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 of Data Science

Computationally Stable QCQP and SDP Multikernel Support Vector Regression Formulations

---

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

A thesis presented by:
Gregorio Alberto Álvarez Álvarez

Thesis Advisor:
Dr. Juan Diego Sánchez Torres

Tlaquepaque, Jalisco, July, 2025

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 of Data Science Approval Form

Thesis Title: **Computationally Stable QCQP and SDP Multikernel Support Vector Regression Formulations**

Author: **Gregorio Alberto Álvarez Álvarez**

Thesis Approved to complete all degree requirements for the Master of Science Degree in Data Science.

Thesis Advisor, **Dr. Juan Diego Sánchez Torres**

Thesis Reader, **Dr. Jorge Alejandro Delgado Aguiñaga**

Thesis Reader, **Dr. Riemann Ruíz Cruz**

Academic Advisor, **Dr. Rocío Carrasco Navarro**

Tlaquepaque, Jalisco, July, 2025

Computationally Stable QCQP and SDP Multikernel Support Vector Regression Formulations

Gregorio Alberto Álvarez Álvarez

Abstract

This study will explore alternative versions of the Multikernel Support Vector Regressor (SVR) algorithm. The two versions that will be explored include a derivation that uses an Objective-to-Constraint transformation to derive a Quadratically Constrained Quadratic Program (QCQP) algorithm with computational advantages over the earlier formulations. For the other approach, an innovative method to filter support vectors is used to increase numerical stability. This approach uses Lagrangian Duality and Semidefinite Programming (SDP) theory to derive a more general formulation. It will be shown that the alternative QCQP and SDP formulations provide computational advantages over their respective prior formulations, offering a more practical alternative to manual kernel design, especially in scenarios where using a multikernel is essential for problem construction, making it an ideal tool for researchers and practitioners.

Dedicated to my parents, who supported me throughout the entire process to where I am now, and to Juan Diego for his support during this work.

Contents

List of Figures

Page

List of Tables

1 Introduction

In recent years, with the increase in computational capacity and the decrease in costs per computational unit [1], a climb in the usage of machine learning algorithms has surged [2]. Support Vector Machines (SVM) have been used due to their capacity to find non-linear correlations between the dependent and independent variables and the stability of the solution due to their convex property [3]. The selection of kernels is an important process for identifying a model that best describes the input data. The task of selecting multiple kernels to capture the internal subpatterns intertwined in the data is more challenging. The Multiple Kernel Learning (MKL) [4] approach addresses this challenge by introducing a set of learnable parameters to appropriately weight each kernel, based on the specific learning task. In this work, we explore several developments in Multiple Kernel Learning (MKL), with a particular emphasis on advancing the optimization of Multikernel Support Vector Regressor (MSVR) formulations. The goal of these new derivations is not only to learn the kernel from the given data, rather than relying on predefinition via domain knowledge, but also to achieve significantly improved computational stability over earlier formulations, enhancing the numerical precision and convergence speed, and feasibility for certain problems. We will detail two specific approaches that build upon the methods first published by Lanckriet et al. [5]. Chapter 2 reviews the theoretical background of SVMs, along with previous derivations of MSVR. Chapter 3 details the modified MSVR formulations developed in this study. Chapter 4 presents two case studies comparing these new formulations with previous ones and established benchmark algorithms.

[1] Guillermo Alexander Loayza-Delgado, Xiomara Luciana Tejada-Montalvo, María Fernanda Carnero-Quispe, and Christian Frederick Gárate-Rodríguez. Machine learning in industrialization: a bibliometric analysis. *DYNA*, 92 (235):28–37, january-march 2025. ISSN 0012-7353

[2] Sakshi Aggarwal. Machine learning algorithms, perspectives, and real-world application: Empirical evidence from united states trade data. MPRA Paper 116579, Indian Institute of Foreign Trade, march 2023. URL https://mpra.ub.uni-muenchen.de/116579/. Posted 04 Mar 2023 09:21 UTC

[3] Shigeo Abe. *Support Vector Machines for Pattern Classification*. Springer, 2nd edition, 2010. ISBN 978-1-84996-098-4

[4] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011. URL https://jmlr.csail.mit.edu/papers/volume12/gonen11a/gonen11a.pdf

[5] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

2 *Preliminaries*

Contents

2.1 *Definitions and auxiliary problems*

2.1.1 *Symbols and Notation*

$\mathbf{x}$ Vector $\mathbf{x}$.

$\mathbf{A}$ Matrix $\mathbf{A}$.

x_i i-th component of 1D vector $\mathbf{x}$.

$\mathbb{R}^n$ Vector of real numbers of size n.

$\mathbb{N}^n$ Vector of integer numbers of size n.

$||\mathbf{x}||^p$ Norm p of vector $\mathbf{x}$.

$\{y_1, y_2, \ldots, y_n\}$ Set of scalars from vector $\boldsymbol{y}$.

$\langle \mathbf{x}, \mathbf{x}' \rangle$ Inner product between two vectors $\mathbf{x}, \mathbf{x}'$

$\mathbf{K}(\mathbf{x}, \mathbf{x}')$ kernel function obtained from vectors $\mathbf{x}, \mathbf{x}'$

$\mathbf{k}_i$ i-th kernel from a subset of kernels which linear combinations results in $\mathbf{K}$

2.1.2 *Background in statistical learning*

Statistical learning is the process where a set of tools is used to understand some dataset [1]. The goal is to find the pattern that the observed data contains, then being able to describe the process or to reproduce the behaviour by finding a generalization of it.

This work will focus on the process of *supervised learning* where there is a set of input-output pairs that are fed into a *Machine learning* algorithm to find a generalized model from the underlying pattern. [2].

Basic terminology will be discussed for the problem of supervised statistical learning, either regression or classification.

Given some examples of the form:

$$\{\mathbf{x}_i, y_i\}_{i=1}^m$$

Where $\mathbf{x}_i \in \mathbb{R}^n$ is known as *data point* or object and $y_i \in \mathbb{R}$ represents the output, target, or label (in the context of classification) related to the point $\mathbf{x}_i$.

[1] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer Texts in Statistics. Springer New York, 2013. ISBN 978-1-4614-7137-0. DOI: 10.1007/978-1-4614-7138-7. URL http://doi.org/10.1007/978-1-4614-7138-7

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006

A data point can be written as follows in extended format

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{in}]^\top$$

Where each entry represents a feature or position of the data point.

The set of output variables $\mathbf{y} = [y_1, y_2, \cdots, y_m]$ is known as the *output, target or dependent variable.*

The goal now is to learn a function $f(x)$ that relates the input and output data,

$$\hat{\mathbf{y}} = f(x).$$

This stage, known as *training* or *learning* in machine learning theory, involves a process where a set of parameters are *optimized* guided by a cost function $l(x, y)$ to adapt the model to the specific problem [3].

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006

2.1.3 Kernel Methods

The kernel is an important part of the SVM theory since it is what makes it possible to find complex frontiers in the associated higher-dimensional feature space. In this context, the existence of the kernel is the result of the computation of the dual form and is defined mathematically as follows:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i)\phi(\mathbf{x}_j) \rangle$$

where $\mathbf{x}_i \in X$ represents a data example, ϕ is the mapping function, and $k(\mathbf{x}_i, \mathbf{x}_j)$ represents the (i, j)-th entry of the kernel matrix. Notice that when using kernel methods [4], it is not necessary to employ the mapping function ϕ or to get the associated coordinates $\phi(\mathbf{x}_i)$ [5] to get information about the new coordinate system and reproduce the properties for new set of data points, instead all the positional information is encoded in the inner product of each pair of samples [6].

[4] The group of methods that use kernels as a step in the overall process

[5] Which in terms, would be very difficult or impossible to compute do the fact that some transformations lead to infinite dimensional feature spaces

[6] Hal Daumé III. From zero to reproducing kernel hilbert spaces in twelve pages or less. Technical report, UMIACS

Hilbert Spaces

In the statistical process of learning, a way to generalize the relationship between some input set ($\{\mathbf{x}|\mathbf{x} \in \mathcal{X}\}$) and output set ($\{y|y \in \mathcal{Y}\}$) can be found by a similarity measure.

The inner product can be seen as a similarity measure, giving information about the difference between samples from the set. This similarity measure can be computed through the use of a function k that maps the set of inputs $\mathcal{X}$ to an inner product space $\mathcal{H}$

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

A Hilbert space $\mathcal{H}$ is a real or complex inner product space that is also a complete metric space. This means that the space has an inner product and metric induced by the inner product. This space has the property of maintaining the rules of linear algebra for finite or infinite spaces.

Reproducible Kernel Hilbert Space (RKHS)

Before, it was explained how a kernel induces the creation of a Hilbert space where the inner product property gives rise to the possibility of embedding new points into this space and making it possible to predict new elements from an output random variable as in the case of the support vector machine. In this subsection, we will explain how a reproducible Hilbert space is built.

Corollary 1 *Given some function $f(\cdot)$ and a kernel function $k(\cdot, \mathbf{x})$ inside the associated RKHS created from the kernel function, the following relationship holds*

$$\langle f(\cdot), k(\cdot, \mathbf{x}) \rangle = f(\mathbf{x})$$

Which states that the evaluation of $\mathbf{x}$ by $f(\cdot)$ can be reproduced by the dot product between $f(\cdot)$ and $k(\cdot, \mathbf{x})$

Proof:

Let's define a mapping function $\phi : \mathcal{X} \to \mathbb{R}$ and

$$\mathbf{x} \in \mathcal{X} \to k(\cdot, \mathbf{x}) = \phi(\mathbf{x})$$

where $k(\cdot, \mathbf{x})$ spans $\mathbf{x}$, and receives a second argument $\mathbf{x}' \in \mathcal{X}$ such that $\mathbf{x}, \mathbf{x}' \to k(\mathbf{x}', \mathbf{x})$, and the inner product between two kernel functions is defined as follows:

$$\langle \phi(\mathbf{x}'), \phi(\mathbf{x}) \rangle = k(\mathbf{x}', \mathbf{x}) = \langle k(\cdot, \mathbf{x}'), k(\cdot, \mathbf{x}) \rangle$$

Now we will create a base from a linear combination of vectors that span all the space created from the images of $\mathbf{x} \in \mathcal{X}$

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x)$$

and another base that spans all the space created from the images of $\mathbf{x}' \in \mathcal{X}$:

$$g(\cdot) = \sum_{j=1}^{n'} \beta_j k(\cdot, \mathbf{x}'_j)$$

Notice that $\alpha_i, \beta_j \in \mathcal{R}$, and n and n' are not necessarily the same.

Computing the inner product between the previously defined vector spaces gives the following result:

$$\begin{aligned}\langle f(\cdot), g(\cdot)\rangle &= \sum_{i=1}^{n}\sum_{j=1}^{n'} \alpha_i \beta_j \langle k(\cdot, \mathbf{x}_i), k(\cdot, \mathbf{x}'_j)\rangle \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n'} \alpha_i \beta_j k(\mathbf{x}'_j, \mathbf{x}_i) \\ &= \sum_{j=1}^{n'} \beta_j f(\mathbf{x}_j) \\ &= \sum_{i=1}^{n} \alpha_i g(\mathbf{x}_i) \end{aligned} \tag{2.1}$$

From (2.1) it can be seen that by the property of the inner product, the product is linear and symmetric. From the previous, the following can be proven as well:

$$\langle f(\cdot), k(\cdot, \mathbf{x}_i)\rangle = f(\mathbf{x})$$

Specifically, there is some self-reproducible property, stated as follows :

$$\langle k(\cdot, \mathbf{x}'), k(\cdot, \mathbf{x})\rangle = k(\mathbf{x}', \mathbf{x})$$

The kernel can then be defined by its two intrinsic properties: matrix symmetry and positive semidefiniteness [7].

[7] Shigeo Abe. *Support Vector Machines for Pattern Classification*. Springer, 2nd edition, 2010. ISBN 978-1-84996-098-4

Multikernel

The multikernel aims to model functions with combined properties like multi-period time series with random noise [8]. We can construct a positive semidefinite multikernel $\mathbf{K}$ by taking a linear combination of a given set of kernels $\mathbf{k}_i \in \mathbb{R}^{n\times n}$, $i = 1, \dots N$ with some weights $\mu_i \in \mathbb{R}$, provided that $\mathbf{x}^\top \mathbf{k}_i \mathbf{x} \geq 0$ for $\mathbf{x} \neq \mathbf{0}$ and $\mu_i \geq 0$

[8] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://gaussianprocess.org/gpml/chapters/RW.pdf

$$\boldsymbol{K} = \sum_{n=1}^{N} \mu_i \boldsymbol{k}_i$$

Common kernels

The kernels listed above are among the most commonly used for regression problems [9].

[9] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://gaussianprocess.org/gpml/chapters/RW.pdf

- *Linear*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}' + c$$

That computes the standard linear product to represent linear relationships in the input space. An offset can be added with $c \geq 0$ ($c \in \mathbb{R}$).

- *Polynomial*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = (\alpha \mathbf{x}^\top \mathbf{x}' + c)^d$$

Aiming to obtain model polynomial interactions up to $d > 0$ degree. Additionally, a scaler $\alpha > 0$ can be added to each term of the polynomial.

- *Radial Basis Function (RBF)*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

Corresponds to an infinite-dimensional feature space. Where $\sigma > 0$. The following parametrization is often used $\gamma = \frac{1}{2\sigma^2}$.

- *Periodic*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{2\sin^2(\pi\|\mathbf{x} - \mathbf{x}'\|/p)}{\ell^2}\right)$$

Usually employed for time series modeling, where a repetitive pattern is present with periodicity $p > 0$ and length-scale $\ell > 0$, with similar effect to $\sigma > 0$ for the radial basis kernel.

- *Matérn*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \frac{1}{2^{\nu-1}\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right)$$

The Matérn kernel can be seen as a generalization of the RBF kernel. The parameter $\nu > 0$ controls the smoothness of the resulting function. K_ν is the modified Bessel function of the second kind, and Γ is the gamma function. Common choices for ν are $\frac{1}{2}$, $\frac{3}{2}$, and $\frac{5}{2}$.

- *Laplacian*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_1}{\ell}\right)$$

This kernel is equivalent to the Matérn kernel with $\nu = \frac{1}{2}$. Its kernel is suitable for modeling data where sharp changes are expected.

- *Rational Quadratic*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\ell^2}\right)^{-\alpha}$$

The Rational Quadratic kernel can be seen as a mixture of RBF kernels with different characteristic length-scales. The parameter $\alpha > 0$ is the scale mixture parameter, which dictates the weighting of these different length-scales.

- *Constant*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \sigma_0^2$$

When combined with other kernels, it represents a bias in the modeled function. The parameter σ_0^2 represents the variance of this constant offset.

- *White/Kronecker Kernel*:

$$\boldsymbol{K}(\mathbf{x}, \mathbf{x}') = \sigma_n^2 \delta(\mathbf{x}, \mathbf{x}')$$

Where $\delta(\mathbf{x}, \mathbf{x}')$ is the Kronecker delta (1 if $\mathbf{x} = \mathbf{x}'$, 0 otherwise). The parameter $\sigma_n^2 > 0$ represents the noise variance. It is commonly added to a set of kernels to account for the noise in the observations. With a varying σ_n^2, it can be utilized as a Ridge regularization term.

2.1.4 *Convex Optimization*

When learning with support vector machines, it is possible to ensure that if the problem is feasible, the optimal solution can be found; this means that no better result can be found for the specific optimization problem. The reason for this to happen is that the mathematical optimization problem (objective function as well as the constraints) is convex (or concave), and a minimum can be found for the minimization problems or a maximum can be found for the maximization problems.

Convex Functions

A convex function is one where the domain of this function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex set and given some $x, y \in f$ and $0 \leq \theta \leq 1$ The following relation holds:

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 + \theta) f(y)$$

This means that any line segment created inside the domain of the convex function from $(x, f(x))$ to $(y, f(y))$ (right side of the equation) must lay on top of the function's graph (left side of the equation) [10].

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

Convex Sets

A set is said to be convex if a line segment created from any two points in the set is inside the set. Given some convex set C, two points inside the set x_1, x_2 and $0 \leq \theta \leq 1$

$$\theta x_1 + (1 - \theta) x_2 \in C$$

Convex Optimization Problems

The convex optimization problems belong to a subclass of the mathematical optimization field where the objective and the constraints

are convex functions, and therefore, the problem is said to be convex. The generalized mathematical definition of a convex problem is written as follows [11]:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) \\ s.t \quad & f_i(\mathbf{x}) \leq b_i \quad , i = 1, 2, 3, \ldots, m \\ & h_i(\mathbf{x}) = 0, \quad i = 1, 2, 3, ..., m \end{aligned}$$

Where $\mathbf{x}$ is the optimization variable or the set of the problem's optimizers. $f_0(\mathbf{x})$ ($\mathbb{R}^n \rightarrow \mathbb{R}$) is the objective function, f_i and $h_i(i)$ ($\mathbb{R}^n \rightarrow \mathbb{R}$), create the set of the problem's constraints.

Lagrangian

See the following generalized linear program

$$\begin{aligned} \max_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s. t.} \quad & g(\mathbf{x})_i \geq 0 \quad i = 0 \ldots N \end{aligned}$$

Where $f_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are the sets of functions that constrain the problem. Its corresponding Lagrangian has the following format

$$\max_{\mathbf{x}} f(\mathbf{x}) + \min_{\mathbf{u}} \left[\sum_{i=1}^{N} u_i g_i(\mathbf{x}) \right] ; u_i \geq 0, i = 0 \ldots N$$

The dual variable u forms the sets that define the constraints of the dual problem.

