Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Department of Mathematics and Physics
Master in Data Science

A Generalized Lagrange Multiplier Method Based for Support Vector Classification

---

THESIS to obtain the **DEGREE** of
MASTER IN DATA SCIENCE

A thesis presented by: **Juan Francisco Toral Cañedo**

Thesis Advisor: **Mtra. Sara Eugenia Rodríguez Reyes**

Thesis Co-Advisor: **Dr. Juan Diego Sánchez Torres**

Tlaquepaque, Jalisco, May, 2023

Instituto Tecnológico
y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Matemáticas y Física

Maestría en Ciencia de Datos

Un Método Generalizado del Multiplicador de Lagrange basado en la Máquina de Vector Soporte para Clasificación

---

TESIS para obtencion del **GRADO** de
MAESTRO EN CIENCIA DE DATOS

Una tesis presentada por: **Juan Francisco Toral Cañedo**

Asesor de tesis: **Mtra. Sara Eugenia Rodríguez Reyes**

Co-asesor de tesis: **Dr. Juan Diego Sánchez Torres**

Tlaquepaque, Jalisco, May, 2023

A Generalized Lagrange Multiplier Method Based for Support Vector Classification

Juan Francisco Toral Cañedo

Abstract

This document aims to present an alternative formulation to the well-known L_1 Support Vector Classifier (In short, SVC). By analyzing the composition of the Support Vector Classifier Lagrangian function and its dual problem, we introduce new terms that will ultimately allow the Support Vector Classifier to have an elastic net regularization structure in its dual formulation through a Generalized Lagrange Multiplier Method (GLMM). This paper also aims to reveal the regularization form already contained in the classical Support Vector Classifier dual problem structure in order to understand the implication of an elastic net regularization form in the Support Vector Classifier, demonstrate any possible enhancements resulting in this new approach, and compare the results to a group of other frequently used methods for classification. This paper intends to demonstrate the viability of the GLMM based for Support Vector Classification with an elastic net Regularization.

Keywords— Support Vector Classifier, Kernel-based methods, Generalized Lagrange Multiplier Method

Método Generalizado del Multiplicador de Lagrange basado en la Máquina de Vector Soporte para Clasificación

Juan Francisco Toral Cañedo

Resumen

El objetivo de este documento es el de presentar una formulación alterna a la ya conocida Máquina de Vector Soporte tipo L_1 (SVC). Analizando la composición del Lagrangiano de dicha máquina asi como su problema dual, se introducen nuevos términos que terminarían ayudando a que la Máquina de Vector Soporte contenga una regularización tipo elastic net en su formulación del problema dual por medio del Método Generalizado del Multiplicador de Lagrange (GLMM). Este documento también tiene como intención revelar el tipo de regularización que ya existe en la formulación clásica de la Máquina de Vector Soporte para el problema de clasificación con el fin de entender la implicación y rol que tendría la regularización tipo elastic net; así como demostrar cualquier posible mejora resultado de este nuevo acercamiento y comparar los resultados contra otros métodos comúnmente empleados para resolver problemas de clasificación. Este documento intenta demostrar la viabilidad del Método Generalizado del Multiplicador de Lagrange basado en la Máquina de Vector Soporte para el problema de clasificación con una regularización tipo elastic net.

***Keywords*—** Máquina de Vector Soporte, Clasificación, Método Generalizado del Multiplicador de Lagrange, Kernel

Contents

List of Figures

List of Tables

This work is dedicated to my dear parents, Juan Francisco Toral Ávila and Marcela Cañedo Sardo, who raised me as someone who enjoys challenges and can overcome any obstacle. To my beloved wife, Estrella Michelle González García, who inspires me to become a better man every day and supports me unconditionally. To my brother Gonzalo Toral Cañedo who tutored me in math and programming to get me started in Data Science, to Carlos Emilio Carranza Ávila who was alongside me most of my academic life sharing the struggle together. To my thesis advisor Sara Rodriguez whose help and guidance got me to this point.

1 Introduction

As we dive into this thesis topic, we must first understand the common use of machine learning classification techniques and their practical benefits. From financial institutions needing safer loan-issuing opportunities based on the information they have on applicants to medical patients being subjected to several tests in order to discover a condition or disease. These real-life scenarios find themselves submitting data sets about the problem they need to solve into numerous calculus that will ultimately classify the desired outcome accordingly.

Any given data set's observations and its variables could be interpreted as vectors, where $x_i \in \mathbb{R}^n$ are vectors of features and $y_i \in \{-1, +1\}$ are its classes or outcome variables, then a set of labeled points (x_i, y_i) could have a new point x assigned to a proper class in a binary classification problem [1].

[1] Dimitry Fradkin and Ilya Muchnik (2006) *Support Vector Machines for Classification*

Support Vector Machines (in short SVM) are supervised learning methods with foundations in Statistical Learning Theory that produce classifiers (or regressors) that can accurately solve problems, especially when they are not linearly separable, meaning that, for the classification problem, two sets of points cannot be separated apart from each other in a two-dimensional space by drawing a single line [2]. By the use of the Lagrangian method to solve a constrained function, the SVM can submit the entry data to a transformation so that the input variables are mapped into a different space where they can be linearly separated; this transformation technique use something called kernels, which will be further explained down this document.

[2] Yoonkyung Lee (2010) *Support Vector Machines For Classification: A Statistical Portrait* The Ohio State University

The concept of regularization in machine learning constitutes a series of techniques that prevent the proposed model from overfitting through a penalization term into the cost function's coefficients. This paper will go over the types of regularization the conventional Support Vector Machines contain, and a new form of regularization will be proposed to attempt to improve the performance of the Support Vector Classifier

(SVC).

This thesis will first visit some known regularization techniques, briefly reviewing the purpose and implication of them. Then it is intended to explain the soft-margin Support Vector classifiers in their traditional form before getting into what the proposal of this document will be, which is a new Support Vector Machine using a Elastic Net regularization method by introducing a Generalized Lagrangian Multiplier Method. Lastly, the proposed machine will be tested using data sets to evaluate the feasibility and performance against other classification methods.

2 Support Vector Classifier and Regularization

Contents

2.1 Ridge Regularization

Also referred to as Tikhonov regularization or L_2 regularization is one of the most commonly used techniques to correct over-fitting in machine learning modeling. The inclusion of a penalty term to the cost function often brings stability to the model by reducing the noise contained in the variables that do not represent our data set the best.

Ridge regularization shrinks the sum of the squared coefficients of the model. By setting this penalty, Ridge regularization reduces the model complexity and multi-collinearity.

The Ridge regularization would be represented by the following penalty form[1]

$$\frac{\lambda}{2}\sum_{k=1}^{m} w_k^2$$

Its Euclidean norm can also represent it,

$$\frac{\lambda}{2}\|w\|_2$$

[1] w represents the coefficients of the model and k=1,...,m. These indicate the direction of the relationship between a predictor variable and the response variable.

The hyperparameter λ is the value that controls the importance of the penalty term, allowing the model to be as flexible or strict as needed; this helps the model to avoid over-fitting. In this form of regularization, coefficients never become zero, thus taking into consideration every variable in the dataset.

2.2 *LASSO Regularization*

Also known as L_1 regularization, LASSO[2] regularization is represented by the following penalty term,

[2] Acronym for Least Absolute Shrinkage and Selection Operator

$$\lambda \sum_{k=1}^{m} |w_k|$$

It can also be written in its Taxicab norm,

$$\lambda \|w\|_1$$

For this regularization, the penalty term is the hyperparameter lambda multiplied by the sum of the absolute value of the coefficients. LASSO's coefficients might shrink to zeros, therefore making it a great regularization method for feature selection[3].

[3] Amy@GrabNGoInfo (2022) *Towardsai.net*

In the classification problem, the penalization term is usually added in the objective function for coefficient shrinkage using a probability distribution for the label or target variable.

2.3 *Elastic Net Regularization*

The elastic net regularization is a form of penalization that combines the L_1 and L_2 types. The effect that this regularization has on the cost function would set some coefficients to zero with their LASSO part, but the variable selection, in this case, is often smoother as the RIDGE portion helps decrease correlation among variables.

The elastic net representation takes the following form:

$$\lambda \left[(1-\omega) \sum \|w_k\|_1 + \frac{\omega}{2} \|w_k\|_2^2 \right] \tag{2.1}$$

For that particular form, the λ is a non-negative primal-parameter, and the value of ω can only acquire values between 0 and 1.

2.4 L_1 Soft-Margin Support Vector Classifier

The L_1 Soft-Margin Support Vector Machine is utilized when the data our model is being trained with is not linearly separable. Most data in real-life scenarios is not linearly separable in the classification context. Then, a non-negative slack variable is introduced ($\xi_k \geq 0$) to the hard-margin SVM, so miss-classification is accepted to propose a hyperplane where feasible solutions exist.

The L_1 SVC follows this optimization problem:

$$\begin{aligned} \min_{w,b,\xi} P(w,b,\xi) &= \frac{1}{2} w^T w + \frac{C}{p} \sum_{k=1}^{m} \xi_k^p \\ \text{s.t. } y_k[w^T \phi(x_k) + b] &\geq 1 - \xi_k \\ \xi_k &\geq 0,\ k = 1, \ldots, N \end{aligned} \tag{2.2}$$

The hyperparameter C determines the trade-off between the maximization of the margin and the minimization of the classification error [4]. And $\phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ represents the mapping of the original input into a high-dimensional dot-product space where the data could be linearly separable[5].

[4] Shigeo Abe. *Support Vector Machines for Pattern Classification*. Springer, second edition, 2004. ISBN 978-1-84996-097-7

[5] This is commonly referred to as *The Feature Space*

Now that the primal problem of the L_1 SVC has been established, the Lagrangian function can be formulated using the objective function from the primal optimization problem and introduce the respective Lagrange multipliers to the constraints of the same problem.

