log(N))$.

3.2 CC-LSSVM: LSSVM meets Class-corner Instance Selection

In some literature, it is supported that pruning and reduced-set strategies for LSSVM work quite well (CARVALHO; BRAGA, 2009; BRABANTER *et al.*, 2010; NETO; BARRETO, 2013; YANG *et al.*, 2014). The only limitation is that if an input sample is a SV (column), its constraint (row) must be in the system since losing that link degrades the final solution.

Instead of solving a full rank system as in Equation 2.5, we adopt a less constrained model via an overdetermined one. Firstly, we select via CCIS the restrictions and SVs before producing an overdetermined system.

3.2.1 The CC-LSSVM formulation

In CC-LSSVM, the $\mathcal{SV} = \text{CCIS}(\mathcal{D}, 0)$ is the set of support vectors while $\mathcal{PS} = \text{CCIS}(\mathcal{D}, P)$ represents the constraints in the model with threshold P. Moreover, we kept the link between the variables and constraints since $\mathcal{SV} \subset \mathcal{PS} \subset \mathcal{D}$, thus, resulting in a possibly smaller and less constrained linear system. We present such a system in the following.

We formulate the linear system in CC-LSVM as $\boldsymbol{\Lambda\omega} = \boldsymbol{\upsilon}$ so that

$$\underbrace{\left[\begin{array}{c|c} 0 & \mathbf{1}^\intercal \\ \hline \mathbf{1} & \boldsymbol{\Psi} \end{array}\right]}_{\boldsymbol{\Lambda}} \underbrace{\left[\begin{array}{c} b^\star \\ \hline \boldsymbol{\alpha}^\star \end{array}\right]}_{\boldsymbol{\omega}} = \underbrace{\left[\begin{array}{c} 0 \\ \hline \mathbf{Y}_{\text{SV}} \end{array}\right]}_{\boldsymbol{\upsilon}}, \tag{3.11}$$

where $\mathbf{Y}_{\text{SV}} = [y_1, y_2, \ldots, y_M]^\intercal$ is a matrix with labels from $\mathcal{SV}$ and

$$\Psi_{i,j} = \begin{cases} \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j), & \text{if } \mathbf{x}_i \neq \mathbf{x}_j; \\ \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) + \gamma^{-1}, & \text{otherwise.} \end{cases}$$

with $\mathbf{x}_i \in \mathcal{PS}$ and $\mathbf{x}_j \in \mathcal{SV}$ and $\gamma \in \mathbb{R}_+$ is the same cost parameter in the original (LS)SVM. Also, since the linear system in CC-LSVM is *possibly* less constrained, one can notice that $M = \text{rank}(\boldsymbol{\Lambda}) \leq \text{rank}(\mathbf{A}) = N$, thus resulting in $\boldsymbol{\omega}, \boldsymbol{\upsilon} \in \mathbb{R}^M$.

3.2.2 The CC-LSSVM learning algorithm

The CC-LSSVM learning algorith requires the solution of Equation 3.11 produced by both support vector and constraint selections. Since $\boldsymbol{\Lambda}$ is not always a full rank matrix, the CC-LSSVM linear system solution is obtained by the usual least-squares estimate, that is,

$$\hat{\boldsymbol{\omega}} = (\boldsymbol{\Lambda}^\intercal \boldsymbol{\Lambda})^{-1} \boldsymbol{\Lambda}^\intercal \boldsymbol{\upsilon}. \tag{3.12}$$

The approximated solution $\hat{\boldsymbol{\omega}}$ yields the dual variables $\boldsymbol{\alpha}^\star$ and the bias $b^\star$ so that out-of-sample predictions can be computed. The algorithm for CC-LSSVM is given in Algorithm 8.

Algorithm 8 The CC-LSSVM learning algorithm

CC-LSSVM-TRAINING$(\mathcal{D}, R, P)$
1 Compute R according to Equation 3.10.
  ▷ Step 1: Support vector and constraint selection.
2 $\mathcal{PS} \leftarrow \text{CCIS}(\mathcal{D}, 0)$.
3 $\mathcal{SV} \leftarrow \text{CCIS}(\mathcal{D}, P)$.
  ▷ Step 2: Estimating Lagrangian multipliers and bias.
4 Build the linear system of Equation 3.11 from $\mathcal{PS}$ and $\mathcal{SV}$.
5 Estimate $\hat{\boldsymbol{\omega}}$ using the usual least-squares.
6 Rescue from $\hat{\boldsymbol{\omega}}$ the dual variables $\boldsymbol{\alpha}^\star$ and the bias $b^\star$.
7 **return** $\mathcal{SV}$, $\boldsymbol{\alpha}^\star$, and $b^\star$.

3.2.3 The CC-LSSVM out-of-sample prediction

In possession of the estimated dual variables $\boldsymbol{\alpha}^\star$, bias $b^\star$, and the SVs from $\mathcal{SV}$, one can build the final predictive model as

$$\hat{y} = f(\mathbf{x}) = \text{sign}\left(\sum_{j=1}^{M} \alpha_j^\star \mathcal{K}(\mathbf{x}, \mathbf{x}_j) + b^\star\right), \mathbf{x}_j \in \mathcal{SV}. \tag{3.13}$$

3.2.4 Computational complexity of CC-LSSVM

As CC-LSSVM consists of two steps: (*i*) the class-corner support vector selection and (*ii*) computing the solution of a less constrained linear system, the overall complexity of CC-LSSVM is given by the sum of the complexity of these two steps.

The support vector selection complexity is driven by computing CCIS; thus, is $\mathcal{O}(N\log(N))$. On the following, the linear system solution is given by $\mathcal{O}(|\mathcal{PS}||\mathcal{SV}|^2)$ since the system in Equation 3.11 is built from $\mathcal{PS}$ and $\mathcal{SV}$. However, neither a sparse solution neither a less constraint model is always possible according to the problem's nature. To simplify our notation, we adopt M as the number of selected SVs for further analysis.

Thus, we have, in fact, **two cases**: the best (and average) and the worst one. In the best case, a less constrained model with fewer SVs is achieved, resulting in $\mathcal{O}(M^3)$. In the worst-case, a fully constrained model is built, resulting in $\mathcal{O}(NM^2)$.

Later, we discuss how M behaves in real-world experiments and how the worst-case affects the overall model performance, especially via parameter tuning. From the carried out experiments, we conclude that the overall complexity of CC-LSSVM runs in $\mathcal{O}(M^3)$ since $\mathcal{O}(N\log(N)) + \mathcal{O}(M^3) \sim \mathcal{O}(M^3)$.

3.3 Experiments and Discussion on CCIS and CC-LSSVM

This section presents the experimental framework followed in this work and the results collected and discussions on them.

3.3.1 Experiment Setup for CCIS and CC-LSSVM

We carried out three types of experiments to evaluate different aspects of our proposal. Our goal is to investigate how CC-LSSVM behaves concerning the following aspects: accuracy, sparseness, support vector selection, and hyperparameter sensitivity. Each of the following experiments addresses such concerns. Moreover, we performed all experiments using a Macbook Pro with an Intel Core i5 2.4 GHz, 8 GB of RAM, and running macOS Sierra 10.12.6 with MATLAB Version 8.3.0.

- **The typical black-box assessment on some data sets**: Here, we are interested in assessing CC-LSSVM against some state-of-the-art dual LSSVM proposals through accuracy and sparseness in a typical black-box test fashion on some "toy-size" and large-size data sets.
- **Empirical decision boundary quality assessment**: This experiment investigates visually the quality of solutions produced by the some state-of-the-art dual LSSVM proposals and CC-LSSVM by empirically evaluating the decision boundaries and, consequently, the support vector selection.
- **Hyperparameter sensitivity**: In this experiment, we intend to identify the influence of hyperparameter tuning concerning the resulting CC-LSSVM model.

3.3.2 Typical black-box assessment for CC-LSSVM

We divided this assessment into two parts.

In the first part, we carried out some simulations on 10 "toy-size" data sets in which 9 out of 10 are real-world data sets from the University of California at Irvine (UCI)

Table 3 – Description of the data sets: name, acronym, input dimensionality and number of training and test samples.

data set	**ACRONYM**	**#Dim**	**# Tr**	**#Te**
Banana	BAN	2	4239	1061
Breast Cancer Winscousin	BCW	9	550	138
German	GER	24	800	200
Haberman's Survival	HAB	3	244	62
Heart	HEA	13	216	54
Hepatitis	HEP	19	65	15
Ionosphere	ION	34	281	70
Pima Indians Diabetes	PID	9	614	154
Ripley	RIP	2	999	250
Two Moon	TMN	2	800	201
Vertebral Column	VCP	6	248	62

Source – Oliveira *et al.* (2018).

Repository (LICHMAN, 2013), while the remaining data set is an artificial one (Two Moon) generated by the Scikit Learn Library (PEDREGOSA *et al.*, 2011), see Table 3 for a summarized description of these data sets.

In the second part, we chose some large-size non-linear problems with a fixed number of SVs $\#\mathcal{SV}$ and adopted the same hyperparameters as those described in Zhou (2016), see Table 6. In this experiment, we compared a different version of CC-LSSVM against the FSA-LSSVM and SSD-LSSVM since other pruning algorithms, such as IP-LSSVM, P-LSSVM, GA-LSSVM, CP-LSSVM, and CCP-LLSSVM, are unsuitable for large-scale training because of memory constraints (they require solving a full rank system).

3.3.2.1 CC-LSSVM for Toy-size problems

In this first part, we carried out some simulations on 11 *toy-size* data sets in which 9 out of 10 are real-world data sets from the University of California at Irvine (UCI) Repository (LICHMAN, 2013). In contrast, the remaining data set is an artificial one, named Two Moon, generated by the Scikit Learn Library (PEDREGOSA *et al.*, 2011), see Table 3 for a summarized description of these data sets. Also, we highlight that we pre-processed all data sets so that we removed samples containing missing values and normalized features by variation (standardization).

We compared three LSSVM variations against CC-LSSVM, namely, the Fast Sparse Approximation scheme for LSSVM (FSA-LSSVM) (JIAO *et al.*, 2007), the Coupled Compressive Pruning for sparse LSSVM (CCP-LSSVM) ((YANG *et al.*, 2014)), and the Sparsified

Table 4 – LSSVM variants and their hyperparameter configurations.

MODEL	DESCRIPTION	PARAMETER VALUE
CC-LSSVM	Number of neighbors.	$K = 16$.
	Variability threshold.	$P = 9$.
	Distance threshold.	R obtained by (Equation 3.10).
FSA-LSSVM	ε-insensitive criterion.	$\varepsilon = 0.5$.
SSD-LSSVM	Subset size.	$\lfloor 4 \times \sqrt{N} \rfloor$ (i.e., $k = 4$).
CCP-LSSVM	Sampling ratio.	$M = 0.3N$.
	Sparse approximation.	Orthogonal Matching Pursuit.
	Compressive sampling.	Random gaussian matrix.

Source – Oliveira *et al.* (2018).

Subsampled Dual LSSVM (SSD-LSSVM) (MALL; SUYKENS, 2015)). All variations arise from the dual formulation of LSSVM, and thus, we employ them for comparison since our proposal C-LSSVM is also based on the same dual formulation. In this comparison, we report the average accuracy (ACC) and sparseness (SPR)[1] over 30 independent realizations. Also, the hyperparameter setting for each variation, including ours, is presented in Table 4.

Concerning the hyperparameter setup of γ and σ (RBF kernel parameter), we optimized them via a 10-fold grid search in a usual LSSVM and used the same optimized hyperparameters for all LSSVM variants. We present the average results for 30 independent realizations for these 10 *toy-size* data sets in Table 5.

Furthermore, we carried out the Friedman test, evaluating both ACC and SPR scores to better distinguish the LSSVM variations. The Friedman test quantifies the consistency of model results when applied in several data sets according to their performance rankings[2]. In this test, the null hypothesis states that there is no statistical difference between all evaluated models. We graphically represent such comparison through the Critical difference plot as presented in Demšar (2006). In this plot, the top line is the axis on which each model's average ranks, while the connected group of models indicated that they are not significantly different. Moreover, the critical difference (CD) is presented above the plot and by adopting the significance level of $\alpha = 0.05$.

[1] The number of non-used training samples as support vectors, i.e., $SPR = 1 - \frac{M}{N}$

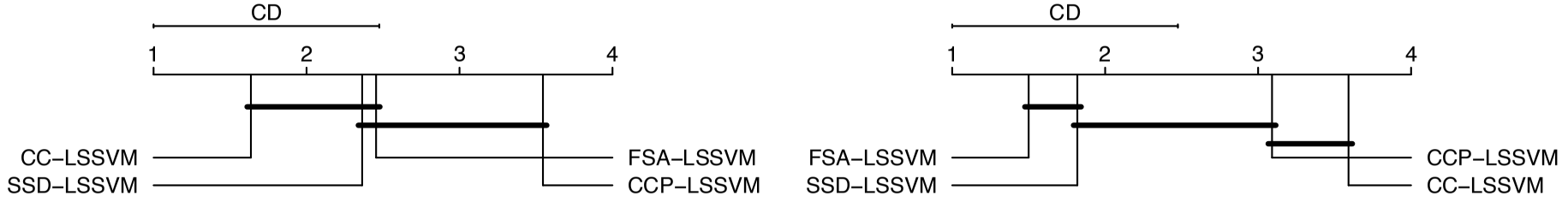
[2] For each data set, the best performing model getting the rank of 1, the second-best rank 2, and so on.

Table 5 – Performance on some *toy-size* data sets. For each data set, the best performing models are in boldface. We recall that all values are the average for 30 independent realizations.

data set	**LSSVM**	**CC-LSSVM**		**FSA-LSSVM**		**SSD-LSSVM**		**CCP-LSSVM**	
	ACC	ACC	SPR	ACC	SPR	ACC	SPR	ACC	SPR
BAN	0.90 ± 0.01	**0.89 ± 0.01**	0.93 ± 0.00	**0.89 ± 0.01**	0.93 ± 0.00	0.90 ± 0.01	**0.96 ± 0.00**	0.68 ± 0.17	0.72 ± 0.00
BCW	0.96 ± 0.02	**0.96 ± 0.01**	0.75 ± 0.01	0.95 ± 0.02	**0.96 ± 0.01**	0.94 ± 0.02	0.84 ± 0.00	0.81 ± 0.06	0.71 ± 0.00
GER	0.75 ± 0.02	**0.75 ± 0.02**	0.00 ± 0.00	0.72 ± 0.02	0.81 ± 0.01	0.70 ± 0.00	**0.86 ± 0.00**	0.60 ± 0.07	0.71 ± 0.00
HAB	0.73 ± 0.03	0.73 ± 0.03	0.31 ± 0.02	**0.75 ± 0.03**	0.79 ± 0.02	0.74 ± 0.03	**0.80 ± 0.01**	0.58 ± 0.16	0.71 ± 0.00
HEA	0.72 ± 0.04	**0.84 ± 0.05**	0.00 ± 0.00	0.56 ± 0.00	**0.87 ± 0.02**	0.57 ± 0.02	0.73 ± 0.00	0.71 ± 0.08	0.71 ± 0.00
HEP	0.60 ± 0.00	**0.70 ± 0.10**	0.00 ± 0.00	0.60 ± 0.00	0.65 ± 0.04	0.60 ± 0.00	0.55 ± 0.02	0.53 ± 0.12	**0.72 ± 0.00**
ION	0.80 ± 0.05	**0.93 ± 0.04**	0.40 ± 0.01	0.68 ± 0.03	**0.85 ± 0.01**	0.69 ± 0.10	0.77 ± 0.00	0.73 ± 0.06	0.71 ± 0.00
PID	0.74 ± 0.03	**0.76 ± 0.02**	0.24 ± 0.01	0.68 ± 0.02	0.83 ± 0.01	0.65 ± 0.00	**0.84 ± 0.00**	0.66 ± 0.04	0.71 ± 0.00
RIP	0.91 ± 0.02	0.90 ± 0.02	0.70 ± 0.01	**0.91 ± 0.02**	**0.92 ± 0.01**	**0.91 ± 0.01**	0.90 ± 0.00	0.75 ± 0.16	0.72 ± 0.00
TMN	1.00 ± 0.00	0.99 ± 0.01	0.97 ± 0.00	0.99 ± 0.01	**0.98 ± 0.01**	**1.00 ± 0.00**	0.89 ± 0.00	0.98 ± 0.04	0.72 ± 0.00
VCP	0.87 ± 0.03	**0.87 ± 0.03**	0.59 ± 0.01	0.84 ± 0.03	**0.92 ± 0.01**	0.85 ± 0.03	0.78 ± 0.01	0.59 ± 0.16	0.71 ± 0.00
AVG RNK.		**1.64**	**3.59**	**2.45**	**1.50**	**2.36**	**1.82**	**3.55**	**3.09**

Source – Oliveira *et al.* (2018).

Figure 7 – Critical difference plots with respect to the accuracy rankings (a) and the sparsity rankings (b) from Table 5. We recall that those variants which are not joined by a bold line can be regarded as different.



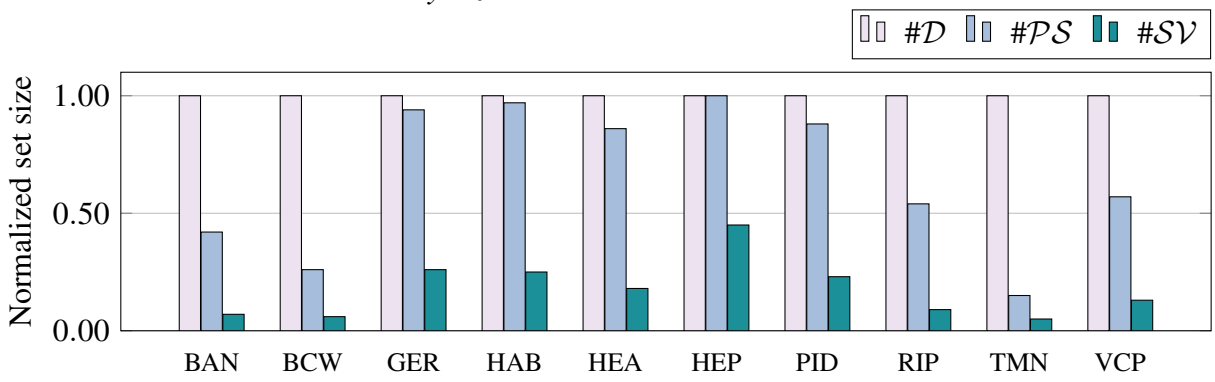
(a) Accuracy rankings.

(b) Sparsity rankings.

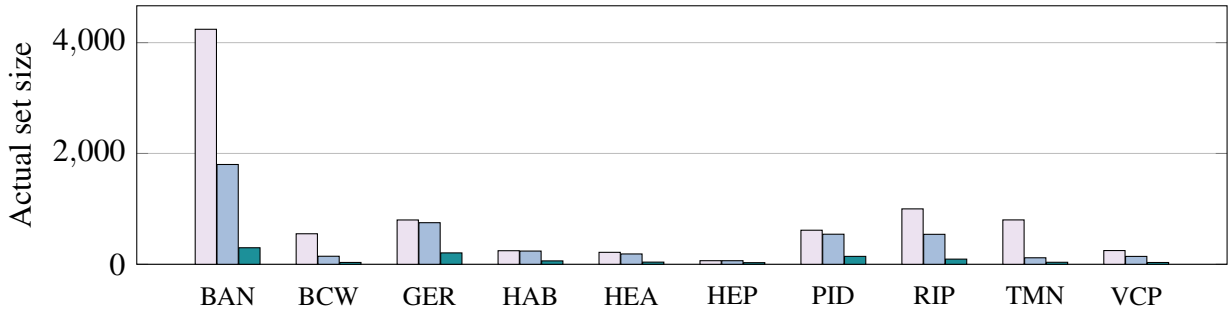
Source – Oliveira *et al.* (2018).

As one can see from Figure 7, CC-LSSVM performed the best rank for the accuracy criteria, thus, showing high generalization performance against other variants. However, CC-LSSVM performed the last for the sparsity criteria. Such a finding indicates that our class-corner support vector selection (via CCIS) can balance between a smaller model and a less constrained one without sacrificing the generalization performance.

Figure 8 – Bar plots showing the original training set $\mathcal{D}$, the number of elements in $\mathcal{PS}$, and the number of support vectors in $\mathcal{SV}$ for CC-LSSVM. We show both the scaled and the actual sizes for each *toy size* data set.



(a) Normalized set sizes for $\mathcal{D}$, $\mathcal{PS}$, and $\mathcal{SV}$.