The problem can then be rearranged in the following way

$$\max_{\mathbf{x}} \min_{\mathbf{u}} f(\mathbf{x}) + \sum_{i=1}^{N} u_i g_i(\mathbf{x}); u_i \geq 0, i = 0 \ldots N$$

Where the Lagrangian is defined as:

$$\mathcal{L}(\mathbf{x}; \mathbf{u}) = f(\mathbf{x}) + \sum_{i=1}^{N} u_i g_i(\mathbf{x})$$

For this problem, u acts as a barrier for the region in the feasible region of the primal problem.

$$\min_{\mathbf{u} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}; \mathbf{u}) = \begin{cases} f_i(\mathbf{x}) & \text{if } g_i(\mathbf{x}) \geq 0, \quad i = 1, \ldots, N \\ -\infty & \text{otherwise.} \end{cases}$$

The order of the maximization and minimization operations can be interchanged if strong duality holds. By analyzing the resulting problem, it is possible to gain additional insights about the problem and to construct the dual formulation [12].

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

[12] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

This transformation yields a lower bound on the value of the original problem. For a specific subset of problems, the values of the min-max and max-min problems are equal. Such problems are said to exhibit strong duality. Strong duality can often be proven using Slater's condition. Specifically, if the inner optimization problem is convex and feasible, then strong duality holds.

Epigraph form

In mathematical programming, the epigraph form is a technique used to reformulate a convex optimization problem into one with a linear objective function.

Given the following convex problem,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s. t.} \quad & g_i(\mathbf{x}) \geq 0 \quad i = 0 \dots N \end{aligned}$$

It's possible to introduce a slack variable t that provides an upper bound for the functional,

$$\begin{aligned} \min_{\mathbf{x}} \quad & t \\ \text{s. t.} \quad & f(\mathbf{x}) - t \leq 0 \\ & g_i(\mathbf{x}) \geq 0 \quad i = 0 \dots N \end{aligned}$$

Where $T = \{t \in T\}$ is the epigraph set that can be geometrically viewed as the volume that fills the convex function. The minimum of this volume is therefore located in the same place as that of the original functional. The optimal solution is achieved when $t = \min f(\mathbf{x})$.

2.1.5 *Objective-to-Constraint Transformation*

The Objective-to-Constraint Transformation consists of moving part of the objective function into the constraints.

Consider the following min-max optimization problem

$$\begin{aligned} \max_{\mathbf{u}} \min_{\mathbf{x}} \quad & f(\mathbf{x}) + \sum_{i=1}^{N} u_i g_i(\mathbf{x}) \\ \text{s. t.} \quad & h_i(\mathbf{x}) = 0, \quad i = 1, \dots M \\ & \mathbf{u} \geq \mathbf{0} \end{aligned}$$

Assuming convexity of the inner problem ($\min_{\mathbf{x}}$) and concavity of the outer problem ($\max_{\mathbf{u}}$), it's possible to switch the order of the problems as in the case of the zero duality gap.

$$\begin{aligned} \min_{\mathbf{x}} \max_{\mathbf{u}} \quad & f(\mathbf{x}) + \sum_{i=1}^{N} u_i g_i(\mathbf{x}) \\ \text{s. t.} \quad & h_i(\mathbf{x}) = 0, \quad i = 1, \dots M \\ & \mathbf{u} \geq \mathbf{0} \end{aligned}$$

Then the max problem can be isolated for terms that depend on the variable that it optimizes:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) + \max_{\mathbf{u}} \left[\sum_{i=1}^{N} u_i g_i(\mathbf{x}) \right] \\ \text{s. t.} \quad & h_i(\mathbf{x}) = 0, \quad i = 1, \dots M \\ & \mathbf{u} \geq \mathbf{0} \end{aligned}$$

Following the inverse process observed in 2.1.4, where the Lagrangian is constructed by moving the constraints to the objective function [13], the following problem emerges:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s. t.} \quad & g_i(\mathbf{x}), \quad i = 1, \dots N \\ & h_i(\mathbf{x}) = 0, \quad i = 1, \dots M \end{aligned}$$

Notice that the vector $\mathbf{u}$ disappears from the problem. Most interior point method solvers address the dual problem, allowing $\mathbf{u}$ to be obtained implicitly.

[13] During the Lagrangian construction, it's also possible to move a subset of constraints to the objective and maintain the rest as constraints. This process is known as Lagrangian relaxation

Bernard Lemaire. Lagrangian relaxation. Technical report, ENS de Lyon, January 2012. URL https://www.ens-lyon.fr/DI/wp-content/uploads/2012/01/LagrangianRelax.pdf

2.1.6 *Quadratically Constrained Quadratic Program*

The Quadratically Constrained Quadratic Program (QCQP) generalizes both Quadratic Programs (QP) and Linear Programs (LP). These problems can be mathematically represented as follows [14]:

$$\begin{aligned} \min_{\mathbf{x}} \quad & (1/2)\mathbf{x}^\top \mathbf{P}_0 \mathbf{x} + \mathbf{q}_0^\top \mathbf{x} + r_0 \\ \text{s. t.} \quad & (1/2)\mathbf{x}^\top \mathbf{P}_i \mathbf{x} + \mathbf{q}_i^\top \mathbf{x} + r_i \leq 0, \quad i = 1, \dots N \\ & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable.
- $\mathbf{P}_i \in \mathbb{R}^{n \times n}, \quad i = 0, \dots N$ is the quadratic part of the objective function and constraints.
- $\mathbf{q}_i \in \mathbb{R}^n, \quad i = 0, \dots N$ represents the linear part of the objective function and constraints.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

- $\mathbf{r}_i \in \mathbb{R}, \quad i = 0, \dots N$ are the scalar constants of the objective function and constraints.
- $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix of coefficients, used for the equality constraint.
- $\mathbf{b} \in \mathbb{R}^m$ is a vector of constants for the equality constraint.

In this problem, both the objective function and some constraints are quadratic. The convexity of the problem depends on whether $\mathbf{P}_i$ for all $i = 0, \dots, N$ are positive semidefinite.

2.1.7 *Second-Order Cone Programming*

The second-cone cone programming (SOCP) is a more general problem than QCQP. The SOCP is equivalent to the QCQP, where all quadratic constraints are squared [15].

$$\begin{aligned} \min_{\mathbf{x}} \quad & f^\top \mathbf{x} \\ \text{s. t.} \quad & ||\mathbf{A}_i \mathbf{x} + \mathbf{b}_i||_2 \leq \mathbf{c}_i^\top \mathbf{x} + d_i \\ & \mathbf{F}\mathbf{x} = \mathbf{g} \end{aligned}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable.
- $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b}_i \in \mathbb{R}^n$, $\mathbf{c} \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$ form the second-order cone constraints
- $\mathbf{F} \in \mathbb{R}^{p \times n}$ and $\mathbf{g} \in \mathbb{R}^p$ constitute the equality constraint

[15] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

2.1.8 *Semidefinite programming*

Semidefinite Programming (SDP) is a generalization of linear and quadratic programming that allows optimization over a wide range of problems. It's a powerful tool that can tackle difficult or impossible problems (see Max-Cut problem [16] or control) through their reformulation as SDP with the help of linear matrix inequalities [17] and efficient interior point methods, due to their convexity property, to find their solution [18].

A semidefinite program is an optimization problem of the form:

[16] Mikhail Nedbai. 4.1 introduction 4.2 max cut local search algorithm. Lecture 4: Local Search, CS880: Approximation and Online Algorithms

[17] Carsten Scherer and Siep Weiland. Linear matrix inequalities in control. Technical report, University of Stuttgart, 2015. URL https://www.imng.uni-stuttgart.de/mst/files/LectureNotes.pdf

[18] Robert M. Freund. Introduction to semidefinite programming (sdp). Technical report, Massachusetts Institute of Technology, March 2004

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{s. t.} \quad & \mathbf{F}(\mathbf{x}) \succeq 0 \end{aligned}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable.
- $\mathbf{c} \in \mathbb{R}^n$ is a cost vector.
- $\mathbf{F}(\mathbf{x}) = \mathbf{F}_0 + \sum_{i=1}^{n} x_i \mathbf{F}_i$ is a Linear Matrix Inequality (LMI).
- $\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_n$ are symmetric matrices of size $p \times p$.
- $\mathbf{F}(\mathbf{x}) \succeq 0$ means that $\mathbf{F}(\mathbf{x})$ is a positive semidefinite matrix.

SDP programs tend to be computationally more demanding than QCQP. Therefore, whenever possible, the problem should be reduced to QCQP (see Lanckriet et al. [19]).

[19] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

2.1.9 *Schur Complement Lemma*

An effective approach to converting a problem into a semidefinite program involves using the Schur Complement Lemma. According to this lemma, if we have the equation $M = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$, where $M \geq 0$ and $\mathbf{A} \succeq \mathbf{0}$, the problem can be restructured as follows [20]:

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}; \quad \mathbf{M} \succeq \mathbf{0} \tag{2.2}$$

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

This is the Schur complement of the top-left block $\mathbf{A}$, there are additional formulations like the Schur complement of the bottom-right block $\mathbf{D}$.

2.1.10 *Block Bootstrap*

Unlike the traditional bootstrap method, which resamples individual observations from the original sample and assumes independence, the block bootstrap is tailored for time series data exhibiting serial correlation.

The block bootstrap works by dividing the time series into consecutive blocks. It then resamples these blocks with replacement and joins them to create a new bootstrap time series. This process preserves the order within blocks, thus maintaining the temporal dependence structure inherent in the data [21].

Choosing the block length is crucial and non-trivial, with no universal rule. While exploring different lengths via sensitivity analysis

[21] Wolfgang Härdle, Joel L. Horowitz, and Jens-Peter Kreiss. Bootstrap methods for time series, August 2001. URL https://nbn-resolving.de/urn:nbn:de:kobv:11-10050152

can offer insights, data-driven selection methods often provide a more systematic approach. An improper length leads to issues. Excessively short blocks can break the dependence structure, causing bias, while overly long blocks limit the resampling variability, reducing the effectiveness of the bootstrap [22].

[22] James G. MacKinnon. Bootstrap Methods in Econometrics, February 2006. URL http://www.econ.queensu.ca/faculty/mackinnon/. Revised, June, 2006

2.1.11 SVC mathematical Background

The Support Vector Classifier (SVC), initially formulated by Vapnik [23], is an algorithm for pattern classification [24], where outputs belong to distinct classes $y_i \in \{-1, 1\}$. The fundamental objective of the SVC is to determine an optimal decision boundary within the feature space 2.1.3.

[23] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998. ISBN 0-471-03003-1

[24] Shigeo Abe. *Support Vector Machines for Pattern Classification*. Springer, 2nd edition, 2010. ISBN 978-1-84996-098-4

At its core, the SVC is solved as a mathematical optimization problem [25]. This involves finding the optimal separating hyperplane that maximizes the margin. The margin is defined as the distance between the hyperplane and the closest data points from each class. The SVC's strategy is therefore to identify the data points nearest to the class boundary (also known as support vectors) and construct a hyperplane, whose corresponding margin maintains the largest possible separation from these critical points.

[25] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

the generalization of a plane to higher dimensions

Figure 2.1 (a) illustrates a separable dataset with one dependent and one independent variable. The optimal hyperplane represented in orange separates the two classes. The frontier is represented by the green dot at $f(\mathbf{x}) = 0$. Figure 2.1 (b) shows a separable dataset with two independent variables. The continuous line represents the frontier, while the dashed lines represent the margin. For the latter case, the separating hyperplane cannot be depicted in two dimensions, requiring an extra dimension for its visualization.

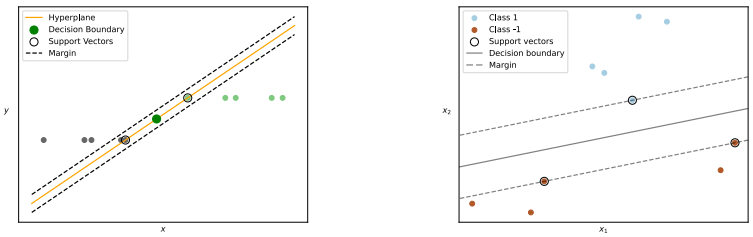

Figure 2.1: Representation of a separable dataset with one dependent and one independent variable (a) and with two independent variables (b).

We will now proceed to detail key concepts related to the SVC formulation, after which the derivation will be presented.

Distance between a point and a plane

For the SVC problem, the equation of the hyperplane that separates the two classes is given by:

$$f(\mathbf{x}) = \boldsymbol{\omega}^\top \mathbf{x} + b \tag{2.3}$$

where $\boldsymbol{\omega} \in \mathbb{R}^n$ is the vector that contains the weights that dictate the orientation of the hyperplane, $\mathbf{x}$ is the set of independent variables, and b represents the distance to the hyperplane, from the origin.

Then, the associated frontier is located at:

$$\boldsymbol{\omega}^\top \mathbf{x} + b = 0$$

The points that are on top of the frontier satisfy the equation:

$$\boldsymbol{\omega}^\top \mathbf{x} + b > 0$$

while the points that are below the frontier satisfy the equation:

$$\boldsymbol{\omega}^\top \mathbf{x} + b < 0$$

Let $\mathbf{x}_m$ be a point that is the closest to the hyperplane on the positive side, then the distance between the point and the hyperplane is given by some distance r:

$$\boldsymbol{\omega}^\top \mathbf{x}_m + b = r$$

Since this equation will be used for an optimization problem, it's possible to scale the parameters $\boldsymbol{\omega}^* = \boldsymbol{\omega}/r$ and $b^* = b/r$ so that the distance r is normalized and the problem is unchanged.

$$\frac{1}{r}[\boldsymbol{\omega}^\top \mathbf{x}_m + b = r] \quad \Rightarrow \quad \boldsymbol{\omega}^{*\top} \mathbf{x}_m + b^* = 1 \tag{2.4}$$

Margin

It can be proven that the distance between a point x_k and the hyperplane is given by:

$$d_k = \frac{|\boldsymbol{\omega}^\top \mathbf{x}_k + b|}{||\boldsymbol{\omega}||_2}$$

where $||\boldsymbol{\omega}||_2$ is the norm of the vector $\boldsymbol{\omega}$, which is perpendicular to the hyperplane.

By (2.4), the distance to the closest positive point is given by:

$$d_m = \frac{1}{||\boldsymbol{\omega}||_2}$$

Therefore, the margin length is given by:

$$M = \frac{2}{||\boldsymbol{\omega}||_2} \tag{2.5}$$

SVC Problem formulation

From (2.4) we can infer that for entries correctly classified, the following condition must be satisfied:

$$\begin{cases} \boldsymbol{\omega}^T \mathbf{x}_k + b \geq 1 & \text{if } y_k = 1 \\ \boldsymbol{\omega}^T \mathbf{x}_k + b \leq -1 & \text{if } y_k = -1 \end{cases}$$

One way to merge both conditions is by:

$$y_k[\boldsymbol{\omega}^T \mathbf{x}_k + b] \geq 1 \tag{2.6}$$

To make the sentence in (2.6) more comprehensible, a generalized example will be portrayed:

Let's suppose that there is a data point $\mathbf{x}_k$ that is correctly classified, then the following inequality holds true

$$\overbrace{y_k}^{\substack{1 \\ -1}} \cdot \overbrace{[\boldsymbol{\omega}^T \mathbf{x}_k + b]}^{\substack{\geq 1 \\ \leq -1}} \geq 1$$

Then satisfying (2.6).

On the contrary, if the data point is incorrectly classified, $y_k f(x_k)$ will not satisfy inequality (2.6)

$$\overbrace{y_k}^{\substack{1 \\ -1}} \cdot \overbrace{[\boldsymbol{\omega}^T \mathbf{x}_k + b]}^{\substack{< 1 \\ > -1}} \not\geq 1$$

Building upon the SVC's objective of maximizing the margin, and considering the margin definition (2.5) and the classification constraint (2.6), the optimization problem can be formulated as follows:

$$\max_{\boldsymbol{\omega}, b} M = \frac{2}{||\boldsymbol{\omega}||_2}$$

Subject to:

$$y_k[\boldsymbol{\omega}^T \mathbf{x}_k + b] \geq 1$$

It's possible to rewrite the previous problem so that instead of maximizing the margin, we minimize the square of the reciprocal of the margin:

$$\begin{aligned} \min_{\boldsymbol{\omega}, b} \quad & \frac{1}{2}||\boldsymbol{\omega}||^2 \\ \text{s. t.} \quad & y_k[\boldsymbol{\omega}^T \mathbf{x}_k + b] \geq 1, \quad i = 1, \ldots, n \end{aligned} \tag{2.7}$$

The constant $\frac{1}{4}$ is added to simplify the solution of the problem, scaling the value of $\boldsymbol{\omega}$ as in (2.4)

Predicting new values

After finding the set of *optimizers* $\boldsymbol{\omega}^*$ and b^* that solve the problem, it's possible to predict the labels of new data points. By using (2.3) to make predictions, we would obtain values in the range $(-\inf, \inf)$, representing values above and below the margin. To determine the labels, we take the sign of the result.

$$f(x) = \text{sign}((\boldsymbol{\omega}^T \mathbf{x}) + b)$$

Soft margin

The previously defined optimization problem (2.7) is able to find optimal separating hyperplanes for datasets where the sets of points are well separated. For datasets where there is no clear linear separation between classes, as shown in Figure 2.2, no optimal solution can be found. By adding a slack variable $\boldsymbol{\xi} \in \mathbb{R}^n$ as an error factor, data points inside the margin $0 < \xi_k < 1$ and misclassified data points for $\xi > 1$ are allowed.