Let the Lagrangian function for the problem (2.2)

$$\begin{aligned} \mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2} w^T w + C \sum_{k=1}^{N} \xi_k \\ & - \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \xi_k \right) - \sum_{k=1}^{N} \eta_k \xi_k \end{aligned} \tag{2.3}$$

For this function, the variables w, b, ξ_k are related to the primal problem that tries to minimize them. In contrast, the Lagrangian problem introduced the non-negative variables α, η that are tied to the constraints of the objective function. Where $\alpha = (\alpha_1, ..., \alpha_m)^T$ and $\eta = (\eta_1, ..., \eta_m)^T$

Support Vector Machines are generally based on quadratic

programming, meaning that their constraints would have linear equality and inequality, and to find an optimal solution, the *Karush-Kuhn-Tucker conditions* (KKT) or First-Order Conditions would be used just for that, these conditions are found by calculating the partial derivative of the Lagrangian function with respect of each primal variable of the problem:

- $\nabla w\mathcal{L} = w - \sum_{k=1}^{N} \alpha_k y_k \phi(x_k) = 0 \rightarrow$w = $\sum_{k=1}^{N} \alpha_k y_k \phi(x_k)$
- $\frac{\partial}{\partial b}\mathcal{L} = \sum_{k=1}^{N} \alpha_k y_k = 0$
- $\frac{\partial}{\partial \xi_k}\mathcal{L} = C - \alpha_k - \eta_k = 0 \rightarrow$C = $\alpha_k + \eta_k$ which implies $\rightarrow \eta_k = C - \alpha_k$

Now that the variables have been canceled due to the First Order conditions, they can be replaced in the Lagrangian function. Replacing the variable C:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + (\alpha_k + \eta_k) \sum_{k=1}^{N} \xi_k \\ &- \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \xi_k \right) \\ &- \sum_{k=1}^{N} \eta_k \xi_k\end{aligned} \tag{2.4}$$

Then, variable w is replaced in the Lagrangian (2.4):

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} \left[\sum_{k=1}^{N} \alpha_k y_k \phi(x_k) \right]^T \left[\sum_{l=1}^{N} \alpha_l y_l \phi(x_l) \right] + (\alpha_k + \eta_k) \sum_{k=1}^{N} \xi_k \\ &- \sum_{k}^{N} \alpha_k \left(y_k \left[\sum_{l=1}^{N} \alpha_l y_l \phi(x_l)^T \phi(x_k) + b \right] - 1 + \xi_k \right) \\ &- \sum_{k=1}^{N} \eta_k \xi_k\end{aligned} \tag{2.5}$$

Once the variables are replaced, they are now grouped and reduced to obtain the following terms:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2}\left[\sum_{k=1}^{N} \alpha_k y_k \phi(x_k)\right]^T \left[\sum_{l=1}^{N} \alpha_l y_l \phi(x_l)\right] + \sum_{k=1}^{N} \alpha_k \xi_k + \sum_{k=1}^{N} \eta_k \xi_k \\ & - \sum_{k}^{N} \alpha_k y_k \phi(x_k)^T \sum_{l=1}^{N} \alpha_l y_l \phi(x_l) + \sum_{k=1}^{N} \alpha_k - \sum_{k=1}^{N} \alpha_k \xi_k \\ & - \sum_{k=1}^{N} \eta_k \xi_k \end{aligned} \tag{2.6}$$

Once the terms were grouped, simply reduce them from (2.6) to obtain the following:

$$\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) = -\frac{1}{2} \sum_{k=1}^{N} \alpha_k \alpha_l y_k y_l \phi(x_k)^T \phi(x_l) + \sum_{k=1}^{N} \alpha_k \tag{2.7}$$

Recalling the primal feasibility conditions from (2.2), which are the constraints of the primal problem:

$$\begin{aligned} y_k[w^T \phi(x_k) + b] \geq 1 - \xi_k, \ k = 1, \ldots, N \\ \xi_k \geq 0, \ k = 1, \ldots, N \end{aligned} \tag{2.8}$$

The Dual Feasibility Conditions come from the Non-negative Lagrangian Multipliers, and taking the First Order Condition, the following can be deducted.

$$\begin{aligned} \alpha_k \geq 0, \ k = 1, \ldots, N \\ \eta_k \geq 0, \ k = 1, \ldots, N \\ C - \alpha_k = \eta_k \geq 0, \rightarrow \ C - \alpha_k \geq 0 \\ 0 \leq \alpha_k \leq C \end{aligned} \tag{2.9}$$

Finally, with the primal and Dual Feasibility conditions established, the following Dual Problem is obtained:

$$\begin{aligned} \max_{\alpha} \frac{1}{2} \sum_{k,l=1}^{N} \alpha_k \alpha_l y_k y_l \phi(x_k)^T \phi(x_l) + \sum_{k=1}^{N} \alpha_k \\ \text{s.t.} \sum_{k=1}^{N} \alpha_k y_k = 0 \\ 0 \leq \alpha_k \leq C, \; k = 1, \ldots, N \end{aligned} \tag{2.10}$$

Strong duality states that the optimal primal and dual objective values are equal under certain conditions. The weak duality creates an optimality gap where the duality gap from the primal objective is greater or equal to zero.

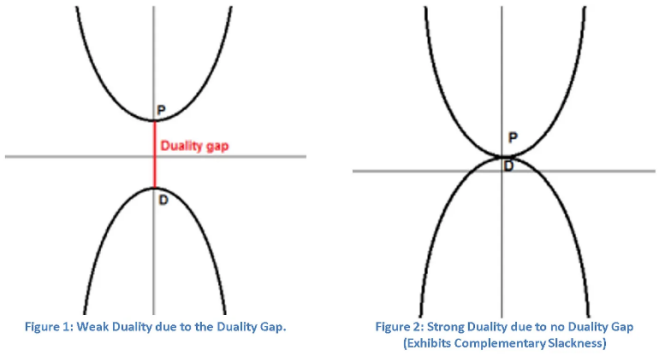


Figure 2.1: Duality Gaps

The concept of Complementary Slackness suggests a relationship between the slackness between the primal problem and the Dual problem, more specifically, the primal constraint and the slackness of the dual variables. Slack in corresponding inequalities must be complementary, and the primal problem must be bounded for the dual to be feasible[6].

[6] J.Birge, F. Loveaux (2011) *Introduction to Stochastic Programming*

The following Complementary Slackness condition displays the relation between the dual variable and the primal constraints from the L_1 SVC that must be satisfied.

$$\begin{aligned} \alpha_k \left[y_k (w^T \phi(x_k) + b) - 1 + \xi_k \right] = 0 \\ \alpha_k \geq 0 \\ \eta_k \geq 0 \\ \xi_k \geq 0 \end{aligned} \tag{2.11}$$

With the constraints of the dual problem and the complementary conditions set, analyzing the possible values for α_k.

1. When α_k = 0 from (2.10) it has the implication that $y_k[w^T\phi(x_k) + b] \geq 1 - \xi_k$ from the primal constraint. Knowing that $\eta_k\xi_k$ = 0, and that $\eta_k = C$, therefore $\xi_k = 0$. In conclusion, when $\alpha_k = 0$, $y_k[w^T\phi(x_k) + b] \geq 1$ and α_k is correctly classified.

2. When $0 \leq \alpha_k \leq C$ from (2.10) it has the implication that $y_k[w^T\phi(x_k) + b] \geq 1 - \xi_k$ from the primal constraint. Knowing that $\eta_k\xi_k$ = 0, in this case, $\eta_k > 0$, therefore $\xi_k = 0$. In conclusion, when $0 \leq \alpha_k \leq C$, $y_k[w^T\phi(x_k) + b] = 1 - \xi_k$, and α_k is correctly classified, and it is an "unbounded" Support Vector.

3. When $\alpha_k = C$ from (2.10) it has the implication that $y_k[w^T\phi(c_k) + b] = 1 - \xi_k$ from the primal constraint and $\xi_k \geq 0$. In this case, there are two possible outcomes:

   - The Support Vector is correctly classified when $yk[w^T\phi(x_k) + b] > 0$ which implicates that $1 - \xi_k \geq 0$ and $0 \leq \xi_k \leq C$. This is a "bounded" Support Vector.
   - The Support Vector is misclassified when $y_k[w^T\phi(x_k) + b] \leq 0$ which implicates that $1 - \xi_k \leq 0$ and $\xi_k \geq 1$. This is also a bounded Support Vector.

   The bounded and unbounded vector refers to those found within the area of possible misclassification that the slack variable allows for the model.

   The decision function for the L_1 SVC is given by:

$$D(x) = \sum_{k \in S} \alpha_k y_k \phi(x_k)^T \phi(x_l) + b \tag{2.12}$$

   Where S is the list of Support Vector Indices, and knowing that α_k is a non-negative for the support vectors, the unbounded α_k is satisfied with $b = y_k - w^T x_k$. And by calculating the b from the unbounded support vectors, the average is taken for a more precise calculation of b.

$$b = \frac{1}{|U|} \sum_{i \in U} (y_k - w^T x_k) \tag{2.13}$$

7

[7] U is the list of the unbounded support vector indices

2.5 L_2 Soft-Margin Support Vector Classifier

The Lagrangian problem (2.2) for the L_2 SVC takes the following formulation:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + \frac{C}{2} \sum_{k=1}^{N} \xi_k^2 \\ & - \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \xi_k \right)\end{aligned} \tag{2.14}$$

Just as the L_1 optimization problem, w, b, ξ are the primal variables, and α are the dual variables associated with the constraints of the primal problem. There is no η primal variable for this problem since there is no constraint for the non-negative ξ since the square of ξ_k guarantees in the objective function that the variable will be non-negative.

The First Order Conditions will be established as follows:

- $\nabla w \mathcal{L} = w - \sum_{k=1}^{N} \alpha_k y_k \phi(x_k) = 0 \rightarrow \text{w} = \sum_{k=1}^{N} \alpha_k y_k \phi(x_k)$ (w.2)
- $\frac{\partial}{\partial b} \mathcal{L} = \sum_{k=1}^{N} \alpha_k y_k = 0$ (b.2)
- $\frac{\partial}{\partial \xi_k} \mathcal{L} = C\xi_k - \alpha_k = 0 \rightarrow C\xi_k = \alpha_k$ which implies $\rightarrow \xi_k = \frac{\alpha_k}{C}$ (xi.2)

Now that the variables have been canceled due to the First Order conditions, they can be replaced in the Lagrangian Function. Replacing the variable ξ_k:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + \frac{C}{2} \sum_{k=1}^{N} \frac{\alpha_k^2}{C^2} \\ & - \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \frac{\alpha_k}{C} \right)\end{aligned} \tag{2.15}$$

This time, before the variable w is replaced in the Lagrangian (2.15) we group and reduce to have the following terms:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + \frac{C}{2} \sum_{k=1}^{N} \frac{\alpha_k^2}{C^2} \\ & - \sum_{k}^{N} \alpha_k y_k w^T \phi(x_k) - \sum_{k=1}^{N} \alpha_k y_k b \\ & + \sum_{k=1}^{N} \alpha_k - \frac{1}{C} \sum_{k=1}^{N} \alpha_k^2\end{aligned} \tag{2.16}$$

Replace the (b.2) to obtain:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + \frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 \\ & - \sum_{k}^{N} \alpha_k y_k w^T \phi(x_k) \\ & + \sum_{k=1}^{N} \alpha_k - \frac{1}{C} \sum_{k=1}^{N} \alpha_k^2\end{aligned} \tag{2.17}$$

Now the w variable is replaced, it gets reduced to the following term:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& -\frac{1}{2} \sum_{k,l=1}^{N} \alpha_k \alpha_{lk} y_k y_l \phi(x_k)^T \phi(x_l) \\ & - \frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 + \sum_{k=1}^{N} \alpha_k\end{aligned} \tag{2.18}$$