(b) Actual set sizes for $\mathcal{D}$, $\mathcal{PS}$, and $\mathcal{SV}$.

Source – Oliveira *et al.* (2018).

Next, we present the sparsity aspect of CC-LSSVM in both *toy-size* and large-size data sets in Figure 8 and Figure 9, respectively. Each bar plot present the scaled and actual training set sizes showing how our class-corner selection works.

Regarding the sparsity aspect of CC-LSSVM, we present in Figure 8 bar plots showing the scaled and actual training set sizes for further analysis. From that, one can see that in small data sets, CC-LSSVM not always reduce much from the training set $\mathcal{D}$ to the prototype set $\mathcal{PS}$. However, the size of $\mathcal{SV}$ is shown to be very small compared to the training set $\mathcal{D}$, i.e., $M \ll N$. Such a finding suggests that our class-corner selection considers both the model size and model generalization capability.

3.3.2.2 CC-LSSVM for large-size data sets

Table 6 – Large-size data set description: input/output dimensionality and number of training/test samples and required sparsity level.

data set	**#Dim**	**# Tr**	**#Te**	$\#\mathcal{SV}$
ADULT	123	32561	16281	300 (0.92%)
IJCNN	22	49990	91790	400 (0.80%)
SHUTTLE	9	43500	14500	200 (0.46%)
USPS8	256	7291	2007	200 (2.07%)
VEHICLE	100	78823	19705	400 (0.51%)

Source – Oliveira *et al.* (2018).

Regarding the second part of this experiment, we assess how the LSSVM variants behave in large-size settings, reporting the average accuracy of over 20 independent realizations. We recall we adopted the same hyperparameters and methodology described in Zhou (2016), see Table 6.

Concerning the LSSVM variations, we only compared our proposal against the FSA-LSSVM and SSD-LSSVM since other pruning algorithms are unsuitable for large-scale training due to memory constraints (they require solving the complete linear system in advance). However, for this experiment, we employed a different version of CC-LSSVM, named Fixed-sized CC-LSSVM (FCC-LSSVM). We perform a greedy class-corner support vector selection in a stratification fashion. For that, we extend the local aspect of our proposal, the fixed-radius near neighbor search, by using stratification and performing greedy class-corner support vector selection in each disjoint set. The stratification strategy splits the data into J (a hyperparameter) disjoint subsets with equal class distribution, that is

$$\mathcal{D} = \bigcup_{i=1}^{J} \mathcal{S}_i. \tag{3.14}$$

For each stratum $\mathcal{S}_i$, we carry out the CCIS to select local subsets of class-corner support vectors and constraints. Once we perform such a process on the strata, we select the *strong* samples based on $\Gamma(\cdot)$ so that we can combine them and estimate the dual variables and bias of FCC-LSSVM. For better comprehension, we present the algorithm for FCC-LSSVM in Algorithm 9.

As one can see in Table 7, the FCC-LSSVM achieved very similar and stable accuracy values against other variants. We support that FCC-LSSVM is stable due to the reported standard deviation scores being too low despite noticing different support vectors at each realization.

Algorithm 9 The FCC-LSSVM learning algorithm

FCC-LSSVM-TRAINING$(\mathcal{D}, R, P, J)$

1 Initialize both $\mathcal{PS} \leftarrow \{\varnothing\}$ and $\mathcal{SV} \leftarrow \{\varnothing\}$.
  ▷ Step 1: Support vector selection.
  ▷ Step 1.1: Stratification.
2 Perform stratification on $\mathcal{D}$ so that $\mathcal{D} \leftarrow \bigcup_{i=1}^{J} \mathcal{S}_i$.
  ▷ Step 1.2: Local strata selections.
3 **for** $i \leftarrow 1$ **to** J
4 **do** Compute R_i according to Equation 3.10 using $\mathcal{S}_i$.
5 $\mathcal{PS} \leftarrow \mathcal{PS} \cup \text{CCIS}(\mathcal{S}_i, 0, R_i)$.
6 $\mathcal{SV} \leftarrow \mathcal{SV} \cup \text{CCIS}(\mathcal{S}_i, P, R_i)$.
  ▷ Step 1.3: Fixed-size selection.
7 Sort descendingly $\mathcal{SV}$ using $\Gamma(\cdot)$ as the value function.
8 Let $\mathcal{SV}^\star$ be a subset of $\mathcal{SV}$ with the first M samples.
  ▷ Step 2: Estimating Lagrangian multipliers and bias.
9 Build the linear system of Equation 3.11 using $\mathcal{PS}$ and $\mathcal{SV}^\star$.
10 Estimate $\hat{\boldsymbol{\omega}}$ using the usual least-squares.
11 Rescue from $\hat{\boldsymbol{\omega}}$ the dual variables $\boldsymbol{\alpha}^\star$ and the bias $b^\star$.
12 **return** $\mathcal{SV}^\star$, $\boldsymbol{\alpha}^\star$, and $b^\star$.

Table 7 – Performance on some large data sets. For each data set, the best performing models are in boldface. We recall that all values are the average for 20 independent realizations.

data set	**FCC-LSSVM**	**FSA-LSSVM**	**SSD-LSSVM**
ADULT	$\mathbf{0.76 \pm 0.00}$	0.65 ± 0.04	$\mathbf{0.76 \pm 0.00}$
IJCNN	0.89 ± 0.00	0.78 ± 0.07	$\mathbf{0.93 \pm 0.00}$
SHUTTLE	0.99 ± 0.00	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
USPS8	0.93 ± 0.00	$\mathbf{0.99 \pm 0.00}$	0.93 ± 0.00
VEHICLE	$\mathbf{0.84 \pm 0.00}$	0.62 ± 0.04	0.54 ± 0.00

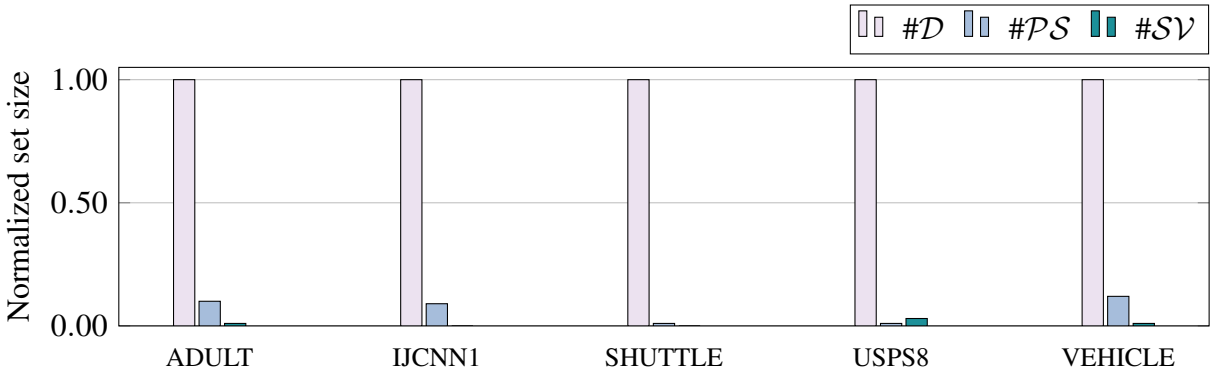
Source – Oliveira *et al.* (2018).

From the sparsity point-of-view depicted in Figure 9, one can see a dramatical reduction in each step, especially from $\mathcal{D}$ to $\mathcal{PS}$. Again, such a finding enforces that our class-corner selection can balance between a smaller model and a less constrained one without sacrificing the model generalization, especially for large-size data sets.

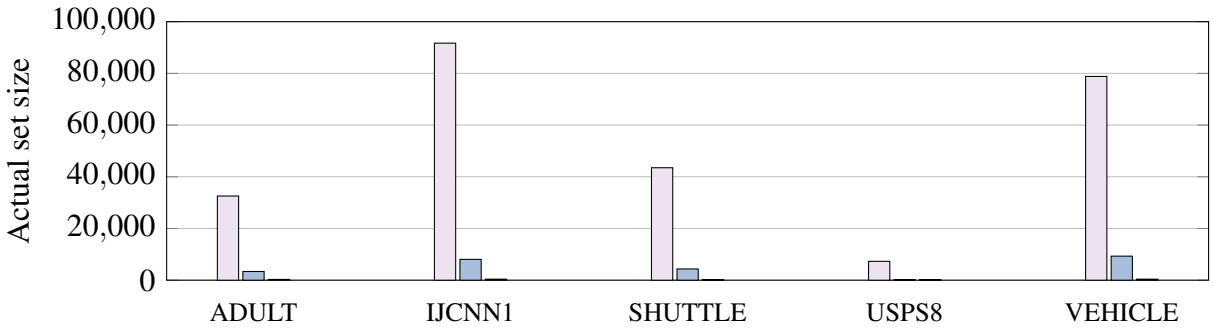
3.3.3 Decision boundary empirical quality assessment for CC-LSSVM

Continuing with the set of experiments, we chose three data sets from the toy problems $\in \mathbb{R}^2$. Such data sets were select by their features with respect to decision boundary analysis, i.e., where $f(\mathbf{x}) = 0$. The first one, namely, Two moons (TMN), is a separable problem,

Figure 9 – Bar plots showing the set sizes for FCC-LSSVM: the training set $\mathcal{D}$, the prototype vectors' set $\mathcal{PS}$, and the support vectors' set $\mathcal{SV}$. We show both the scaled and the actual sizes for each large-size data set.



(a) Normalized set sizes for $\mathcal{D}$, $\mathcal{PS}$, and $\mathcal{SV}$.



(b) Actual set sizes for $\mathcal{D}$, $\mathcal{PS}$, and $\mathcal{SV}$.

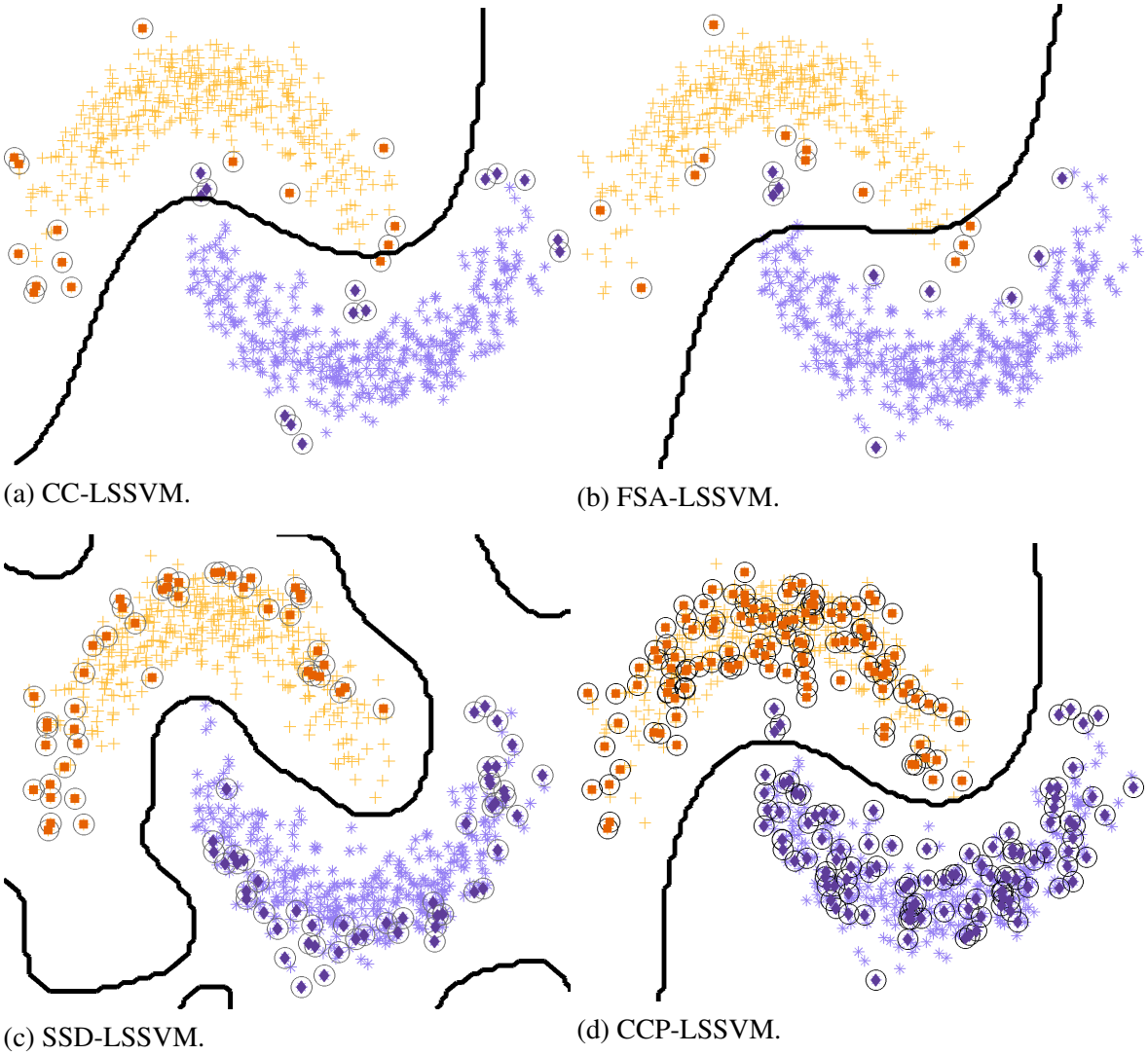
Source – Oliveira *et al.* (2018).

while the other two, namely, Ripley (RIP) and Banana (BAN), have low and moderate class overlapping, respectively.

As one can see in Figure 10, all variants produced proper decision boundaries to separate the data. Such a finding is strong evidence of high generalization in all LSSVM variants, despite their formulation. The noticeable difference across all LSSVM variants is the SV selection in which CC-LSSVM and FSA-LSSVM required fewer support vectors, all located near the decision boundary. In a different direction, SSD-LSSVM found most SVs somehow far from the decision boundary while required more SVs than CC-LSSVM and FSA-LSSVM. As for the CCP-LSSVM, although we set its sampling ratio to 30%, its sparse approximation could not get rid of more SVs, thus resulting in a less sparse solution.

As for the Ripley data set depicted in Figure 11, we highlight that most models produced satisfactory decision boundaries even though there are a little class overlapping and some outliers. In this data set, CC-LSSVM and FSA-LSSVM kept most SVs near the boundary decision; however, CC-LSSVM required less than FSA-LSSVM. Also, SSD-LSSVM attained

Figure 10 – Two moon data set and the produced decision boundaries by some LSSVM variants.



(a) CC-LSSVM.

(b) FSA-LSSVM.

(c) SSD-LSSVM.

(d) CCP-LSSVM.

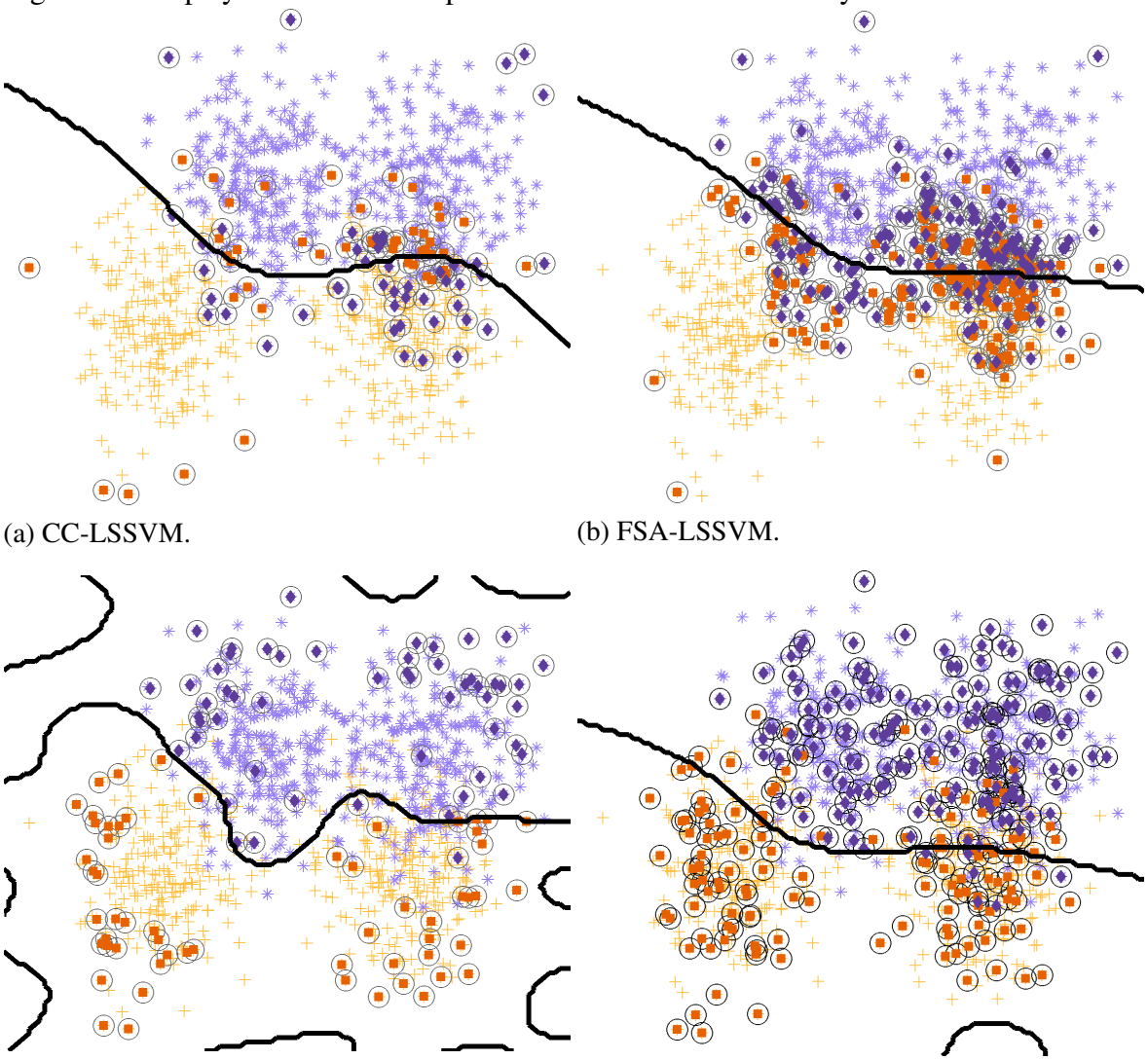
Source – Oliveira *et al.* (2018).

fewer SVs near the boundary decision while CCP-LSSVM achieved the sparsest solution.

Regarding the Banana data set depicted in Figure 12, one can see that most models produced similar decision boundaries, except for FSA-LSSVM. Additionally, due to the severe overlapping between classes right at the top, these models required more SVs in overlapping regions. In this data set, CC-LSSVM and FSA-LSSVM chose SVs near the boundary decision while SSD-LSSVM selected SVs far from the decision boundary.

From this empirical analysis, we noticed that CC-LSSVM is sensitive to the degree of overlap between classes. As the overlapping increases, e.g., Banana, CC-LSSVM selected more SVs; thus, the sparsity decreases but not as much as the other variants, namely, CCP-LSSVM and FSA-LSSVM. On the other hand, when it comes to separable problems (e.g., Two moons and Ripley), CC-LSSVM required fewer SVs than the others.

Figure 11 – Ripley data set and the produced decision boundaries by some LSSVM variants.



(a) CC-LSSVM. (b) FSA-LSSVM.

(c) SSD-LSSVM. (d) CCP-LSSVM.