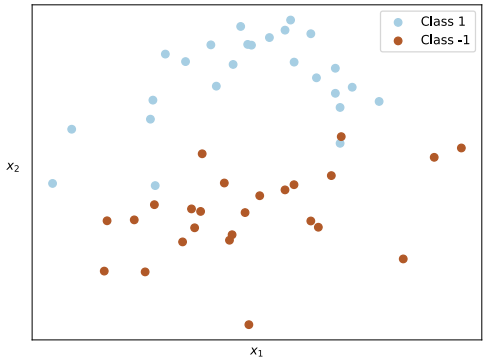

Figure 2.2: Diagram of non-separable dataset.

$$y_k[\boldsymbol{\omega}^T \mathbf{x}_k + b] \geq 1 - \xi_k$$

By introducing this slack variable, the margin can potentially increase indefinitely. To restrict this effect, a term is added to the problem's cost function. This last term is also known as *penalization* term.

$$\frac{c}{p} \sum_{k=1}^{N} \xi_k^p$$

Where c is the weighting hyperparameter to ponder the effect of the slack variable. Larger values of c shadow the effect of the margin maximization term, therefore allowing to find solutions with larger error. The value of p is normally chosen between 1 and 2. When $p = 1$ the problem is known as L1 SVC, and L2 SVC when $p = 2$.

The final problem can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \frac{1}{2}||\boldsymbol{\omega}||_2^2 + \frac{c}{p}\sum_{k=1}^{N}\xi_k^p \\ \text{s. t.} \quad & y_k[\boldsymbol{\omega}^\top \mathbf{x}_k + b] \geq 1 - \xi_k \\ & \xi_k \geq 0 \end{aligned} \tag{2.8}$$

Figure 2.3 illustrates the set of support vectors, now including the subset of data points inside the margin or on the opposite side of the boundary.

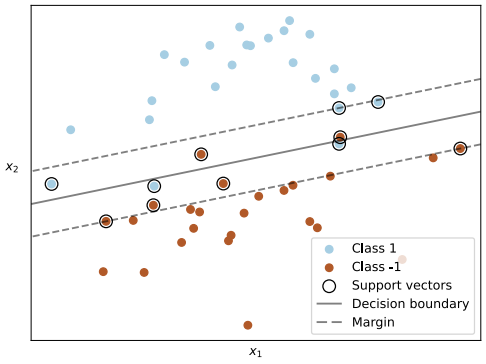

Figure 2.3: Diagram of non-separable dataset with optimal linear hyperplane.

Non-linear functions

As explained in Section 2.1.11, a hyperplane for non-separable datasets can be found by using a penalization term. Another way to achieve separability, without penalization, is by transforming the independent variable $\mathbf{x}$ so that separation is possible in the new feature space. The generalized SVC problem can then be rewritten as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \frac{1}{2}||\boldsymbol{\omega}||_2^2 + \frac{c}{2}\sum_{k=1}^{N}\xi_k^p \\ \text{s. t.} \quad & y_k[\boldsymbol{\omega}^T\boldsymbol{\varphi}(\mathbf{x}_k) + b] \geq 1 - \xi_k, \quad k = 1,\ldots,N \\ & \xi_k \geq 0, \quad k = 1,\ldots,N \end{aligned} \tag{2.9}$$

Where $\varphi : \mathbf{R}^m \to \mathbf{R}^m$ is the transformation function. An example of this transformation is the square function or the sine function. Notice that it is necessary to have information about the data distribution to suggest the best transformation.

The penalization term is commonly used to increase the generalization of the obtained models by finding a balance between bias and variance when complex frontiers are found.

Figure 2.4: Diagram of non-linear dataset with optimal non-linear separating hyperplane.

2.1.12 SVR Mathematical background

In the previous sections the SVM classification problem (SVC) was derived. In the current section, we will derive the formulation for the regression problem (Support Vector Regressor or SVR).

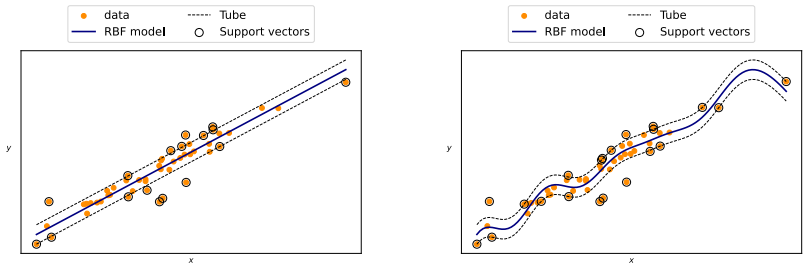

Figure 2.5: Representation regression on dataset using SVR's linear kernel (a) and non-linear kernel (b).

The data set for a regression problem is one where the set of input variables $\mathbf{x} \in \mathbb{R}^{m \times n}$ and the target variable $\mathbf{y} \in \mathbb{R}^m$ are both continuous.

The goal of the regression problem is find a function $f(\mathbf{x})$ that captures the trend and periodicity of the data (see figure 2.5), enabling it to extrapolate or interpolate for new values. A function in the subsequent way can be proposed:

$$\mathbf{y} = f(\mathbf{x}) + \mathbf{e}$$

Where $\mathbf{e} \in \mathbb{R}^n$ corresponds to an error vector and $f(\mathbf{x})$ can be defined with a set of parameters as follows:

$$f(\mathbf{x}) = \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}) + b \tag{2.10}$$

For the Linear Regression problem, where the error is considered to be normally distributed, the cost function or error term is given by the following equations:

$$L(\mathbf{x}, \mathbf{y}) = ||f(\mathbf{x}) - \mathbf{y}||^p = ||\mathbf{e}||^p$$

p is usually selected between 1 and 2. To find the optimal set of parameters, the following optimization problem can be constructed. :

In the linear regression problem, the constraints are incorporated into the problem's functional, transforming it into an unconstrained optimization problem $\min_\theta L(x, y) = \min_\theta ||f(\mathbf{x}) - \mathbf{y}||^p$.

$$\min_{!,b} \sum_{\in N} e_k^p$$
$$\text{s. t.} \quad e_k = f(\mathbf{x}_k) - y_k$$

SVR cost function

In Support Vector Regression theory, the ϵ-insensitive cost function (Introduced by Vapnik [26]), unlike the linear regression cost function,

[26] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998. ISBN 0-471-03003-1

this function penalizes only the terms outside a tube of radius ϵ, as illustrated in the Figure 2.6

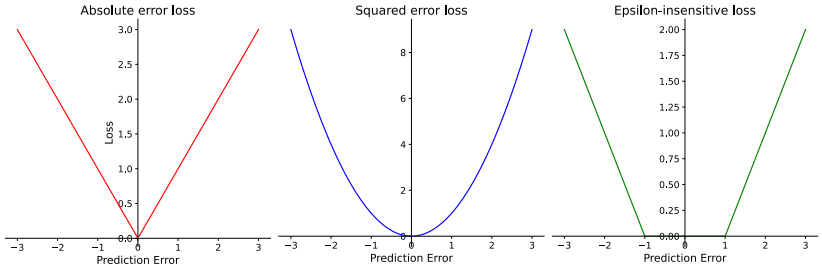

Figure 2.6: Regression loss function. From left to right: Absolute error loss, Squared error loss and Epsilon-insensitive loss

The ϵ cost-insensitive function is expressed as follows

$$L(\mathbf{x}_k, y_k) = \begin{cases} 0 & \text{if } |y_k - f(\mathbf{x}_k)| \leq \epsilon \\ |y_k - f(\mathbf{x}_k)| - \epsilon & \text{otherwise} \end{cases} \tag{2.11}$$

With the previous cost function (2.11), an optimization problem can be proposed,

$$\begin{aligned} \min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & \sum_{k=1}^{N} (\xi_k + \xi_k^*)^p \\ \text{s. t.} \quad & y_k \leq \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b + \epsilon + \xi_k, \quad k = 1, \ldots, N \\ & y_k \geq \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b - \epsilon - \xi_k^*, \quad k = 1, \ldots, N \\ & \xi_k, \xi_k^* \geq 0, \quad k = 1, \ldots, N \end{aligned} \tag{2.12}$$

To better understand (2.12), let's break it down into parts:

$$\begin{aligned} y_k &\leq \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b + \epsilon, \quad k = 1, \ldots, N \\ y_k &\geq \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b - \epsilon, \quad k = 1, \ldots, N \end{aligned}$$

The previous two constraints only allow for data points $f(\mathbf{x}_k)$ inside or on the tube where $|y_k - f(\mathbf{x}_k)| \leq \epsilon$ as expressed in (2.11) first case.

The corresponding ξ_k and ξ_k^* positive slack variables or positive and negative error terms become active when the data points lie outside of the margin, then representing the effect of the second case in (2.11).

The goal of the problem is to minimize the error given by the data points, then the minimum of the functional,$\sum_{k=1}^{n} (\xi_k + \xi_k^*)^p$, can be computed.

SVR Margin

As shown in Figure 2.7, with (2.12) for datasets and a value of ϵ where all data points lie inside the tube, an infinite number of tubes satisfy optimality to the problem.

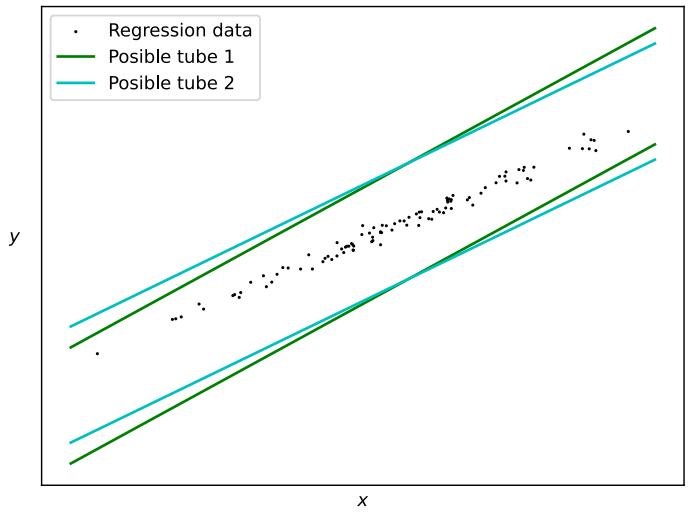

Figure 2.7: Subset of optimal tubes for regression using (2.12)

This set of tubes does not necessarily generalize the data pattern. Assuming that the model with the most generalization is given by the tube with maximum margin, the following optimization can be proposed:

$$\begin{aligned} \min_{\omega,b,\xi,\xi^*} \quad & \frac{1}{2}||\omega||^2 + C\sum_{k=1}^{N}(\xi_k + \xi_k^*)^p \\ \text{s. t.} \quad & y_k \leq \omega^\top \varphi(x_k) + b + \epsilon + \xi_i, \quad k = 1,\dots,N \\ & y_k \geq \omega^\top \varphi(x_k) + b - \epsilon - \xi_i^*, \quad k = 1,\dots,N \\ & \xi_k, \xi_k^* \geq 0, \quad k = 1,\dots,N \end{aligned} \tag{2.13}$$

For this problem, the added term, helps in finding the maximum margin so to center the resulting function, as in the case of the previously explored linear regression errors.

Following (2.11) format, we can propose a new formulation for (2.13):

$$\min J = \min L(x,y) + \frac{1}{2}||\omega||^2$$

2.2 *SVM QCQP formulations*

This module introduces foundational concepts for deriving Quadratically Constrained Quadratic Program (QCQP) Multikernel Support Vector Machines, based on the method presented in [27]. This method offers computational advantages in terms of memory usage

[27] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

and convergence time compared to the approach described in section 2.3, and will be discussed in detail in the next module. The goal of this section is to explore the dual formulation of the problems outlined in the previous section and to introduce the Multikernel QCQP derivations. The derivation for the SVR case will be presented as a precursor to those in chapter 3.

2.2.1 *Dual SVC L1 formulation*

The dual formulation of the SVC leads to the introduction of the kernel matrix [28], taking advantage of non-linear complex transformations. In this section, we will derive the SVC dual formulation for the L1 variants.

[28] Detailed in section 2.1.3

Recalling (2.9), the L1 formulation is writtien as follows:

$$\begin{aligned} \min_{\boldsymbol{\omega},b} \quad & \frac{1}{2}||\boldsymbol{\omega}||_2^2 + c\sum_{k=1}^{N}\xi_k \\ \text{s. t.} \quad & y_k[\boldsymbol{\omega}^T\boldsymbol{\varphi}(\mathbf{x}_k)+b] \geq 1-\xi_k \\ & \xi_k \geq 0 \end{aligned} \tag{2.14}$$

The lagrangian of (2.14) is given by:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\omega},b,\boldsymbol{\xi};\boldsymbol{\alpha},\boldsymbol{\nu}) = & \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{\omega} + c\sum_{k=1}^{N}\xi_k \\ & - \sum_{k=1}^{N}\alpha_k\left[y_k(\boldsymbol{\omega}^T\boldsymbol{\varphi}(\mathbf{x}_k)+b-1)+\xi_k\right] - \sum_{k=1}^{N}\nu_k\xi_k \end{aligned} \tag{2.15}$$

$\alpha_k \in \mathbf{R}_{\geq 0}$ and $\nu_k \in \mathbf{R}_{\geq 0}$ are the Lagrange multiplier that corresponds to the first and second set of constraints (See 2.1.4).

Notice that each term was added to the Lagrangian so that the direction of the gradient leads to an optimal value. As an example, let us look at the second term:

$$-\underbrace{\sum_{k=1}^{N}\nu_k}_{\geq 0}\cdot\underbrace{\xi_k}_{\geq 0} < 0$$

Since this is a minimization problem, which implies a convex cost function, the product Langrange multiplier-constraint is added so that the direction of the gradient, which will be computed in a subsequent process, adds to a negative value. In the case of a maximization problem with inequality constraints, each product has to be added

as a positive term so that the gradient is able to find a maximum during the optimization process.

Karush Kuhn Tucker conditions

Now we will derive the Karush Kuhn Tucker (KKT) conditions, that when satisfied, strong duality holds [29]. This means that the primal and dual problems share the same optimal value. A more concise explanation about the topic can be found in [30].

[29] As an intermediate step to this process, the order of the problems ($\min_{\omega,b}\max_{\alpha}\mathcal{L}$) are swapped ($\max_{\alpha}\min_{\omega,b}\mathcal{L}$) to analyze the primal problem (see 2.1.4.). Strong duality holds since the interior problem is convex

[30] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7

KKT1 or First order conditions: This condition states that the optimal value is found when the gradient of every primal variable is equal to zero.

$$\nabla_{\omega}\mathcal{L} = \omega - \sum_{k=1}^{N}\alpha_k y_k \varphi(\mathbf{x}_k) = 0 \Rightarrow \omega = \sum_{l=1}^{N}\alpha_k y_k \varphi(\mathbf{x}_k) \tag{2.16}$$

$$\nabla_{b}\mathcal{L} = -\sum_{k=1}^{N}\alpha_k y_k = 0 \Rightarrow \sum_{k=1}^{N}\alpha_k y_k = 0 \tag{2.17}$$

$$\nabla_{\xi_k}\mathcal{L} = C - \alpha_k - \nu_k = 0 \Rightarrow \alpha_k + \nu_k = C \tag{2.18}$$

KKT2 or primal feasibility condition: This conditon requires for the existence of a *feasible region* in the primal space. This means that there are some region where the primal conditions are satisfied. When there is no feasible region, the problem is said to be non-viable.

$$\begin{aligned} &y_k[\omega^{\top}\varphi(\mathbf{x}_k) + b] \geq 1 - \xi_k \\ &\xi_k \geq 0 \end{aligned} \tag{2.19}$$

It can be observed that these constraints were implicitly embedded during the construction of the Lagrangian.

KKT3 or dual feasibility condition: In the same way as the previous condition, this condition requires the existence of a feasible region where the dual conditions are satisfied.

$$\alpha_k, \nu_k \geq 0$$

KKT4 or complementary slackness condition: The condition states that when a primal constraint is active (with value greater or equal to zero), its corresponding dual is inactive (its value is zero) and vice versa.

$$\alpha_k\{y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\boldsymbol{x}_k)+b]-1+\xi_k\}=0 \tag{2.20}$$

$$\nu_k\xi_k=0 \tag{2.21}$$

In section 2.2.1, we will get a more detailed explanation about the information that each of these conditions gives.

KKT analysis

Recalling (2.18) from KKT1, along with the conditions from KKT2, KKT3, and KKT4, we will conduct further analysis to gain more insight into the state of the problem under different conditions for α. From (2.18), we find that the value of α ranges between 0 and C to maintain the non-negativity of the dual variables.

CASE 1: $\alpha_k = 0$

$\alpha_k = 0$ implies that $y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] \geq 1-\xi_k$ in (2.20) and $\nu_k = c$ in (2.18). From the latter, since $C > 0$, then $\nu_k > 0$. From (2.21) it can be infered that $\xi_k = 0$; and therefore $y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] \geq 1$. This can be summarized as follows:

$$\alpha_k = 0 \Rightarrow \begin{cases} (2.20)\ \ y_k[\boldsymbol{\omega}\boldsymbol{\varphi}(\mathbf{x}_k)+b] \geq 1-\xi_k \Rightarrow \\ \\ (2.18)\ \ C=\nu_k \Rightarrow \nu_k > 0 \Rightarrow (2.21)\ \ \xi_k = 0 \Rightarrow \end{cases} \boxed{y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] \geq 1}$$

The enclosed result indicates that for $\alpha_k = 0$, the corresponding data point x_k is located on the correct side of the margin.