If $k = l$:

$$\begin{aligned}& - \frac{1}{2} \sum_{k=1}^{N} \alpha_k^2 y_k^2 \phi(x_k) \phi(x_k) \\ & - \frac{1}{2} \sum_{k,l=1}^{N} \alpha_k \alpha_l y_k y_l \phi(x_k)^T \phi(x_l) \\ & - \frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 + \sum_{k=1}^{N} \alpha_k\end{aligned} \tag{2.19}$$

If $k \neq l$:

$$
\begin{aligned}
&- \frac{1}{2} \sum_{k,l=1}^{N} \alpha_k \alpha_l y_k y_l \phi(x_k)^T \phi(x_l) \\
&- \frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 + \sum_{k=1}^{N} \alpha_k
\end{aligned}
\tag{2.20}
$$

Following the case where $k = l$ the terms can be grouped into this form:

$$
\begin{aligned}
\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) = &- \frac{1}{2} \sum_{k,l=1}^{N} \alpha_k \alpha_{lk} y_k y_l \phi(x_k)]^T \phi(x_l) \\
&- \frac{1}{2} \sum_{k=1}^{N} \alpha_k^2 + (y_k^2 \phi(x_k) \phi(x_k) + \frac{1}{C}) \\
&+ \sum_{k=1}^{N} \alpha_k
\end{aligned}
\tag{2.21}
$$

Kronecker's delta function contains two variables; given the fact that the problem can contain both cases where variable k is equal or different than l, this function can gain the value 1 if $k = l$ or 0 if $k \neq l$:

$$
\delta_{k,l} = \begin{cases} 1 \text{ if } k = l \\ 0 \text{ if } k \neq l \end{cases}
\tag{2.22}
$$

The Dual Feasibility conditions are the non-negative Lagrangian Multipliers related to the primal problem, and taking the First Order Condition, the following can be deducted (x1.2):

$$
\begin{aligned}
\alpha_k \geq 0, \ k = 1, \ldots, N \\
C\xi_k = \alpha_k
\end{aligned}
\tag{2.23}
$$

The following dual problem is obtained:

$$\begin{aligned} \max_{\alpha} -\frac{1}{2}\alpha_l\alpha_l y_k y_l \left[\phi(x_k)^T\phi(x_l) + \frac{\delta_{kj}}{C}\right] + \sum_{k=1}^{N}\alpha_k \\ \text{s.t. } \sum_{k=1}^{N}\alpha_k y_k = 0 \\ \alpha_k \geq 0,\ k = 1,\ldots,N \end{aligned} \tag{2.24}$$

The following Complementary Slackness condition displays the relation between the dual variable and the primal constraints from the L_2 SVC that must be satisfied.

$$\begin{aligned} \alpha_k \left[y_k(w^T\phi(x_k) + b) - 1 + \xi_k\right] = 0 \\ C\xi_k \geq \alpha_k \end{aligned} \tag{2.25}$$

With the constraints of the dual problem and the complementary conditions set, analyzing the possible values for α_k.

1. When α_k = 0 from (3.38) it has the implication that $y_k[w^T\phi(x_k) + b] \geq 1 - \xi_k$ from the primal constraint. Knowing that $C\xi_k = \alpha_k$, therefore $\xi_k = 0$. In conclusion, when $\alpha_k = 0$, $y_k[w^T\phi(x_k) + b] \geq 1$ and α_k is correctly classified.
2. When $\alpha_k \geq 0$ from (3.38), it is given that $y_k[w^T\phi(x_k) + b] - 1 = 0$.

   Just like the L_1 SVC, the decision function for the L_2 SVC is given by:

$$D(x) = \sum_{k\in S}\alpha_k y_k \phi(x_k)^T\phi(x_l) + b \tag{2.26}$$

   Unlike the case of the L_1 SVC, the L_2 machine doesn't contain bounded support vectors since the upper bound given by C is now removed. To calculate the bias for $\alpha_k \geq 0$ we have the following:

$$b = \frac{1}{|S|}\sum_{l\in U}\left(y_l - \sum_{k\in S}\alpha_k y_k\left(\phi(x_k)^T\phi(x_l) + \frac{\delta_{kl}}{C}\right)\right) \tag{2.27}$$

8

[8] Since there are no unbounded support vectors, the list of indices of support vector is taken instead

3 Generalized Lagrangian-based Support Vector Machines for Classification

Contents

3.1 Generalized Lagrangian Multiplier Method

The Lagrange multiplier method helps to connect constrained optimization and saddle-point problems since saddle points of Lagrangians provide solutions to corresponding constrained optimization problems, as in the case of the Support Vector Classifier, which is based on this saddle-point dynamics.

The Generalized Lagrange Multiplier Method (GLMM) reduces the duality gap between primal and dual problems for non-convex optimization.

Considering a constrained optimization problem with one equality and one inequality constraints

$$\begin{aligned} &\inf_{x\in\mathbb{R}^n} f(x) \\ &\text{s.t. } g(x) \leq 0,\ h(x) = 0 \end{aligned} \tag{3.1}$$

Denoting the feasibility set $\{x|g(x) \leq 0, h(x) = 0\}$ as $\mathcal{F}$

Adopting the GLMM proposed in [1],

$$\mathcal{L}(x, \tau, \nu) = f(x) + \mathcal{G}(\tau, g(x)) + \mathcal{H}(\nu, h(x)) \tag{3.2}$$

[1] Mengmou Li. *Generalized Lagrangian Multiplier Method and KKT Conditions With an Application to Distributed Optimization*, volume 66. 2019. DOI: 10.1109/TCSII.2018.2842085

where $\mathcal{G}(\tau, g(x))$ is a function of $g(x)$ and τ, satisfying

(a) Monotonically increasing with respect to $g(x)$.

(b) Concave with respect to λ, if $g(x) \leq 0$.

(c) $\sup_\tau \mathcal{G}(\tau, g(x)) = 0$, $\forall g(x) \leq 0$, and $\sup_\tau \mathcal{G}(\tau, g(x)) = +\infty$, $\forall g(x) > 0$;
similarly, $\mathcal{H}(\nu, h(x))$ is a function of $h(x)$ and ν, satisfying

(d) Concave with respect to ν, if $h(x) = 0$.

(e) $\sup_\nu \mathcal{H}(\nu, 0) = 0$, and $\sup_\nu \mathcal{H}(\nu, h(x)) = +\infty$, $\forall h(x) \neq 0$.

The concavity of τ is relaxed; $\mathcal{G}(0, g(x))$, $\mathcal{G}(\tau, 0)$ are not required to be zero.
Assuming $\mathcal{G}(\tau, g(x))$ and $\mathcal{H}(\nu, h(x))$ are continuous and differentiable.

Theorem 1. *Assume the constraints are strictly feasible. If a generalized Lagrangian $\mathcal{L}(x, \tau, \nu)$ under conditions (1)-(5) is closed and proper, it satisfies strong duality between primal and dual problems, i.e.,*

$$\inf_x \sup_{\tau,\nu} \mathcal{L}(x, \tau, \nu) = \sup_{\tau,\nu} \inf_x \mathcal{L}(x, \tau, \nu) \tag{3.3}$$

Proof The primal function is $\mathcal{L}_p(x) = \sup_{\tau,\nu} \mathcal{L}(x, \tau, \nu)$, the primal problem $\inf_x \mathcal{L}_p(x)$ is equivalent to the original problem (3.2) by conditions (3) and (5),

$$\inf_x \mathcal{L}_P(x) = \inf\{f(x), +\infty\} = \inf_x f(x), \forall x \in \mathcal{F} \tag{3.4}$$

Since $\mathcal{L}(x, \tau, \nu)$ is closed, there exists (x^*, τ^*, ν^*) as a solution to (3.4). The dual function is $\mathcal{L}_D(\tau, \nu) = \inf_x \mathcal{L}(x, \tau, \nu) = \inf_x \{f(x) + \mathcal{G}(\tau, g(x)) + \mathcal{H}(\nu, h(x))\}$. Since $\mathcal{L}(x, \tau, \nu)$ is proper, $\mathcal{L}(x, \tau, \nu) > -\infty$, and x' minimizes $\mathcal{L}(x, \tau, \nu)$ for fixed τ and ν, then its gradient $\nabla_x \mathcal{L}(x, \tau, \nu)$ vanishes at x',

$$\nabla_x f\left(x'\right) + \nabla_x \mathcal{G}\left(\tau, g\left(x'\right)\right) + \nabla_x \mathcal{H}\left(\nu, h\left(x'\right)\right) = \mathbf{0}_n \tag{3.5}$$

where $x'(\tau, \nu)$ is a function of τ and ν. Equation (3.5) is guaranteed to have solution for any feasible (τ, ν) including (τ^*, ν^*). Therefore by (3.4) and (3.5),

$$\inf_x \sup_{\tau,\nu} \mathcal{L}(x, \tau, \nu) = \mathcal{L}\left(x'\left(\tau^*, \nu^*\right), \tau^*, \nu^*\right) \tag{3.6}$$

and due to max-min inequality [2]

$$\mathcal{L}\left(x'\left(\tau^*, \nu^*\right), \tau^*, \nu^*\right) \leq \sup_{\tau,\nu} \mathcal{L}_D(\tau, \nu) \leq \inf_x \sup_{\tau,\nu} \mathcal{L}(x, \tau, \nu) \tag{3.7}$$

$\inf_x \sup_{\tau,\nu} \mathcal{L}(x, \tau, \nu) = \sup_{\tau,\nu} \inf_x \mathcal{L}(x, \tau, \nu)$

[2] S. Boyd and L.Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 978-0-521-83378-3

Remark 1. *Being closed and proper is a sufficient condition for strong duality. Besides, one major difference between the classic and generalized Lagrangians is that a dual function is concave concerning its multipliers in the classic Lagrange multiplier method. In contrast, a generalized $\mathcal{L}_D(\tau, \nu)$ is not necessarily concave with respect to (τ, ν) but a function with upper bound according to the proof.*

KKT Conditions for Generalized Lagrangians The generalized KKT conditions of the GLMM for the optimization problem can be derived

$$g\left(x^*\right) \leq 0 \tag{3.8}$$

$$\mathcal{G}\left(\tau^*, g\left(x^*\right)\right) = 0 \tag{3.9}$$

$$h\left(x^*\right) = 0 \tag{3.10}$$

$$\mathcal{H}\left(\nu^*, h\left(x^*\right)\right) = 0 \tag{3.11}$$

$$\nabla_x f\left(x^*\right) + \nabla_x \mathcal{G}\left(\tau^*, g\left(x^*\right)\right) + \nabla_x \mathcal{H}\left(x^*, g\left(x^*\right)\right) = \mathbf{0}_n. \tag{3.12}$$

3.2 A GLMM Reformulation for the L_1 SVC

There is an attempt to use a Generalized Lagrange Multiplier Method or the L_1 SVC to add an elastic net Penalization term to capitalize on the benefits this regularization brings to the coefficients.

It was determined that it is best to add the elastic net term to the Lagrangian directly for the Dual Problem to be assembled and the conditions satisfied.