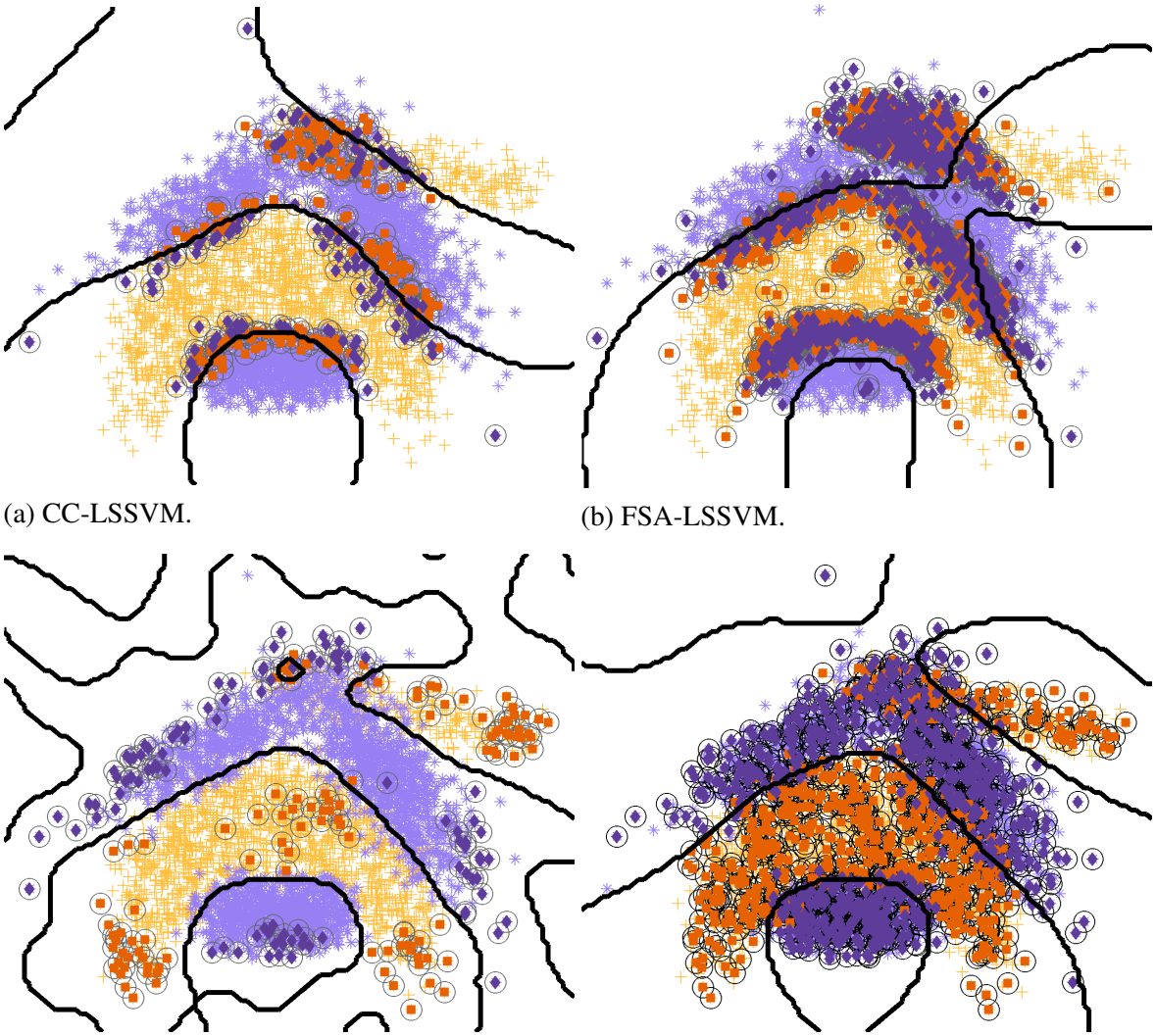
Source – Oliveira *et al.* (2018).

3.3.4 The hyperparameter influence for CC-LSSVM

To investigate the hyperparameter influence, we varied both P and R for the Ripley data set, see Figure 13. The more both variability threshold and distance threshold increase, the more the sparsity increases while the accuracy decreases. On the other hand, the more both variability threshold and distance threshold decreases, the more the accuracy increases while the sparsity decreases dramatically.

Concerning the SVs selection analysis, we noticed that CCISS imposes sparseness in both *toy-size* and large-size data sets. At the same time, it achieved higher or similar accuracy values, especially for large-size data sets alongside dramatical sparsity results. Thus, we support that CCISS is more suitable for large-size data sets.

Figure 12 – Banana data set and the produced decision boundaries by some LSSVM variants.



(a) CC-LSSVM. (b) FSA-LSSVM.

(c) SSD-LSSVM. (d) CCP-LSSVM.

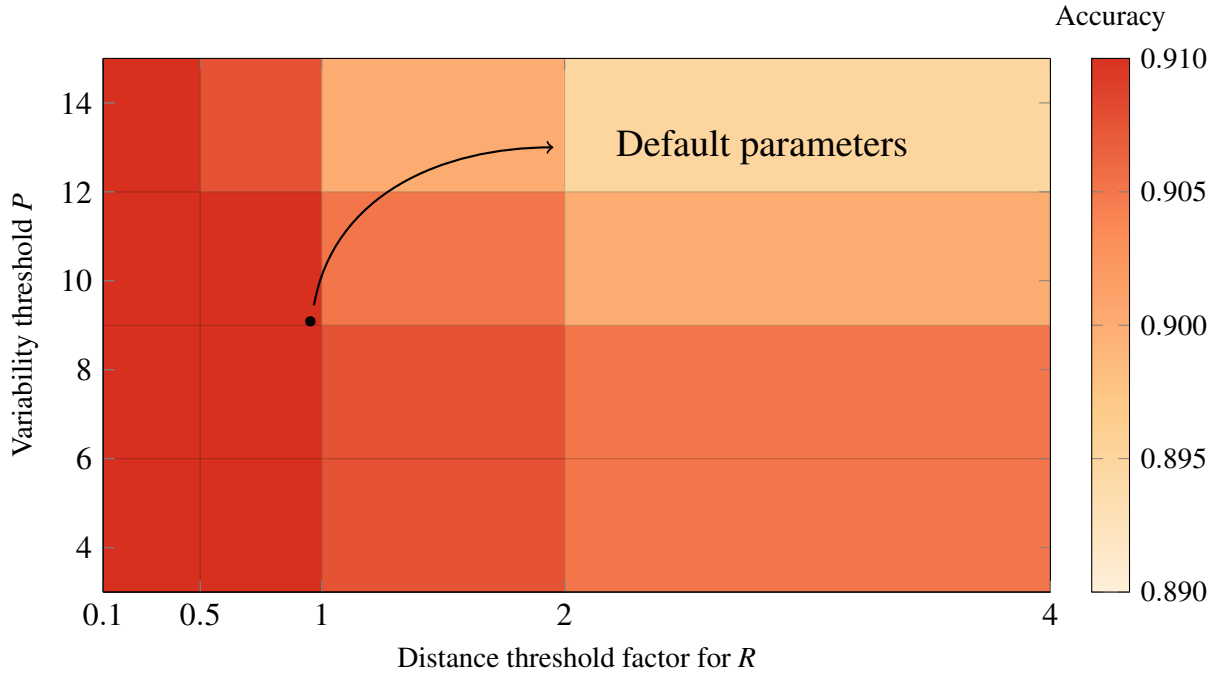
Source – Oliveira *et al.* (2018).

As one can see in Table 8, CC-LSSVM is a competitive variant since it kept polynomial costs concerning memory, training, and prediction costs. Such a finding is very interesting because when combined with higher/similar accuracy rates and sparsity against other LSSVM variants, such a formulation is beneficial for the LSSVM model.

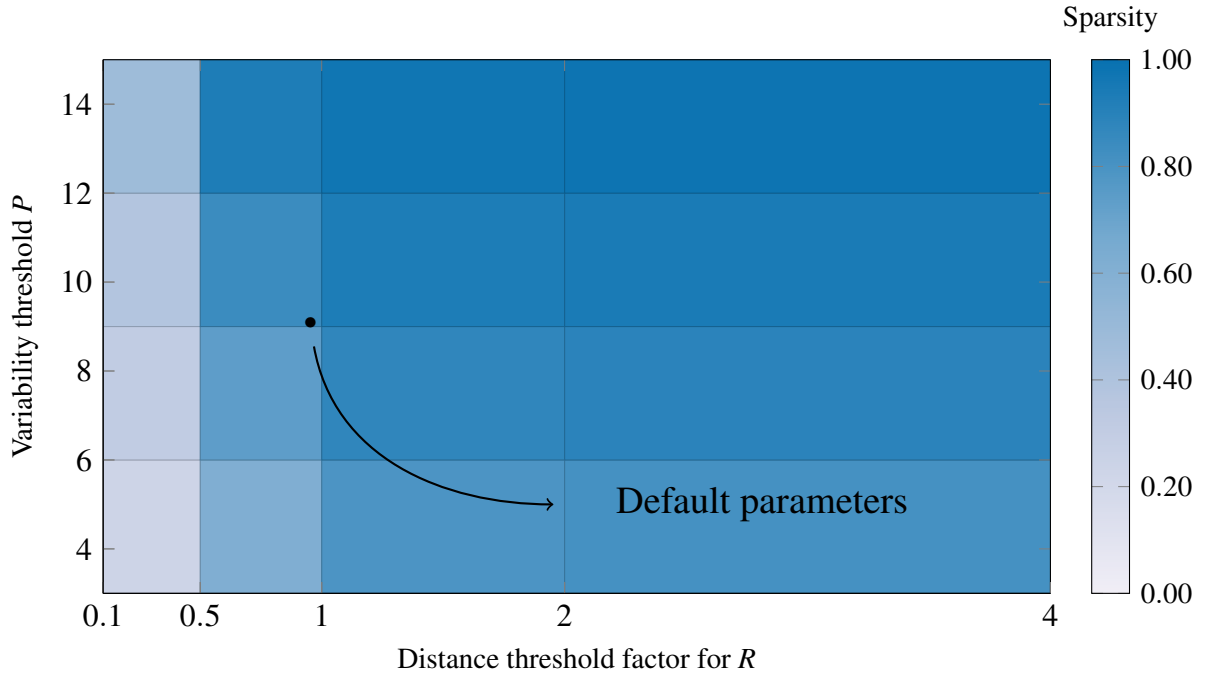
3.4 Concluding remarks for CC-LSSVM

This chapter presented two contributions of this thesis: the Class Corner Instance Selection and Class-corner Least-Squares Support Vector Machine. In our first contribution, we mainly extended the definition of a corner in FAST, an image corner detector, and then applied the same reasoning to support vector selection. At the same time, in the second contribution, we

Figure 13 – Hyperparameter influence concerning both Accuracy and Sparsity in CC-LSSVM using Ripley data set. The black point stands for the default values empirically determined. The variability threshold (y-axis) is P, while the distance threshold (x-axis) is a factor of R.



(a) Accuracy experiment.



(b) Sparsity experiment.

Source – Adapted from Oliveira *et al.* (2018).

Table 8 – LSSVM variant comparison in terms of memory requirement, training cost, and prediction cost. We recall that N stands for the cardinality of the training set, while M is the number of support vectors.

MODEL	MEMORY	TRAINING	PREDICTION
LSSVM (DUAL)	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$
FSA-LSSVM (JIAO *et al.*, 2007)	$\mathcal{O}(M^2)$	$\mathcal{O}(M^3)$	$\mathcal{O}(M)$
CCP-LSSVM (YANG *et al.*, 2014)	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$	$\mathcal{O}(M)$
SSD-LSSVM (MALL; SUYKENS, 2015)	$\mathcal{O}(MN)$	$\mathcal{O}(M^2N)$	$\mathcal{O}(M)$
CC-LSSVM	$\mathcal{O}(MN)$	$\mathcal{O}(M^2N)$	$\mathcal{O}(M)$
FSOCC-LSSVM	$\mathcal{O}(MN)$	$\mathcal{O}(M^2N)$	$\mathcal{O}(M)$

Source – Oliveira *et al.* (2018).

proposed a less constrained model.

We validate both proposals concomitantly through three types of experiments, evaluating different aspects: accuracy, sparseness, support vector selection, and hyperparameter sensitivity. In the first set of experiments, we performed accuracy and sparseness measurements in a typical black-box evaluation fashion on some *toy-size* and large-size data sets. Later, in the second set of experiments, we analyzed the support vector selection as sparseness. After, we performed an empirical decision boundary analysis in which we analyzed the SVs' localization and the resulting decision boundaries. Finally, in the third and last set of experiments, we addressed the hyperparameter influence in the face of the trade-off between sparsity and accuracy.

Regarding the *toy-size* results, our proposal achieved higher accuracy rates with moderate, and sometimes similar, sparsity against other LSSVM variations. As for the large data sets, our proposal accomplished similar or higher accuracy rates against other variants, all with low variance on large-size data sets. We noticed from the empirical decision boundary analysis that our proposal achieved less complex decision boundaries due to the SVs' localization near the class boundary. Moreover, regarding the hyperparameters (variability threshold and distance threshold), we noticed a positive correlation between them and sparsity scores alongside their negative correlation and accuracy.

Finally, after carrying out an analysis regarding the model complexity, we support that our proposal is a competitive variant since it kept a polynomial cost concerning memory, training, and prediction costs alongside higher and similar accuracy rates and sparsity scores compared to other LSSVM variants.

3.4.1 Shortcomings

Choosing an appropriate value for K and P in K-NN is, for sure, the main drawback in CISS. Also, one should consider high-dimensional data and the effects of the curse of dimensionality because the Euclidean distance becomes meaningless as the dimension of the data increases significantly. For this scenario, we suggest adopting any prior dimensionality reduction technique to overcome such a drawback.

Also, we highlight that CCIS presents a higher computational cost since it needs all samples to perform instance selection. However, apart from the additional computation during the learning phase, such a decremental algorithm results in a model that requires less memory usage, which is well worth the computational savings during the execution that follows.

3.4.2 Future works

Continuing exploring the image processing domain, we are exploring concepts related to scale-space filtering to adopt other strategies to identify other class corners/relevant samples to build instance-based learners. Also, we are investigating manners to deal with the shortcomings that affect its performance, such as choosing appropriate values for K (especially for high-dimensional data) and how to adopt an incremental instead of the batch/decremental one.

Moreover, we intend to tackle some instance-based learners formulations to perform both tasks, namely, instance selection and parameter learning, during training.

4 REDUCING COMPLEXITY BY REGULARIZATION

Our second proposal to deal with the complexity of learning models explored the model regularization perspective. To do so, we built an instance-based regularized system to impose sparseness not by selection but by using weighted information into the model. Here, we adopted the MLM framework and derived a novel formulation of it. We named it *Lightweight* Minimal Learning Machine (LW-MLM).

The key idea is to assign weights based on instance relevance instead of simply discarding them as previously done (through selection). We opt for such a pathway because treating such a process of selecting instances and computing the mapping between distance matrices as a sparse resolution problem of multiple-response linear systems is an NP-Hard problem.

For sparse learning strategies, we refer the reader to the following works, namely, Donoho and Elad (2003), Fuchs (2004), and Tropp (2004) and for some applications related to image super-resolution (YANG *et al.*, 2010), order preservation (NI *et al.*, 2015), and multitask learning in image emotion distribution prediction (ZHAO *et al.*, 2017).

4.1 The Lightweight Minimal Learning Machine

To derive LW-MLM, we posed the MLM model under two mains constraints: the reconstruction constraint – error term – and the model complexity constraint – norm term. The reconstruction constraint requires consistency between the recovered mapping from the input to the output spaces. In contrast, the model complexity constraint assumes that the model can represent such a mapping sparsely in an appropriately chosen overcomplete dictionary and recover its sparse representation.

4.1.1 LW-MLM Formulation

The LW-MLM formulation has the following learning cost function:

$$\min_{\mathbf{B}} \; \mathcal{J}_{\mathrm{LW}}(\mathbf{B},\mathbf{P}) = ||\mathbf{DB}-\mathbf{\Delta}||_{\mathcal{F}}^2 + ||\mathbf{PB}||_{\mathcal{F}}^2 \tag{4.1}$$

which yields the following solution (see Appendix A):

$$\hat{\mathbf{B}}_{\mathrm{LW}} = (\mathbf{D}^{\intercal}\mathbf{D}+\mathbf{P}^{\intercal}\mathbf{P})^{-1}\mathbf{D}^{\intercal}\mathbf{\Delta} \tag{4.2}$$

where $\mathbf{P}$ acts as a regularization matrix, weighting the coefficients associated with mapping the samples.

Although it sounds inappropriate to have a hyperparameter $\mathbf{P} \in \mathbb{R}^{N \times N}$, it is actually to fill the gap due to the MLM multiresponse system. In contrast, $\mathbf{P}$ can be derived by a regularization vector $\mathbf{p} \in \mathbb{R}^{N}$ by adopting $\mathbf{P}$ as a diagonal matrix as shown as follows:

$$\mathbf{P} = \text{diag}(\mathbf{p}). \tag{4.3}$$

So, one must establish a feasible way to provide proper weight values for $\mathbf{P}$. Next, we address some strategies to do it so.

4.1.2 On the selection of P

In this section, we will describe some efforts to produce feasible regularization matrices. In the following, we focus on the vector notation since $\mathbf{P}$ can be derived by a regularization vector $\mathbf{p}$.

4.1.2.1 By the normal random-based one

Related to the compressive sensing described in Yang *et al.* (2014), we seek to obtain $\mathbf{P}$ by assigning randomly values to its diagonal, letting the other indexes with 0. In this case, $\mathbf{p}$ is defined as $p_i \sim \mathcal{N}(0,1)$.

4.1.2.2 By the nonlinear parts of f

Sousa Júnior (2014) presented an instance selection algorithm based on distance computations and non-parametric hypothesis testing. By analyzing the *p-values* from the non-parametric hypothesis test, the algorithm indicates which samples correspond to the most linear part of the target function $f(\cdot)$. Consequently, one can find the less linear ones.

Firstly, the author assumes that $f(\cdot)$ satisfies the Weierstrass continuity definition[1]. Then, the author also assumes the distributions of distance measurements on the input and output spaces are equal (matches) up to a normalization factor if one considers a neighborhood of f such that f is linear. Next, one must normalize the measurements to have zero-mean and unit variance before applying the non-parametric Kolmogorov-Smirnov hypothesis test (KS2Sample)

[1] $f(x)$ is continuous at $x = x_0$ if $\forall \varepsilon > 0 \; \exists \; \delta > 0$ such that for every x in the domain of f, $|x - x_0| < \delta \implies |f(x) - f(x_0)| < \varepsilon$

(MASSEY, 1951). The null hypothesis in such a test is that normalized measurements are samples from the same distribution. Then, one builds a ranking criterion based on the resulting *p-values* for each measurement and select some instances based on a threshold.

As we are not interested in selecting samples, we simply apply the KS2Sample test and employ the resulting *p-values* as regularization vector. Let us denote $\phi_k(\mathbf{x}_i)$ and $\psi_k(\mathbf{y}_i)$ the column vectors that encompass the distance measurements from their argument to its k neighbors (nearest points). For sake of simplicity, consider in the following description and notation only the case of $\phi_k(\mathbf{x}_i)$ since a similar description and notation for $\psi_k(\mathbf{y}_i)$ can be done straightforwardly:

$$\phi_k(\mathbf{x}) = \left[\mathrm{d}(\mathbf{x}_{i_1},\mathbf{x}),\mathrm{d}(\mathbf{x}_{i_2},\mathbf{x}),\ldots,\mathrm{d}(\mathbf{x}_{i_K},\mathbf{x})\right]^{\mathsf{T}} \tag{4.4}$$

such that $0 < \mathrm{d}(\mathbf{x}_{i_1},\mathbf{x}) \leq \cdots \leq \mathrm{d}(\mathbf{x}_{i_K},\mathbf{x}) \leq \cdots \leq \mathrm{d}(\mathbf{x}_{i_N},\mathbf{x}) \; \forall \, \mathbf{x}_{i_*} \in \mathcal{D}$, where the indexes $i_1, i_2, \ldots, i_N$ form a permutation of $\{1,\ldots,N\}$. Also, consider $\overline{\phi_k}(\mathbf{x}_i)$ and $\overline{\psi_k}(\mathbf{y}_i)$ the normalized measurements from $\phi_k(\mathbf{x}_i)$ and $\psi_k(\mathbf{y}_i)$, respectively. Now, let us define the KS2Sample test result for a given sample pair $(\mathbf{x}_i,\mathbf{y}_i)$ from their normalized distance measurements as:

$$\Pi(\mathbf{x},\mathbf{y}) = \mathrm{KS2Sample}(\overline{\phi_k}(\mathbf{x}),\overline{\psi_k}(\mathbf{y})). \tag{4.5}$$

Regarding the values, by applying the KS2Sample test result directly, the weights would behave similarly to the previous strategy when we adopted normal distributed random weights since the *p-values* $\in [0,1]$. Therefore, to overcome such a situation, we adopted a transformation to address such a condition by penalizing the samples with *p-values* below a given linear threshold t, that is,

$$\mathcal{KS}^k_{t,\lambda}(\mathbf{x},\mathbf{y}) = \begin{cases} \Pi(\mathbf{x},\mathbf{y}) + \lambda, & \text{if } \Pi(\mathbf{x},\mathbf{y}) < t, \\ \Pi(\mathbf{x},\mathbf{y}), & \text{otherwise.} \end{cases} \tag{4.6}$$

In possession of such a definition, we derive a $p_i = \mathcal{KS}^k_{t,\lambda}(\mathbf{x}_i,\mathbf{y}_i)$. Also, we support the Otsu's method (SEZGIN; SANKUR, 2004) is suitable to find a feasible threshold t from the $\{\Pi(\mathbf{x}_i,\mathbf{y}_i)\}_{i=1}^N$.