CASE 2: $0 < \alpha_k < C$

$$0 < \alpha_k < c \Rightarrow \begin{cases} (2.20)\ \ y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] = 1-\xi_k \Rightarrow \\ \\ (2.18)\ \ \nu_k > 0 \Rightarrow (2.21)\ \ \xi_k = 0 \Rightarrow \end{cases} \boxed{y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] = 1}$$

$0 < \alpha_k < C$ implies that x_k lies exactly on the margin. The set of associated data points is referred to as *unbounded support vectors*.

CASE 3: $\alpha_k = c$

$$\alpha_k = C \Rightarrow \begin{cases} (2.20)\ \ y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] = 1-\xi_k \Rightarrow \\ \\ (2.18)\ \ \nu_k = 0 \Rightarrow (2.21)\ \ \xi_k > 0 \Rightarrow \end{cases} \boxed{y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b] \leq 1}$$

$\alpha_k = c$ indicates that x_k is either inside of the margin or on the wrong side of the frontier. These points are called *bounded support vectors*.

Building dual formulation

Substituting kkt conditions (2.16), (2.17) and (2.18) into the lagrangian (2.15),

$$\begin{aligned}\mathcal{D}(\boldsymbol{\alpha}) =& \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k \boldsymbol{\varphi}(\mathbf{x}_l)^\top \boldsymbol{\varphi}(\mathbf{x}_k) + C\sum_{k=1}^{N}\xi_k \\ &- \sum_{k=1}^{\alpha}\alpha_k y_k \boldsymbol{\varphi}(\mathbf{x}_k)^\top \sum_{l=1}^{N}\alpha_l y_l \boldsymbol{\varphi}(\mathbf{x}_l) - b\sum_{k=1}^{N}\alpha_k y_k + \sum_{k=1}^{N}\alpha_k \\ &- \sum_{k=1}^{N}\alpha_k\xi_k - \sum_{k=1}^{N}(-\alpha_k + C)\xi_k\end{aligned} \tag{2.22}$$

To simplify (2.22) lets analyze each term:

$$\begin{aligned}\mathcal{D}(\boldsymbol{\alpha}) =& \underbrace{\frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k \boldsymbol{\varphi}(\mathbf{x}_l)^\top \boldsymbol{\varphi}(\mathbf{x}_k)}_{a} + \underbrace{C\sum_{k=1}^{N}\xi_k}_{b} \\ &\underbrace{- \sum_{k=1}^{\alpha}\alpha_k y_k \boldsymbol{\varphi}(\mathbf{x}_k)^\top \sum_{l=1}^{N}\alpha_l y_l \boldsymbol{\varphi}(\mathbf{x}_l)}_{c=2a} \underbrace{- b\sum_{k=1}^{N}\alpha_k y_k}_{d=0} + \underbrace{\sum_{k=1}^{N}\alpha_k}_{e} \\ &\underbrace{- \sum_{k=1}^{N}\alpha_k\xi_k}_{f} + \underbrace{\sum_{k=1}^{N}\alpha_k\xi_k}_{g=-f} \underbrace{- C\sum_{k=1}^{N}\xi_k}_{h=-b}\end{aligned}$$

It is evident that several terms are duplicated or can be canceled out. Simplifying leads to the following equation:

$$\mathcal{D}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k \boldsymbol{\varphi}(\mathbf{x}_l)^\top \boldsymbol{\varphi}(\mathbf{x}_k) + \sum_{k=1}^{N}\alpha_k \tag{2.23}$$

Equation (2.23) is referred to as the dual Lagrangian. The original problem can be expressed in a compact form as follows:

$$D(\boldsymbol{\alpha}) = \min_{\boldsymbol{\omega}, b} L(\boldsymbol{\omega}, b; \boldsymbol{\alpha})$$

Since the dual problem is a lower bound of the primal problem where the objective function is concave, it's possible to formulate (2.23) as an unconstrained optimization problem:

$$\max_{\boldsymbol{\alpha}} \mathcal{D}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k \boldsymbol{\varphi}(\mathbf{x}_l)^\top \boldsymbol{\varphi}(\mathbf{x}_k) + \sum_{k=1}^{N}\alpha_k$$

Which is the functional of the maximization problem. By changing the sign of the functional and defining $\boldsymbol{\varphi}(\mathbf{x}_l)^T\boldsymbol{\varphi}(\mathbf{x}_k) = k(\mathbf{x}_k, \mathbf{x}_l)$

$$\min_{\alpha} \mathcal{D}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k k(\mathbf{x}_k, \mathbf{x}_l) - \sum_{k=1}^{N}\alpha_k$$

Adding constraint for the upper and lower bound of $\boldsymbol{\alpha}$ derived from (2.18), and (2.17) from KKT1 we get the following optimization problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k k(\mathbf{x}_k, \mathbf{x}_l) - \sum_{k=1}^{N}\alpha_k \\ s.t. \quad & \sum_{k=1}^{N}\alpha_k y_k = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned} \tag{2.24}$$

Considering that all KKT conditions are met, strong duality is satisfied, and the optimal value can be found for the original problem (2.14).

(2.24) CAN BE CASTED INTO A VECTOR FORMAT:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top (\mathbf{Y} \circ \mathbf{K})\, \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned} \tag{2.25}$$

The $\circ$ symbol accounts for the Hadamard product and $\mathbf{Y} = \mathbf{y}\mathbf{y}^\top$.

Predicting new values from the dual formulation

As a result of the optimization process, a set of support vectors $\mathbf{x}_s$ are recovered where the value of $0 < \alpha_k \leq C$ [31].

[31] This is the set of data points that construct the decision function (see 2.1.11 and 2.2.1)

Suppose that we want to predict a new set of labels $\hat{\mathbf{y}}_t$ that correspond to a set of data points $\mathbf{x}_t$.

To predict new values, we can use the following decision function:

$$f(\mathbf{x}_t) = \boldsymbol{\omega}_s^\top \boldsymbol{\varphi}(\mathbf{x}_t) + b \tag{2.26}$$

And to classify the entries, the following criteria should be followed:

$$\begin{cases} \text{Class 1 if } f(\mathbf{x}_t) < 0 \\ \text{Class 2 if } f(\mathbf{x}_t) > 0 \end{cases} \tag{2.27}$$

From here the value of $\boldsymbol{\omega}_s$ can be obtained from the vector format of (2.16) written as follows:

$$\boldsymbol{\omega}_s = \mathbf{Y}_s \boldsymbol{\alpha}_s \boldsymbol{\varphi}(\mathbf{x}_s) \tag{2.28}$$

Substituting (2.28) into (2.26) results in the following equation:

$$f(\mathbf{x}_t) = \boldsymbol{Y}_s \boldsymbol{\alpha}_s \boldsymbol{\varphi}(\mathbf{x}_s)\boldsymbol{\varphi}(\mathbf{x}_t) + b$$

Adding the sign function to replace (2.27) and simplifying the notation:

$$f(\mathbf{x}) = \text{sign}\left(\boldsymbol{\alpha}\left(\boldsymbol{Y} \circ \mathbf{k}(\mathbf{x}_s, \mathbf{x}_t)\right) + b\right)$$

For the RKHS theory, introduced in 2.1.3 we know that by creating the kernel, it's possible to find a solution for the new values $\mathbf{x}_t$ with the information contained on the support vectors $\mathbf{x}_s$

Compute b

To obtain the value of b we can use:

$$y_k[\boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b] = 1$$

Which corresponds to the subset of support vectors $x_k \in U$ that are on the margin (unbounded support vectors).

To solve for b, recall that $y_k \in \{1, -1\}$, therefore:

$$\frac{y_k}{y_k}[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] = \frac{1}{y_k} \Rightarrow \boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k$$

$$b = y_k - \boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k)$$

Which encapsulates the effect of data point x_k in b. To account for the effect of every support vector, the average of the whole subset is computed:

$$b = \frac{1}{|U|} \sum_{\in U} y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k)$$

Substituting the value of $\boldsymbol{\omega}$:

$$b = \frac{1}{|U|} \sum_{\in U} \left(y_k - \sum_{\in S} \alpha_l y_l k(\mathbf{x}_l, \mathbf{x}_k) \right)$$

Here $\mathbf{x}_l \in S$ corresponds to the whole set of support vectors.

2.2.2 *Dual SVC L2 formulation*

Lets define the primal SVC-L2 from (2.8) with $p = 2$.

$$\begin{aligned} \min_{\boldsymbol{\omega}, b} \quad & \frac{1}{2}||\boldsymbol{\omega}||_2^2 + \frac{C}{2} \sum_{k=1}^{N} \xi_k^2 \\ \text{s. t.} \quad & y_k[\boldsymbol{\omega}^\top \mathbf{x}_k + b] \geq 1 - \xi_k, \quad k = 1, \cdots, N \end{aligned}$$

Notice that the constraint $\xi_k \geq 0$ is not included in this problem, since it is implicitly embedded in the penalization term. By squaring ξ_k, only positive values are allowed.

Lagrangian

$$\mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \nu) = \frac{1}{2}\boldsymbol{\omega}^\top \boldsymbol{\omega} + \frac{C}{2}\sum_{k=1}^{N} \xi_k^2 - \sum_{k=1}^{N} \alpha_k \left(y_k \left[\boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b \right] - 1 + \xi_k \right) \tag{2.29}$$

Karush Kuhn Tucker conditions

KKT1

$$\nabla_{\boldsymbol{\omega}} \mathcal{L} = \boldsymbol{\omega} - \sum_{k=1}^{N} \alpha_k y_k \boldsymbol{\varphi}(\mathbf{x}_k) = 0 \Rightarrow \boldsymbol{\omega} = \sum_{l=1}^{N} \alpha_k y_k \boldsymbol{\varphi}(\mathbf{x}_k) \tag{2.30}$$

$$\nabla_b \mathcal{L} = -\sum_{k=1}^{N} \alpha_k y_k = 0 \Rightarrow \sum_{k=1}^{N} \alpha_k y_k = 0 \tag{2.31}$$

$$\nabla_{\xi_k} \mathcal{L} = C\xi_k - \alpha_k = 0 \Rightarrow C\xi_k = \alpha_k \tag{2.32}$$

(2.32) shows that $\xi_k \geq 0$ since $\alpha_k \geq 0$.

KKT2

$$y_k[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] \geq 1 - \xi_k$$

KKT3

$$\alpha_k \geq 0 \tag{2.33}$$

KKT4

$$\alpha_k \{ y_k[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] - 1 + \xi_k \} = 0$$

KKT analysis

Case 1: $\alpha_k = 0$

$$\alpha_k = 0 \Rightarrow \begin{cases} y_k[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] \geq 1 - \xi_k \Rightarrow & \boxed{y_k[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] \geq 1} \\ \xi_k = 0 \Rightarrow & \end{cases}$$

The enclosed result indicates that for $\alpha_k = 0$, the corresponding data point $\mathbf{x}_k$ is located on the correct side of the margin.

Case 2: $\alpha_k \geq 0$

$$\alpha_k \geq 0 \Rightarrow \begin{cases} y_k[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] = 1 - \xi_k \Rightarrow & \boxed{y_k[\boldsymbol{\omega}^T \boldsymbol{\varphi}(\mathbf{x}_k) + b] \leq 1} \\ \alpha_k \geq 0 \Rightarrow \xi_k \geq 0 \Rightarrow & \end{cases}$$

$\alpha_k \geq 0$ indicates that $\mathbf{x}_k$ is either inside of the margin or on the wrong side of the frontier.

The previous results indicate that the support vectors correspond to the set of α_i where $\alpha_i > 0$.

Building dual formulation

Substituting (2.30), (2.31), and (2.32) into (2.29)

$$\begin{aligned} \mathcal{D}(\boldsymbol{\alpha}) =& \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_l \alpha_k y_l y_k \boldsymbol{\varphi}(\mathbf{x}_l)^\top \boldsymbol{\varphi}(\mathbf{x}_k) + \frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 \\ & - \sum_{k=1}^{\alpha} \alpha_k y_k \boldsymbol{\varphi}(\mathbf{x}_k)^\top \sum_{l=1}^{N} \alpha_l y_l \boldsymbol{\varphi}(\mathbf{x}_l) - b \sum_{k=1}^{N} \alpha_k y_k + \sum_{k=1}^{N} \alpha_k \\ & - \frac{1}{C} \sum_{k=1}^{N} \alpha_k^2 \end{aligned} \tag{2.34}$$

After simplifying (2.34) and substituting $\boldsymbol{\varphi}(\mathbf{x}_l)^\top \boldsymbol{\varphi}(\mathbf{x}_k) = k(\mathbf{x}_k, \mathbf{x}_l)$.

$$\mathcal{D}(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_l \alpha_k y_l y_k k(\mathbf{x}_k, \mathbf{x}_l) - \frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 + \sum_{k=1}^{N} \alpha_k \tag{2.35}$$

Since the size of the k-ithm and l-ithm sets are both size N, it's possible to break down the following term from equation (2.35)

$$-\frac{1}{2C} \sum_{k=1}^{N} \alpha_k^2 = -\frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_l \alpha_k \left(\frac{\delta_{kl}}{C} \right)$$

Inserting them back into (2.35) results in the following equation:

$$\mathcal{D}(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_l \alpha_k y_l y_k k(\mathbf{x}_k, \mathbf{x}_l) - \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_l \alpha_k \left(\frac{\delta_{kl}}{C} \right) + \sum_{k=1}^{N} \alpha_k$$

With the previous change in format, it's possible to simplify (2.35) by joining the first and second term in the following way:

$$\max_{\boldsymbol{\alpha}} \mathcal{D}(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_l \alpha_k y_l y_k \left(k(\mathbf{x}_k, \mathbf{x}_l) + \frac{\delta_{kl}}{C} \right) + \sum_{k=1}^{N} \alpha_k$$

The previous is the unconstrained *Wolfe dual*. The dual problem is built by adding the constrains (2.31) and (2.33),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} & \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N}\alpha_l\alpha_k y_l y_k\left(k(\mathbf{x}_k,\mathbf{x}_l)+\frac{\delta_{kl}}{C}\right)-\sum_{k=1}^{N}\alpha_k \\ s.t. & \sum_{k=1}^{N}\alpha_k y_k = 0 \\ & \alpha_k \geq 0 \quad k=1,\cdots,N \end{aligned} \tag{2.36}$$

Unlike the L1 formulation, the L2 formulation incorporates a $(1/C)$ term added to each term of the kernel trace. For the problem to be feasible, the kernel matrix must be positive semidefinite. In situations where the kernel matrix is not positive semidefinite, such as when there's correlation between the independent variables, a solution can be achieved by adding this term with an adjusted C value [32]. This approach enhances the stability of the computation.

[32] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2nd edition, 2002. ISBN 0-89871-521-0

(2.36) can be cast into a vector format in the following way:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} & \frac{1}{2}\boldsymbol{\alpha}^\top\left[\mathbf{Y}\circ\left(\mathbf{K}+\mathbf{I}\frac{1}{C}\right)\right]\boldsymbol{\alpha}-\boldsymbol{\alpha}^\top\mathbf{1} \\ s.t. & \boldsymbol{\alpha}^\top\mathbf{y}=0 \\ & \alpha_k\geq 0 \quad k=1,\cdots,N \end{aligned} \tag{2.37}$$

To predict new values, it's possible to derive it from the primal constraint (2.19) for the subset of support vectors $\mathbf{x}_k \in \mathbf{S}$, expressed as follows:

A more detailed derivation for the L1 decision function is provided in the previous section 2.2.1

$$y_k[\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b]=1$$

Solving for b, substituting the value of $\boldsymbol{\omega}$ found in (2.30) and averaging over the set of all unbounded support vectors results in the following equation:

$$b=\frac{1}{|\mathbf{S}|}\sum_{k\in S}y_k-\sum_{l\in S}\alpha_l y_l\left(k(\mathbf{x}_k,\mathbf{x}_l)+\frac{\delta_{kl}}{C}\right)$$

Notice that the $\frac{\delta_{kl}}{C}$ term was added to the kernel term as in the training process to maintain consistency.

2.2.3 *Dual SVR L1 formulation*

In this section we will derive the dual formulation for the SVR problem, expressed in 2.1.12

From (2.13) with $p=1$, we get,

$$\begin{aligned}
\min_{\boldsymbol{\omega},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad & \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{k=1}^{N}(\xi_k + \xi_k^*) \\
\text{subject to} \quad & \xi_k + \epsilon \geq y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b, \quad k = 1, \ldots, N \\
& \xi_k^* + \epsilon \geq \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b - y_k, \quad k = 1, \ldots, N \\
& \xi_k, \xi_k^* \geq 0, \quad k = 1, \ldots, N
\end{aligned}$$

Lagrangian

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\nu}, \boldsymbol{\nu}^*) = & \frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{k=1}^{N}(\xi_k + \xi_k^*) \\
& - \sum_{k=1}^{N}\alpha_k[\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b] \\
& - \sum_{k=1}^{N}\alpha_k^*[\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b] \\
& - \sum_{k=1}^{N}\nu_k\xi_k - \sum_{k=1}^{N}\nu_k^*\xi_k^*
\end{aligned}$$

Karush Kuhn Tucker conditions

KKT1

$$\begin{aligned}
\nabla_\omega \mathcal{L} &= \boldsymbol{\omega} - \sum_{k=1}^{N}\alpha_k\boldsymbol{\varphi}(\mathbf{x}_k) + \sum_{k=1}^{N}\alpha_k^*\boldsymbol{\varphi}(\mathbf{x}_k) = 0 \\
&\Rightarrow \boldsymbol{\omega} = \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\boldsymbol{\varphi}(\mathbf{x}_k) \\
\frac{\partial \mathcal{L}}{\partial b} &= -\sum_{k=1}^{N}\alpha_k + \sum_{k=1}^{N}\alpha_k^* = 0 \Rightarrow \quad \sum_{k=1}^{N}\alpha_k = \sum_{k=1}^{N}\alpha_k^* \\
\frac{\partial \mathcal{L}}{\partial \xi_k} &= C - \alpha_k - \nu_k = 0 \Rightarrow \quad \nu_k = C - \alpha_k \qquad (2.38) \\
\frac{\partial \mathcal{L}}{\partial \xi_k^*} &= C - \alpha_k^* - \nu_k^* = 0 \Rightarrow \quad \nu_k^* = C - \alpha_k^* \qquad (2.39)
\end{aligned}$$

KKT2

$$\begin{aligned}
& \xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b \geq 0 \\
& \xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b \geq 0 \\
& \xi_k \geq 0 \\
& \xi_k^* \geq 0
\end{aligned}$$

KKT3

$$\alpha_k \geq 0$$
$$\alpha_k^* \geq 0$$
$$\nu_k \geq 0$$
$$\nu_k^* \geq 0$$

KKT4

$$\alpha_k[\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b] = 0 \tag{2.40}$$
$$\alpha_k^*[\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b] = 0 \tag{2.41}$$
$$\nu_k \xi_k = 0$$
$$\nu_k^* \xi_k^* = 0$$

From the previous results, it can be noted that the primal constraint (2.49) is associated with α_k and (2.50) is associated with α_k^*. Immediately it's visible that $\alpha_k \alpha_k^* = 0$, since when α_k is active, meaning that $(y_k, \mathbf{x}_k)$ is outside of one side of the tube, α_k^* is inactive because the data point cannot be on the other side of the tube.