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + C \sum_{k=1}^{N} \xi_k \\ &- \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \xi_k \right) \\ &- \sum_{k=1}^{N} \eta_k \xi_k \\ &+ \lambda \left[(1-\omega) \sum_{j=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{j=1}^{N} \alpha_k^2 \right]\end{aligned} \tag{3.13}$$

The added term keeps a convex formulation where $\lambda \geq 0$ and $0 \leq \omega \leq 1$.

The proposed term has such a formulation that the configuration of the hyperparameters λ and ω at specific values will make the dual problem retake its original formulation from the classic L_1 SVC.

Having established the values the hyperparameters to the added term in the Lagrangian can be, it is required to verify that the proposal meets the definition of the GLMM.

Monotonically Decreasing with respect to $g(x)$:

$g_a(x) = [y_k(w^T\phi(x_k) + b) - 1 + \xi_k]$

$g_b(x) = \xi_k$

$g_c(x) = \lambda \left[(1-\omega)\sum_{k=1}^{N} \alpha_k + \frac{\omega}{2}\sum_{k=1}^{N} \alpha_k^2\right]$

As:

$g(x) = \langle g_a(x), g_b(x), g_c(x)\rangle$

And,

$$\nabla g(x)\mathcal{G}(\tau, g(x)) = \left| \begin{array}{c} \nabla g_a(x)\mathcal{G}(\tau, g(x)) \\ \nabla g_b(x)\mathcal{G}(\tau, g(x)) \\ \nabla g_c(x\mathcal{G}(\tau, g(x)) \end{array} \right| \tag{3.14}$$

Now every expression gets the partial derivative with respect to their Lagrange multiplier to obtain the following:

- $\alpha g_a(x) = -\alpha_k[y_k(w^T\phi(x_k) + b) - 1 + \xi_k]$
  - $\nabla g_a(x)\mathcal{G}(\alpha, g(x) = -\alpha_k$
- $\eta g_b(x) = -\eta_k\xi_k$
  - $\nabla g_b(x)\mathcal{G}(\eta, g(x) = -\eta_k$

The first condition of monotonically decreasing with respect to

$g(x)$ is met since the lagrangian multipliers are negative.

The second condition being convex with respect to α, η if $g(x) \geq 0$ is obtained by getting the second partial derivative for all expressions of the Lagrangian:

Looking at the results obtained after getting the second partial derivative, it can be concluded that the equation is convex with respect to α, η is $g(x) \geq 0$ since the results were non-negative.

The third condition is $\sup_\tau \mathcal{G}(\tau, g(x)) = 0$, $\forall g(x) \leq 0$, and $\sup_\tau \mathcal{G}(\tau, g(x)) = +\infty$, $\forall g(x) > 0$.

- $\sup_\tau \mathcal{G}(\tau, g(x)) = 0$, $\forall g(x) \leq 0$. Since the Lagrange multipliers are non-negative, and having $g(x) \leq 0$, the lower boundary is set by zero. Therefore, the $\sup_\tau \mathcal{G}(\tau, g(x)) = 0$.

- $\sup_\tau \mathcal{G}(\tau, g(x)) = +\infty$, $\forall g(x) > 0$. Since the Lagrangian multipliers are non-negative, and having $g(x) > 0$, the set has no upper bound. Therefore, the $\sup_\tau \mathcal{G}(\tau, g(x)) = \infty$.

  Once the proposed Lagrangian has met the Generalized Lagrange Multiplier Method conditions, it is now time to proceed with the First Order Conditions of our problem.

(a) $\nabla w \mathcal{L} = w - \sum_{k=1}^{N} \alpha_k y_k \phi(x_k) = 0 \rightarrow \mathrm{w} = \sum_{k=1}^{N} \alpha_k y_k \phi(x_k)$

(b) $\frac{\partial}{\partial b} \mathcal{L} = \sum_{k=1}^{N} \alpha_k y_k = 0$

(c) $\frac{\partial}{\partial \xi_k} \mathcal{L} = C - \alpha_k - \eta_k = 0 \rightarrow \mathrm{C} = \alpha_k + \eta_k$ which implies $\rightarrow \eta_k = C - \alpha_k$

Variable C Replaced in the Lagrangian:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}w^T w + (\alpha_k+\eta_k)\sum_{k=1}^{N}\xi_k \\ & - \sum_{k}^{N}\alpha_k(y_k[w^T\phi(x_k)+b]-1+\xi_k) \\ & - \sum_{k=1}^{N}\eta_k\xi_k \\ & + \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.15}$$

Then, variable w is replaced in the Lagrangian (3.15):

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}[\sum_{k=1}^{N}\alpha_k y_k\phi(x_k)]^T[\sum_{l=1}^{N}\alpha_l y_l\phi(x_l)] + (\alpha_k+\eta_k)\sum_{k=1}^{N}\xi_k \\ & - \sum_{k}^{N}\alpha_k(y_k[\sum_{l=1}^{N}\alpha_l y_l\phi(x_l)^T\phi(x_k)+b]-1+\xi_k) \\ & - \sum_{k=1}^{N}\eta_k\xi_k \\ & + \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.16}$$

Once the variables are replaced, they are now grouped and reduced to obtain the following terms:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}[\sum_{k=1}^{N}\alpha_k y_k\phi(x_k)]^T[\sum_{l=1}^{N}\alpha_l y_l\phi(x_l)] + \sum_{k=1}^{N}\alpha_k\xi_k + \sum_{k=1}^{N}\eta_k\xi_k \\ & - \sum_{k}^{N}\alpha_k y_k\phi(x_k)^T\sum_{l=1}^{N}\alpha_l y_l\phi(x_l) + \sum_{k=1}^{N}\alpha_k - \sum_{k=1}^{N}\alpha_k\xi_k \\ & - \sum_{k=1}^{N}\eta_k\xi_k \\ & + \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.17}$$

After the terms in (3.17) are grouped and reduced, the following

is obtained:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) = &-\frac{1}{2}\sum_{k=1}^{N}\alpha_k\alpha_l y_k y_l \phi(x_k)]^T\phi(x_l) + \sum_{k=1}^{N}\alpha_k \\ &+\lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.18}$$

For the equation (3.18) we observe the added term providing the elastic net penalization to the dual problem, however, there is still a redundant term in form of $\sum \alpha_k$ that was already providing a LASSO penalization to the dual function, making it possible for some α_k to be = 0. In order for it to be taken out of the dual,problem a slight modification to the original Lagrangian formulation is proposed.

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =&\frac{1}{2}w^T w + C\sum_{k=1}^{N}\xi_k \\ &-\sum_{k}^{N}\alpha_k\left(y_k\left[w^T\phi(x_k)+b\right]-1+\xi_k+1\right) \\ &-\sum_{k=1}^{N}\eta_k\xi_k \\ &+\lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.19}$$

For the equation (3.19) a trivial solution is incorporated to the original Lagrangian in the form of a $+1$ to one of the primal constraints, hoping that this will take out the redundant $\sum \alpha_k$ from the Dual Problem. To corroborate what has been proposed, the same previous steps to get to the Dual Problem are followed:

The first step in grouping variables and reducing terms is replacing the C in (3.19):

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}w^T w + (\alpha_k+\eta_k)\sum_{k=1}^{N}\xi_k \\ &- \sum_{k}^{N}\alpha_k(y_k[w^T\phi(x_k)+b]-1+\xi_k+1) \\ &- \sum_{k=1}^{N}\eta_k\xi_k \\ &+ \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.20}$$

Variable w replaced in the Lagrangian (3.20):

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}[\sum_{k=1}^{N}\alpha_k y_k\phi(x_k)]^T[\sum_{l=1}^{N}\alpha_l y_l\phi(x_l)] + (\alpha_k+\eta_k)\sum_{k=1}^{N}\xi_k \\ &- \sum_{k}^{N}\alpha_k(y_k[\sum_{l=1}^{N}\alpha_l y_l\phi(x_l)^T\phi(x_k)+b]-1+\xi_k+1) \\ &- \sum_{k=1}^{N}\eta_k\xi_k \\ &+ \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.21}$$

The equation's variables are now grouped and reduced to obtain the following term:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}[\sum_{k=1}^{N}\alpha_k y_k\phi(x_k)]^T[\sum_{l=1}^{N}\alpha_l y_l\phi(x_l)] + \sum_{k=1}^{N}\alpha_k\xi_k + \sum_{k=1}^{N}\eta_k\xi_k \\ &- \sum_{k}^{N}\alpha_k y_k\phi(x_k)^T\sum_{l=1}^{N}\alpha_l y_l\phi(x_l) + \sum_{k=1}^{N}\alpha_k - \sum_{k=1}^{N}\alpha_k\xi_k - \sum_{k=1}^{N}\alpha_k \\ &- \sum_{k=1}^{N}\eta_k\xi_k \\ &+ \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.22}$$

Now that the term $\sum\alpha_k$ has been canceled out from the equation

(3.22), the function is reduced to the following form:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) = &-\frac{1}{2}\sum_{k=1}^{N}\alpha_k\alpha_l y_k y_l \phi(x_k)^T\phi(x_l) \\ &+\lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.23}$$

We recall the primal feasibility conditions from the original primal formulation:

$$\begin{aligned} y_k[w^T\phi(x_k)+b] &\geq 1-\xi_k,\ k=1,\ldots,N \\ \xi_k &\geq 0,\ k=1,\ldots,N \end{aligned} \tag{3.24}$$

The Dual Feasibility Conditions come from the Non-negative Lagrangian Multipliers; these do not change from the classical L_1 SVC formulation.

$$\begin{aligned} \alpha_k &\geq 0,\ k=1,\ldots,N \\ \eta_k &\geq 0,\ k=1,\ldots,N \\ C-\alpha_k = \eta_k \geq 0, \rightarrow\ C-\alpha_k &\geq 0 \\ 0 \leq \alpha_k &\leq C \end{aligned} \tag{3.25}$$

Finally, with the primal and dual feasibility conditions established, the following Dual Problem is obtained:

$$\begin{aligned}\max_{\alpha}\ &\frac{1}{2}\sum_{k,l=1}^{N}\alpha_k\alpha_l y_k y_l \phi(x_k)^T\phi(x_l) + \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right] \\ &\text{s.t.}\ \sum_{k=1}^{N}\alpha_k y_k = 0 \\ &0 \leq \alpha_k \leq C,\ k=1,\ldots,N\end{aligned} \tag{3.26}$$

he following Complementary Slackness condition displays the relation between the dual variable and the primal constraints from the L_1 SVC that must be satisfied.

$$\begin{aligned} \alpha_k \left[y_k(w^T \phi(x_k) + b) - 1 + \xi_k \right] &= 0 \\ \alpha_k &\geq 0 \\ \eta_k &\geq 0 \\ \xi_k &\geq 0 \end{aligned} \tag{3.27}$$

By using the complementary slackness conditions, b can be calculated as the SVC classical form:

$$b = \frac{1}{|U|} \sum_{i \in U} (y_k - w^T x_k) \tag{3.28}$$

3.3 A GLMM Reformulation for the L_2 SVC

$$\begin{aligned} \mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) = & \frac{1}{2} w^T w + \frac{C}{2} \sum_{k=1}^{N} \xi_k^2 \\ & - \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \xi_k \right) \\ & + \lambda \left[(1 - \omega) \sum_{j=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{j=1}^{N} \alpha_k^2 \right] \end{aligned} \tag{3.29}$$

Just as it was done with the L_1 GLMM SVC, it will be verified if the proposal on the L_2 machine meets the definition of the GLMM.