4.1.3 Speeding up the out-of-sample prediction

The *lightweight* in LW-MLM is not just related to the regularized values in $\mathbf{B}$ (linear mapping coefficients) but also related to the speedup out-of-sample prediction. Since we employ

all samples in the LW-MLM learning algorithm, we believe that LW-MLM learns the whole known geometric structure, i.e., the domain knowledge is "fully" represented in **B**. With that in mind, we truly support that **B** by itself. Thus, in this setting, LW-MLM can successfully map the linear structure.

Such an assumption encourages us to discard some components from the out-of-sample prediction procedure since most RP projections will result in zero error. First, let us define the discard function $\kappa : \mathbb{R}^N \rightarrow \mathbb{R}^K$ as

$$\kappa(\mathbf{a}) = (a_{i_1}, a_{i_2}, \ldots, a_{i_K})^\intercal, \tag{4.7}$$

where $i_1, i_2, \ldots, i_K, \ldots, i_N$ form a random permutation of $\{1, \ldots, N\}$. Now, the location of $\hat{\mathbf{y}}$ can be estimated by minimizing the following objective function:

$$\hat{\mathbf{y}} = h(\mathbf{x}) = \arg\min_{\mathbf{y}} \parallel \kappa(\Psi(\mathbf{y}) - \Phi(\mathbf{x})\,\mathbf{B}) \parallel_2 . \tag{4.8}$$

Later, we discuss such a speedup procedure in the experiments while showing the relationship through varying the number of components and the prediction error associated.

4.2 Tikhonov regularization and Heteroscedasticity in the LW-MLM framework

It is well known that minimization of a sum-of-squares error (SSE) metric corresponds to maximum likelihood estimation of the parameters of a regression model, where the target data are assumed to be realizations of some deterministic process that has been corrupted by additive Gaussian noise with constant variance (i.e., a homoscedastic noise process) (CAWLEY *et al.*, 2004 apud BISHOP *et al.*, 1995). However, it was reported that the noise models in some real-world applications do satisfy heteroscedasticity (CAWLEY *et al.*, 2004). Heteroscedasticity is the opposite of homoscedasticity. Heteroscedasticity refers to data for which the variance of the dependent variable is unequal across the range of independent variables.

Up to this point, the MLM framework had mainly formulations regarding homoscedasticity (see Karkkainen (2019), Dias *et al.* (2018), Dias *et al.* (2019)), but *w*MLM (GOMES *et al.*, 2015) and Rank-MLM (ALENCAR *et al.*, 2015). The *w*MLM induces such a heteroscedastic behavior in the error term directly since it employs weighted least squares (a specialization of generalized least squares) to find its solution. In short, it goes from the MLM system in Equation 2.10 to the weighted least-squares seen in Equation 2.18 by penalizing the norm of the solution **B** by weighting it with some positive constant λ. The authors of *w*MLM

proposed two ways of configuring its weights in imbalanced classification and classification with reject option. On the other hand, Rank-MLM and LW-MLM induce the same heteroscedastic behavior in the (norm) complexity term to balance the model complexity and generalization. In summary, Rank-MLM and LW-MLM go from the MLM system in Equation 2.10 the same Tikhonov regularization strategy, see Equation 2.18 and Equation 4.2, but they differ when it comes to model flexibility since Rank-MLM can be seen as a particular case of LW-MLM ($\mathbf{P} = \lambda \mathbf{I}$).

Moreover, the regularization terms can be understood to be encoding priors on the parameters. This is particularly beneficial and adds more flexibility to the model since the choice of prior is one of the most critical parts of the Bayesian inference workflow. In contrast, most models have often fallen back on vague priors, such as standard Gaussians.

4.2.1 LW-MLM Algorithms for training and out-of-sample procedures

For sake of simplicity, consider the following Algorithm 10 and Algorithm 11 to describe the training and prediction procedures in LW-MLM, respectively.

Algorithm 10 Training procedure in LW-MLM.

LW-MLM-TRAINING$(\mathcal{D}, \mathbf{P})$

▷ Here, $\mathbf{P}$ is given by one of the strategies presented in subsection 4.1.2.

1 Compute both distance pairwise matrices $\mathbf{D}$ and $\mathbf{\Delta}$ from $\mathcal{D}$, respectively.
2 Compute $\hat{\mathbf{B}} = (\mathbf{D}^\intercal \mathbf{D} + \mathbf{P}^\intercal \mathbf{P})^{-1} \mathbf{D}^\intercal \mathbf{\Delta}$.
3 **return** $\hat{\mathbf{B}}$.

Algorithm 11 Prediction procedure in LW-MLM.

LW-MLM-PREDICT$(\mathbf{x}^\star, \hat{\mathbf{B}})$

▷ Here, $\kappa(\cdot)$ randomly selects some components from its argument.

1 Project $\mathbf{x}^\star$ to the output space by computing $\Phi(\mathbf{x}^\star)\,\hat{\mathbf{B}}$.
2 Compute $\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} \parallel \kappa\left(\Psi(\mathbf{y}) - \Phi(\mathbf{x}^\star)\,\hat{\mathbf{B}}\right) \parallel_2$.
3 **return** $\hat{\mathbf{y}}$.

4.3 Experiments and Discussion for LW-MLM

This section presents the experimental framework followed in this chapter, together with the results collected and discussions on them.

4.3.1 Experiment Setup

We carried out some experiments to evaluate different aspects of LW-MLM. Our goal is to investigate how LW-MLM behaves concerning the following aspects: the prediction error, the goodness-of-fit of estimated vs. measured values (without being scaled by the output scale), the model complexity (via norm of matrix **B**), the influence of the RPs number in the out-of-sample prediction and, finally, the prediction error in high dimensional feature spaces. The following experiments address such concerns:

a) **Typical black-box assessment**: Here, we assess LW-MLM against some variants of MLM, analyzing the prediction error and the goodness-of-fit of estimated vs. measured values in a typical black-box test fashion on some data sets;

b) **Visual qualitative analysis**: In this experiment, we examine the importance of regularization by analyzing the level of complexity (via *smoothness*) of the estimated function while we identify the difference between LW-MLM and the other formulations empirically;

c) **The relevance of RPs in the prediction step**: This experiment examines the influence of the number of RPs during out-of-sample prediction in the resulting LW-MLM model;

d) **The prediction error in high dimensional feature**: Here, we conclude our experiments by analyzing how LW-MLM deals with high dimension data sets, i.e., $\mathbf{x} \in \mathbb{R}^D$ s.t. D from ≈ 100 up to 4000.

4.3.2 Typical black-box assessment

This assessment has two perspectives for analysis purposes: the first is related to the RMSE, while the second is related to the R^2. In both experiments, we used real-world benchmark data sets from UCI data sets (LICHMAN, 2013) and the Luis Torgo collection (TORGO, 2005). data set name, acronym, number of training samples (#*Tr*), number of testing samples (#*Te*) and input dimensionality (#*Dim*) about the aforementioned data sets are presented in Table 9.

Also, we highlight we pre-processed all data sets so that we removed samples containing missing values, and we normalized all features by variation (standardization).

Table 9 – Data set descriptions for black-box assessment.

data set	**ACRONYM**	**# DIM**	**# TR**	**# TE**
Abalone	ABA	8	3000	1177
AutoMPG	MPG	7	350	42
Boston housing	BTH	13	400	106
concrete	CON	8	700	330
cpu_act	CPU	12	5000	3192
delta ailerons	DAI	5	5000	2129
delta elevators	DEL	6	6000	3517
kinematics	KIN	8	4500	3692
motor_UPDRS	MUP	20	3000	2875
puma8NH	8NH	8	4500	3692
stock	STO	9	600	350
total_UPDRS	TUP	20	3000	2875
winequality_red	WRE	11	1000	599
winequality_white	WWH	11	3500	1398

Source – Florencio V *et al.* (2020).

We assessed LW-MLM's performance alongside three other MLM variants. The first one, named Full-MLM (FL-MLM), employs all the training set as Reference Points. The second variant is the Random-MLM, in which it randomly selects K reference points from the training data. Finally, the third variant is the Rank-MLM adds up a regularization term to its cost function (ALENCAR *et al.*, 2015).

Also, we investigate how LW-MLM behaves regarding adopting three different mechanisms of generating $\mathbf{P}$. For that, we derived three other versions of LW-MLM, namely, LW-MLM-1, LW-MLM-2, and LW-MLM-3. LW-MLM-1. LW-MLM-1 employs a diagonal matrix with random values from a normal distribution with zero mean and unit variance. In contrast, LW-MLM-2 and LW-MLM-3 both employ a diagonal matrix with values assigned via the KS-2-Sample-Test described in Equation 4.6, but adopting different linearity thresholds as described in Table 10.

We also employed a combination of the k-fold cross-validation and holdout methods in the experiments. The holdout method with the training and testing division used by Zhao *et al.* (2012) was employed to estimate the performance metrics. For Random-MLM and Rank-MLM, we choose the hyperparameters by grid search combined with 5-fold cross-validation yielding the best combinations with higher R^2 values from 30 independent runs. See Table 10 for the

hyperparameter value space employed for each model.

Table 10 – MLM variants and their hyperparameters configurations.

MODEL	DESCRIPTION	HYPERPARAMETER VALUE
Full-MLM	K: Number of RPs.	$K = N$.
Random-MLM	K: Number of RPs chosen randomly from a range through grid search and cross-validation.	$K \in \{\lfloor 0.05 \times i \times N \rfloor\}_{i=1}^{10}$
Rank-MLM	K: Number of RPs chosen randomly from a range through grid search and cross-validation	$K \in \{\lfloor 0.05 \times i \times N \rfloor\}_{i=1}^{10}$
	C : Regularization parameter optimized by grid search and cross-validation.	$C \in \{2^{-4}, 2^{-3}, \ldots, 2^{1}, 2^{2}\}$.
LW-MLM-1	**P**: Diagonal matrix with random values from a normal distribution with zero mean and unit variance.	$P_{i,j} = \begin{cases} \mathcal{N}(0,1), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$.
LW-MLM-2	**P**: Diagonal matrix with values assigned via the KS-2-Sample-Test described in Equation 4.6.	$P_{i,j} = \begin{cases} \mathcal{KS}_0^k(\mathbf{x}_i, \mathbf{y}_i), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$.
	k: Number of nearest points to compute distance.	$k = \log_{10}(5 \times N)$.
LW-MLM-3	**P**: Diagonal matrix with values assigned via the KS-2-Sample-Test described in Equation 4.6.	$P_{i,j} = \begin{cases} \mathcal{KS}_t^k(\mathbf{x}_i, \mathbf{y}_i), & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$.
	k: Number of nearest points to compute distance.	$k = \log_{10}(5 \times N)$.
	t: Linearity threshold.	$t = \mathcal{OTSU}_k(\mathbf{X}, \mathbf{Y})$.

Source – Florencio V *et al.* (2020).

Regarding the statistical analysis of the results, we carried out the Friedman test to assess all performance metrics, namely, the RMSE, the R^2, and the norm of matrix **B** to better distinguish the comparison of the MLM models over 30 independent realizations. To do it so, we graphically represent such a comparison through the Critical Difference (CD) plot as the one presented in Demšar (2006).

4.3.2.1 Prediction error as performance measurement

As one can see in Table 11, we reported the RMSE metrics regarding the black-box experiments for all data sets and models. In such an experiment, after performing the Friedman test, we noticed that all models are equivalent since all are connected, see the CD plot in Figure 14.

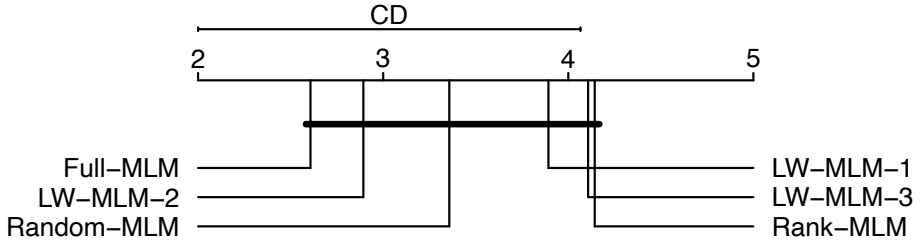
Table 11 – Black-box experiment results for RMSE. We recall that the variants which are not joined by a bold line can be regarded as different in the CD plot bellow.

data sets	FULL-MLM	RANDOM-MLM	RANK-MLM	LW-MLM-1	LW-MLM-2	LW-MLM-3
ABA	2.24 ± 0.07	2.14 ± 0.06	2.12 ± 0.06	2.13 ± 0.06	2.22 ± 0.07	2.22 ± 0.07
MPG	2.64 ± 0.53	2.69 ± 0.50	2.62 ± 0.49	2.61 ± 0.47	2.57 ± 0.48	2.58 ± 0.48
BTH	3.11 ± 0.48	3.47 ± 0.60	3.57 ± 0.57	3.54 ± 0.52	3.31 ± 0.50	3.80 ± 0.49
CON	5.76 ± 0.53	6.40 ± 0.46	7.25 ± 0.38	7.21 ± 0.38	7.37 ± 0.41	7.94 ± 0.35
CPU	2.81 ± 0.06	2.86 ± 0.07	2.94 ± 0.07	2.91 ± 0.06	2.81 ± 0.06	2.81 ± 0.06
DAI	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$
DEL	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$
KIN	0.08 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	0.09 ± 0.00	0.10 ± 0.00	0.11 ± 0.00
MUP	2.11 ± 0.06	2.34 ± 0.06	2.68 ± 0.05	2.65 ± 0.05	2.49 ± 0.06	2.52 ± 0.06
8NH	3.41 ± 0.04	3.36 ± 0.04	3.33 ± 0.04	3.33 ± 0.04	3.32 ± 0.03	3.37 ± 0.04
STO	0.66 ± 0.04	0.74 ± 0.04	0.83 ± 0.05	0.83 ± 0.05	0.79 ± 0.04	0.85 ± 0.05
TUP	2.85 ± 0.08	3.17 ± 0.07	3.68 ± 0.06	3.66 ± 0.07	3.48 ± 0.07	3.97 ± 0.07
WRE	0.61 ± 0.02	0.62 ± 0.02	0.62 ± 0.01	0.62 ± 0.02	0.61 ± 0.02	0.61 ± 0.02
WWH	0.61 ± 0.02	0.65 ± 0.01	0.67 ± 0.01	0.67 ± 0.01	0.61 ± 0.02	0.61 ± 0.02

* The values are not exactly zero but rather too small.

Source – Florencio V *et al.* (2020).

Figure 14 – Critical Difference plots regarding RMSE for black-box experiments.



Source – Florencio V *et al.* (2020).

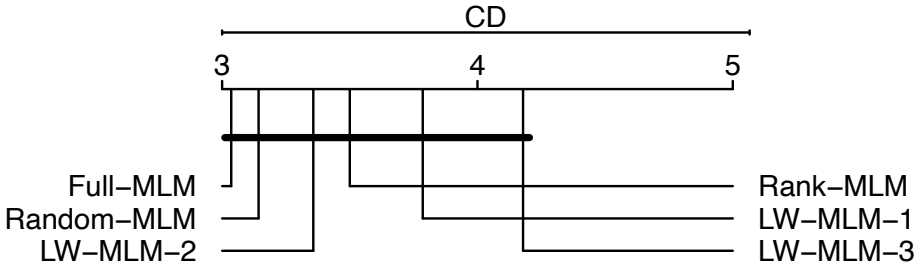
Table 12 – Black-box experiment results for R^2. Again, we recall that the variants which are not joined by a bold line can be regarded as different in the CD plot bellow.

data sets	FULL-MLM	RANDOM-MLM	RANK-MLM	LW-MLM-1	LW-MLM-2	LW-MLM-3
ABA	0.53 ± 0.02	0.57 ± 0.02	0.58 ± 0.02	0.57 ± 0.02	0.54 ± 0.02	0.54 ± 0.02
MPG	0.89 ± 0.05	0.88 ± 0.04	0.89 ± 0.04	0.89 ± 0.04	0.89 ± 0.04	0.89 ± 0.04
BTH	0.88 ± 0.04	0.85 ± 0.05	0.85 ± 0.05	0.85 ± 0.05	0.87 ± 0.04	0.83 ± 0.05
CON	0.88 ± 0.02	0.85 ± 0.02	0.81 ± 0.02	0.81 ± 0.02	0.81 ± 0.02	0.78 ± 0.02
CPU	0.98 ± 0.00	0.98 ± 0.00	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00
DAI	0.68 ± 0.01	0.71 ± 0.01	0.71 ± 0.01	0.71 ± 0.01	0.69 ± 0.01	0.69 ± 0.01
DEL	0.59 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.64 ± 0.01	0.59 ± 0.01	0.59 ± 0.01
KIN	0.90 ± 0.00	0.89 ± 0.00	0.88 ± 0.00	0.88 ± 0.00	0.87 ± 0.00	0.83 ± 0.01
MUP	0.93 ± 0.00	0.92 ± 0.00	0.89 ± 0.00	0.90 ± 0.00	0.91 ± 0.00	0.91 ± 0.01
8NH	0.63 ± 0.01	0.64 ± 0.01	0.65 ± 0.01	0.65 ± 0.01	0.65 ± 0.01	0.64 ± 0.01
STO	0.99 ± 0.00	0.99 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.99 ± 0.00	0.98 ± 0.00
TUP	0.93 ± 0.00	0.91 ± 0.00	0.88 ± 0.01	0.89 ± 0.01	0.90 ± 0.00	0.87 ± 0.01
WRE	0.44 ± 0.03	0.40 ± 0.03	0.40 ± 0.03	0.41 ± 0.03	0.44 ± 0.03	0.44 ± 0.03
WWH	0.53 ± 0.02	0.47 ± 0.02	0.43 ± 0.02	0.43 ± 0.02	0.53 ± 0.02	0.53 ± 0.02

Source – Florencio V *et al.* (2020).

Such a finding is very interesting because it shows that such a new formulation did not sacrifice the generalization performance.

Figure 15 – Critical Difference plots regarding R^2 for black-box experiments.



Source – Florencio V *et al.* (2020).

Table 13 – Black-box experiment results for NORM. The matrix B norm value is scaled between 0 and 1 for each data set. Once again, we recall that the variants which are not joined by a bold line can be regarded as different in the CD plot bellow.

data sets	FULL-MLM	RANDOM-MLM	RANK-MLM	LW-MLM-1	LW-MLM-2	LW-MLM-3
ABA	1.00 ± 0.00	0.03 ± 0.01	0.01 ± 0.00	0.07 ± 0.00	0.84 ± 0.02	0.84 ± 0.02
MPG	1.00 ± 0.00	0.28 ± 0.12	0.07 ± 0.01	0.14 ± 0.02	0.24 ± 0.02	0.24 ± 0.03
BTH	1.00 ± 0.00	0.49 ± 0.06	0.20 ± 0.02	0.32 ± 0.04	0.36 ± 0.02	0.14 ± 0.08
CON	1.00 ± 0.00	0.33 ± 0.32	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$	$0.00 \pm 0.0^*$
CPU	1.00 ± 0.00	0.43 ± 0.02	0.03 ± 0.00	0.11 ± 0.01	0.82 ± 0.01	0.82 ± 0.01
DAI	1.00 ± 0.00	0.04 ± 0.02	0.02 ± 0.01	0.10 ± 0.01	0.92 ± 0.01	0.92 ± 0.01
DEL	1.00 ± 0.00	0.02 ± 0.00	0.01 ± 0.00	0.09 ± 0.00	0.96 ± 0.02	0.96 ± 0.02
KIN	1.00 ± 0.00	0.53 ± 0.01	0.31 ± 0.00	0.45 ± 0.01	0.31 ± 0.01	0.12 ± 0.00
MUP	1.00 ± 0.00	0.59 ± 0.02	0.23 ± 0.00	0.44 ± 0.01	0.51 ± 0.02	0.50 ± 0.02
8NH	1.00 ± 0.00	0.23 ± 0.06	0.15 ± 0.02	0.32 ± 0.01	0.21 ± 0.01	0.07 ± 0.00
STO	1.00 ± 0.00	0.51 ± 0.02	0.13 ± 0.00	0.26 ± 0.01	0.29 ± 0.02	0.15 ± 0.01
TUP	1.00 ± 0.00	0.60 ± 0.02	0.23 ± 0.00	0.45 ± 0.02	0.45 ± 0.01	0.20 ± 0.00
WRE	1.00 ± 0.00	0.43 ± 0.10	0.09 ± 0.02	0.20 ± 0.02	0.96 ± 0.03	0.96 ± 0.03
WWH	1.00 ± 0.00	0.47 ± 0.02	0.07 ± 0.00	0.19 ± 0.01	0.99 ± 0.02	0.99 ± 0.02

* The values are not exactly zero but rather too small, specially after the scaling.