KKT Analysis

To derive the following results, the previous KKT conditions and (2.39) are taken into consideration:

Case 1: $\alpha_k = 0$ *and* $\alpha_k^* = 0$

$$\begin{aligned} \xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b > 0 \Rightarrow \quad & \epsilon + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b > y_k \\ \xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b > 0 \Rightarrow \quad & -\epsilon + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b < y_k \end{aligned}$$

$\alpha_k, \alpha_k^* = 0$ implies $(y_k, \mathbf{x}_k)$ inside the margin.

Case 2: $0 < \alpha_k < C$ *or* $0 < \alpha_k^* < C$

$$\begin{aligned} \xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b = 0 \Rightarrow \quad & \epsilon + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k \\ \xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b = 0 \Rightarrow \quad & -\epsilon + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k \end{aligned} \tag{2.42}$$

$0 < \alpha_k, \alpha_k^* < C$ implies $(y_k, \mathbf{x}_k)$ on the margin.

Case 3: $\alpha_k = C$ *or* $\alpha_k^* = C$

$$\begin{aligned} \xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b = 0 \Rightarrow \quad & \xi_k + \epsilon + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k \\ \xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b = 0 \Rightarrow \quad & -\xi_k - \epsilon + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k \end{aligned}$$

$\alpha_k = C$ or $\alpha_k^* = C$ implies $(y_k, \mathbf{x}_k)$ outside the margin.

To simplify the substitution process during the dual construction, a two-step process will be performed.

Substituting (2.38) and (2.39) into the Lagrangian function ($\boldsymbol{\omega}$ is not substituted):

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}||\boldsymbol{\omega}||^2 + C\sum_{k=1}^{N}(\xi_k + \xi_k^*) \\ &- \sum_{k=1}^{N}\alpha_k\xi_k - \sum_{k=1}^{N}\alpha_k^*\xi_k^* \\ &- \sum_{k=1}^{N}\alpha_k\epsilon + \sum_{k=1}^{N}\alpha_k y_k - \sum_{k=1}^{N}\alpha_k\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - \sum_{k=1}^{N}\alpha_k b \\ &- \sum_{k=1}^{N}\alpha_k^*\epsilon - \sum_{k=1}^{N}\alpha_k^* y_k + \sum_{k=1}^{N}\alpha_k^*\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + \sum_{k=1}^{N}\alpha_k^* b \\ &- \sum_{k=1}^{N}(C-\alpha_k^*)\xi_k - \sum_{k=1}^{N}(C-\alpha_k^*)\xi_k^*\end{aligned}$$

Simplifying

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}||\boldsymbol{\omega}||^2 \\ &- \sum_{k=1}^{N}\alpha_k\epsilon + \sum_{k=1}^{N}\alpha_k y_k - \sum_{k=1}^{N}\alpha_k\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) \\ &- \sum_{k=1}^{N}\alpha_k^*\epsilon - \sum_{k=1}^{N}\alpha_k^* y_k + \sum_{k=1}^{N}\alpha_k^*\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)\end{aligned}$$

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}||\boldsymbol{\omega}||^2 - \sum_{k=1}^{N}(\alpha_k + \alpha_k^*)\epsilon + \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)y_k \\ &- \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)\end{aligned}$$

Substituting $\boldsymbol{\omega}$

$$\begin{aligned}\mathcal{D} = &\frac{1}{2}\sum_{k,l=1}^{N}(\alpha_k - \alpha_k^*)^\top(\alpha_l - \alpha_l^*)\boldsymbol{\varphi}(\mathbf{x}_l)^\top\boldsymbol{\varphi}(\mathbf{x}_k) \\ &- \sum_{k=1}^{N}(\alpha_k + \alpha_k^*)\epsilon \\ &+ \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)y_k - \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)(\alpha_l - \alpha_l^*)\boldsymbol{\varphi}(\mathbf{x}_l)^\top\boldsymbol{\varphi}(\mathbf{x}_k)\end{aligned}$$

Simplifying

$$\begin{aligned}\mathcal{D}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) = &-\frac{1}{2}\sum_{k,l=1}^{N}(\alpha_k - \alpha_k^*)^\top(\alpha_l - \alpha_l^*)\boldsymbol{\varphi}(\mathbf{x}_l)^\top\boldsymbol{\varphi}(\mathbf{x}_k) \\ &- \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\epsilon + \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)y_k\end{aligned}$$

Dual problem:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{1}{2}\sum_{k,l=1}^{N}(\alpha_k-\alpha_k^*)^\top(\alpha_l-\alpha_l^*)k(\mathbf{x}_l,\mathbf{x}_k)\\ & -\epsilon\sum_{k=1}^{N}(\alpha_k+\alpha_k^*)+\sum_{k=1}^{N}y_k(\alpha_k-\alpha_k^*)\\ \text{s.t.} \quad & \sum_{k=1}^{N}(\alpha_k-\alpha_k^*)=0\\ & 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N\end{aligned}$$

Vector format:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\ & -\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0\\ & 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N\end{aligned}\tag{2.43}$$

To obtain b, from (2.42) we get two sets of conditions associated with the dual unbounded conditions:

$$\begin{aligned}\epsilon+\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b=y_k \quad &\Rightarrow \quad b=y_k-\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)-\epsilon\\ -\epsilon+\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+b=y_k \quad &\Rightarrow \quad b=y_k-\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)+\epsilon\end{aligned}$$

To compute b, it is necessary to compute the average of both sets

$$\begin{aligned}b_\alpha &= \frac{1}{|U|}\sum_{k=1}^{U}\left(y_k-\frac{1}{|S|}\sum_{l=1}^{S}[(\alpha_l-\alpha_l^*)k(\mathbf{x}_l,\mathbf{x}_k)]+\epsilon\right)\text{ if } 0<\alpha_k\leq C\\ b_{\alpha^*} &= \frac{1}{|U^*|}\sum_{k=1}^{U^*}\left(y_k-\frac{1}{|S^*|}\sum_{l=1}^{S^*}[(\alpha_l-\alpha_l^*)k(\mathbf{x}_l,\mathbf{x}_k)]-\epsilon\right)\text{ if } 0<\alpha_k^*\leq C\end{aligned}\tag{2.44}$$

The U set corresponds to the associated unbounded conditions $0<\alpha<C$, while the S set corresponds to the set of conditions associated with all support vectors where $\alpha>0$. Recalling that $\alpha\alpha^*=0$, it's possible to simplify (2.44),

$$\begin{aligned}b_\alpha &= \frac{1}{|U|}\sum_{k=1}^{U}\left(y_k-\frac{1}{|S|}\sum_{l=1}^{S}\alpha_l k(\mathbf{x}_l,\mathbf{x}_k)+\epsilon\right)\text{ if } 0<\alpha_k\leq C\\ b_{\alpha^*} &= \frac{1}{|U^*|}\sum_{k=1}^{U^*}\left(y_k+\frac{1}{|S^*|}\sum_{l=1}^{S^*}\alpha_l^* k(\mathbf{x}_l,\mathbf{x}_k)-\epsilon\right)\text{ if } 0<\alpha_k^*\leq C\end{aligned}$$

Geometrically speaking, S and U correspond to the set of support vectors and unbounded support vectors for one side of the tube, while S^* and U^* correspond to the set and subset of the opposite side.

The result in (2.44) is averaged to get a single value of b.

$$b = \frac{1}{2}\left(b_{\alpha} + b_{\alpha^*}\right)$$

To predict a new subset $\mathbf{x}$, we can use (2.10), which computes the reproducing function or the margin of the tube for the new values,

$$f(\mathbf{x}) = \frac{1}{|S^*|}\sum_{l=1}^{S^*}(\alpha_l - \alpha_l^*)k(\mathbf{x}_l, \mathbf{x}) + b$$

2.2.4 *Dual SVR-L2 formulation*

(2.13) WITH $p = 2$:

$$\begin{aligned} \min_{\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad & \frac{1}{2}||\boldsymbol{\omega}||^2 + \frac{C}{2}\sum_{k=1}^{N}(\xi_k + \xi_k^*)^2 \\ \text{s. t.} \quad & \xi_k + \epsilon \geq y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b, \quad k = 1, \ldots, N \\ & \xi_k^* + \epsilon \geq \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b - y_k, \quad k = 1, \ldots, N \end{aligned}$$

Notice that the constraint $\xi_k, \xi_k^* \geq 0$ is not used here, since the sum of squares in the functional encapsulates the non-negativity of the slack variables.

Lagrangian

$$\begin{aligned} \mathcal{L}(\boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\nu}, \boldsymbol{\nu}^*) = & \frac{1}{2}||\boldsymbol{\omega}||^2 + \frac{C}{2}\sum_{k=1}^{N}(\xi_k + \xi_k^*)^2 \\ & - \sum_{k=1}^{N}\alpha_k[\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) + b] \\ & - \sum_{k=1}^{N}\alpha_k^*[\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top \boldsymbol{\varphi}(\mathbf{x}_k) - b] \end{aligned}$$

Karush Kuhn Tucker conditions

KKT1

$$\nabla_{\boldsymbol{\omega}}\mathcal{L} = \boldsymbol{\omega} - \sum_{k=1}^{N}\alpha_k\boldsymbol{\varphi}(\mathbf{x}_k) + \sum_{k=1}^{N}\alpha_k^*\boldsymbol{\varphi}(\mathbf{x}_k) = 0$$

$$\Rightarrow \boldsymbol{\omega} = \sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\boldsymbol{\varphi}(\mathbf{x}_k) \tag{2.45}$$

$$\frac{\partial\mathcal{L}}{\partial b} = -\sum_{k=1}^{N}\alpha_k + \sum_{k=1}^{N}\alpha_k^* = 0 \Rightarrow \sum_{k=1}^{N}\alpha_k = \sum_{k=1}^{N}\alpha_k^*$$

$$\Rightarrow \sum_{k=1}^{N}(\alpha_k - \alpha_k^*) = 0 \tag{2.46}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_k} = C\xi_k - \alpha_k = 0 \Rightarrow \quad \xi_k = \alpha_k/C \tag{2.47}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_k^*} = C\xi_k - \alpha_k^* = 0 \Rightarrow \quad \xi_k^* = \alpha_k^*/C \tag{2.48}$$

KKT2

$$\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b \geq 0$$
$$\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - b \geq 0$$

KKT3

$$\alpha_k \geq 0$$
$$\alpha_k^* \geq 0$$

KKT4

$$\alpha_k[\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b] = 0 \tag{2.49}$$
$$\alpha_k^*[\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - b] = 0 \tag{2.50}$$

KKT Analysis

Case 1: $\alpha_k = 0$ *and* $\alpha_k^* = 0$

$$\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b > 0 \quad \Rightarrow \quad \epsilon + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b > y_k$$
$$\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - b > 0 \quad \Rightarrow \quad -\epsilon + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b < y_k$$

$\alpha_k, \alpha_k^* = 0$ implies $(y_k, \mathbf{x}_k)$ inside the margin.

Case 2: $\alpha_k > 0$ *or* $\alpha_k^* > 0$

$$\xi_k + \epsilon - y_k + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b = 0 \quad \Rightarrow \quad \xi_k + \epsilon + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k$$
$$\xi_k^* + \epsilon + y_k - \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - b = 0 \quad \Rightarrow \quad -\xi_k - \epsilon + \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + b = y_k \tag{2.51}$$

$\alpha_k > 0$ or $\alpha_k^* > 0$ implies $(y_k, \mathbf{x}_k)$ outside the margin.

SVR L2 Dual formulation

Substituting (2.47), (2.48) into the Lagrangian function:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}||\boldsymbol{\omega}||^2 + \frac{C}{2}\sum_{k=1}^{N}\left(\frac{\alpha_k}{C} + \frac{\alpha_k^*}{C}\right)^2 \\ &- \frac{1}{C}\sum_{k=1}^{N}\alpha_k^2 - \frac{1}{C}\sum_{k=1}^{N}\alpha_k^{*2} \\ &- \sum_{k=1}^{N}\alpha_k\epsilon + \sum_{k=1}^{N}\alpha_k y_k - \sum_{k=1}^{N}\alpha_k\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - \sum_{k=1}^{N}\alpha_k b \\ &- \sum_{k=1}^{N}\alpha_k^*\epsilon - \sum_{k=1}^{N}\alpha_k^* y_k + \sum_{k=1}^{N}\alpha_k^*\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + \sum_{k=1}^{N}\alpha_k^* b\end{aligned} \tag{2.52}$$

Rearrenging (2.52)

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}||\boldsymbol{\omega}||^2 + \frac{1}{2C}\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)^2 \\ &- \epsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + y_k\sum_{k=1}^{N}(\alpha_k - \alpha_k^*) - b\sum_{k=1}^{N}(\alpha_k - \alpha_k^*) \\ &- \sum_{k=1}^{N}\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)\,(\alpha_k - \alpha_k^*)\end{aligned}$$

From (2.46) it is clear that the term that contains b can be eliminated. Arrenging the squared term differently:

$$\begin{aligned}\mathcal{L} = &\frac{1}{2}||\boldsymbol{\omega}||^2 - \frac{1}{2C}\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)^\top(\alpha_k - \alpha_k^*) \\ &- \sum_{k=1}^{N}\boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k)\,(\alpha_k - \alpha_k^*) \\ &- \epsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + y_k\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\end{aligned} \tag{2.53}$$

Substituting (2.45) in (2.53) results in the following equation:

$$\begin{aligned}\mathcal{D} = &\frac{1}{2}\sum_{k,l=1}^{n}(\alpha_l - \alpha_l^*)^\top(\alpha_k - \alpha_k^*)\,\boldsymbol{\varphi}(\mathbf{x}_l)^\top\boldsymbol{\varphi}(\mathbf{x}_k) \\ &- \frac{1}{2C}\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)^\top(\alpha_k - \alpha_k^*) \\ &- \sum_{k,l=1}^{n}(\alpha_l - \alpha_l^*)^\top\boldsymbol{\varphi}(\mathbf{x}_l)^\top\boldsymbol{\varphi}(\mathbf{x}_k)\,(\alpha_k - \alpha_k^*) \\ &- \epsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + y_k\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\end{aligned} \tag{2.54}$$

Simplifying (2.54):

$$\begin{aligned}\mathcal{D} = &-\frac{1}{2}\sum_{k,l=1}^{N}\left(\alpha_l - \alpha_l^*\right)^\top\left(\alpha_k - \alpha_k^*\right)\boldsymbol{\varphi}(\mathbf{x}_l)^\top\boldsymbol{\varphi}(\mathbf{x}_k)\\ &-\frac{1}{2C}\sum_{k=1}^{N}\left(\alpha_k - \alpha_k^*\right)^\top\left(\alpha_k - \alpha_k^*\right)\\ &-\epsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + y_k\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\end{aligned} \tag{2.55}$$

Arrenging (2.55) in the standard format,

$$\begin{aligned}\mathcal{D}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*; \boldsymbol{\omega}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*) = &-\epsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + y_k\sum_{k=1}^{N}(\alpha_k - \alpha_k^*)\\ &-\frac{1}{2}\sum_{k,l=1}^{N}\left(\alpha_l - \alpha_l^*\right)^\top\left(\alpha_k - \alpha_k^*\right)\left(k(l,k) + \frac{\delta_{l,k}}{C}\right)\end{aligned}$$

Adding (2.46) and KKT3 conditions, and setting up the optimization problem,

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}\quad &-\frac{1}{2}\sum_{k,l=1}^{N}\left(\alpha_l - \alpha_l^*\right)^\top\left(\alpha_k - \alpha_k^*\right)\left(k(l,k) + \frac{\delta_{l,k}}{C}\right)\\ &-\epsilon\sum_{k=1}^{N}(\alpha_k + \alpha_k^*) + \sum_{k=1}^{N} y_k(\alpha_k - \alpha_k^*)\\ \text{s.t.}\quad &\sum_{k=1}^{N}(\alpha_k - \alpha_k^*) = 0\\ &\alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N\end{aligned} \tag{2.56}$$

Then (2.56) in vector format,

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*}\quad &-\frac{1}{2}\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right)^\top\left(K + I\frac{1}{C}\right)\left(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\right)\\ &-\epsilon\mathbf{1}^\top(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) + y^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)\\ \text{s.t.}\quad &\mathbf{1}^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0\\ &\alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N\end{aligned} \tag{2.57}$$

From the KKT analysis, we can see that, compared to the SVR-L1 formulation, there is no set of constraints where $y = f(x) \pm \epsilon$. To obtain b, we will consider (2.51). Solving for b:

$$\begin{aligned}b &= y_k - \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) - \epsilon - \xi_k\\ b &= y_k - \boldsymbol{\omega}^\top\boldsymbol{\varphi}(\mathbf{x}_k) + \epsilon + \xi_k^*\end{aligned}$$

Which are associated with $\alpha_k, \alpha_k^* > 0$; therefore, to compute the average, all support vectors will be considered. Substituting the values $\boldsymbol{\omega}$, $\boldsymbol{\xi}$, and $\boldsymbol{\xi}^*$ and computing the mean over the set of support vectors:

$$b_{\boldsymbol{\alpha}} = \frac{1}{|S|}\sum_{k=1}^{S} y_k - (\alpha_l - \alpha_l^*)\left(k(x_l, x_k) + \frac{\delta_{lk}}{C}\right) - \epsilon - \frac{\alpha_k}{C} \quad \text{if } \alpha_k > 0$$

$$b_{\alpha^*} = \frac{1}{|S^*|}\sum_{k=1}^{S^*} y_k - (\alpha_l - \alpha_l^*)\left(k(x_l, x_k) + \frac{\delta_{lk}}{C}\right) + \epsilon + \frac{\alpha_k^*}{C} \quad \text{if } \alpha_k^* > 0$$

$$b = \frac{1}{2}\left(b_{\boldsymbol{\alpha}} + b_{\boldsymbol{\alpha}^*}\right)$$

To predict a new subset x we can use (2.10) which computes the reproducing function or the margin of the tube for the new values,

$$f(x) = \frac{1}{|S^*|}\sum_{l=1}^{S^*}(\alpha_l - \alpha_l^*)k(x_l, x) + b$$

2.2.5 *SVC QCQP L1 Multikernel Formulation*

In this and the following section, we will present a method to derive the SVC formulation with automatic kernel weights adjustments for the multikernel problem 2.1.3. This method is an alternative to the Semidefinite Programming method, which is computationally more demanding and will be explained in detail in section 2.3.9.