Monotonically Decreasing with respect to $g(x)$:

- $g_a(x) = [y_k(w^T \phi(x_k) + b) - 1 + \xi_k]$
- $g_b(x) = \xi_k$
- $g_c(x) = \lambda \left[(1 - \omega) \sum_{k=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{k=1}^{N} \alpha_k^2 \right]$

As:

$g(x) = \langle g_a(x), g_b(x), g_c(x) \rangle$

And,

$$\nabla g(x)\mathcal{G}(\tau, g(x)) = \left| \begin{array}{c} \nabla g_a(x)\mathcal{G}(\tau, g(x)) \\ \nabla g_b(x)\mathcal{G}(\tau, g(x)) \\ \nabla g_c(x = \mathcal{G}(\tau, g(x)) \end{array} \right| \tag{3.30}$$

Now every expression gets the partial derivative with respect to their Lagrange multiplier to obtain the following:

$$\alpha g_a(x) = -\alpha_k[y_k(w^T\phi(x_k) + b) - 1 + \xi_k]$$

$$\nabla g_a(x)\mathcal{G}(\alpha, g(x) = -\alpha_k$$

$$\eta g_b(x) = -\eta_k\xi_k$$

$$\nabla g_b(x)\mathcal{G}(\eta, g(x) = -\eta_k$$

The first condition of monotonically decreasing with respect to $g(x)$ is met since the lagrangian multipliers are negative.

The second condition being convex with respect to α, η if $g(x) \geq 0$ is obtained by getting the second partial derivative for all expressions of the Lagrangian:

$$\alpha g_i(x) = -\alpha_k[y_k(w^T\phi(x_k) + b) - 1 + \xi_k]$$
$$\nabla^2 g_i(x)\mathcal{G}(\alpha, g(x)) = 0$$

$$\eta g_j(x) = -\eta_k\xi_k$$

$$\nabla^2 g_j(x)\mathcal{G}(\eta, g(x) = 0$$

$$\eta g_m(x) = -\lambda\left[(1-\omega)\sum_{k=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{k=1}^{N}\alpha_k^2\right]$$

$$\nabla^2 g_m(x)\mathcal{G}(\lambda, \omega, g(x) = \lambda\omega$$

Looking at the results obtained after getting the second partial derivative, it can be concluded that the equation is convex with respect to α, η is $g(x) \geq 0$ since the results were non-negative.

The third condition is $\sup_\tau \mathcal{G}(\tau, g(x)) = 0$, $\forall g(x) \leq 0$, and $\sup_\tau \mathcal{G}(\tau, g(x)) = +\infty$, $\forall g(x) > 0$.

- $\sup_\tau \mathcal{G}(\tau, g(x)) = 0, \forall g(x) \leq 0$. Since the Lagrange multipliers are non-negative, and having $g(x) \leq 0$, the lower boundary is set by zero. Therefore, the $\sup_\tau \mathcal{G}(\tau, g(x)) = 0$.

- $\sup_\tau \mathcal{G}(\tau, g(x)) = +\infty, \forall g(x) > 0$. Since the Lagrangian multipliers are non-negative, and having $g(x) > 0$, the set has no upper bound. Therefore, the $\sup_\tau \mathcal{G}(\tau, g(x)) = \infty$.
  Once the proposed Lagrangian has met the Generalized Lagrange Multiplier Method conditions, it is now time to proceed with the First Order Conditions of our problem.
  * $\nabla w \mathcal{L} = w - \sum_{k=1}^{N} \alpha_k y_k \phi(x_k) = 0 \rightarrow \mathrm{w} = \sum_{k=1}^{N} \alpha_k y_k \phi(x_k)$
  * $\frac{\partial}{\partial b}\mathcal{L} = \sum_{k=1}^{N} \alpha_k y_k = 0$
  * $\frac{\partial}{\partial \xi_k}\mathcal{L} = C - \alpha_k - \eta_k = 0 \rightarrow \mathrm{C} = \alpha_k + \eta_k$ which implies $\rightarrow \eta_k = C - \alpha_k$

  The First Order Conditions will be established as follows:

  * $\nabla w \mathcal{L} = w - \sum_{k=1}^{N} \alpha_k y_k \phi(x_k) = 0 \rightarrow \mathrm{w} = \sum_{k=1}^{N} \alpha_k y_k \phi(x_k)$
  * $\frac{\partial}{\partial b}\mathcal{L} = \sum_{k=1}^{N} \alpha_k y_k = 0$
  * $\frac{\partial}{\partial \xi_k}\mathcal{L} = C\xi_k - \alpha_k = 0 \rightarrow \mathrm{C}\xi_k = \alpha_k$ which implies $\rightarrow \xi_k = \alpha_{k\,\overline{C}}$

  Now that the variables have been canceled due to the First Order conditions, they can be replaced in the Lagrangian Function. Replacing the variable ξ_k:

$$\begin{aligned}\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) =& \frac{1}{2} w^T w + \frac{C}{2} \sum_{k=1}^{N} \frac{\alpha_k^2}{C^2} \\ & - \sum_{k}^{N} \alpha_k \left(y_k \left[w^T \phi(x_k) + b \right] - 1 + \frac{\alpha_k}{C} \right) \\ & + \lambda \left[(1-\omega) \sum_{j=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{j=1}^{N} \alpha_k^2 \right]\end{aligned} \tag{3.31}$$

  Before replacing the variable w in the Lagrangian (3.31) the equation is grouped and reduced to have the following expression:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}w^T w + \frac{C}{2}\sum_{k=1}^{N}\frac{\alpha_k^2}{C^2} \\ &- \sum_{k}^{N}\alpha_k y_k w^T\phi(x_k) - \sum_{k=1}^{N}\alpha_k y_k b \\ &+ \sum_{k=1}^{N}\alpha_k - \frac{1}{C}\sum_{k=1}^{N}\alpha_k^2 \\ &+ \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.32}$$

Replace the First Order Condition on b to obtain:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& \frac{1}{2}w^T w + \frac{1}{2C}\sum_{k=1}^{N}\alpha_k^2 \\ &- \sum_{k}^{N}\alpha_k y_k w^T\phi(x_k) \\ &+ \sum_{k=1}^{N}\alpha_k - \frac{1}{C}\sum_{k=1}^{N}\alpha_k^2 \\ &+ \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.33}$$

Now the w variable is replaced, it gets reduced to the following term:

$$\begin{aligned}\mathcal{L}(w,b,\xi_k,\alpha_k,\eta_k) =& -\frac{1}{2}\sum_{k,l=1}^{N}\alpha_k\alpha_{lk}y_k y_l\phi(x_k)^T\phi(x_l) \\ &- \frac{1}{2C}\sum_{k=1}^{N}\alpha_k^2 + \sum_{k=1}^{N}\alpha_k \\ &+ \lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right]\end{aligned} \tag{3.34}$$

If $k = l$:

$$
\begin{aligned}
&- \frac{1}{2}\sum_{k=1}^{N} \alpha_k^2 y_k^2 \phi(x_k)\phi(x_k) \\
&- \frac{1}{2}\sum_{k,l=1}^{N} \alpha_k \alpha_l y_k y_l \phi(x_k)^T \phi(x_l) \\
&- \frac{1}{2C}\sum_{k=1}^{N} \alpha_k^2 + \sum_{k=1}^{N} \alpha_k \\
&+ \lambda \left[(1-\omega) \sum_{j=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{j=1}^{N} \alpha_k^2 \right]
\end{aligned}
\tag{3.35}
$$

If $k \neq l$:

$$
\begin{aligned}
&- \frac{1}{2}\sum_{k,l=1}^{N} \alpha_k \alpha_l y_k y_l \phi(x_k)^T \phi(x_l) \\
&- \frac{1}{2C}\sum_{k=1}^{N} \alpha_k^2 + \sum_{k=1}^{N} \alpha_k \\
&+ \lambda \left[(1-\omega) \sum_{j=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{j=1}^{N} \alpha_k^2 \right]
\end{aligned}
\tag{3.36}
$$

Following the case where $k = l$ the terms can be grouped into this form:

$$
\begin{aligned}
\mathcal{L}(w, b, \xi_k, \alpha_k, \eta_k) = &- \frac{1}{2}\sum_{k,l=1}^{N} \alpha_k \alpha_{lk} y_k y_l \phi(x_k)]^T \phi(x_l) \\
&- \frac{1}{2}\sum_{k=1}^{N} \alpha_k^2 + (y_k^2 \phi(x_k)\phi(x_k) + \frac{1}{C}) \\
&+ \sum_{k=1}^{N} \alpha_k \\
&+ \lambda \left[(1-\omega) \sum_{j=1}^{N} \alpha_k + \frac{\omega}{2} \sum_{j=1}^{N} \alpha_k^2 \right]
\end{aligned}
\tag{3.37}
$$

The following dual problem is obtained:

$$\begin{aligned}\max_{\alpha} -\frac{1}{2}\alpha_l\alpha_l y_k y_l \left[\phi(x_k)^T\phi(x_l) + \frac{\delta_{kj}}{C}\right] + \sum_{k=1}^{N}\alpha_k \\ +\lambda\left[(1-\omega)\sum_{j=1}^{N}\alpha_k + \frac{\omega}{2}\sum_{j=1}^{N}\alpha_k^2\right] \\ \text{s.t. } \sum_{k=1}^{N}\alpha_k y_k = 0 \\ \alpha_k \geq 0,\ k = 1,\ldots,N\end{aligned} \tag{3.38}$$

We can observe that this new proposal of SVC based on the original L_2 SVC does not offer a new structure since it is another L_2 SVC with the same elastic net regularization and without the box constraints. Therefore, we will not use this new SVC proposal for further study.

4 Application

Contents

4.1 Types of kernels

Kernel functions are useful in various contexts since they offer a straightforward transition from linearity to non-linearity for algorithms that can be expressed as dot products.

They transform data into higher-dimensional spaces, making it simpler to separate it there. This mapping can result in infinite-dimensional spaces because there are no restrictions on its shape.

The choice of a kernel depends on the problem because it depends on what we are trying to model. For example, a polynomial kernel allows modeling feature conjunctions up to the polynomial's order. Radial basis functions allow picking out circles (or hyper-spheres) – in contrast with the Linear kernel, which allows only picking out outlines (or hyper-planes).