Source – Florencio V *et al.* (2020).

4.3.2.2 Goodness-of-fit as performance measurement

Another good way to compare the regressors is the coefficient of determination R^2 since it does not scale according to the output order of magnitude. Table 12 shows these results for all models. As one can see, the results are very similar, and, again, all models can be regarded as equivalents since all are connected, see Figure 15.

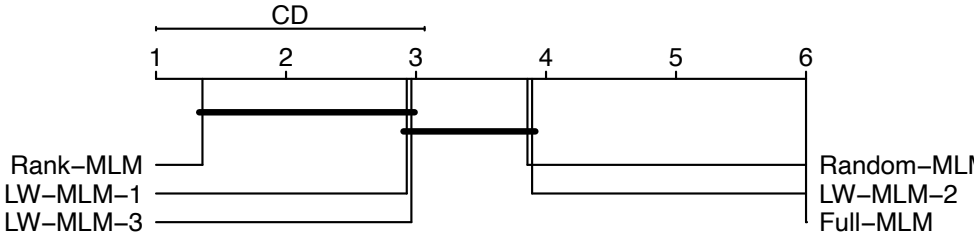
4.3.2.3 The norm as performance measurement

Lastly, Table 13 shows the norm of matrix $\mathbf{B}$. The values were scaled between 0 and 1 by subtracting 1 from division by the $\| \mathbf{B} \|_{\mathcal{F}}$ of Full-MLM (that is $\text{NORM}(x) = 1 - x \| \mathbf{B} \|_{\mathcal{F}}{}^{-1}$) since it acts as an upper bound[2] because in other models either $\mathbf{D}$ and/or $\mathbf{\Delta}$ are not squared matrices nor they present any regularization factor.

From that, it is possible to conclude that LW-MLM reduces the model complexity,

[2] $\| \mathbf{B} \|_{\mathcal{F}} = \| \mathbf{D}^{-1}\mathbf{\Delta} \|_{\mathcal{F}} \geq \| \mathbf{D}^{-1} \|_{\mathcal{F}} \| \mathbf{\Delta} \|_{\mathcal{F}}$.

Figure 16 – Critical Difference plots regarding NORM for black-box experiments.



Source – Florencio V *et al.* (2020).

achieving feasible results without sacrificing the model generalization, especially when analyzing the overall results from other metrics, see Figure 16.

4.3.3 Visual qualitative analysis

In this experiments, we chose three data sets from the regression problems $f : \mathbb{R} \to \mathbb{R}$, two artificial and a real-world one. Such data sets were select by their features concerning the learned function. The first two data sets, namely, Artificial I and Artificial II, are regression problems generated by known functions. The first function has a uniform error, while the second is a very distinct non-linear function, both with 200 samples. The third function is a real-world data set called by mcycle containing 133 samples from a simulated motorcycle accident used to test crash helmets (SILVERMAN, 1985). We summarize such data sets in the following:

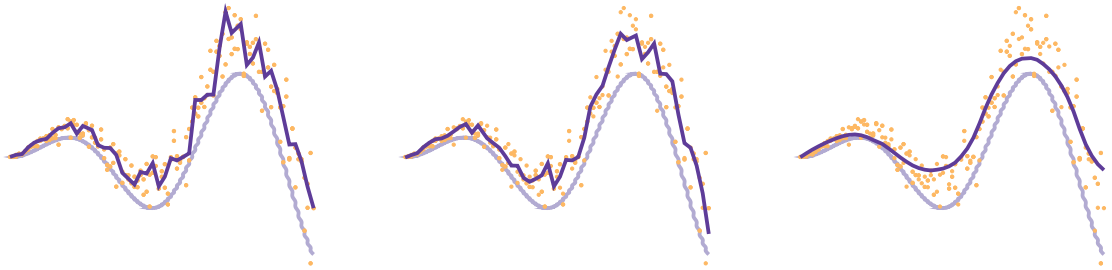
a) **Artificial I**. $y_i = i \times \sin(1.5 \times x_i) + u$, s.t. $u \sim \mathcal{U}(0.250, 0.500), i = 1, \ldots, 200$;
b) **Artificial II**. $y_i = \sum_{j=1,3,5,\ldots}^{\infty} \frac{\sin(j \times x_i)}{j} + u$, s.t. $u \sim \mathcal{U}(0.025, 0.050), i = 1, \ldots, 200$;
c) **Mcycle**. 133 observations from a simulated motorcycle accident, used to test crash helmets (SILVERMAN, 1985).

As for the hyperparameters to produce such visualizations, we performed a grid search with cross-validation to determine them. Moreover, for visualization purposes, we kept the RPs with the same color as the training samples.

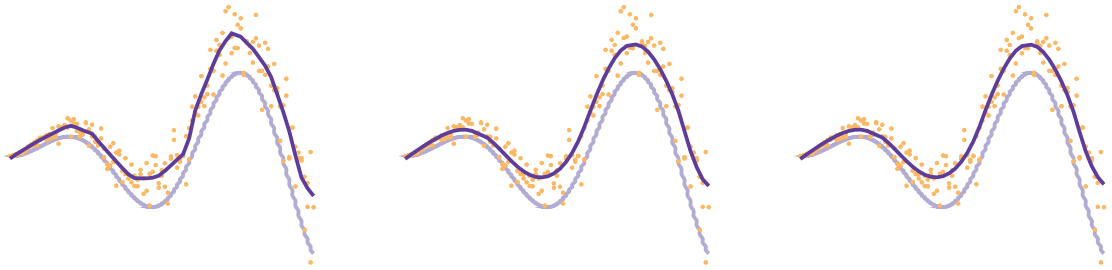
As one can see in Figure 17, one can notice that all models produced feasible estimated functions. However, the ones that somewhat employed regularization, namely, Rank-MLM and LW-MLM, were the ones with proper results and less over-fitting. Such an analysis is critical because if we only use RMSE or R^2 scores to assess such models, we would not easily identify the over-fitted models. An interesting fact regarding Random-MLM is that the random selection of RPs somewhat caused less complex results, acting as a regularization factor in the model.

Figure 17 – Artificial data set I.

(a) Full-MLM. (b) Random-MLM. (c) Rank-MLM.



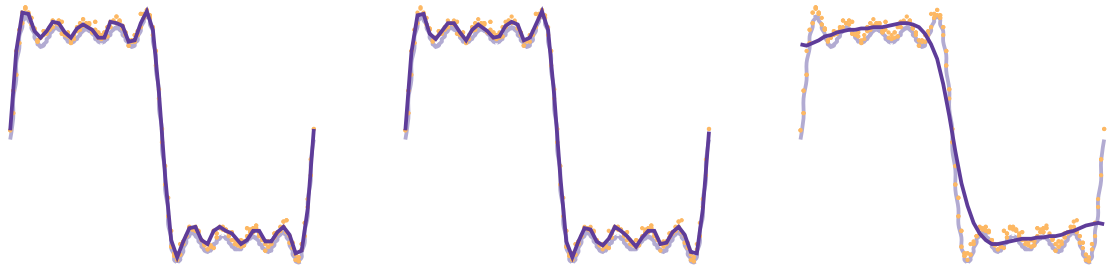
(d) LW-MLM 1. (e) LW-MLM 2. (f) LW-MLM 3.



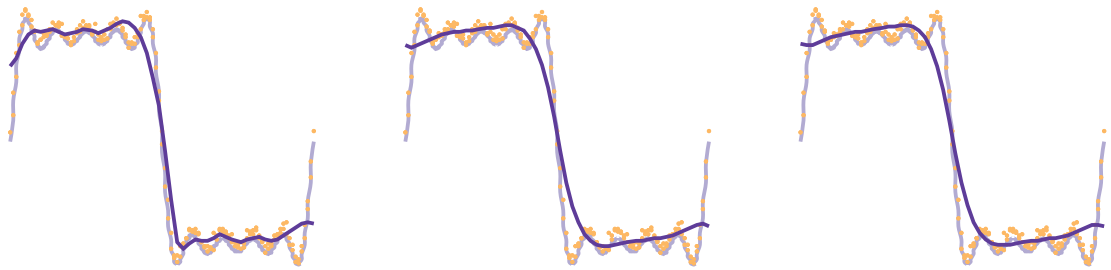
Source – Florencio V *et al.* (2020).

Figure 18 – Artificial data set II.

(a) Full-MLM (b) Random-MLM. (c) Rank-MLM



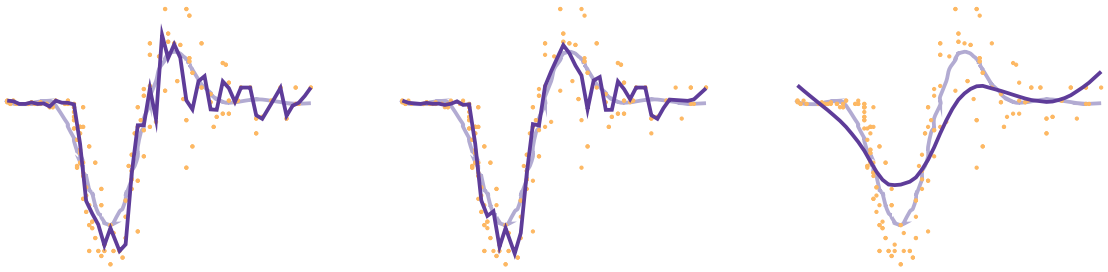
(d) LW-MLM 1. (e) LW-MLM 2. (f) LW-MLM 3.



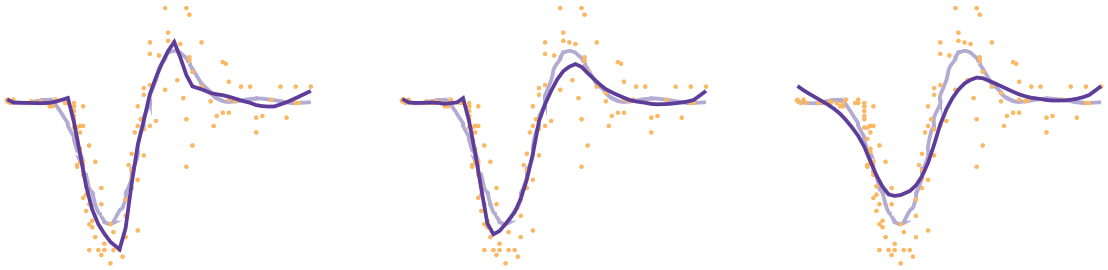
Source – Florencio V *et al.* (2020).

Figure 19 – Mcycle data set.

(a) Full-MLM. (b) Random-MLM. (c) Rank-MLM.



(d) LW-MLM 1. (e) LW-MLM 2. (f) LW-MLM 3.



Source – Florencio V *et al.* (2020).

As for the second artificial data set, presented in Figure 18, one can notice that the regularization variants, namely, Rank-MLM and LW-MLM, impaired the model performance, especially near the "wave" patterns. However, they kept the overall behavior of the learned functions.

Finally, regarding the Mcycle data set depicted Figure 19, one can notice a case where both regularization and overfit add up difficulties regarding the learning process. Both Full-MLM and Random-MLM presented aspects of overfitting because of the heteroscedasticity scenario. Thus, adopting all (Full-MLM) or some (Random-MLM) RPs is not enough, indicating the need for regularization.

Concerning the regularized variants, namely, Rank-MLM and LW-MLM, one can notice smoother functions. However, the homoscedasticity behavior embedded into Rank-MLM did not achieve a proper fit in contrast to LW-MLM variants. From such a finding, we truly support that such a regularization approach alongside the heteroscedasticity is beneficial to the model.

4.3.4 The relevance of RPs during out-of-sample prediction

As one of the pillars in LW-MLM is the *lightweight* pillar, which we support that it learns the whole known geometric structure of the data itself, thus, encouraging us to discard some RPs in the out-of-sample prediction. In this experiment, we analyze how such a discard influences the error in LW-MLM.

Additionally, since we are focusing on regression tasks whose output $\in \mathbb{R}$, we need at least two RPs due to the multilateration setting in MLM. Thus, we vary the quantity of RPs from 2 points and, then, we increase it by a step of multiples of 5% of the actual data set size to examine how the model behaves regarding the RMSE.

In each plot, see Figures 20 and 21, we show the RMSE for 30 independent runs (green points) and the connected (blue) ones represent the average for the 30 runs. Also, we highlight that according to the quantity of RPs, one can see LW-MLM acting as Random-MLM or even Rank-MLM for values between 5% and 50%. When it reaches 100%, one can consider LW-MLM as Full-MLM but taking into account some regularization.

4.3.5 LW-MLM performance at high dimension data sets

This experiment evaluates how LW-MLM deals with some high dimension data sets in comparison with other regressors. Here, the primary goal is to verify how the high dimension affects the error. We pretty much performed the same typical black-box assessment presented in subsection 4.3.2 on three data sets in four settings (The residential data set has two outputs; thus, we split it into two sets), see Table 14. However, we did employ some other regressors from the literature, namely, the k-Nearest Neighbor Regressor (*k*NN), the Random Forest (RF), and the Support Vector Regressor (SVR).

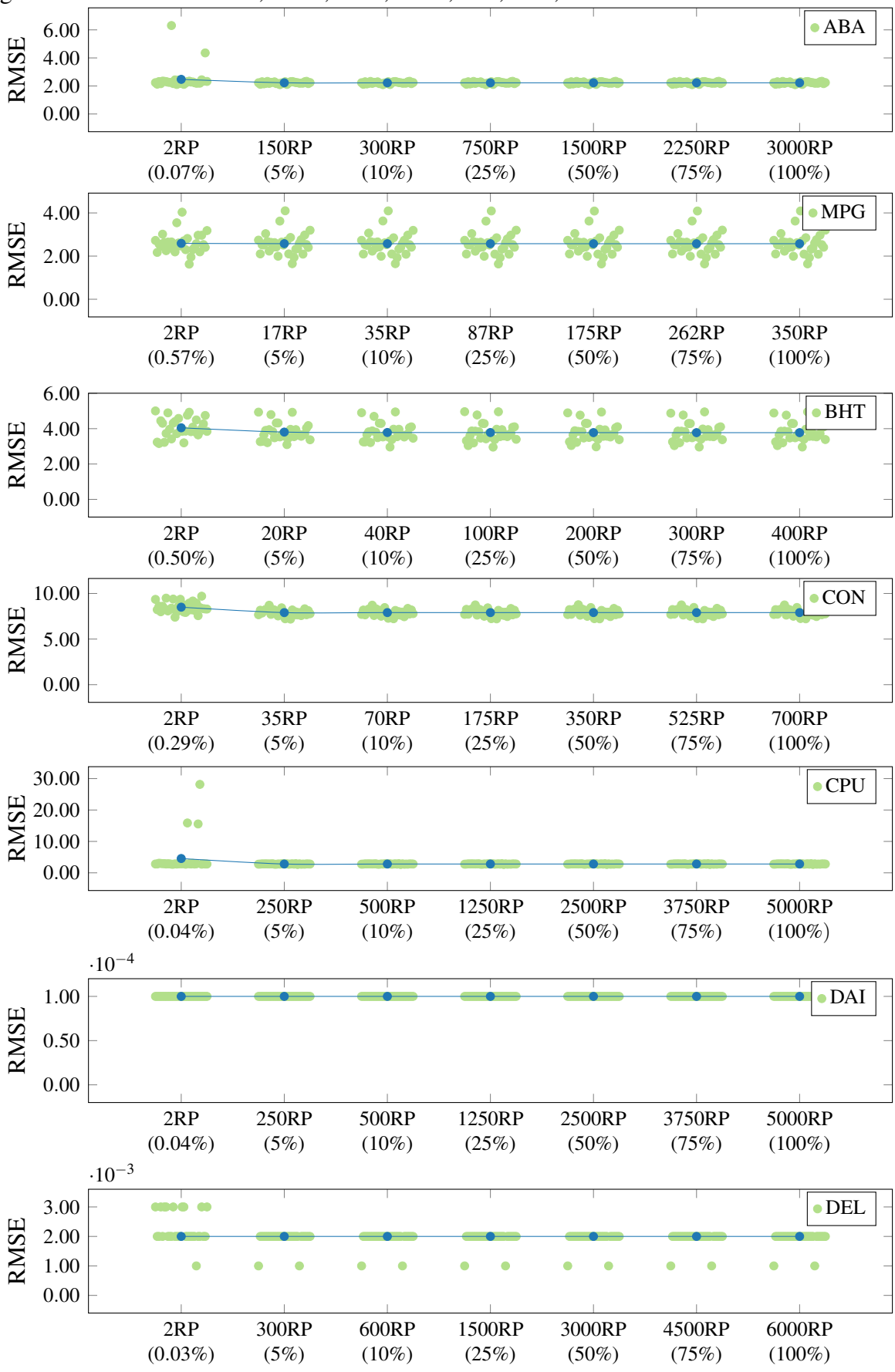
Table 14 – High dimension data set descriptions.

data set	**ACRONYM**	**# DIM**	**# TR**	**# TE**
Residential (output: Sales price)	RS	105	297	75
Residential (output: Construction price)	RC	105	297	75
Communities Crime	CC	147	1772	443
Riboflavin	RB	4088	50	21

Source – Florencio V *et al.* (2020).

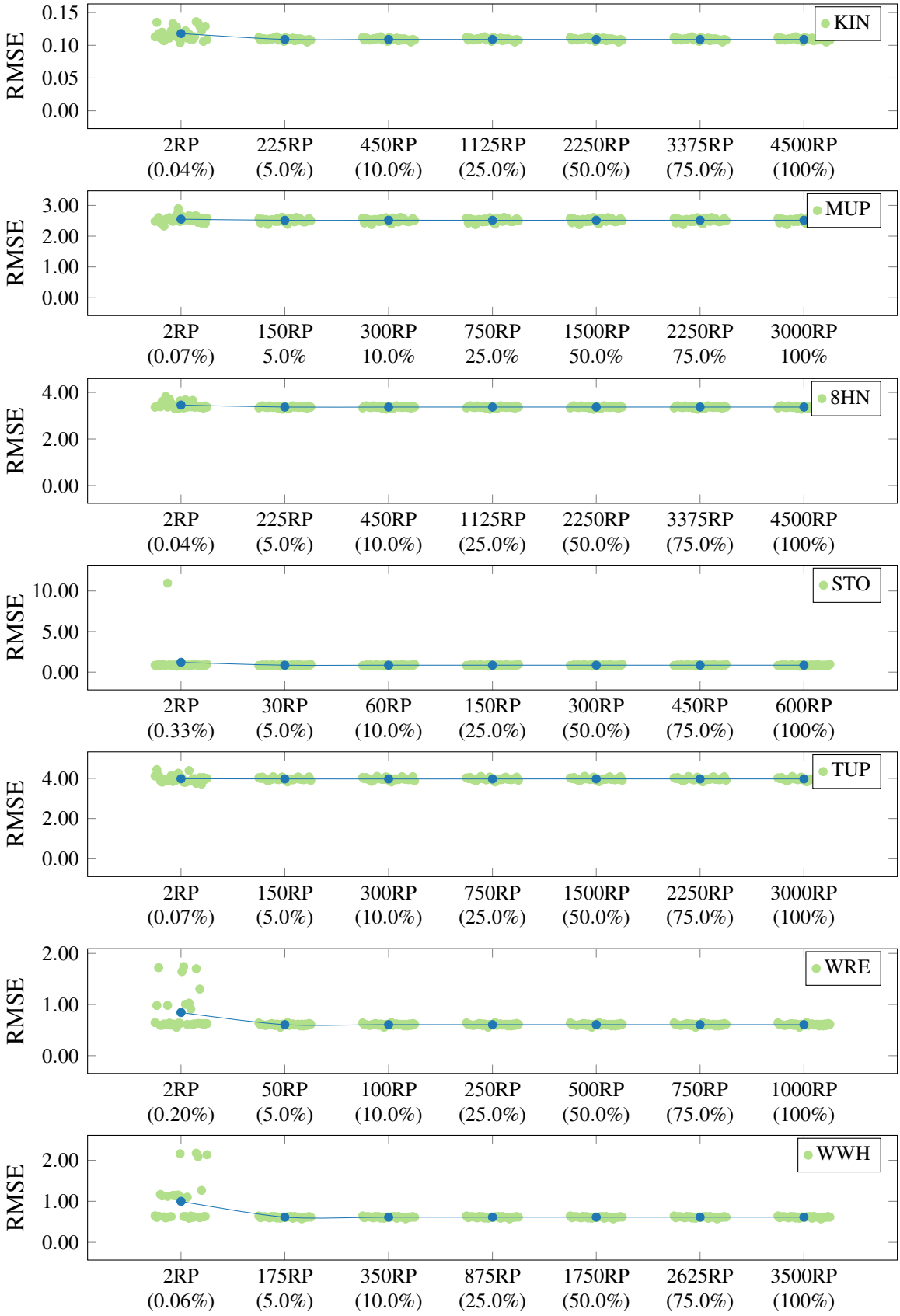
As one can see in Table 16, we reported the RMSE and R^2 metrics regarding the black-box experiments for all data sets presented in Table 14. In such an experiment, we

Figure 20 – RMSE for ABA, MPG, BHT, CON, CPI, DAI, and DEL data sets.