The Quadratically Constrained Quadratic Program (QCQP) version was first derived in [33], where a multilevel problem is constructed by nesting the dual formulation (2.25), obtained in section 2.2.1, with $\mathbf{K}$ as a linear combination of kernels, $\mathbf{K} = \sum_{i=1}^{M} \mu_i \mathbf{k}_i$, into an outer problem where $\boldsymbol{\mu}$ is optimized. This multi-level problem mimics a $\min - \max$ problem.

[33] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

$$\begin{aligned} \Omega(\mathbf{K}) = \max_{\boldsymbol{\alpha}} \quad & -\boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \sum_{i=1}^{M} \mu_i \mathbf{k}_i\right) \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C, \quad k = 1, \cdots, N \end{aligned}$$

The inner problem designates a performance measure $\Omega(\mathbf{K})$ that checks the *quality* of the kernel for a given set of labels. A constraint can be added to limit the search space for possible μ_i. The one presented in (Lanckriet et al.) involves setting the trace of the kernels (i.e., $\text{trace}(\mathbf{K}) = c$). Because the trace is a linear operator, we have $r = \sum_{i=1}^{M} \text{trace}(\mathbf{k}_i) = \text{trace}\left(\sum_{i=1}^{M} \mathbf{k}_i\right)$. The resulting nested problem appears as follows:

$$\min_{\boldsymbol{\mu}} \Omega(\mathbf{K}) \; ; \text{s.t } \mu_i \geq 0 \; \forall i, \boldsymbol{\mu}^\top \mathbf{r} = c$$

In extended format,

$$\begin{aligned}\min_{\boldsymbol{\mu}}\max_{\boldsymbol{\alpha}} \quad & -\boldsymbol{\alpha}^\top\left(\mathbf{Y}\circ\sum_{i=1}^{M}\mu_i\mathbf{k}_i\right)\boldsymbol{\alpha}+2\boldsymbol{\alpha}^\top\mathbf{1}\\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0\\ & 0\leq\alpha_k\leq C,\quad k=1,\cdots,N\\ & \mu_i\geq 0,\quad i=1\cdots,M\\ & \boldsymbol{\mu}^\top\mathbf{r}=c\end{aligned}$$

Since the inner problem is convex (concave) with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$, and the outer problem is convex (linear) with respect to $\boldsymbol{\mu}$, strong duality holds, allowing the order of the problems to be switched without changing the optimal value,

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\min_{\boldsymbol{\mu}} \quad & -\boldsymbol{\alpha}^\top\left(\mathbf{Y}\circ\sum_{i=1}^{M}\mu_i\mathbf{k}_i\right)\boldsymbol{\alpha}+2\boldsymbol{\alpha}^\top\mathbf{1}\\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0\\ & 0\leq\alpha_k\leq C,\quad k=1,\cdots,N\\ & \mu_i\geq 0,\quad i=1\cdots,M\\ & \boldsymbol{\mu}^\top\mathbf{r}=c\end{aligned}$$

Isolating the set of terms that explicitly depend on $\boldsymbol{\mu}$

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & 2\boldsymbol{\alpha}^\top\mathbf{1}+\min_{\boldsymbol{\mu}}\left[-\boldsymbol{\alpha}^\top\left(\mathbf{Y}\circ\sum_{i=1}^{M}\mu_i\mathbf{k}_i\right)\boldsymbol{\alpha}\right]\\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0\\ & 0\leq\alpha_k\leq C,\quad k=1,\cdots,N\\ & \mu_i\geq 0,\quad i=1\cdots,M\\ & \boldsymbol{\mu}^\top\mathbf{r}=c\end{aligned}$$

$\min_{\boldsymbol{\mu}}$ to $\max_{\boldsymbol{\mu}}$ and sum rearrangement

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & 2\boldsymbol{\alpha}^\top\mathbf{1}-\max_{\boldsymbol{\mu}}\left[\sum_{i=1}^{M}\mu_i\boldsymbol{\alpha}^\top\left(\mathbf{Y}\circ\mathbf{k}_i\right)\boldsymbol{\alpha}\right]\\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0\\ & 0\leq\alpha_k\leq C,\quad k=1,\cdots,N\\ & \mu_i\geq 0,\quad i=1\cdots,M\\ & \boldsymbol{\mu}^\top\mathbf{r}=c\end{aligned}\tag{2.58}$$

Substituting the last two constraints from (2.58) in μ_i [34] and modifying the maximization variable for an index as the problem does not explicitly depend on $\boldsymbol{\mu}$ [35]

[34] $\mu_i=\frac{c}{r_i}$ ensures non-negativity for all $c\geq 0$

[35] Notice that this format is similar to the one obtained through the Lagrangian construction, where a set of dual variables is chosen to limit the set of the problem's constraints $\min_x f(x)$ s.t $g_i(x)\leq 0\rightarrow\min_x f(x)+\max_u u_i g_i(x)$

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & 2\boldsymbol{\alpha}^\top \mathbf{1} - \max_i \left[\frac{c}{r_i} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_i \right) \boldsymbol{\alpha} \right] \\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C, \quad k = 1, \cdots, N \end{aligned} \tag{2.59}$$

Because $\boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ k_i \right) \boldsymbol{\alpha}$ is convex, we can introduce an epigraph variable (see 2.1.4) to transform the maximization within the problem's functional into a constraint.

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & 2\boldsymbol{\alpha}^\top \mathbf{1} - ct \\ \text{s.t.} \quad & t \geq \frac{1}{r_i} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_i \right) \boldsymbol{\alpha}, \quad i = 1, \cdots, M \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C, \quad k = 1, \cdots, N \end{aligned} \tag{2.60}$$

In order to understand the aforementioned transformation, the subsequent explanation will be presented.

Consider the extended list of constraints presented in (2.60)

$$\begin{bmatrix} & \\ & \frac{1}{r_1} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_1 \right) \boldsymbol{\alpha}, \\ & \frac{1}{r_2} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_2 \right) \boldsymbol{\alpha}, \\ & \vdots \\ & \frac{1}{r_N} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_N \right) \boldsymbol{\alpha}, \\ & \end{bmatrix} \tag{2.61}$$

A set of epigraphs can be computed for the i-th term

$$\begin{aligned} t_1 &\geq \frac{1}{r_1} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_1 \right) \boldsymbol{\alpha} \\ t_2 &\geq \frac{1}{r_2} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_2 \right) \boldsymbol{\alpha} \\ &\vdots \\ t_n &\geq \frac{1}{r_N} \boldsymbol{\alpha}^\top \left(\mathbf{Y} \circ \mathbf{k}_N \right) \boldsymbol{\alpha} \end{aligned}$$

By identifying a value t that bounds all t_i from above, we can find the maximum epigraph, which corresponds to the maximum of (2.61).

The worst-case complexity of this problem is around $O(mn^3)$, which is comparably more efficient than the equivalent problem presented in 2.3.9 which has a worst-case complexity of about $O\left((m+n)^2 n^{2.5}\right)$ [36].

[36] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

2.2.6 SVC QCQP L2 Multikernel Formulation

Considering the SVC L2 problem as a performance metric,

$$\begin{aligned}\omega(\boldsymbol{\mu}) = \max_{\boldsymbol{\alpha}} \quad & -\boldsymbol{\alpha}^\top \left[\mathbf{Y} \circ \left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i + \frac{1}{C}\mathbf{I}\right)\right] \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N\end{aligned}$$

The entire problem can be outlined as follows:

$$\begin{aligned}\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha}} \quad & -\boldsymbol{\alpha}^\top \left[\mathbf{Y} \circ \left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i + \frac{1}{C}\mathbf{I}\right)\right] \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N \\ & \mu_i \geq 0 \quad i = 1, \cdots, M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c\end{aligned}$$

Isolating $\boldsymbol{\mu}$ dependent term

Notice that $\mathbf{Y} \circ \mathbf{I} = \mathbf{I}$

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{C}\boldsymbol{\alpha}^\top \boldsymbol{\alpha} - c \max_i \left[\boldsymbol{\alpha}^\top \left(\frac{1}{r_i}\mathbf{Y} \circ \mathbf{k}_i\right) \boldsymbol{\alpha}\right] \\ \text{s.t.} \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N\end{aligned} \tag{2.62}$$

And reformulating (2.63) using epigraph

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top \mathbf{1} - \frac{1}{C}\boldsymbol{\alpha}^\top \boldsymbol{\alpha} - ct \\ \text{s.t.} \quad & t \geq \frac{1}{r_i}\boldsymbol{\alpha}^\top (\mathbf{Y} \circ \mathbf{k}_i)\, \boldsymbol{\alpha}, \quad i = 1, \cdots, M \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N\end{aligned} \tag{2.63}$$

2.2.7 SVR QCQP L1 Multikernel Formulation

Following the approach in 2.2.5, we will derive the Quadratically Constrained Quadratic Program within the SVR framework. Let us define the following performance metric:

$$\begin{aligned}\Omega(\mathbf{K}) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \le \alpha_k, \alpha_k^* \le C, \quad k = 1, \cdots, N\end{aligned}$$

The resulting nested problem appears as follows:

$$\min_{\boldsymbol{\mu}} \Omega(\mathbf{K}) \; ; \text{s.t } \mu_i \ge 0 \; \forall i, \boldsymbol{\mu}^\top \mathbf{r} = c \tag{2.64}$$

Where $c \ge 0$. (2.64) in extended format

$$\begin{aligned}\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \le \alpha_k, \alpha_k^* \le C, \quad k = 1, \cdots, N \\ & \mu_i \ge 0, \quad i = 1, \cdots, M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c\end{aligned}$$

Since the inner problem is convex (concave) with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$, and the outer problem is convex (linear) with respect to $\boldsymbol{\mu}$, strong duality holds, and it's possible to switch the order of the problems,

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \min_{\boldsymbol{\mu}} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \le \alpha_k, \alpha_k^* \le C, \quad k = 1, \cdots, N \\ & \mu_i \ge 0, \quad i = 1, \cdots, M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c\end{aligned} \tag{2.65}$$

Isolating the set of terms that explicitly depend on $\boldsymbol{\mu}$

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & + \min_{\boldsymbol{\mu}} \left[-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \right] \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \le \alpha_k, \alpha_k^* \le C, \quad k = 1, \cdots, N \\ & \mu_i \ge 0, \quad i = 1, \cdots, M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c\end{aligned}$$

$\min_{\mu} \rightarrow \max_{\mu}$ and sum rearrangement

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -\max_{\boldsymbol{\mu}} \left[\sum_{i=1}^{M} \mu_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\right] \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k=1,\cdots,N \\ & \mu_i \geq 0, \quad i=1,\cdots,M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c \end{aligned} \tag{2.66}$$

Substituting the last two constraints from (2.66) in μ_i and modifying the maximization variable for an index as the problem does not explicitly depend on $\boldsymbol{\mu}$

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -\max_{i} \left[\frac{c}{r_i}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\right] \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k=1,\cdots,N \end{aligned} \tag{2.67}$$

Because $(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top k_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)$ is convex, we can introduce an epigraph variable to transform the maximization within the problem's functional into a constraint.

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-ct \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & t \geq \frac{1}{r_i}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*), \quad i=1,\cdots,M \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k=1,\cdots,N \end{aligned} \tag{2.68}$$

The transition between (2.67) and (2.68) is explained in detail in section 2.3.9.

2.2.8 *SVR QCQP L2 Multikernel formulation*

Considering the SVR L1 problem as a performance metric

$$\begin{aligned}\Omega(K) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i + \mathbf{I}\frac{1}{C}\right)(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N\end{aligned}$$

It's possible to obtain the complete problem

$$\begin{aligned}\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i + \mathbf{I}\frac{1}{C}\right)(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N \\ & \mu_i \geq 0, \quad i = 1, \cdots, M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c\end{aligned}$$

Separating μ dependent terms

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - \frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -\max_{\boldsymbol{\mu}} \left[\sum_{i=1}^{M} \mu_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\right] \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N \\ & \mu_i \geq 0, \quad i = 1, \cdots, M \\ & \boldsymbol{\mu}^\top \mathbf{r} = c\end{aligned} \tag{2.69}$$

Substituting $\mu_i = \frac{c}{r_i}$ and $\max_\mu \rightarrow \max_i$

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - \frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -\max_{i} \left[\frac{1}{r_i}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\right] \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N\end{aligned}$$

using the epigraph approach to reformulate (2.69)

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-\frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-ct \\ \text{s.t.} \quad & t \geq \frac{1}{r_i}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \quad i=1,\cdots,M \\ & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & \alpha_k,\alpha_k^* \geq 0, \quad k=1,\cdots,N\end{aligned} \tag{2.70}$$

2.3 SVM SDP formulations

This module provides the necessary background to derive Multikernel Semidefinite Programming (SDP) Support Vector Machines, offering a different approximation than that discussed previously. The formulations in this section have disadvantages in memory usage and convergence time compared to those in the previous module. However, they can be used as a single kernel in the same manner as (2.43) and (2.57), and as multikernels like (2.68) and (2.70). This closes the gap for the formulation verification process[37] during the transition from single kernel SVR to multikernel SVR. This foundation aids in introducing innovative formulation variants detailed in chapter 3.

[37] This process includes checking the number of support vectors, their values, and corresponding predictions in one-to-one experiments

2.3.1 SVC SDP L1 formulation

Based on the problem obtained in section 2.2.1, we will derive new formulations based on the SDP theory (see 2.1.8).

Considering,

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top(\mathbf{Y}\circ\mathbf{K})\boldsymbol{\alpha}-\boldsymbol{\alpha}^\top\mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0 \\ & 0\leq\alpha_k\leq C \quad k=1,\cdots,N\end{aligned}$$

Using $\mathbf{G}=\mathbf{Y}\circ\mathbf{K}$, and multiplying the functional by 2, which does not alter the solution, we get:

$$\begin{aligned}\min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top\mathbf{G}\boldsymbol{\alpha}-2\boldsymbol{\alpha}^\top\mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0 \\ & 0\leq\alpha_k\leq C \quad k=1,\cdots,N\end{aligned} \tag{2.71}$$

We can introduce a slack variable t that provides an upper bound for the functional (see 2.1.4)

$$t \geq \boldsymbol{\alpha}^\top\mathbf{G}\boldsymbol{\alpha}-2\boldsymbol{\alpha}^\top\mathbf{1} \tag{2.72}$$

Inserting (2.72) into (2.71):

$$\begin{aligned} \min_{\boldsymbol{\alpha},t} \quad & t \\ s.t. \quad & t \geq \boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{1} \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned}$$

Rearranging the first constraint we get,

$$\begin{aligned} \min_{\boldsymbol{\alpha},t} \quad & t \\ s.t. \quad & t - \boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{1} \geq 0 \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned} \tag{2.73}$$

SDP formulation

Let's define the following decomposition for $\mathbf{G}$:

$$\mathbf{G} = \mathbf{P}\mathbf{P}^\top \tag{2.74}$$

While this derivation will focus on the Cholesky decomposition, it's important to note that other $\mathbf{P}\mathbf{P}^\top$ decompositions can also be used.