A few kernel functions will be mentioned, along with some of their properties.

4.1.1 Linear Kernel

This one is the simplest kernel function. It is given by the inner product $\langle x, y \rangle$ plus an optional constant **c**.

$$k(x, y) = x^T y + c \tag{4.1}$$

4.1.2 Polynomial Kernel

The polynomial kernel is a non-stationary kernel. These are well-suited for problems where all the training data is normalized.

Adjustable parameters are the slope α, the constant term **c** and the polynomial degree **d**.

$$k(x, y) = \left(\alpha x^T y + c\right)^d \tag{4.2}$$

4.1.3 Radial Basis Kernel

The radial basis function kernel, or RBF kernel, is a popular kernel function used in various kernelized learning algorithms. The adjustable parameter sigma significantly impacts the kernel's efficiency and should be fine-tuned to the problem at hand. When the exponential is overestimated, it behaves almost linearly, and the higher-dimensional projection loses its non-linear power. On the other hand, if the function is undervalued, it will lack regularization, and the decision boundary will be extremely sensitive.

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right) \tag{4.3}$$

4.1.4 Kernel Implementation

Gaussian (RBF) (4.3) Kernel will be implemented in this paper in **Python**, where the free parameter in the model is **sigma**.

4.2 Grid Search for hyperparameter Tuning

It isn't easy to select optimal hyperparameter values, C, and σ values for Gaussian (RBF) kernels along with λ and ω for the regularization.

There is no way of knowing in advance which value is appropriate for the hyperparameters. A Grid Search approach will be used for automatic parameter selection. This can be achieved with Python's **GridSearchCV** package.

Grid Search is simply an exhaustive search that considers all parameter value combinations. Some performance metric, typically determined by cross-validation on the training set, must direct a grid search algorithm.

Since the parameter space of a machine learning algorithm may include real-valued or unbounded value spaces, it may be necessary to manually set bounds and discretize certain parameters before applying the grid search.

Although the grid search method is the simplest to use and comprehend, it is sadly ineffective when there are a lot of parameters.

4.3 Implementation Issues L_1 Extended SVC

Proceeding with the new GLMM SVC Python implementation using the formulation in (2.10).

The free hyperparameters $\mathbf{c}$, σ and the new parameters λ and ω will be later tuned using grid search optimization algorithms. Code seen in appedinx .1.

The implementation is important for this thesis to verify that the proposal of the elastic net regularization can converge with this extended SVC. Also, this will help us prove that the formulation of the new model has a set of parameters that allow to return to the classical SVC when $\lambda = 1$ and $\omega = 0$.

5 Examples

Contents

To test the performance of the model proposed in this work, two known data sets are used to solve a classification problem. The Extended-Lagrangian-SVC Python implementation is based on the formulation in (2.10). The C, ω, and λ free hyperparameters were tuned using an Exhaustive Grid Search [1] and a Radial Based Kernel.

[1] scikit-learn developers (BSD License). Gridsearchcv, 2007 - 2023. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Four performance measures are going to be used for performance evaluation, which are based on the confusion matrix[2]:

[2] Source: Understanding Confusion Matrix by Sarang Narkhede on Towards Data Science

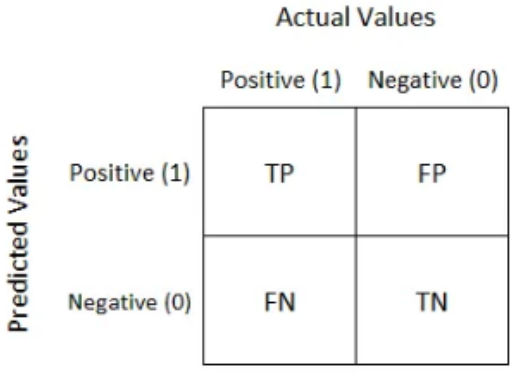


Figure 5.1: Confusion Matrix

* *Accuracy*: measures how often the model is correct in its predictions. It is calculated as the number of correct predictions divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

* *Precision*: measures how many predicted positive results are actually positive. It is calculated as the number of true positives divided by the number of true positives plus false positives.

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

* *Recall*: measures how many actual positive results are correctly predicted as positive. It is calculated as the number of true positives divided by the number of true positives plus false negatives.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

* *F1 Score*: is a harmonic mean of precision and recall, providing a single metric that balances precision and recall.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.4}$$

5.1 Case Analysis: Breast Cancer Dataset

The well-known Breast Cancer dataset is used; it is a classic and very easy binary classification dataset. This dataset can be accessed from the *scikit-learn* library [3]. The objective is to predict if a breast mass is cancerous or not.

[3] Olvi L. Mangasarian Dr. William H. Wolberg, W. Nick Street. sklearn.datasets.load_breast_cancer, 1995. URL https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

5.1.1 Attribute Information

There are 569 samples and 30 numeric, predictive attributes and the target variable (class).

(a) ID number
(b) Diagnosis (M = malignant, B = benign)
(c) Ten real-valued features are computed for each cell nucleus:
    * radius (mean of distances from the center to points on the perimeter)
    * texture (standard deviation of gray-scale values)
    * perimeter

* area
* smoothness (local variation in radius lengths)
* compactness (perimeter^2 / area - 1.0)
* concavity (severity of concave portions of the contour)
* concave points (number of concave portions of the contour)
* symmetry
* fractal dimension ("coastline approximation" - 1)

5.1.2 *Model Evaluation*

The model proposed in this work is implemented in Python and is compared against the classical SVC model and decision tree-based models such as Random Forest[4,5] and XGBoost[6]. Decision tree-based models have been selected as a tool for comparison due to them being highly interpretable and their exhaustive search of the best partition possible through a defined criterion. The partitioning of the data into training and test data sets using a ratio of 70% and 30%, respectively.

The comparison between the models was made using the performance metrics *Accuracy*, *Precision*, *Recall*, and *F*1, where the summary of the results obtained is presented in Table 5.1. The table shows that the proposed model improves metrics compared to the commonly applied SVC.

After performing an Exhaustive Grid Search to find the better hyperparameters, these were the ones that offered the better results. Code seen in appendix .2, appendix .3 and appendix .4[7]:

* SVC
  - kernel = RBF
  - C = 2
  - γ = 0.05
* Extended SVC
  - kernel = RBF
  - C = 5
  - λ = 0.5
  - ω = 0.1
  - γ = 0.05
* Random Forest
  - n estimators = 100
  - criterion = entropy

[4] Random forest is a form of a bagged tree model that uses an ensemble method consisting creating random features in every individual tree ensuring low correlation between them, to finally average the predictions typically with highly accurate results.

[5] IBM. What is a random forest? URL https://www.ibm.com/topics/random-forest

[6] XGBoost stands for eXtreme Gradient Boosting and it is a model that aims to greatly optimize the loss function, using a weak learner to make predictions and minimize the loss when adding trees to the ensemble method by a gradient descent procedure.

[7] The γ* represents the parameter of the XGBoost Model. The γ represents the parameter of the RBF Kernel

- min samples split = 2
- min samples leaf = 2

* XGBoost
  - max depth = 5
  - learning rate = 0.1
  - γ^* = 0.25
  - reg lambda = 0
  - scale pos weight = 3

Table 5.1: Metric Performance Comparison for Breast Cancer Dataset

Model/Metric	Acc.	Prec.	Rec.	F1
SVC	0.9707	0.9813	0.9722	0.9767
Extended SVC	0.9766	0.9814	0.9814	0.9814
Random Forest	0.9649	0.9555	0.9907	0.9727
XGBoost	0.9707	0.9639	0.9907	0.9771

This document's proposal outperformed the classical SVC and the decision tree-based models by a small margin for this dataset. As stated in Chapter 3, by mirroring the C and γ hyperparameters from the classical SVC and setting the values of the Extended SVC $\lambda = 1$ and $\omega = 0$, the scores obtained are identical to those of the SVC.

5.2 *Case Analysis: Parkinsons Dataset*

For this following dataset, a binary target variable set comprises a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD). This dataset can be accessed from the *UCI Machine Learning Repository* library [8]. Each column in the table is a particular voice measure, and each row corresponds to one of 195 voice recordings from the subjects. The status column is the target variable containing whether the subject has Parkinson's or not.

[8] Eric J. Hunter Lorraine O. Ramig Max A. Little, Patrick E. McSharry. Parkinsons data set, 2008. URL https://archive.ics.uci.edu/ml/datasets/parkinsons

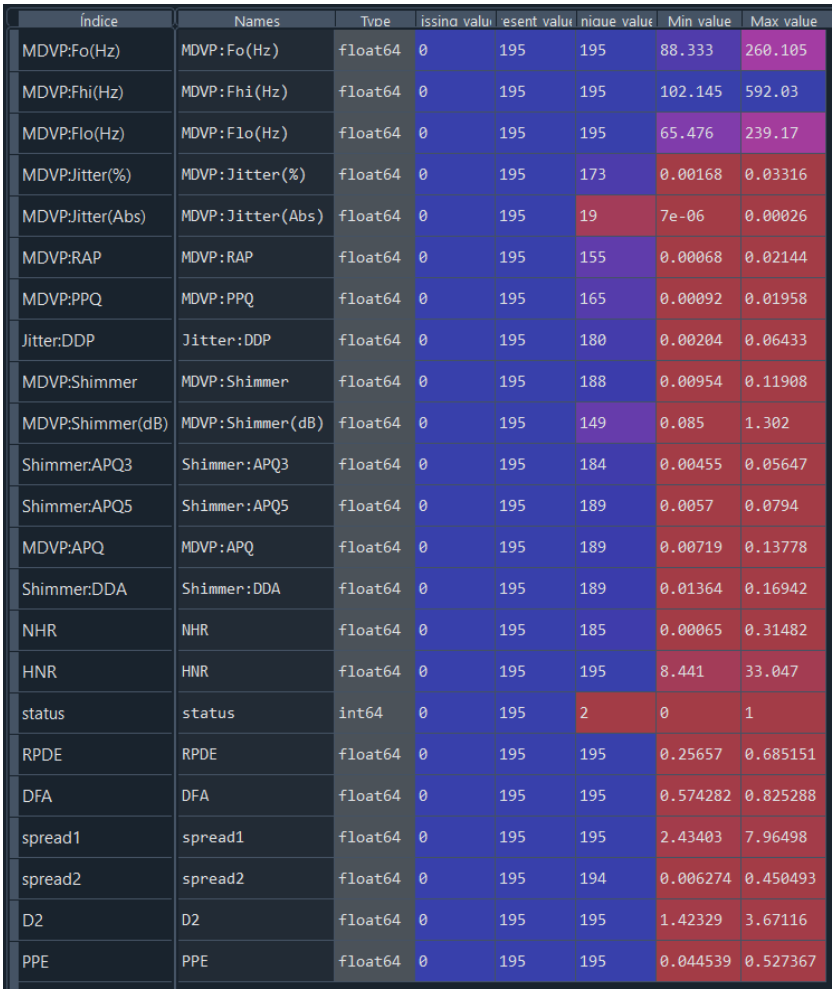
5.2.1 *Attribute Information*

The target variable of this data set is 'status'. Health status of the subject (one) - Parkinson's, (zero) - healthy.