Source – Florencio V *et al.* (2020).

Figure 21 – RMSE for KIN, MUP, 8NH, STO, TUP, WRE, and WWH data sets.



Source – Florencio V *et al.* (2020).

Table 15 – Hyperparameters' space in the high dimension data set experiment.

MODEL	DESCRIPTION	HYPERPARAMETER SPACE
RF	d: the maximum depth of the tree.	$d \in \{10, 30, 80, 100\}$.
	f: Max features.	$f \in \{2, 3\}$
	l: Minimum number of leaf node samples.	$l \in \{3, 4, 5\}$
	s: Number of minimum samples to split.	$s \in \{8, 10, 12\}$
	n: Number of trees in the forest.	$n \in \{100, 200, 300\}$
SVR	C: Regularization parameter.	$C \in \{10^i\}_{i=0}^{3}$
	σ: RBF kernel coefficient.	$\sigma \in \{10^i\}_{i=-2}^{2}$
kNN	k: Number of nearest neighbors to compute distance.	$k \in \{3, 5, 9, 13, 15, 25, 40\}$.

Source – Florencio V *et al.* (2020).

Table 16 – Black-box experiment results for RMSE and R^2 in high dimension data sets.

		FULL-MLM	RF	kNN	SVR	LW-MLM-1	LW-MLM-2	LW-MLM-3
RMSE	RS	**318.89**	736.70	396.97	575.75	**378.09**	**378.35**	**381.64**
	RC	**47.34**	84.26	89.38	**39.46**	52.44	51.82	53.20
	CC	**871.78**	1558.03	1859.21	1243.90	985.36	935.71	989.50
	RB	**0.59**	0.83	0.80	0.74	**0.59**	**0.59**	**0.59**
R^2	RS	**0.93**	0.61	0.53	0.77	**0.90**	**0.90**	**0.90**
	RC	**0.92**	0.73	0.70	**0.94**	**0.90**	**0.90**	**0.90**
	CC	**0.90**	0.66	0.51	0.78	**0.88**	**0.89**	**0.88**
	RB	**0.67**	0.15	0.18	0.32	**0.67**	**0.67**	**0.67**

Source – Florencio V *et al.* (2020).

noticed that models that rely on regular distance computations, namely, kNN and Random Forest, performed worse in these high dimensional data sets. Simultaneously, the LW-MLM variants mirrored the best models in both RMSE and R^2, achieving similar results. Such a finding is exciting because it shows that such a new formulation can also deal with high dimension data sets.

4.3.6 Computational complexity analysis of MLM variants from the experiments

To simplify our notation, we adopt M as the number of RPs used in the out-of-sample prediction, while N is the number of samples in the data set. Since we employ all data set into the training step, the memory used is in the order of $\mathcal{O}(N)$ for both **D** and **P**, and the linear system solution is given by $\mathcal{O}(N^3)$ which is pretty much the same as the Full-MLM. However, the out-of-sample prediction step has less cost because fewer RPs are employed (see subsection 4.3.4), thus, resulting in $\mathcal{O}(M)$ where M is the number of RPs used in the out-of-sample prediction.

Furthermore, it is worth mentioning that we neglected the cost of building **P** and the optimization process since they are asymptotically smaller than the system resolution.

Table 17 – MLM variant comparison in terms of memory requirement, training cost, and out-of-sample prediction cost. We recall that N stands for the cardinality of the training set, while M is the number of RPs ($N \geq M$).

MODEL	MEMORY	TRAINING	PREDICTION
Full-MLM	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$	$\mathcal{O}(N)$
Random-MLM	$\mathcal{O}(NM)$	$\mathcal{O}(NM^2)$	$\mathcal{O}(M)$
Rank-MLM	$\mathcal{O}(NM)$	$\mathcal{O}(NM^2)$	$\mathcal{O}(M)$
LW-MLM	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3)$	$\mathcal{O}(M)$

Source – Florencio V *et al.* (2020).

As one can see in Table 17, LW-MLM is a competitive variant since it kept polynomial costs concerning memory, training, and prediction costs. Although LW-MLM has a similar cost to the Full-MLM during training, it dramatically reduced out-of-sample prediction cost, staying similar to the other variants. Such finding is exciting because when combined with higher/similar error rates, RPs quantity, and norm scores against other MLM variants, it suggests that such a formulation is beneficial for the MLM model.

4.4 Concluding remarks on LW-MLM

This Chapter presented a novel formulation for the MLM model based on a regularization matrix named Lightweight Minimal Learning Machine (LW-MLM). We mainly extend the definition of the cost function in the original version of MLM and then embed a speeded-up procedure into the out-of-sample prediction.

We validate LW-MLM through four types of experiments, evaluating different aspects: prediction error, the goodness-of-fit of estimated vs. measured values, the model complexity (via norm of matrix **B**), the influence of the RPs number in the out-of-sample predict, and, finally, the performance in high dimensional data sets.

In the first set of experiments, we assessed LW-MLM and some MLM variants through the prediction error and the goodness-of-fit of estimated vs. measured values in a typical black-box test fashion on some data sets. In this typical black-box assessment, LW-MLM achieved similar or higher accuracy rates against other variants, all with low variance and seen as statistically equivalent.

Next, in the second set of experiments, we visually assessed the MLM variants on

three regression problems. From such an empirical analysis, we support that some problems can take advantage of the regularization approach due to such a formulation in LW-MLM. At the same time, other models do not perform precisely.

In the third set of experiments, in which we addressed the relevance of RPs during prediction, we highlighted the stability property of LW-MLM concerning such a selection even when fewer points are employed. Through the results, we noticed the *lightweight* aspect in LW-MLM is beneficial for the MLM model since it kept it a competitive variant with a polynomial cost concerning memory, training. Additionally, it dramatically reduced out-of-sample prediction cost while staying similar to the other variants regarding the error prediction.

Finally, we investigated how LW-MLM deals with high-dimensional data sets in the last set of experiments. LW-MLM variants mirrored the best models, in both RMSE and R^2, achieving similar results to some State-of-the-art models.

4.4.1 Shortcomings

Since we do not discard any sample during training, the memory and training cost might be prohibitive for some problems, as shown in Table 17, they are $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$, respectively. Also, we again have to deal with the effects of the curse of dimensionality because the distance matrices, namely, $\mathbf{D}$ and $\mathbf{\Delta}$, have meaningless distances as the dimension of the data increases significantly.

4.4.2 Future works

Nowadays, we are exploring other formulations for LW-MLM as a means to improve the current approach. Also, we are investigating manners to deal with the shortcomings that affect its performance, such as the choice of appropriate values for $\mathbf{P}$, especially for large data sets, and metric learning strategies during training as well.

5 REDUCING COMPLEXITY BY CLASS-CORNER NEARNESS

Our first attempt to deal with learning models' complexity was to explore the instance selection perspective through CCIS. Later, we introduced the Lightweight framework for MLM, reducing model complexity and speeding up the out-of-sample prediction.

This chapter introduces the class-corner concept (via CCIS) to derive a novel LW-MLM variant. We name such a derivation that embodies CCIS into LW-MLM as the Class-Corner Lightweight Minimal Leaning Machine (CCLW-MLM). To achieve a *Lightweight* fashion for MLM, we present a way to produce **P** via CCIS by regularizing samples' closeness concerning the class corners. Such a proposition results in a model in which the class-corner samples will have higher regularization costs, while those far from the corners will be less penalized.

5.1 Measuring class-corner nearness

Given a datset $\mathcal{D}$ and a class-corner set $\mathcal{PS}$, such that $\mathcal{PS} \subset \mathcal{D}$, we then define the Nearest Class-Corner Distance NCD$(\cdot)$ of a given sample $\mathbf{x}$ as:

$$\text{NCD}(\mathbf{x}) = \min \left\{ ||\mathbf{x} - \mathbf{x}_j||_2 \right\}, \forall \mathbf{x}_j \in \mathcal{PS}. \tag{5.1}$$

Next, we derive a distance factor, named class-corner nearness, as a regularization cost for each sample concerning the closest class-corner. To do so, we must define the maximum cost by class-corner nearness as the maximum distance of a query sample to the class-corners for all samples available, that is,

$$\zeta = \max \left\{ \text{NCD}(\mathbf{x}_i) \right\}, \forall \mathbf{x}_i \in \mathcal{D}, \tag{5.2}$$

because it acts as an upper bound so that finally, we derive the cost by class-corner nearness of a given sample $\mathbf{x}$ as:

$$\varsigma(\mathbf{x}) = \zeta - \text{NCD}(\mathbf{x}). \tag{5.3}$$

The reasoning behind such a proposition results in the class-corner samples will have higher regularization costs, while it penalizes less the samples far from the corners.

5.2 Class-Corner-Nearness-Aware Minimal Learning Machines

To derive a LW-MLM variant, we explore the suitability of employing the above-mentioned class-corner nearness to produce **P** for LW-MLMs. We name such a derivation as the

Class-Corner Lightweight Minimal Leaning Machine (CCLW-MLM) and in the following, we present how the learning and out-of-sample prediction works.

5.2.1 Learning algorithm and out-of-sample prediction for CCLW-MLM

Both learning algorithms and out-of-sample prediction in CCLW-MLM follow the same framework presented in LW-MLM. However, the MLM model takes advantage of the known output space, i.e., the already known labels, in classification tasks. Therefore, one can simply avoid the multilateration procedure during prediction by merely replacing it by directly applying the known labels into the cost function. Firstly, let us define $\mathcal{Y}^{\star}$ as the set with known labels (encoded in One-Hot-Encoding) such that

$$\mathcal{Y}^{\star} = \left\{\mathbf{y}^{(i)}\right\}_{i=1}^{C}, \text{ such that } \mathbf{y}^{(i)} = \left[\mathbb{1}[i=1], \mathbb{1}[i=2], \ldots, \mathbb{1}[i=C]\right] \tag{5.4}$$

then, we rewrite the out-of-sample prediction as:

$$h(\mathbf{x}) = \underset{\mathbf{y}^{(i)} \in \mathcal{Y}^{\star}}{\arg\min} \parallel \Psi\left(\mathbf{y}^{(i)}\right) - \Phi(\mathbf{x})\,\mathbf{B} \parallel_2 . \tag{5.5}$$

We present the learning algorithm for CCLW-MLM in Algorithm 12:

Algorithm 12 The CCLW-MLM learning algorithm

CCLW-MLM-TRAINING($\mathcal{D}$)

▷ Step 1: Computing the class-corners.
1 $\mathcal{PS} \leftarrow \text{CCIS}(\mathcal{D}, 0)$.
▷ Step 2: Computing the class-corner nearness cost.
2 $N \leftarrow |\mathcal{D}|$.
3 $\zeta \leftarrow \max\left\{\text{NCD}(\mathbf{x}_i)\right\}, \forall \mathbf{x}_i \in \mathcal{D}$.
4 $\mathbf{p} \leftarrow \text{Zeros}(1, N)$.
5 $p_i \leftarrow \zeta - \text{NCD}(\mathbf{x}_i)$.
6 $\mathbf{P} \leftarrow \text{diag}(\mathbf{p})$.
▷ Step 3: Estimating the system solution.
7 **return** LW-MLM-TRAINING($\mathcal{D}, \mathbf{P}$).

5.3 Experiments and Discussion for CCLW-MLM

This section presents the experimental framework followed in this work, together with the results collected and discussions on them.

5.3.1 Experiment Setup for CCLW-MLM

We carried out two types of experiments to evaluate different aspects of our proposal. Our goal is to investigate how CCLW-MLM behaves with respect to the following aspects: accuracy and sparseness[1]. Each of the following experiments addresses such concerns. Moreover, we highlight all experiments were performed using a Mac Mini with an 3.6 GHz Intel Core i3 Quad-Core, 8 GB of RAM, and running macOS Catalina 10.15.6 with Python 3.7.3.

a) **The typical black-box assessment on some data sets**: Here, we are interested in assessing CCLW-MLM against some of the state-of-the-art MLM variants that employ regularization through accuracy and sparseness in a typical black-box test fashion on the same *toy-size* data sets presented in subsection 3.3.2.

b) **Empirical decision boundary quality assessment**: This experiment investigates visually the quality of solutions produced by the MLM variants that employ regularization by empirically evaluating the decision boundaries.

5.3.2 Typical black-box assessment for CCLW-MLM

Here, we employ the same data sets from Table 3 and compared CCLW-MLM against the Full-MLM, Rank-MLM, and Random-MLM. All variations but Random-MLM arise from regularized strategies into the model formulation that have a unique solution. Thus, we employ them for comparison since our proposal is also based on a similar formulation. In this comparison, we report the average accuracy (ACC) and sparseness via $||\mathbf{B}||_{\mathcal{F}}$ over 30 independent realizations. The hyperparameter setting for each variation, including ours, is presented in Table 18. We report the average results for 30 independent realizations for these 10 toy-size data sets in Table 19. Also, to better distinguish the MLM variations comparison, we carried out the Friedman test evaluating both ACC and $||\mathbf{B}||_{\mathcal{F}}$ scores and showed the results graphically through the Critical difference plot adopting the significance level of $\alpha = 0.05$ in Figure 22.

As one can see in Table 19, we noticed that all models scored very close values regarding the ACC, while the most stand out result was the one related to the $||\mathbf{B}||_{\mathcal{F}}$, in which Rank-MLM and CCLW-MLM outperformed the others. Such a finding is exciting because it shows that such a new formulation did not sacrifice the generalization performance.

With respect to the Critical difference plots depicted in Figure 22, one can see that

[1] Again, we adopted the scaled between 0 and 1 by subtracting 1 from division by the $\| \mathbf{B} \|_{\mathcal{F}}$ of Full-MLM (that is $\text{NORM}(x) = 1 - x\| \mathbf{B} \|_{\mathcal{F}}{}^{-1}$) since it acts as an upper bound.

Table 18 – MLM variants and their hyperparameters configurations.

MODEL	DESCRIPTION	HYPERPARAMETER SPACE
Full-MLM	None.	None.
Random-MLM	K: Number of RPs chosen randomly from a range through grid search and random cross-validation.	$k \sim \mathcal{U}(0.05, 0.5)$ so that $K = \lfloor k \times N \rfloor$.
Rank-MLM	C : Regularization parameter optimized by grid search and cross-validation.	$\log(C) \sim \mathcal{U}(\log(1e-3), \log(1e2))$.
CCLW-MLM	**P**: Diagonal matrix with random values from the complements of their closeness to the corners in $\mathcal{PS}$.	See Algorithm 12.

Source – Own authorship.

regarding the accuracy all models are seen as equivalents. However, we highlight that the regularized variants, Rank-MLM and CCLW-MLM, appear to be in the best rank. From such a finding, we support that model regularization is indeed advantageous for MLM. On the other hand, when analyzing the norm rank, we see a different perspective. There, we noticed two groups of equivalence: one with Full-MLM, Random-MLM, and Rank-MLM; and the second with only Rank-MLM and CCLW-MLM.

With respect to the Critical difference plots depicted in Figure 22, one find two interesintg findinds. Regarding the accuracy ranking, we noticed two groups of equivalence: one with Full-MLM, Random-MLM, and Rank-MLM; and the second with only Rank-MLM and CCLW-MLM. From such a finding, we support that model regularization is indeed advantageous for MLM since they are placed in best ranks alongside the Random-MLM (low rank system).

On the other hand, when analyzing the norm ranking, all models are seen as equivalents

5.3.3 Empirical decision boundary quality assessment for CCLW-MLM

Here, in this set of experiments, we repeated the same assessment in subsection 3.3.3 with respect to decision boundary analysis, i.e., where $f(x) = 0$, in three data sets. The first one, namely, Two moon (TMN), is a separable problem, while the other two, namely, Ripley (RIP), and Banana (BAN), have low data and moderate class overlapping, respectively.

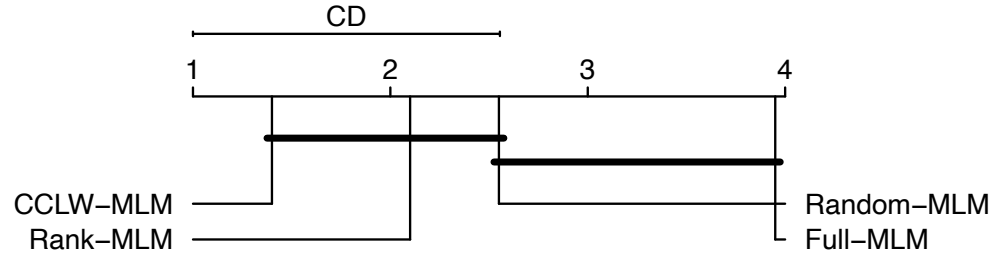
In Figure 23, one can see all variants produced proper decision boundaries able to

Table 19 – Black-box experiment results for ACC and NORM using FULL-MLM, Random-MLM, Rank-MLM, and CCLW-MLM.

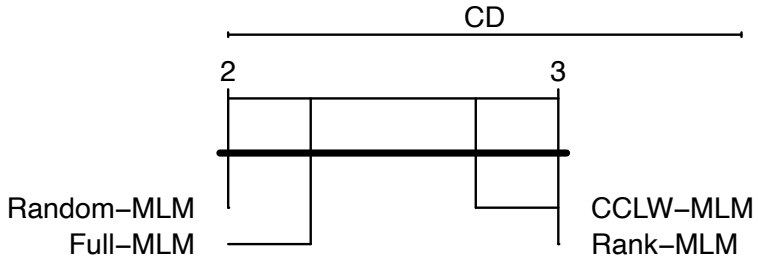
data set	**FULL-MLM**	**Random-MLM**		**Rank-MLM**		**CCLW-MLM**	
	ACC	ACC	NORM	ACC	NORM	ACC	NORM
BAN	0.88 ± 0.00	0.89 ± 0.00	0.86 ± 0.00	0.90 ± 0.00	0.99 ± 0.00	0.90 ± 0.00	0.99 ± 0.00
BCW	0.97 ± 0.00	0.97 ± 0.01	0.77 ± 0.00	0.97 ± 0.00	0.05 ± 0.00	0.96 ± 0.00	0.92 ± 0.00
GER	0.74 ± 0.00	0.74 ± 0.01	0.82 ± 0.00	0.74 ± 0.00	0.43 ± 0.00	0.74 ± 0.00	0.87 ± 0.00
HAB	0.74 ± 0.00	0.74 ± 0.01	0.93 ± 0.00	0.76 ± 0.00	0.99 ± 0.00	0.75 ± 0.00	0.97 ± 0.00
HEA	0.82 ± 0.00	0.82 ± 0.01	0.74 ± 0.00	0.81 ± 0.00	0.83 ± 0.00	0.82 ± 0.00	0.72 ± 0.00
ION	0.91 ± 0.00	0.88 ± 0.03	0.56 ± 0.00	0.91 ± 0.00	0.30 ± 0.00	0.90 ± 0.00	0.76 ± 0.00
PID	0.72 ± 0.00	0.74 ± 0.01	0.73 ± 0.00	0.74 ± 0.00	0.78 ± 0.00	0.76 ± 0.00	0.94 ± 0.00
RIP	0.88 ± 0.00	0.89 ± 0.00	0.91 ± 0.00	0.89 ± 0.00	0.99 ± 0.00	0.89 ± 0.00	0.99 ± 0.00
TMN	1.00 ± 0.00	0.96 ± 0.03	0.00 ± 0.00	1.00 ± 0.00	0.40 ± 0.00	1.00 ± 0.00	0.47 ± 0.00
VCP	0.79 ± 0.00	0.82 ± 0.02	0.75 ± 0.00	0.83 ± 0.00	0.68 ± 0.00	0.83 ± 0.00	0.91 ± 0.00

The values regarding the standard deviation are not zero, but very small ones.
Source – Own authorship.