Inserting (2.74) in (2.73),

$$\begin{aligned} \min_{\boldsymbol{\alpha},t} \quad & t \\ s.t. \quad & t - \boldsymbol{\alpha}^\top \mathbf{P}\mathbf{P}^\top \boldsymbol{\alpha} + 2\boldsymbol{\alpha}^\top \mathbf{1} \geq 0 \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned} \tag{2.75}$$

Taking the Schur Complement Lemma 2.1.9 into consideration, we could visualize the first constraint in (2.75) as outlined,

$$\underbrace{t + 2\boldsymbol{\alpha}^\top \mathbf{1}}_{D} - \underbrace{\boldsymbol{\alpha}^\top \mathbf{P}}_{C} \underbrace{\mathbf{I}^{-1}}_{A^{-1}} \underbrace{\mathbf{P}^\top \boldsymbol{\alpha}}_{B}$$

Which clearly shows that this constraint could be converted to the matricial format (see (2.2)). Reformulating (2.75),

$$\begin{aligned} \min_{\boldsymbol{\alpha},t} \quad & t \\ s.t. \quad & \begin{pmatrix} \mathbf{I} & \mathbf{P}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{P} & t + 2\boldsymbol{\alpha}^\top \mathbf{1} \end{pmatrix} \succeq 0 \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned}$$

As $I^{-1} = I$. It can clearly be seen that the decomposition expressed in (2.74) was used to simplify the computation of the inverse.

The decision function is computed in the same way as expressed in section 2.2.1.

2.3.2 SVC SDP L2 formulation

In section 2.2.2 the problem below was obtained:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top \left[\mathbf{Y} \circ \left(\mathbf{K} + \mathbf{I}\frac{1}{c}\right)\right] \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N \end{aligned}$$

Using $\mathbf{P}\mathbf{P}^\top = \left[\mathbf{Y} \circ \left(\mathbf{K} + \mathbf{I}\frac{1}{c}\right)\right]$ and multiplying the functional by 2:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top \mathbf{P}\mathbf{P}^\top \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N \end{aligned}$$

Following the processes outlined in section 2.3.1, the subsequent problem can be derived:

$$\begin{aligned} \min_{\boldsymbol{\alpha}, t} \quad & t \\ s.t. \quad & \begin{pmatrix} \mathbf{I} & \mathbf{P}^\top \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top \mathbf{P} & t + 2\boldsymbol{\alpha}^\top \mathbf{1} \end{pmatrix} \succeq 0 \\ & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N \end{aligned}$$

2.3.3 SVR SDP L1 formulation

Following the derivation of Semidefinite Programs for SVC formulations (as seen in section 2.3.1 and 2.3.2), the formulations for the SVR problems will be derived in the subsequent two sections.

In section 3.1.6 we derived the SVR L1 dual problem,

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \quad -(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) - 2\epsilon(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$$

$$\begin{aligned} \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \end{aligned}$$

That can be cast as a minimization problem,

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) - 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \end{aligned} \tag{2.76}$$

From here, it's possible to obtain the epigraph formulation of (2.76),

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,t} \quad & t \\ \text{s.t.} \quad & t \geq (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) - 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \end{aligned}$$

Reformulating the first constraint and applying the Cholesky decomposition on $\mathbf{K}$,

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,t} \quad & t \\ \text{s.t.} \quad & t + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{P}\mathbf{P}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) \geq 0 \\ & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \end{aligned} \tag{2.77}$$

It's possible to visualize the reformulated constraint as the Schur complement lemma; and therefore, (2.78) can be rewritten as follows:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,t} \quad & t \\ \text{s.t.} \quad & \begin{pmatrix} \mathbf{I} & \mathbf{P}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{P} & t + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \end{pmatrix} \succeq 0 \\ & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \end{aligned} \tag{2.78}$$

2.3.4 *SVR SDP L2 formulation*

The SVR L2 dual formulation, obtained in section 2.3.2, is expressed below:

$$\begin{aligned}\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \left(\mathbf{K}+\mathbf{I}\frac{1}{C}\right)(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & \alpha_k,\alpha_k^*\geq 0, \quad k=1,\cdots,N\end{aligned}$$

Recasting as a minimization problem,

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \left(\mathbf{K}+\mathbf{I}\frac{1}{C}\right)(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & +2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)-2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & \alpha_k,\alpha_k^*\geq 0, \quad k=1,\cdots,N\end{aligned} \tag{2.79}$$

Using $\mathbf{P}\mathbf{P}^\top = \left(\mathbf{K}+\mathbf{I}\frac{1}{C}\right)$, and formatting (2.79) as an epigraph formulation,

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,t} \quad & t \\ \text{s.t.} \quad & t+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{P}\mathbf{P}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)\geq 0 \\ & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & \alpha_k,\alpha_k^*\geq 0, \quad k=1,\cdots,N\end{aligned} \tag{2.80}$$

And then (2.80) as an SDP format,

$$\begin{aligned}\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,t} \quad & t \\ \text{s.t.} \quad & \begin{pmatrix}\mathbf{I} & \mathbf{P}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{P} & t+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\end{pmatrix}\succeq 0 \\ & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\ & \alpha_k,\alpha_k^*\geq 0, \quad k=1,\cdots,N\end{aligned}$$

2.3.5 *SVC L1 Dual SDP Formulation*

The dual formulation for the SVC problem was obtained in section 2.2.1 and 2.2.2. In this and the following section, we will compute a dual formulation of the previously computed dual problems, aiming at the construction of the SDP formulation.

Recalling (2.25),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top (\mathbf{Y} \circ \mathbf{K}) \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \cdots, N \end{aligned} \tag{2.81}$$

The Lagrangian from (2.81),

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}; \boldsymbol{\eta}, \boldsymbol{\delta}, \lambda) = & \boldsymbol{\alpha}^\top (\mathbf{Y} \circ \mathbf{K}) \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{1} - 2\lambda(\boldsymbol{\alpha}^\top \mathbf{y}) \\ & - 2\boldsymbol{\eta}^\top (\mathbf{1}C - \boldsymbol{\alpha}) - 2\boldsymbol{\delta}^\top \boldsymbol{\alpha} \end{aligned} \tag{2.82}$$

Where $\boldsymbol{\delta} \in \mathbb{R}^n$, $\boldsymbol{\eta} \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ correspond to the set of dual variables.

KKT1

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}} = & 2 (\mathbf{Y} \circ \mathbf{K}) \boldsymbol{\alpha} - 2\mathbf{1} - 2\lambda \mathbf{y} + 2\boldsymbol{\eta} - 2\boldsymbol{\delta} = 0 \\ & 2 (\mathbf{Y} \circ \mathbf{K}) \boldsymbol{\alpha} = 2\mathbf{1} + 2\lambda \mathbf{y} - 2\boldsymbol{\eta} + 2\boldsymbol{\delta} \\ & \boldsymbol{\alpha} = (\mathbf{Y} \circ \mathbf{K})^{-1} [\mathbf{1} + \lambda \mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}] \end{aligned} \tag{2.83}$$

KKT2

$$\begin{aligned} \alpha_k &\geq 0 \\ C - \alpha_k &\geq 0 \end{aligned}$$

KKT3

$$\begin{aligned} \eta_k &\geq 0 \\ \delta_k &\geq 0 \end{aligned}$$

KKT4

$$\begin{aligned} \eta_k \alpha_k &= 0 \\ (C - \alpha_k)\delta_k &= 0 \end{aligned}$$

For this particular case, we will not delve into the KKT analysis since it does not result in a clear relationship between the original problem and the values in the variables $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$.

Simplifying (2.82),

$$\mathcal{L}(\boldsymbol{\alpha}; \boldsymbol{\eta}, \boldsymbol{\delta}, \lambda) = \boldsymbol{\alpha}^\top (\mathbf{Y} \circ \mathbf{K}) \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top (\mathbf{1} + \lambda \mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}) - 2C\boldsymbol{\eta}^\top \mathbf{1} \tag{2.84}$$

Substituting (2.83) into (2.84),

$$\begin{aligned}\mathcal{D} =&\{(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\}^{\top}(\mathbf{Y}\circ\mathbf{K})\{(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}-\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\}\\ &-2\{(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\}^{\top}(\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta})\\ &-2C\boldsymbol{\eta}^{\top}\mathbf{1}\end{aligned} \tag{2.85}$$

From (2.85) it's possible to see that,

$$\{(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\}^{\top}=[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}$$

Since $(\mathbf{Y}\circ\mathbf{K})^{-1\top} = (\mathbf{Y}\circ\mathbf{K})^{-1}$ due to its symmetric property. Subsequently we can simplify (2.85) as above,

$$\begin{aligned}\mathcal{D}(\boldsymbol{\eta},\boldsymbol{\delta},\lambda;\boldsymbol{\alpha}) =&[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\\ &-2[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\\ &-2C\boldsymbol{\eta}^{\top}\mathbf{1}\end{aligned}$$

Since $(\mathbf{Y}\circ\mathbf{K})(\mathbf{Y}\circ\mathbf{K})^{-1}=\mathbf{I}$. From here it's possible to see that the second term is negative two times the first term; and therefore we get,

$$\begin{aligned}\mathcal{D}(\boldsymbol{\eta},\boldsymbol{\delta},\lambda;\boldsymbol{\alpha}) =&-[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]\\ &-2C\boldsymbol{\eta}^{\top}\mathbf{1}\end{aligned}$$

This is the unconstrained Wolfe dual of (2.81). The new problem is then formatted as follows:

$$\begin{aligned}\max_{\boldsymbol{\eta},\boldsymbol{\delta},\lambda}\quad&-[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]-2C\boldsymbol{\eta}^{\top}\mathbf{1}\\ s.t.\quad&\eta_k,\delta_k\geq 0,\quad k=1,\cdots,N\end{aligned} \tag{2.86}$$

Formatting (2.86) as a minimization problem,

$$\begin{aligned}\min_{\boldsymbol{\eta},\boldsymbol{\delta},\lambda}\quad&[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]+2C\boldsymbol{\eta}^{\top}\mathbf{1}\\ s.t.\quad&\eta_k,\delta_k\geq 0,\quad k=1,\cdots,N\end{aligned} \tag{2.87}$$

THE EPIGRAPH FORMULATION of (2.87) is written as follows:

$$\begin{aligned}\min_{\boldsymbol{\eta},\boldsymbol{\delta},\lambda,t}\quad&t\\ s.t.\quad&[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]^{\top}(\mathbf{Y}\circ\mathbf{K})^{-1}[\mathbf{1}+\lambda\mathbf{y}-\boldsymbol{\eta}+\boldsymbol{\delta}]+2C\boldsymbol{\eta}^{\top}\mathbf{1}\leq t\\ &\eta_k,\delta_k\geq 0,\quad k=1,\cdots,N\end{aligned} \tag{2.88}$$

Reformulating the first constraint in (2.89) it's possible to observe the Schur complement lemma:

$$\underbrace{t - 2C\boldsymbol{\eta}^\top \mathbf{1}}_{D} - \underbrace{[\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]^\top}_{B} \underbrace{(\mathbf{Y} \circ \mathbf{K})^{-1}}_{A^{-1}} \underbrace{[\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]}_{C} \geq 0$$

The modified equation is written as below:

$$\begin{aligned} \min_{\boldsymbol{\eta},\delta,\lambda,t} \quad & t \\ s.t. \quad & \begin{bmatrix} (\mathbf{Y} \circ \mathbf{K}) & [\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]^\top \\ [\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}] & t - 2C\boldsymbol{\eta}^\top \mathbf{1} \end{bmatrix} \succeq 0 \\ & \eta_k, \delta_k \geq 0, \quad k = 1, \cdots, N \end{aligned} \tag{2.89}$$

Notice that the use of Schur complement lemma aids to formulate the constraint as an LMI. The previous constraint in (2.89) can be visualized as follows:

$$\begin{bmatrix} (\mathbf{Y} \circ \mathbf{K}) & [\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]^\top \\ [\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}] & t - 2C\boldsymbol{\eta}^\top \mathbf{1} \end{bmatrix} = \begin{bmatrix} (\mathbf{Y} \circ \mathbf{K}) & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} +$$

$$\begin{bmatrix} \mathbf{0}^{n\times n} & [\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]^\top \\ \mathbf{0}^\top & 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0}^{n\times n} & \mathbf{0} \\ [\mathbf{1} + \lambda\mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]^\top & 0 \end{bmatrix} + \tag{2.90}$$

$$\begin{bmatrix} \mathbf{0}^{n\times n} & \mathbf{0} \\ \mathbf{0} & t - 2C\boldsymbol{\eta}^\top \mathbf{1} \end{bmatrix}$$

Where each term of the matrix is linear and must be convex after the learning process . This property is crucial for the multikernel SVC formulation, explained in 2.3.9.

Notice that it's possible to break down (2.90) further, by using a matrix for every term of the $\mathbf{Y} \circ \mathbf{K}$ matrix. More information on LMI can be found in 2.1.8

2.3.6 SVC L2 Dual SDP Formulation

Obtaining (2.37),

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top \left[\mathbf{Y} \circ \left(\mathbf{K} + \mathbf{I}\frac{1}{c}\right)\right] \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N \end{aligned}$$

For visual simplification, let's define $\mathbf{G} = \left[\mathbf{Y} \circ \left(\mathbf{K} + \mathbf{I}\frac{1}{c}\right)\right]$ and multiply the functional by 2:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & \alpha_k \geq 0 \quad k = 1, \cdots, N \end{aligned}$$

Lagrangian

$$\mathcal{L}(\boldsymbol{\alpha}; \boldsymbol{\eta}, \lambda) = \boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \mathbf{1} - 2\lambda(\boldsymbol{\alpha}^\top \mathbf{y}) - 2\boldsymbol{\eta}^\top \boldsymbol{\alpha}$$

KKT1

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}} =& 2\mathbf{G}\boldsymbol{\alpha} - 2\mathbf{1} - 2\lambda\mathbf{y} - 2\boldsymbol{\eta} = 0 \\ & 2\mathbf{G}\boldsymbol{\alpha} = 2\mathbf{1} + 2\lambda\mathbf{y} + 2\boldsymbol{\eta} \\ & \boldsymbol{\alpha} = \mathbf{G}^{-1}\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right] \end{aligned} \tag{2.91}$$

KKT2

$$\alpha_k \geq 0$$

KKT3

$$\eta_k \geq 0$$

KKT4

$$\eta_k \alpha_k = 0$$

Rearranging the Lagrangian to simplify the substitution process,

$$\mathcal{L}(\boldsymbol{\alpha}; \boldsymbol{\eta}, \lambda) = \boldsymbol{\alpha}^\top \mathbf{G}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \left(\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right)$$

Substituting (2.91) in the previous Lagrangian:

$$\begin{aligned} \mathcal{D}(\boldsymbol{\eta}, \lambda; \boldsymbol{\alpha}) = & \left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right]^\top \mathbf{G}^{-1}\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right] \\ & - 2\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right]^\top \mathbf{G}^{-1}\left[\mathbf{1} + \lambda\mathbf{y} + 2\boldsymbol{\eta}\right] \end{aligned}$$

Simplifying,

$$\mathcal{D}(\boldsymbol{\eta}, \lambda; \boldsymbol{\alpha}) = -\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right]^\top \mathbf{G}^{-1}\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right]$$

Setting up the problem,

$$\begin{aligned} \max_{\boldsymbol{\eta}, \lambda} \quad & -\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right]^\top \mathbf{G}^{-1}\left[\mathbf{1} + \lambda\mathbf{y} + \boldsymbol{\eta}\right] \\ s.t. \quad & \eta_k \geq 0, \quad k = 1, \cdots, N \end{aligned}$$

The previous problem as a minimization problem:

$$\begin{aligned} \min_{\boldsymbol{\eta},\lambda} \quad & [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}]^\top \mathbf{G}^{-1} [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}] \\ s.t. \quad & \eta_k \geq 0, \quad k = 1, \cdots, N \end{aligned}$$

Similar to the L1 formulation, we can also derive its epigraph form:

$$\begin{aligned} \min_{\boldsymbol{\eta},\lambda} \quad & t \\ s.t. \quad & t - [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}]^\top \mathbf{G}^{-1} [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}] \geq 0 \\ & \eta_k \geq 0, \quad k = 1, \cdots, N \end{aligned}$$

And the SDP formulation:

$$\begin{aligned} \min_{\boldsymbol{\eta},\lambda} \quad & t \\ s.t. \quad & \begin{bmatrix} \mathbf{G} & [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}]^\top \\ [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}] & t \end{bmatrix} \succeq 0 \\ & \eta_k \geq 0, \quad k = 1, \cdots, N \end{aligned}$$

2.3.7 *SVR L1 Dual SDP Formulation*

This section will derive the SVR SDP formulation specifically for a single kernel.