(a) name - ASCII subject name and recording number.

(b) MDVP: Fo(Hz) - Average vocal fundamental frequency.
(c) MDVP: Fhi(Hz) - Maximum vocal fundamental frequency.
(d) MDVP: Flo(Hz) - Minimum vocal fundamental frequency.
(e) MDVP: Jitter(Percentage), MDVP: Jitter(Abs), MDVP: RAP, MDVP: PPQ, Jitter: DDP - Several measures of variation in fundamental frequency.
(f) MDVP: Shimmer, MDVP: Shimmer(dB), Shimmer: APQ3, Shimmer: APQ5, MDVP: APQ, Shimmer: DDA - Several measures of variation in amplitude.
(g) NHR, HNR - Two measures of noise ratio to tonal components in the voice.
(h) RPDE, D2 - Two nonlinear dynamical complexity measures.
(i) DFA - Signal fractal scaling exponent.
(j) spread1,spread2, PPE - Three nonlinear measures of fundamental frequency variation.

A brief quality report was performed on the Parkinson's dataset to check for missing values and/or asymmetry. Code seen in appendix .5



Índice	Names	Type	issing valu	esent valu	nique value	Min value	Max value
MDVP:Fo(Hz)	MDVP:Fo(Hz)	float64	0	195	195	88.333	260.105
MDVP:Fhi(Hz)	MDVP:Fhi(Hz)	float64	0	195	195	102.145	592.03
MDVP:Flo(Hz)	MDVP:Flo(Hz)	float64	0	195	195	65.476	239.17
MDVP:Jitter(%)	MDVP:Jitter(%)	float64	0	195	173	0.00168	0.03316
MDVP:Jitter(Abs)	MDVP:Jitter(Abs)	float64	0	195	19	7e-06	0.00026
MDVP:RAP	MDVP:RAP	float64	0	195	155	0.00068	0.02144
MDVP:PPQ	MDVP:PPQ	float64	0	195	165	0.00092	0.01958
Jitter:DDP	Jitter:DDP	float64	0	195	180	0.00204	0.06433
MDVP:Shimmer	MDVP:Shimmer	float64	0	195	188	0.00954	0.11908
MDVP:Shimmer(dB)	MDVP:Shimmer(dB)	float64	0	195	149	0.085	1.302
Shimmer:APQ3	Shimmer:APQ3	float64	0	195	184	0.00455	0.05647
Shimmer:APQ5	Shimmer:APQ5	float64	0	195	189	0.0057	0.0794
MDVP:APQ	MDVP:APQ	float64	0	195	189	0.00719	0.13778
Shimmer:DDA	Shimmer:DDA	float64	0	195	189	0.01364	0.16942
NHR	NHR	float64	0	195	185	0.00065	0.31482
HNR	HNR	float64	0	195	195	8.441	33.047
status	status	int64	0	195	2	0	1
RPDE	RPDE	float64	0	195	195	0.25657	0.685151
DFA	DFA	float64	0	195	195	0.574282	0.825288
spread1	spread1	float64	0	195	195	2.43403	7.96498
spread2	spread2	float64	0	195	194	0.006274	0.450493
D2	D2	float64	0	195	195	1.42329	3.67116
PPE	PPE	float64	0	195	195	0.044539	0.527367

Figure 5.2: Parkinsons Quality Report

There were no missing values in this dataset, and there were some variables with right-skew distribution; however, due to some variables having a normal distribution, it was decided not to treat such asymmetries before running the models.

The target variable under the label of 'status' was imbalanced. 1: 147, 0: 48. To handle such imbalance, an oversampling method was implemented by applying the Adaptive Synthetic Sampling (ADASYN) approach [9]. This technique works according to the density of the minority class instances, generating more synthetic samples in the feature space with a low density of minority samples. Code seen in appendix .6

[9] Kavish111, (2022) *Handling Imbalanced Data with Imbalance-Learn in Python* Analyticsvidhya.com

After having the classes re-balanced at 1: 147, 0: 146, the data was partitioned into Train and Test subsets of 70% and 30%. A Scaling through standardization was applied to the subsets of data once separated as well to improve model convergence.

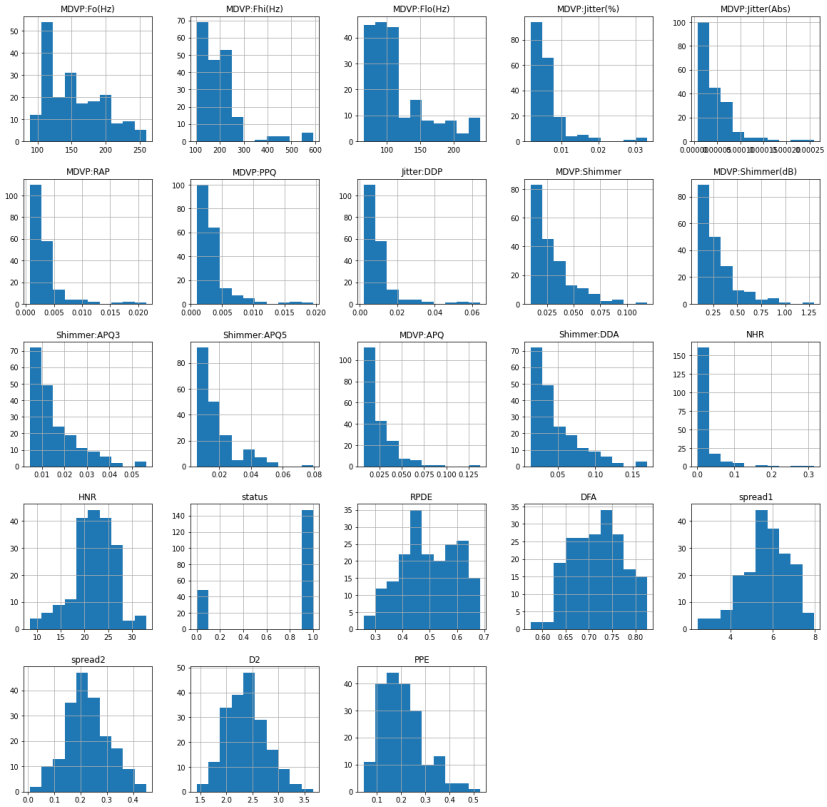


Figure 5.3: Parkinsons Input Variables Skewness

Also, after performing an Exhaustive Grid Search to find the better hyperparameters, these were the configurations that offered the better results:

* SVC
  - kernel = RBF
  - C = 5
  - γ = 0.25
* Extended SVC
  - kernel = RBF
  - C = 2

- λ = 1.5
- ω = 0.1
- γ^* = 0.4

* Random Forest
  - n estimators = 500
  - criterion = entropy
  - min samples split = 2
  - min samples leaf = 2
* XGBoost
  - max depth = 4
  - learning rate = 0.1
  - γ^{**} = 0
  - reg lambda = 0
  - scale pos weight = 1

Table 5.2: Metric Performance Comparison for Parkinsons Dataset

Model/Metric	Acc.	Prec.	Rec.	F1
SVC	0.9772	0.975	0.975	0.975
Extended SVC	0.9756	1	0.9767	0.9876
Random Forest	0.9318	0.9047	0.95	0.9268
XGBoost	0.9204	0.8837	0.95	0.9156

According to what can be observed in table 5.2, the Extended SVC model with the GLMM outperformed the classical SVC by a small margin in all metrics, and the SVC models' scores were notably better than the decision tree-based models as well.

5.3 *Case Analysis: Diabetes Dataset*

The final dataset to put the proposed model to the test is a binary target variable set with eight different health indicators with 768 respective instances. This dataset can be accessed from the *Kaggle Repository* [10]. The Outcome column is the target variable containing whether the subject has Diabetes or not.

[10] National Institute of Diabetes, Digestive, and Vincent Sigillito Kidney Diseases. Diabetes dataser, 1990. URL https://www.kaggle.com/datasets/mathchi/diabetes-data-set

5.3.1 *Attribute Information*

The target variable of this data set is 'Outcome': Health status of the subject (one) - Diabetes, (zero) - healthy.

(a) Pregnancies.
(b) Glucose.
(c) BloodPressure.
(d) SkinThickness.
(e) Insulin.
(f) BMI.
(g) DiabetesPedigreeFunction.
(h) Age.
(i) Outcome.

A brief quality report was performed on the Diabetes dataset to check for any missing values and/or asymmetry.

The diabetes dataset had no missing values, and there were some variables with right-skew distribution; however, due to some variables having a normal distribution, it was decided not to treat such asymmetries before running the models. If skewness was most pronounced among variables, a Box-Cox transformation is recommended to mitigate such asymmetries.

Índice	Names	Type	issing valu	esent valu	nique valu	Min value	Max value
Pregnancies	Pregnancies	int64	0	768	17	0	17
Glucose	Glucose	int64	0	768	136	0	199
BloodPressure	BloodPressure	int64	0	768	47	0	122
SkinThickness	SkinThickness	int64	0	768	51	0	99
Insulin	Insulin	int64	0	768	186	0	846
BMI	BMI	float64	0	768	248	0	67.1
DiabetesPedigreeFunction	DiabetesPedigreeFunction	float64	0	768	517	0.078	2.42
Age	Age	int64	0	768	52	21	81
Outcome	Outcome	int64	0	768	2	0	1

Figure 5.4: Diabetes Quality Report

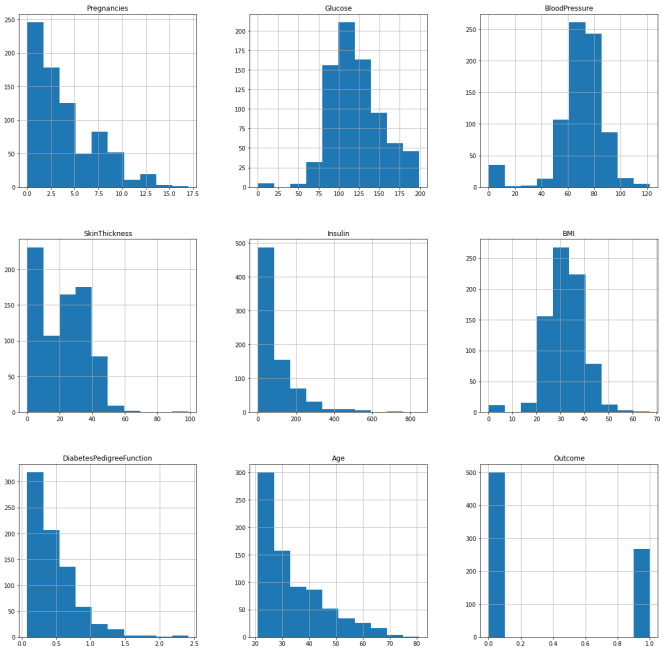


Figure 5.5: Diabetes Input Variables Skewness

The target variable under the label of 'Outcome' was imbalanced. 1: 268, 0: 500. To handle such imbalance, a different oversampling method was introduced by applying the Synthetic Minority Oversampling Technique (SMOTE) approach [11]. This technique picks a random instance from the minority class to have it find its K-Nearest Neighbors from the rest of the minority instances. Once a neighbor gets chosen randomly, it draws the line between those two to have new synthetic examples generated using a convex combination.