Figure 22 – Critical difference plots with respect to the accuracy rankings (a) and the sparsity rankings (b) from Table 5. We recall that those variants which are not joined by a bold line can be regarded as different.



(a) Accuracy rankings.



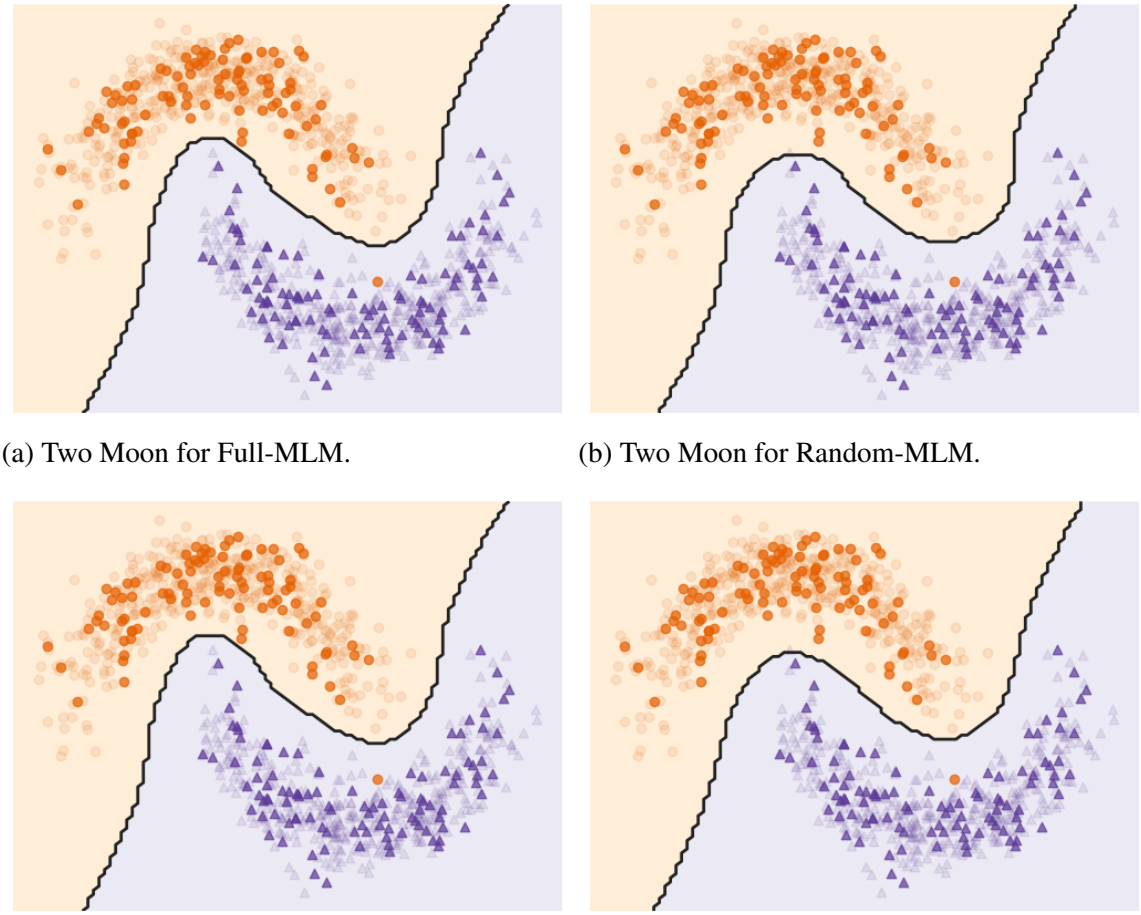
(b) NORM rankings.

Source – Own authorship.

separate the data. Perhaps, the most prominent feature from this setting is the decision boundary in CCLW-MLM, where we can see that it is very close to the samples. However, as shown in Table 19, CCLW-MLM achieved maximum accuracy, i.e., (100%) for this data set.

Regarding the Ripley data set depicted in Figure 24, we highlight that, again, all MLM variants produced satisfactory decision boundaries even though there is a little class overlapping and presence of some outliers in the data. However, the most notable trait from such a data set is all other MLM variants tried to include the left upper part from the bottom set, but CCLW-MLM.

Figure 23 – Results for Two Moon data set.



(a) Two Moon for Full-MLM.

(b) Two Moon for Random-MLM.

(c) Two Moon for Rank-MLM.

(d) Two Moon for CCLW-MLM.

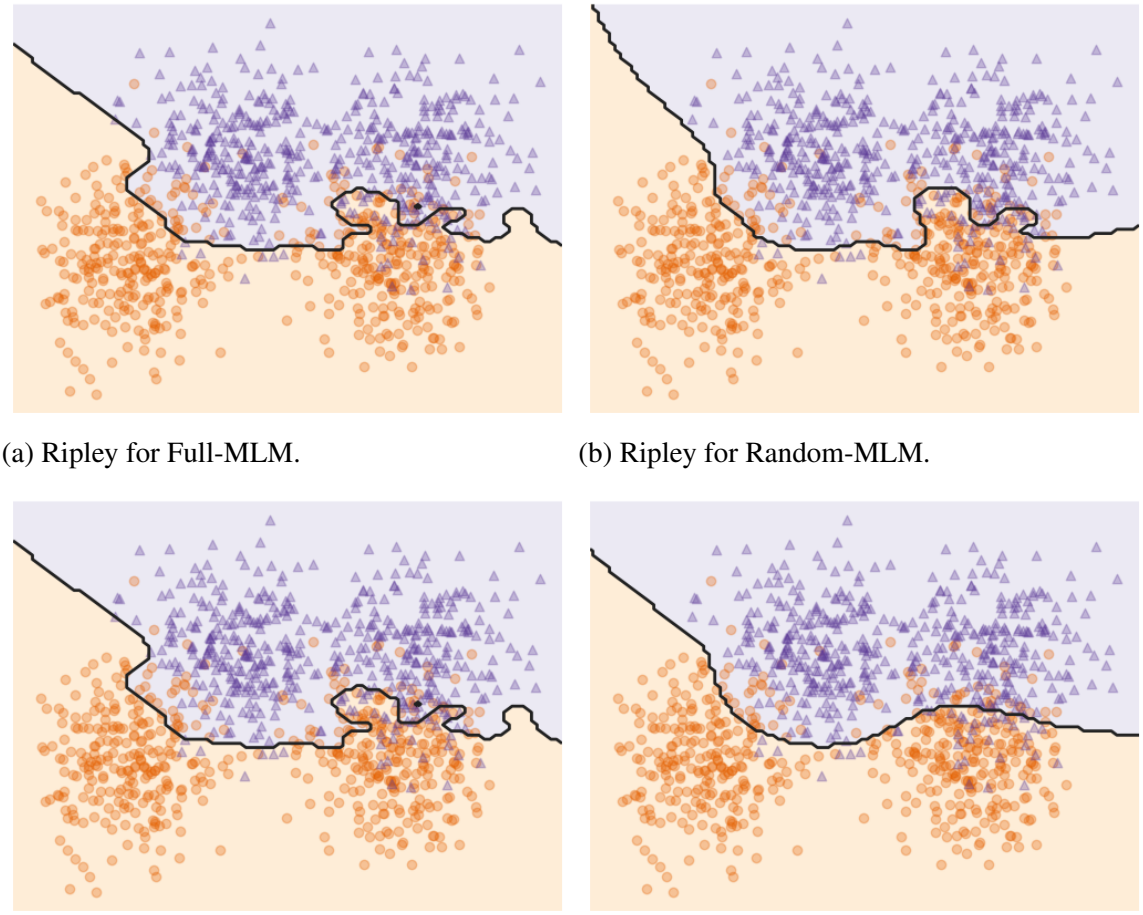
Source – Own authorship.

Also, from a hyperparameter optimization perspective, we noticed that due to the hyperparameter space in Rank-MLM, sometimes the Rank-MLM chooses to select $C \approx 0$ to achieve better results regarding accuracy. In this setting, the Rank-MLM behaves similarly to Full-MLM.

As for the final data set, Banana, depicted in Figure 25, one can see that the models produced similar decision boundaries. Additionally, due to the severe overlapping between classes right at the top, one can notice that some overfitting occurred in most models. However, CCLW-MLM has shown the less complex decision boundary among all.

From such an empirical analysis, we noticed that CCLW-MLM is the one MLM variant that achieved less elaborate decision boundaries. Also, as the overlapping between classes increases, see Ripley and Banana (Figures 25 and 24), CCLW-MLM has shown its good ability for generalization. Finally, when it comes to problems that require a less complex decision boundary (as the Two Moon, depicted in Figure 23), the CCLW-MLM resulting model decision

Figure 24 – Results for Ripley data set.



(a) Ripley for Full-MLM.

(b) Ripley for Random-MLM.

(c) Ripley for Rank-MLM.

(d) Ripley for CCLW-MLM.

Source – Own authorship.

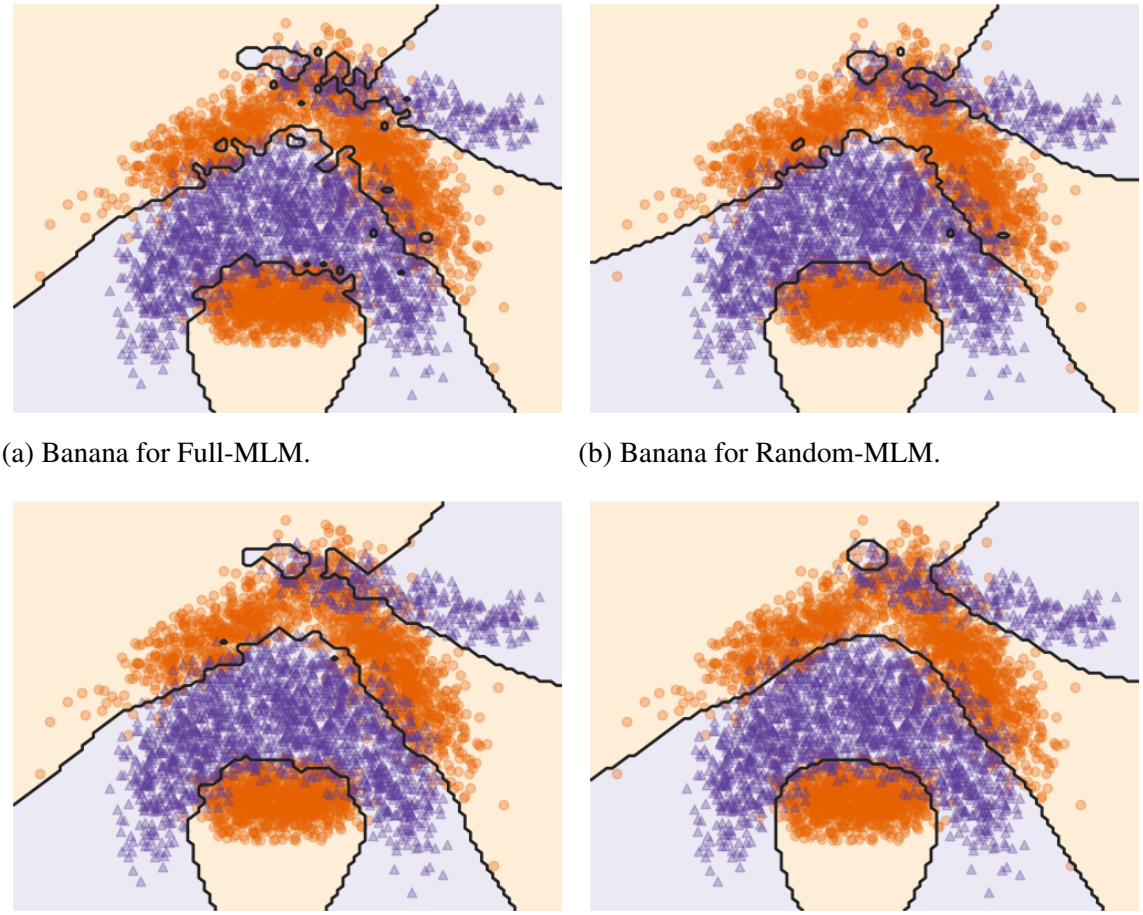
boundary might be too fit for the data, but still achieving good generalization performance.

Moreover, another perspective realized previously in Figure 19 and now in Figures 24 and 25 when analyzing the results from LW-MLM and Rank-MLM suggests that regularized solutions, achieved through similar constrained hypothesis spaces, provide suitable fits to some functions. Even though both LW-MLM and Rank-MLM using the same regularization strategy, LW-MLM is more flexible than Rank-MLM. Such a finding is because Rank-MLM can be seen as a particular case of LW-MLM when adopting $\mathbf{P} = \lambda \mathbf{I}$.

5.4 Concluding remarks on CCLW-MLM

In this chapter, we revisited the *Lightweight* Minimal Leaning Machine and formulated a variant for classification tasks by employing CCIS. We named it Class-Corner Lightweight Minimal Leaning Machine (CCLW-MLM). A key point in CCLW-MLM is that regularization arises from cost by class-corner nearness, a concept we defined before having the final model.

Figure 25 – Results for Banana data set.



(a) Banana for Full-MLM.

(b) Banana for Random-MLM.

(c) Banana for Rank-MLM.

(d) Banana for CCLW-MLM.

Source – Own authorship.

After we carried out some experiments, we noticed such a new formulation did not sacrifice the generalization performance while keeping higher sparsity scores (less complexity), achieving higher ranks than other variants that also employ some regularization. Thus, becoming a desirable formulation to deal with classification tasks.

5.4.1 Shortcomings

The class-corner nearness measure highly relies upon distance computations. Once again, we recall the effects of the curse of dimensionality because the Euclidean distance becomes meaningless as the dimension of the data increases significantly. Also, because we do not discard any sample during training, the memory and training cost might be prohibitive for some problems. Thus, some other strategies regarding instance selection might take place in the formulation.

5.4.2 Future work

As future work, we are focusing on employing the class-corner nearness measure into other distance-based classifiers and investigating the aspects of dealing with low-rank systems in LW-MLM.

6 CONCLUDING REMARKS

The contribution presented in this thesis is four-part and investigated the model complexity reduction in two Instance-based learners, namely: The Least-Squares Support vector machine and the Minimal Learning Machine.

The common idea behind all the solutions is to reduce the complexity in Instance-based learners from instance selection, treating it as a regularization task. Thus, excluding our first contribution, an instance selection algorithm itself, we modified the design of such LSSVM and MLM algorithms to embed such a complexity reduction.

Chapter 3 presented the first well-succeded attempt to reduce the LSSVM complexity by selecting class-corner data points. Based on FAST, an image processing algorithm for corner detection, this thesis's first contribution is formulated and named Class Corner Instance Selection (CCIS). It deals with the instance selection problem by choosing data points near the boundary of classes. From that, we extend a pruned and reduced set LSSVM model, after named CC-LSSVM.

In Chapter 4, we dealt with reducing complexity in MLMs after applying regularization. Such a formulation derived our third contribution, named Lightweight Minimal Learning Machine, which took advantage of such a strategy to learn in a restricted hypothesis space and generate a faster model for predicting regression tasks.

Chapter 5 revisited the LW-MLM and formulated this thesis's final contribution by combining CCIS to LW-MLM. We named it Class-Corner Lightweight Minimal Leaning Machine because it deals with classification tasks straightforwardly.

We carried out some experiments to evaluate each contribution's different aspects, investigating how they behave concerning the following aspects: the prediction error, the goodness-of-fit of estimated vs. measured values, the model complexity, the influence of the parameters, and the learned models' empirical visual analysis.

Even though our contributions strongly rely on distance computations, thus being suffering from the Dimensionality curse, they consistently outperformed the other models in artificial and real-world scenarios. This thesis's apparent unfolding is to directly apply metric learning methods to derive more algorithms with consistent hypotheses.

REFERENCES

AGGARWAL, C. **Data classification**: algorithms and applications. London, United Kingdom: Taylor & Francis, 2014. (Chapman & Hall/CRC Data Mining and Knowledge Discovery Series). ISBN 9781466586741. Available at: https://books.google.com.br/books?id=gJhBBAAAQBAJ.

AHA, D.; KIBLER, D.; ALBERT, M. Instance-based learning algorithms. **Machine Learning**, 6, n. 1, p. 37–66, 1991. ISSN 0885-6125.

ALENCAR, A. S. C.; CALDAS, W. L.; GOMES, J. P. P.; SOUZA JR., A. H.; AGUILAR, P. A. C.; RODRIGUES, C.; FRANCO, W.; CASTRO, M. F. de; ANDRADE, R. M. C. MLM-Rank: a ranking algorithm based on the minimal learning machine. In: IEEE. **2015 Brazilian Conference on Intelligent Systems (BRACIS 2015)**. Rio Grande do Norte, Brazil, 2015. p. 305–309. ISBN 978-1-5090-0016-6.

ANDO, R.; ZHANG, T. A framework for learning predictive structures from multiple tasks and unlabeled data. **Journal of Machine Learning Research**, 6, p. 1817–1853, 2005. ISSN 1532-4435.

ANGIULLI, F. Fast nearest neighbor condensation for large data sets classification. **IEEE Transactions On Knowledge And Data Engineering**, 19, n. 11, p. 1450–1464, 2007.

AYINDE, B. O.; INANC, T.; ZURADA, J. M. Regularizing deep neural networks by enhancing diversity in feature extraction. **IEEE Transactions on Neural Networks and Learning Systems**, 30, n. 9, p. 2650–2661, 2019. ISSN 2162-237X.

BELKIN, M.; HSU, D.; MA, S.; MANDAL, S. Reconciling modern machine-learning practice and the classical bias-variance trade-off. **Proceedings of the National Academy of Sciences of the United States of America**, 116, n. 32, p. 15849–15854, 2019. ISSN 0027-8424.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York, USA: Springer, 2016. (Information Science and Statistics). ISBN 9781493938438.

BISHOP, C. M.; BISHOP, P. N. C. C. M.; HINTON, G.; PRESS, O. U. **Neural Networks for Pattern Recognition**. New York, USA: Clarendon Press, 1995. (Advanced Texts in Econometrics). ISBN 9780198538646.

BRABANTER, K. D.; BRABANTER, J. D.; SUYKENS, J. A. K.; MOOR, B. D. Optimized fixed-size kernel models for large data sets. **Computational Statistics & Data Analysis**, 54, n. 6, p. 1484–1504, 2010. ISSN 0167-9473.

CALDAS, W. L.; GOMES, J. P. P.; MESQUITA, D. P. P. Fast Co-MLM: an efficient semi-supervised co-training method based on the minimal learning machine. **New Generation Computing**, 36, n. 1, p. 41–58, 2018. ISSN 0288-3635.

CARVALHO, B. P. R.; BRAGA, A. P. IP-LSSVM: a two-step sparse classifier. **Pattern Recognition Letters**, 30, n. 16, p. 1507–1515, 2009. ISSN 0167-8655.

CAWLEY, G.; TALBOT, N.; FOXALL, R.; DORLING, S.; MANDIC, D. Heteroscedastic kernel ridge regression. **Neurocomputing**, 57, p. 105–124, 2004. ISSN 0925-2312.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, 20, n. 3, p. 273–297, 1995. ISSN 0885-6125.

COTTER, S.; RAO, B.; ENGAN, K.; KREUTZ-DELGADO, K. Sparse solutions to linear inverse problems with multiple measurement vectors. **IEEE Transactions on Signal Processing**, 53, n. 7, p. 2477–2488, 2005. ISSN 1053-587X.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, 13, n. 1, p. 21+, 1967. ISSN 0018-9448.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, 7, p. 1–30, 2006. ISSN 1532-4435.

DIAS, M. L. D.; FREIRE, A. L.; JUNIOR, A. H. S.; ROCHA NETO, A. R. da; GOMES, J. P. P. Sparse minimal learning machines via l(1/2) norm regularization. In: IEEE. **2018 Brazilian Conference on Intelligent Systems (BRACIS 2018)**. São Paulo, Brazil, 2018. p. 206–211. ISBN 978-1-5386-8023-0.

DIAS, M. L. D.; SOUSA, L. S. de; NETO, A. R. da R.; MATTOS, C. L. C.; GOMES, J. P. P.; KÄRKKÄINEN, T. Sparse minimal learning machine using a diversity measure minimization. In: **Proceedings of 27th European Symposium on Artificial Neural Networks**. Bruges, Belgium: ESANN, 2019.

DING, S.; ZHANG, N.; ZHANG, X.; WU, F. Twin support vector machine: theory, algorithm and applications. **Neural Computing & Applications**, 28, n. 11, p. 3119–3130, 2017. ISSN 0941-0643.

DING, S.; ZHANG, X.; AN, Y.; XUE, Y. Weighted linear loss multiple birth support vector machine based on information granulation for multi-class classification. **Pattern Recognition**, 67, p. 32–46, 2017. ISSN 0031-3203.

DONOHO, D.; ELAD, M. Optimally sparse representation in general (nonorthogonal) dictionaries via l(1) minimization. **Proceedings of the National Academy of Sciences of the United States of America**, 100, n. 5, p. 2197–2202, 2003. ISSN 0027-8424.

EBUCHI, F.; KITAMURA, T. Fast sparse least squares support vector machines by block addition. In: **Advances in Neural Networks - ISNN 2017**. Cham, Germany: Springer International Publishing, 2017. p. 60–70. ISBN 978-3-319-59072-1.

FAN, Y.-t.; WU, W.; YANG, W.-y.; FAN, Q.-w.; WANG, J. A pruning algorithm with l(1/2) regularizer for extreme learning machine. **Journal of Zhejiang University-Science C-Computers & Electronics**, 15, n. 2, p. 119–125, 2014. ISSN 1869-1951.

FLORENCIO V, J. A.; OLIVEIRA, S. A. F.; GOMES, J. P. P.; ROCHA NETO, A. R. A new perspective for minimal learning machines: a lightweight approach. **Neurocomputing**, 401, p. 308–319, 2020. ISSN 0925-2312.

FUCHS, J. On sparse representations in arbitrary redundant bases. **IEEE Transactions on Information Theory**, 50, n. 6, p. 1341–1344, 2004. ISSN 0018-9448.

GARCIA, S.; DERRAC, J.; CANO, J. R.; HERRERA, F. Prototype selection for nearest neighbor classification. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 34, n. 3, p. 417–435, 2012.

GEEBELEN, D.; SUYKENS, J. A. K.; VANDEWALLE, J. Reducing the number of support vectors of SVM classifiers using the smoothed separable case approximation. **IEEE Transactions on Neural Networks and Learning Systems**, 23, n. 4, p. 682–688, 2012. ISSN 2162-237X.

GOMES, J. P. P.; MESQUITA, D. P. P.; FREIRE, A.; JÚNIOR, A. H. S.; KÄRKKÄINEN, T. A robust minimal learning machine based on the m-estimator. In: **Proceedings of 25th European Symposium on Artificial Neural Networks**. Bruges, Belgium: ESANN, 2017.

GOMES, J. P. P.; SOUZA JR., A. H.; CORONA, F.; Rocha Neto, A. R. A cost sensitive minimal learning machine for pattern classification. In: Arik, S and Huang, T and Lai, WK and Liu, Q (Ed.). **Neural Information Processing, PT I**. Istanbul, Turkey: Springer International Publishing, 2015. (Lecture Notes in Computer Science, 9489), p. 557–564. ISSN 0302-9743.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge, USA: MIT Press, 2016. (Adaptive Computation and Machine Learning series). ISBN 9780262035613.

GORODNITSKY, I.; RAO, B. Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm. **IEEE Transactions on Signal Processing**, 45, n. 3, p. 600–616, 1997. ISSN 1053-587X.

HAMALAINEN, J.; ALENCAR, A. S. C.; KARKKAINEN, T.; MATTOS, C. L. C.; JUNIOR, A. H. S.; GOMES, J. P. P. Minimal learning machine: theoretical results and clustering-based reference point selection. **Journal of Machine Learning Research**, 21, 2020. ISSN 1532-4435.

HASSIBI, B.; STORK, D.; WOLFF, G. Optimal brain surgeon and general network pruning. In: IEEE. **IEEE International Conference on Neural Networks**. San Francisco, USA, 1993. p. 293–299. ISBN 0-7803-0999-5.

HUESMANN, K.; RODRIGUEZ, L. G.; LINSEN, L.; RISSE, B. The impact of activation sparsity on overfitting in convolutional neural networks. In: **Pattern Recognition. ICPR International Workshops and Challenges**. Milan, Italy: Springer International Publishing, 2021. p. 130–145.

JANKOWSKI, N.; GROCHOWSKI, M. Comparison of instances selection algorithms I. Algorithms survey. In: Rutkowski, L and Siekmann, J and Tadeusiewicz, R and Zadeh, LA (Ed.). **Artificial Intelligence and Soft Computing - ICAISC 2004**. Zakopane, Poland: Springer Berlin Heidelberg, 2004. (Lecture Notes in Artificial Intelligence, 3070), p. 598–603. ISBN 3-540-22123-9. ISSN 0302-9743.

JAYADEVA; KHEMCHANDANI, R.; CHANDRA, S. Twin support vector machines for pattern classification. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 29, n. 5, p. 905–910, 2007. ISSN 0162-8828.

JIAO, L.; BO, L.; WANG, L. Fast sparse approximation for least squares support vector machine. **IEEE Transactions on Neural Networks**, 18, n. 3, p. 685–697, 2007. ISSN 1045-9227.

KAHAKI, S. M. M.; NORDIN, M. J.; ASHTARI, A. H. Contour-based corner detection and classification by using mean projection transform. **Sensors**, 14, n. 3, p. 4126–4143, 2014. ISSN 1424-8220.

KARKKAINEN, T. Extreme minimal learning machine: ridge regression with distance-based basis. **Neurocomputing**, 342, n. SI, p. 33–48, 2019. ISSN 0925-2312.

KARSMAKERS, P.; PELCKMANS, K.; BRABANTER, K. D.; HAMME, H. V.; SUYKENS, J. A. K. Sparse conjugate directions pursuit with application to fixed-size kernel models. **Machine Learning**, 85, n. 1-2, SI, p. 109–148, 2011. ISSN 0885-6125.

KUMAR, M. A.; GOPAL, M. Least squares twin support vector machines for pattern classification. **Expert Systems with Applications**, 36, n. 4, p. 7535–7543, 2009. ISSN 0957-4174.

LECUN, Y.; DENKER, J. S.; SOLLA, S. A. Optimal brain damage. In: TOURETZKY, D. S. (Ed.). **Advances in Neural Information Processing Systems 2**. San Francisco, USA: Morgan Kaufmann Publishers Inc, 1990. p. 598–605.

LI, Y.; LIN, C.; ZHANG, W. Improved sparse least-squares support vector machine classifiers. **Neurocomputing**, 69, n. 13-15, p. 1655–1658, 2006. ISSN 0925-2312.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Available at: http://archive.ics.uci.edu/ml. Accessed on: Jun, 23th 2021.

LIU, Z.; SUN, M.; ZHOU, T.; HUANG, G.; DARRELL, T. Rethinking the value of network pruning. In: **7th International Conference on Learning Representations, ICLR**. New Orleans, USA: OpenReview.net, 2019.

MACKAY, D. Bayesian interpolation. **Neural Computation**, 4, n. 3, p. 415–447, 1992. ISSN 0899-7667.

MALL, R.; SUYKENS, J. A. K. Very sparse LSSVM reductions for large-scale data. **IEEE Transactions on Neural Networks and Learning Systems**, 26, n. 5, p. 1086–1097, 2015. ISSN 2162-237X.

MASSEY, F. J. The kolmogorov-smirnov test for goodness of fit. **Journal of the American Statistical Association**, 46, n. 253, p. 68–78, 1951. ISSN 0162-1459.

MESQUITA, D. P. P.; GOMES, J. P. P.; JUNIOR, A. H. S. Ensemble of efficient minimal learning machines for classification and regression. **Neural Processing Letters**, 46, n. 3, p. 751–766, 2017. ISSN 1370-4621.

NAKKIRAN, P.; KAPLUN, G.; BANSAL, Y.; YANG, T.; BARAK, B.; SUTSKEVER, I. **Deep double descent**: where bigger models and more data hurt. 2019.

NETO, A. R. R.; BARRETO, G. A. Opposite maps: vector quantization algorithms for building reduced-set SVM and LSSVM classifiers. **Neural Processing Letters**, 37, n. 1, SI, p. 3–19, 2013. ISSN 1370-4621.

NI, B.; MOULIN, P.; YAN, S. Order preserving sparse coding. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 37, n. 8, p. 1615–1628, 2015. ISSN 0162-8828.

NIEWIADOMSKA-SZYNKIEWICZ, E.; MARKS, M. Optimization schemes for wireless sensor network localization. **International Journal of Applied Mathematics and Computer Science**, Walter de Gruyter & Co., Hawthorne, USA, v. 19, n. 2, p. 291–302, 2009. ISSN 1641-876X.

OLIVEIRA, S. A. F.; GOMES, J. P. P.; NETO, A. R. R. Sparse least-squares support vector machines via accelerated segmented test: a dual approach. **Neurocomputing**, 321, p. 308–320, 2018. ISSN 0925-2312.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: machine learning in python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PEKALSKA, E.; DUIN, R.; PACLIK, P. Prototype selection for dissimilarity-based classifiers. **Pattern Recognition**, 39, n. 2, p. 189–208, 2006.

PETERSEN, K. B.; PEDERSEN, M. S. *et al.* The matrix cookbook. **Technical University of Denmark**, v. 7, n. 15, p. 510, 2008.

POGGIO, T.; LIAO, Q.; BANBURSKI, A. Complexity control by gradient descent in deep networks. **Nature Communications**, 11, n. 1, 2020. ISSN 2041-1723.

RAO, B.; KREUTZ-DELGADO, K. An affine scaling methodology for best basis selection. **IEEE Transactions on Signal Processing**, 47, n. 1, p. 187–200, 1999. ISSN 1053-587X.

ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: **Computer Vision – ECCV 2006**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. 3951, n. 1, p. 430–443. ISBN 3-540-33832-2. ISSN 0302-9743.

ROSTEN, E.; PORTER, R.; DRUMMOND, T. Faster and better: a machine learning approach to corner detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 32, n. 1, p. 105–119, 2010. ISSN 0162-8828.

RUSSELL, S.; RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. Hoboken, USA: Pearson, 2020. (Pearson series in artificial intelligence). ISBN 9780134610993.

SANTOS, J. D. A.; BARRETO, G. A. A regularized estimation framework for online sparse LSSVR models. **Neurocomputing**, 238, p. 114–125, 2017. ISSN 0925-2312.

SANTOS, J. D. A.; BARRETO, G. A. Novel sparse LSSVR models in primal weight space for robust system identification with outliers. **Journal of Process Control**, ELSEVIER SCI LTD, OXON, ENGLAND, 67, n. SI, p. 129–140, 2018. ISSN 0959-1524.

SEZGIN, M.; SANKUR, B. Survey over image thresholding techniques and quantitative performance evaluation. **Journal of Electronic Imaging**, 13, n. 1, p. 146–168, 2004. ISSN 1017-9909.

SILVA, D. A.; SILVA, J. P.; NETO, A. R. R. Novel approaches using evolutionary computation for sparse least square support vector machines. **Neurocomputing**, 168, p. 908–916, 2015. ISSN 0925-2312.

SILVERMAN, B. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. **Journal of The Royal Statistical Society Series B-Statistical Methodology**, 47, n. 1, p. 1–52, 1985. ISSN 1369-7412.

SIVANANDAM, S.; DEEPA, S. **Introduction to Genetic Algorithms**. New York, USA: Springer Berlin Heidelberg, 2007. ISBN 9783540731900.

SOBER, E. **Ockham's razors**: a user's manual. Cambridge, United Kingdom: Cambridge University Press, 2015. ISBN 9781316368534.

SOUSA JÚNIOR, A. H. **Regional models and minimal learning machines for nonlinear dynamic system identification**. Phd Thesis (PPGETI/UFC) — Department of Teleinformatics Engineering, Federal University of Ceará, Fortaleza, Brazil, 2014.

SOUZA JÚNIOR, A. H.; CORONA, F.; BARRETO, G. A.; MICHE, Y.; LENDASSE, A. Minimal learning machine: a novel supervised distance-based approach for regression and classification. **Neurocomputing**, 164, p. 34–44, 2015. ISSN 0925-2312.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, 15, p. 1929–1958, 2014. ISSN 1532-4435.

SUYKENS, J.; VANDEWALLE, J. Least squares support vector machine classifiers. **Neural Processing Letters**, 9, n. 3, p. 293–300, 1999. ISSN 1370-4621.

SUYKENS, J. A. K.; LUKAS, L.; VANDEWALLE, J. Sparse least squares support vector machine classifiers. In: **Proceedings of 8th European Symposium on Artificial Neural Networks**. Bruges, Belgium: ESANN, 2000. p. 37–42.

THEODORIDIS, S. **Machine learning**: a bayesian and optimization perspective. Cambridge, USA: Elsevier Science, 2020. ISBN 9780128188040.

TIPPING, M. Sparse bayesian learning and the relevance vector machine. **Journal of Machine Learning Research**, 1, n. 3, p. 211–244, 2001. ISSN 1532-4435.

TORGO, L. **Regression datasets**. 2005. Available at: "www.liaad.up.pt/~ltorgo/Regression/DataSets.html". Accessed on: July, 13th 2021.

TROPP, J. Greed is good: algorithmic results for sparse approximation. **IEEE Transactions on Information Theory**, 50, n. 10, p. 2231–2242, 2004. ISSN 0018-9448.

TROPP, J. A.; GILBERT, A. C. Signal recovery from random measurements via orthogonal matching pursuit. **IEEE Transactions on Information Theory**, 53, n. 12, p. 4655–4666, 2007. ISSN 0018-9448.

VEENMAN, C.; REINDERS, M. The nearest subclass classifier: a compromise between the nearest mean and nearest neighbor classifier. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, 27, n. 9, p. 1417–1429, 2005. ISSN 0162-8828.

VIEIRA, D. C. d. S.; NETO, A. R. R.; RODRIGUES, A. W. d. O. Sparse least squares support vector regression via multiresponse sparse regression. In: IEEE. **2016 International Joint Conference on Neural Networks (IJCNN)**. Vancouver, Canada, 2016. p. 3218–3225. ISBN 978-1-5090-0619-9. ISSN 2161-4393.

WILSON, D.; MARTINEZ, T. Reduction techniques for instance-based learning algorithms. **Machine Learning**, 38, n. 3, p. 257–286, 2000.

YANG, J.; WRIGHT, J.; HUANG, T. S.; MA, Y. Image super-resolution via sparse representation. **IEEE Transactions on Image Processing**, 19, n. 11, p. 2861–2873, 2010. ISSN 1057-7149.

YANG, L.; YANG, S.; ZHANG, R.; JIN, H. Sparse least square support vector machine via coupled compressive pruning. **Neurocomputing**, 131, p. 77–86, 2014. ISSN 0925-2312.

ZHAO, S.; YAO, H.; GAO, Y.; JI, R.; DING, G. Continuous probability distribution prediction of image emotions via multitask shared sparse regression. **IEEE Transactions on Multimedia**, 19, n. 3, p. 632–645, 2017. ISSN 1520-9210.

ZHAO, Y.-P.; SUN, J.-G.; DU, Z.-H.; ZHANG, Z.-A.; ZHANG, Y.-C.; ZHANG, H.-B. An improved recursive reduced least squares support vector regression. **Neurocomputing**, 87, p. 1–9, 2012. ISSN 0925-2312.

ZHOU, H.; LAN, J.; LIU, R.; YOSINSKI, J. Deconstructing lottery tickets: zeros, signs, and the supermask. **Advances in Neural Information Processing Systems**, v. 32, p. 3597–3607, 2019.

ZHOU, S. Sparse LSSVM in primal using cholesky factorization for large-scale problems. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, Piscataway, USA, 27, n. 4, p. 783–795, 2016. ISSN 2162-237X.

APPENDIX A – SOLUTION OF LIGHTWEIGHT MLM LINEAR SYSTEM

Firstly, we rewrite Equation 4.1 as

$$\frac{\partial}{\partial \mathbf{B}}\mathcal{J}(\mathbf{B}) = \frac{\partial}{\partial \mathbf{B}} \parallel \mathbf{DB}-\mathbf{\Delta} \parallel_{\mathcal{F}}^{2} + \parallel \mathbf{PB} \parallel_{\mathcal{F}}^{2} = \frac{\partial}{\partial \mathbf{B}} \parallel \mathbf{DB}-\mathbf{\Delta} \parallel_{\mathcal{F}}^{2} + \frac{\partial}{\partial \mathbf{B}} \parallel \mathbf{PB} \parallel_{\mathcal{F}}^{2}. \tag{A.1}$$

By employing the following properties taken from the Matrix Cookbook (PETERSEN *et al.*, 2008):

a) Properties n°103 and n°104: $\frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}\{\mathbf{A}\mathbf{X}^{\intercal}\} = \frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}\{\mathbf{X}^{\intercal}\mathbf{A}\} = \mathbf{A}$;

b) Property n°108: $\frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}\{\mathbf{X}^{\intercal}\mathbf{B}\mathbf{X}\} = \mathbf{B}\mathbf{X}+\mathbf{B}^{\intercal}\mathbf{X}$;

c) Property n°115 : $\frac{\partial}{\partial \mathbf{X}} \parallel \mathbf{X} \parallel_{\mathcal{F}}^{2} = \frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}\left\{\mathbf{X}\mathbf{X}^{H}\right\} = \frac{\partial}{\partial \mathbf{X}}\mathrm{Tr}\left\{\mathbf{X}^{H}\mathbf{X}\right\} = 2\mathbf{X}$.

where $\mathbf{X}^{H}$ is the transposed and complex conjugated matrix (Hermitian) of $\mathbf{X}$, one obtains:

$$\frac{\partial}{\partial \mathbf{B}}\mathcal{J}(\mathbf{B}) = \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{(\mathbf{DB}-\mathbf{\Delta})^{\intercal}(\mathbf{DB}-\mathbf{\Delta})\} + \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{(\mathbf{PB})^{\intercal}\mathbf{PB}\}. \tag{A.2}$$

$$= \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{\mathbf{B}^{\intercal}\mathbf{D}^{\intercal}\mathbf{DB}-\mathbf{B}^{\intercal}\mathbf{D}^{\intercal}\mathbf{\Delta}-\mathbf{\Delta}^{\intercal}\mathbf{DB}+\mathbf{\Delta}^{\intercal}\mathbf{\Delta}\} + \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{\mathbf{B}^{\intercal}\mathbf{P}^{\intercal}\mathbf{PB}\}. \tag{A.3}$$

As $\partial(\cdot)$ and $\mathrm{Tr}\{\cdot\}$ are linear operators, we rewrite the above equation as follows:

$$\frac{\partial}{\partial \mathbf{B}}\mathcal{J}(\mathbf{B}) = \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{\mathbf{B}^{\intercal}\mathbf{D}^{\intercal}\mathbf{DB}\} - \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{\mathbf{B}^{\intercal}\mathbf{D}^{\intercal}\mathbf{\Delta}\} - \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{\mathbf{\Delta}^{\intercal}\mathbf{DB}\} + \frac{\partial}{\partial \mathbf{B}}\mathrm{Tr}\{\mathbf{B}^{\intercal}\mathbf{P}^{\intercal}\mathbf{PB}\}. \tag{A.4}$$

$$= \mathbf{D}^{\intercal}\mathbf{DB} + (\mathbf{D}^{\intercal}\mathbf{D})^{\intercal}\mathbf{B} - \mathbf{D}^{\intercal}\mathbf{\Delta} - (\mathbf{\Delta}^{\intercal}\mathbf{D})^{\intercal} + \mathbf{P}^{\intercal}\mathbf{PB} + (\mathbf{P}^{\intercal}\mathbf{P})^{\intercal}\mathbf{B}. \tag{A.5}$$

Since $\mathbf{D}, \mathbf{\Delta}$, and $\mathbf{P}$ are symmetric, we express $\frac{\partial}{\partial \mathbf{B}}\mathcal{J}(\mathbf{B})$ as follows:

$$\frac{\partial}{\partial \mathbf{B}}\mathcal{J}(\mathbf{B}) = 2\mathbf{D}^{\intercal}\mathbf{DB} - 2\mathbf{D}^{\intercal}\mathbf{\Delta} + 2\mathbf{P}^{\intercal}\mathbf{PB}. \tag{A.6}$$

Finally, by setting $\frac{\partial}{\partial \mathbf{B}}\mathcal{J}(\mathbf{B}) = \mathbf{0}$ to achieve the optimum solution, one obtains:

$$\hat{\mathbf{B}} = (\mathbf{D}^{\intercal}\mathbf{D} + \mathbf{P}^{\intercal}\mathbf{P})^{-1}\mathbf{D}^{\intercal}\mathbf{\Delta}. \tag{A.7}$$