[11] Kavish111, (2022) *Handling Imbalanced Data with Imbalance-Learn in Python* Analyticsvidhya.com

After having the classes re-balanced at 1: 500, 0: 500, the data was partitioned into Train and Test subsets of 70% and 30%. A Scaling through standardization was applied to the subsets of data once separated as well to improve model convergence. After performing an Exhaustive Grid Search to find the better hyperparameters, these were the configurations that offered the better results:

* SVC
  - kernel = RBF
  - C = 5
  - $\gamma = 1$
* Extended SVC

  - kernel = RBF
  - C = 1
  - λ = 1.5
  - ω = 0
  - γ = 0.5
* Random Forest
  - n estimators = 200
  - criterion = entropy
  - min samples split = 2
  - min samples leaf = 2
* XGBoost
  - max depth = 4
  - learning rate = 0.1
  - γ^* = 0
  - reg lambda = 0
  - scale pos weight = 3

Table 5.3: Metric Performance Comparison for Diabetes Dataset

Model/Metric	Acc.	Prec.	Rec.	F1
SVC	0.8166	0.7704	0.9155	0.8367
Extended SVC	0.8166	7894	0.8766	0.8307
Random Forest	0.8433	0.8057	0.9155	0.8571
XGBoost	0.8233	0.7643	0.9480	0.8463

This time, decision tree-based models performed slightly better in terms of metrics than the SVC models. Nevertheless, the difference between them was not drastic, ruling out the SVC models for their utilization.

6 Conclusion

This thesis proposed a generalized Lagrange multiplier method and derived generalized KKT conditions for support vector classification, which includes a weighted elastic net regularization structure. The incorporation of a GLMM to introduce an elastic net penalty term proved beneficial to the problem since its formulation allows a combination of their parameters to return to the classic Support Vector Classifier, which means the worst-case scenario for this proposed machine, is the original formulation. It was shown that the extended Lagrange SVC model outperformed the classic SVC models in predicting several datasets. A disadvantage of this new model proposal would be the increasing time of the optimization for the new hyperparameters λ and ω but on the other side, the advantage is that this new elastic net structure gives the possibility to reduce the correlation and the number of support vectors used to create the model.

Appendices

Contents

.1 L_1 Extended SVC code

```
1
2  class SVM_cvxpy:
3
4      def __init__(self, C = 0.1,lamda=0.5, gama=0.1, kernel = "
       linear", **kernel_param):
5          import cvxpy as cp
6          import numpy as np
7          from sklearn.metrics.pairwise import pairwise_kernels
8          from sklearn.utils import check_X_y, check_array
9          self.cp = cp
10         self.np = np
11         self.C = C
12         self.lamda = lamda
13         self.gama = gama
14         self.kernel = kernel
15         self.pairwise_kernels = pairwise_kernels
16         self.kernel_param = kernel_param
17         self.check_X_y = check_X_y
18         self.check_array = check_array
19
20
21     def fit(self, X, y):
22         X, y = self.check_X_y(X, y)
23         # hyperparameters
24         C = self.C
25         lamda = self.lamda
26         gama = self.gama
27         kernel = self.kernel
28         pairwise_kernels = self.pairwise_kernels
```

```
        cp = self.cp
        np = self.np

        # Useful parameters
        ydim = y.shape[0]

        #Preprocessing of label
        a = np.unique(y); c = np.array([1, -1])
        y = np.where(y == a[0], c[0], c[1])
        self.y = y

        # Matrices for the optimizer
        K = pairwise_kernels(X, X, metric = kernel, **self.kernel_param)
        Y = np.outer(y, y)
        Q = np.multiply(Y,K) #Kernel
        q = np.ones(ydim) #primal (alpha)
        A = y.T #first constraint
        b = 0 #first constraint
        noneg = cp.Parameter(nonneg=True)
        G = np.concatenate((np.identity(ydim), -np.identity(ydim))) #second constraint
        h_ = np.concatenate((C*np.ones(ydim), np.zeros(ydim))); #second constraint
        h = h_.reshape(-1, 1)

        # loss function and constraints
        beta = cp.Variable((ydim,1))

        min_fun = 0.5*cp.quad_form(beta, Q) - lamda*((1-gama*q.T) @ beta - gama*q/2 @ beta**2)
        objective = cp.Minimize(min_fun)
        constraints = [A @ beta == b, G @ beta <= h]

        # Solver and solution
        prob = cp.Problem(objective,constraints)
        result = prob.solve()

        # support vectors
        beta_1 = np.array(beta.value)
        indx = abs(beta_1) > 5e-3
        beta_sv = beta_1[indx]
        x_sv = X[indx[:,0],:]
        y_sv = y[indx[:,0]]

        # get w_phi and b
        k_sv = pairwise_kernels(x_sv, x_sv, metric = kernel, **self.kernel_param)
        w = (beta_sv*y_sv).reshape(-1,1)

        w_phi = w.T @ k_sv
        b= np.mean(y_sv - w_phi)

        # keep on memory for better use
        self.x_sv = x_sv; self.w = w; self.b = b; self.a=a; self.c=c
        self.beta_sv = beta_sv;
        return self

    def predict(self, X_test):

```

```
        np = self.np
        c = self.c
        a = self.a
        # rename coefficients
        b = self.b; x_sv = self.x_sv
         # create new kernel
        k_test = self.pairwise_kernels(x_sv, X_test, metric = self.
    kernel, **self.kernel_param)
         # multiply w and kernel
        w_phi = self.w.T @ k_test
        # predict new data
        predict1 = np.sign(w_phi + b)
         # rename to original labels
        predict2 = np.where(predict1 == c[0], a[0], a[1])
        return predict2

    def coef_(self):

        np = self.np
        if self.kernel == "linear":
            w = self.w; x_sv = self.x_sv
            w = np.sum(np.multiply(w, x_sv), axis = 0)
            return w, self.b
        else:
            return self.w, self.x_sv,  self.b
```

Listing 1: L_1^ϵ-SVC

.2 *Grid Search*

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
param_grid = {'C': [1, 2, 5, 10],
              'gamma': [1, 0.5, 0.1, 0.05, 0.01, 0.005],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3,
    n_jobs=-1)

# fitting the model for grid search
grid.fit(X_train, y_train)

# print best parameter after tuning
print(grid.best_params_)
grid_predictions = grid.predict(X_test)

# print classification report
print(classification_report(y_test, grid_predictions))
```

Listing 2: Classic SVC Grid Search:

.3 *Random Forest Grid Search*

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(criterion='entropy',
                                max_depth=None,
                                min_samples_split=2,
                                min_samples_leaf=2,
                                max_features='auto',
                                bootstrap=True,
                                oob_score=False,
                                random_state=42,
                                verbose=1)

param_grid = {
    "n_estimators": [50, 100, 200, 500],
}

grid_cv = GridSearchCV(random_forest, param_grid, n_jobs=-1, cv=3,
     scoring="accuracy")
grid_cv.fit(X_train,y_train)
```

Listing 3: Random Forest Grid Search:

.4 XGBoost Grid Search

```
import xgboost as xgb


# Init classifier
xgb_cl = xgb.XGBClassifier(objective="binary:logistic")

param_grid = {
    "max_depth": [3, 4, 5, 7],
    "learning_rate": [0.1, 0.01, 0.05],
    "gamma": [0, 0.25, 1],
    "reg_lambda": [0, 1, 10],
    "scale_pos_weight": [1, 3, 5],
}

grid_cv = GridSearchCV(xgb_cl, param_grid, n_jobs=-1, cv=3, scoring
     ="accuracy")
grid_cv.fit(X_train,y_train)

print(grid_cv.best_score_)
print(grid_cv.best_params_)

final_cl = xgb.XGBClassifier(
    **grid_cv.best_params_,
    objective="binary:logistic",
)

final_cl.fit(X_train, y_train)

preds = final_cl.predict(X_test)
```

Listing 4: XGBoost Grid Search:

.5 Data Quality Report

```
def dqr(data):
    # List of database variables
    cols = pd.DataFrame(list(data.columns.values),
                            columns=['Names'],
                            index=list(data.columns.values))
    # List of data types
    dtyp = pd.DataFrame(data.dtypes,columns=['Type'])
    # List of missing data
    misval = pd.DataFrame(data.isnull().sum(),
                                    columns=['Missing_values'])
    # List of present data
    presval = pd.DataFrame(data.count(),
                                    columns=['Present_values'])
    # List of unique values
    unival = pd.DataFrame(columns=['Unique_values'])
    # List of min values
    minval = pd.DataFrame(columns=['Min_value'])
    # List of max values
    maxval = pd.DataFrame(columns=['Max_value'])
    for col in list(data.columns.values):
        unival.loc[col] = [data[col].nunique()]
        try:
            minval.loc[col] = [data[col].min()]
            maxval.loc[col] = [data[col].max()]
        except:
            pass

    # Join the tables and return the result
    return cols.join(dtyp).join(misval).join(presval).join(unival).
     join(minval).join(maxval)
```

Listing 5: Data Quality Report

.6 ADASYN

```

from imblearn.over_sampling import ADASYN
from collections import Counter
ad= ADASYN()
X,y = ad.fit_resample(X,y)
```

Listing 6: ADASYN Oversampling

Bibliography

Shigeo Abe. *Support Vector Machines for Pattern Classification*. Springer, second edition, 2004. ISBN 978-1-84996-097-7.

S. Boyd and L.Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 978-0-521-83378-3.

Olvi L. Mangasarian Dr. William H. Wolberg, W. Nick Street. sklearn.datasets.load_breast_cancer, 1995. URL https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html.

IBM. What is a random forest? URL https://www.ibm.com/topics/random-forest.

Mengmou Li. *Generalized Lagrangian Multiplier Method and KKT Conditions With an Application to Distributed Optimization*, volume 66. 2019. DOI: 10.1109/TCSII.2018.2842085.

Eric J. Hunter Lorraine O. Ramig Max A. Little, Patrick E. McSharry. Parkinsons data set, 2008. URL https://archive.ics.uci.edu/ml/datasets/parkinsons.

National Institute of Diabetes, Digestive, and Vincent Sigillito Kidney Diseases. Diabetes dataser, 1990. URL https://www.kaggle.com/datasets/mathchi/diabetes-data-set.

scikit-learn developers (BSD License). Gridsearchcv, 2007 - 2023. URL https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.