Considering the SVR L1 dual formulation (2.43),

$$\begin{aligned} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\epsilon\mathbf{1}(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \end{aligned} \tag{2.92}$$

The Lagrangian of the problem has the following format:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha};\lambda,\boldsymbol{\eta},\boldsymbol{\eta}^*,\boldsymbol{\delta},\boldsymbol{\delta}^*) = & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\epsilon\mathbf{1}(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) \\ & + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2\lambda\mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2\boldsymbol{\eta}^\top\boldsymbol{\alpha} \\ & + 2\boldsymbol{\eta}^{*\top}\boldsymbol{\alpha}^* + 2\boldsymbol{\delta}^\top(\mathbf{1}C-\boldsymbol{\alpha}) \\ & + 2\boldsymbol{\delta}^{*\top}(\mathbf{1}C-\boldsymbol{\alpha}^*) \end{aligned} \tag{2.93}$$

KKT1

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\alpha}} = -2\mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\epsilon\mathbf{1} + 2\mathbf{y} + 2\boldsymbol{\eta} - 2\boldsymbol{\delta} + 2\lambda\mathbf{1} = 0 \\ (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = \mathbf{K}^{-1}(-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) \end{aligned} \tag{2.94}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^*} &= 2\mathbf{K}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) - 2\epsilon\mathbf{1} + 2\mathbf{y} + 2\boldsymbol{\eta}^* - 2\boldsymbol{\delta}^* - 2\lambda\mathbf{1} = 0 \\ (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) &= \mathbf{K}^{-1}(\epsilon\mathbf{1} + \mathbf{y} - \boldsymbol{\eta}^* + \boldsymbol{\delta}^* + \lambda\mathbf{1})\end{aligned} \tag{2.95}$$

KKT2

$$\begin{aligned}\alpha_k &\geq 0 \\ C - \alpha_k &\geq 0 \\ \alpha_k^* &\geq 0 \\ C - \alpha_k^* &\geq 0\end{aligned}$$

KKT3

$$\begin{aligned}\eta_k &\geq 0 \\ \delta_k &\geq 0 \\ \eta_k^* &\geq 0 \\ \delta_k^* &\geq 0\end{aligned}$$

KKT4

$$\begin{aligned}\eta_k\alpha_k &= 0 \\ \delta_k(C - \alpha_k) &= 0 \\ \eta_k^*\alpha_k^* &= 0 \\ \delta_k^*(C - \alpha_k) &= 0\end{aligned}$$

To simplify the dual Lagrangian formulation, we will use the following complementary equality from the computation of (2.95) - (2.94):

$$\begin{aligned}&\mathbf{K}^{-1}(\epsilon\mathbf{1} + \mathbf{y} - \boldsymbol{\eta}^* + \boldsymbol{\delta}^* + \lambda\mathbf{1}) = \mathbf{K}^{-1}(-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) \\ &2\epsilon\mathbf{1} - \boldsymbol{\eta}^* - \boldsymbol{\eta} + \boldsymbol{\delta}^* + \boldsymbol{\delta} = 0 \\ &(\boldsymbol{\eta} - \boldsymbol{\delta}) + (\boldsymbol{\eta}^* - \boldsymbol{\delta}^*) = 2\epsilon\mathbf{1} \\ &\boxed{(\boldsymbol{\eta}^* - \boldsymbol{\delta}^*) = -(\boldsymbol{\eta} - \boldsymbol{\delta}) + 2\epsilon\mathbf{1}} \\ &\boldsymbol{\delta}^* - (\boldsymbol{\eta} - \boldsymbol{\delta}) + 2\epsilon\mathbf{1} = \boldsymbol{\eta}^* \\ &\boldsymbol{\delta}^* - (\boldsymbol{\eta} - \boldsymbol{\delta}) + 2\epsilon\mathbf{1} \geq 0 \\ &\boxed{\boldsymbol{\eta} - (\boldsymbol{\delta}^* + \boldsymbol{\delta}) \leq 2\epsilon\mathbf{1}}\end{aligned} \tag{2.96}$$

Simplifying the Lagrangian in (2.93):

$$\begin{aligned}\mathcal{L} = &-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\mathbf{1}^\top\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)\\ &+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+2(\boldsymbol{\eta}-\boldsymbol{\delta})^\top\boldsymbol{\alpha}+2(\boldsymbol{\eta}^*-\boldsymbol{\delta}^*)^\top\boldsymbol{\alpha}^*\\ &+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}+2\lambda(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{1}\end{aligned} \tag{2.97}$$

Substituting the resulting equation from (2.96) into (2.97):

$$\begin{aligned}\mathcal{L} = &-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)\\ &+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+2(\boldsymbol{\eta}-\boldsymbol{\delta})^\top\boldsymbol{\alpha}+2\left[2\epsilon\mathbf{1}-(\boldsymbol{\eta}-\boldsymbol{\delta})\right]^\top\boldsymbol{\alpha}^*\\ &+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}+2\lambda(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{1}\end{aligned}$$

Expanding the second and fifth terms:

$$\begin{aligned}\mathcal{L} = &-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - 2\epsilon\mathbf{1}^\top\boldsymbol{\alpha} - 2\epsilon\mathbf{1}^\top\boldsymbol{\alpha}^*\\ &+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+2(\boldsymbol{\eta}-\boldsymbol{\delta})^\top\boldsymbol{\alpha}+4\epsilon\mathbf{1}^\top\boldsymbol{\alpha}-2(\boldsymbol{\eta}-\boldsymbol{\delta})^\top\boldsymbol{\alpha}^*\\ &+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}+2\lambda(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{1}\end{aligned} \tag{2.98}$$

Simplifying (2.98):

$$\begin{aligned}\mathcal{L} = &-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\ &-2(\boldsymbol{\eta}-\boldsymbol{\delta})^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}+2\lambda(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{1}\end{aligned} \tag{2.99}$$

Rearranging (2.99):

$$\begin{aligned}\mathcal{L} = &-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{K}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + 2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}\\ &+(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\left[2\epsilon\mathbf{1}+2\mathbf{y}-2\boldsymbol{\eta}+2\boldsymbol{\delta}+2\lambda\right]\end{aligned}$$

Substituting (2.94):

$$\begin{aligned}\mathcal{D} = &\left[\mathbf{K}^{-1}(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\right]^\top\mathbf{K}\left[\mathbf{K}^{-1}(-\epsilon\mathbf{1}+\mathbf{y}-\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\right]\\ &+2\left[\mathbf{K}^{-1}(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\right]^\top\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\\ &+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}\end{aligned}$$

Simplifying:

$$\begin{aligned}\mathcal{D} = &\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\mathbf{K}^{-1}\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\\ &+2\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\mathbf{K}^{-1}\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\\ &+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}\end{aligned}$$

$$\begin{aligned}\mathcal{D} = &\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\mathbf{K}^{-1}\left[-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}\right]\\ &+2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}\end{aligned}$$

Constructing the optimization problem:

$$\begin{aligned}\min_{\delta,\eta,\delta^*,\lambda} \quad & (-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\,\mathbf{K}^{-1}\,(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}) \\ & +2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\eta}-(\boldsymbol{\delta}^*-\boldsymbol{\delta})\geq 2\epsilon\mathbf{1} \\ & \boldsymbol{\delta},\boldsymbol{\delta}^*,\boldsymbol{\eta}\geq\mathbf{0}\end{aligned} \tag{2.100}$$

Where $\boldsymbol{\eta}\geq\mathbf{0}$ is equivalent to $\eta_i\geq 0,\quad i=1,\dots N$. (2.100) as a maximization problem:

$$\begin{aligned}\max_{\delta,\eta,\delta^*,\lambda} \quad & -(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\,\mathbf{K}^{-1}\,(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}) \\ & +2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\eta}-(\boldsymbol{\delta}^*-\boldsymbol{\delta})\geq 2\epsilon\mathbf{1} \\ & \boldsymbol{\delta},\boldsymbol{\delta}^*,\boldsymbol{\eta}\geq\mathbf{0}\end{aligned}$$

To compute b to predict new values, $(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)$ can be obtained from (2.94) by first filtering the set of support vectors and values as shown in 3.1.6.

Now we will introduce the SDP formulation for the Support Vector Regression model. The derivation follows a similar pattern from the one used in the SVC counterpart.

$$\begin{aligned}\min_{\eta,\delta,\delta^*,\lambda} \quad & (-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\,\mathbf{K}^{-1}\,(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}) \\ & +2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1} \\ \text{s.t.} \quad & \boldsymbol{\eta}-(\boldsymbol{\delta}^*-\boldsymbol{\delta})\geq 2\epsilon\mathbf{1} \\ & \boldsymbol{\delta},\boldsymbol{\delta}^*,\boldsymbol{\eta}\geq\mathbf{0}\end{aligned} \tag{2.101}$$

Because the function in (2.101) is convex with respect to the problem's parameters, we can introduce an epigraph variable to represent it as a constraint in the following way:

$$\begin{aligned}\min_{\eta,\delta,\delta^*,\lambda} \quad & t \\ \text{s.t.} \quad & -(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1})\,\mathbf{K}^{-1}\,(-\epsilon\mathbf{1}+\mathbf{y}+\boldsymbol{\eta}-\boldsymbol{\delta}+\lambda\mathbf{1}) \\ & -2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top\mathbf{1}+t\geq 0 \\ & \boldsymbol{\eta}-(\boldsymbol{\delta}^*-\boldsymbol{\delta})\geq 2\epsilon\mathbf{1} \\ & \boldsymbol{\delta},\boldsymbol{\delta}^*,\boldsymbol{\eta}\geq\mathbf{0}\end{aligned}$$

Pulling out the first constraint in 2.3.7 for observation,

$$\underbrace{t - 2C(\delta + \delta^*)^\top \mathbf{1}}_{D} - \underbrace{(-\epsilon\mathbf{1} + \mathbf{y} + \eta - \delta + \lambda\mathbf{1})}_{B} \underbrace{\mathbf{K}^{-1}}_{A^{-1}} \underbrace{(-\epsilon\mathbf{1} + \mathbf{y} + \eta - \delta + \lambda\mathbf{1})}_{C} \geq 0$$

The Schur complement lemma can be observed, allowing for rearrangement as follows:

$$\begin{aligned} \min_{\eta,\delta,\delta^*,\lambda} \quad & t \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{K} & (-\epsilon\mathbf{1} + \mathbf{y} + \eta - \delta + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \eta - \delta + \lambda\mathbf{1}) & t - 2C(\delta + \delta^*)^\top \mathbf{1} \end{bmatrix} \succeq 0 \\ & \eta - (\delta^* - \delta) \geq 2\epsilon\mathbf{1} \\ & \delta, \delta^*, \eta \geq \mathbf{0} \end{aligned} \tag{2.102}$$

As discussed in Section 2.3.5, the first constraint in (2.102) exhibits linearity for each of its terms, a property that allows for efficient solution using interior point methods.

2.3.8 *SVR L2 Dual SDP formulation*

Obtaining SVR L2 dual formulation (2.57) from section 3.1.7:

$$\begin{aligned} \max_{\alpha,\alpha^*} \quad & -\frac{1}{2}(\alpha - \alpha^*)^\top \left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right)(\alpha - \alpha^*) \\ & -\epsilon\mathbf{1}^\top(\alpha + \alpha^*) + \mathbf{y}^\top(\alpha - \alpha^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\alpha - \alpha^*) = 0 \\ & \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N \end{aligned} \tag{2.103}$$

Multiplying the functional from (2.103) by -2 and defining

$$\mathbf{G} = \left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right),$$

$$\begin{aligned} \min_{\alpha,\alpha^*} \quad & (\alpha - \alpha^*)^\top \mathbf{G}(\alpha - \alpha^*) - 2\epsilon\mathbf{1}^\top(\alpha + \alpha^*) + 2\mathbf{y}^\top(\alpha - \alpha^*) \\ \text{s.t.} \quad & \mathbf{1}^\top(\alpha - \alpha^*) = 0 \\ & \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, n \end{aligned}$$

Lagrangian

$$\begin{aligned} \mathcal{L}(\alpha, \alpha^*; \lambda, \eta, \eta^*) = & -(\alpha - \alpha^*)^\top \mathbf{G}(\alpha - \alpha^*) \\ & - 2\epsilon(\alpha - \alpha^*)^\top \mathbf{1} + 2\mathbf{y}^\top(\alpha - \alpha^*) \\ & + 2\lambda(\alpha - \alpha^*)^\top \mathbf{1} + 2\eta^\top \alpha + 2\eta^{*\top} \alpha^* \end{aligned}$$

KKT1

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}} &= -2\mathbf{G}\boldsymbol{\alpha} + 2\mathbf{G}\boldsymbol{\alpha}^* - 2\epsilon\mathbf{1} + 2\mathbf{y}^\top + 2\lambda\mathbf{1} + 2\boldsymbol{\eta} \\ (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) &= \mathbf{G}^{-1}(-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta})\end{aligned} \tag{2.104}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\alpha}^*} &= 2\mathbf{G}\boldsymbol{\alpha} - 2\mathbf{G}\boldsymbol{\alpha}^* - 2\epsilon\mathbf{1} + 2\mathbf{y}^\top + 2\lambda\mathbf{1} + 2\boldsymbol{\eta}^* \\ (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) &= \mathbf{G}^{-1}(\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} - \boldsymbol{\eta}^*)\end{aligned} \tag{2.105}$$

KKT2

$$\begin{aligned}\alpha_k &\geq 0 \\ \alpha_k^* &\geq 0\end{aligned}$$

KKT3

$$\begin{aligned}\eta_k &\geq 0 \\ \eta_k^* &\geq 0\end{aligned}$$

KKT4

$$\begin{aligned}\eta_k^\top \alpha_k &= 0 \\ \eta_k^{*\top} \alpha_k^* &= 0\end{aligned}$$

To simplify the dual Lagrangian formulation, we will use the following equality:

(2.104) + (2.105)

$$\begin{aligned}\mathbf{0} &= \mathbf{G}^{-1}(-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta}) + \mathbf{G}^{-1}(-\epsilon\mathbf{1} - \mathbf{y}^\top - \lambda\mathbf{1} + \boldsymbol{\eta}^*) \\ \mathbf{0} &= (-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta} - \epsilon\mathbf{1} - \mathbf{y}^\top - \lambda\mathbf{1} + \boldsymbol{\eta}^*) \\ \mathbf{0} &= -2\epsilon\mathbf{1} + \boldsymbol{\eta} + \boldsymbol{\eta}^* \\ \boxed{\boldsymbol{\eta}^* &= -\boldsymbol{\eta} + 2\epsilon\mathbf{1}}\end{aligned} \tag{2.106}$$

From (2.106) it's possible to derive the following results:

$$\begin{aligned}-\boldsymbol{\eta} + 2\epsilon\mathbf{1} &\geq \mathbf{0} \\ -\boldsymbol{\eta} &\geq -2\epsilon\mathbf{1} \\ \boldsymbol{\eta} &\leq 2\epsilon\mathbf{1}\end{aligned} \tag{2.107}$$

Substituting (2.106) into the lagrangian:

$$\begin{aligned}\mathcal{L} &= -(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \mathbf{G}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + 2\epsilon(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \mathbf{1} + 2\mathbf{y}^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\ &\quad + 2\lambda(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \mathbf{1} - 2\boldsymbol{\eta}^{*\top}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\ &= -(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \mathbf{G}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) + 2(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \left[\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} - \boldsymbol{\eta}^{*\top}\right]\end{aligned}$$

Substituting (2.104):

$$\begin{aligned}\mathcal{D} = &- \left[\mathbf{G}^{-1}(-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta})\right]^\top \mathbf{G} \left[\mathbf{G}^{-1}(-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta})\right] \\ &+ 2\left[\mathbf{G}^{-1}(-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta})\right]^\top \left[-\epsilon\mathbf{1} + \mathbf{y}^\top + \lambda\mathbf{1} + \boldsymbol{\eta}\right]\end{aligned}$$

$$\mathcal{D} = (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \mathbf{G}^{-1} (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) \tag{2.108}$$

Creating the problem from (2.108), KKT3 conditions and (2.107) and substituting the value of $\mathbf{G}$:

$$\begin{aligned}\max_{\boldsymbol{\eta},\lambda} \quad & (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right)^{-1} (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\eta} \leq 2\epsilon\end{aligned} \tag{2.109}$$

(2.109) as a minimization problem:

$$\begin{aligned}\min_{\boldsymbol{\eta},\lambda} \quad & -(-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right)^{-1} (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\eta} \leq 2\epsilon\end{aligned}$$

Once again, to compute b to predict new values, $(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$ can be obtained from (2.104). The support values and vector can be filtered as demonstrated in 3.1.7.

Computing the epigraph format,

$$\begin{aligned}\min_{\boldsymbol{\eta},\lambda} \quad & t \\ & t + (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right)^{-1} (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) \geq 0 \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\eta} \leq 2\epsilon\end{aligned}$$

And using the Schur complement lemma to obtain the SDP format:

$$\begin{aligned}\min_{\boldsymbol{\eta},\lambda} \quad & t \\ & \begin{bmatrix} -\left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right) & (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) & t \end{bmatrix} \succeq 0 \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\eta} \leq 2\epsilon\end{aligned} \tag{2.110}$$

The fact that this is a reformulation of (2.57) enables the prediction of new values using the outcomes detailed in 2.2.4.

From a computational standpoint, L2 formulation (2.110) offers an advantage by training $n + 1$ parameters, whereas the L1 formulation (2.102) requires $3n + 1$ parameters.

2.3.9 SVC SDP L1 Multikernel formulation

In this section, we will briefly illustrate the procedure described in [38], a variation of the QCQP SVC multikernel, defined in section 2.2.5.

[38] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

The SDP formulation, with constraints expressed as a set of Linear Matrix Inequalities (LMIs), enables the incorporation of a multikernel, as each term within the corresponding LMI is linear and convex. (See (2.90)).

Let us define

$$\mathbf{K} = \sum_{i=1}^{M} \mu_i \mathbf{k}_i$$

and (2.25) as a performance measure:

$$\min_{\boldsymbol{\mu}} \omega(\mathbf{K}) \; ; \text{s.t } \boldsymbol{\mu} \geq \mathbf{0}, \text{trace}(\mathbf{K}) = c \tag{2.111}$$

Where $c \geq 0$. (2.111) in extended format:

$$\begin{aligned} \min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\boldsymbol{\alpha}^\top (\mathbf{Y} \circ \mathbf{K})\, \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \mathbf{1} \\ s.t. \quad & \boldsymbol{\alpha}^\top \mathbf{y} = 0 \\ & 0 \leq \alpha_k \leq C \quad k = 1, \dots N \\ & \mu_i \geq 0 \quad i = 1, \dots M \\ & \mathbf{K} = \sum_{i=1}^{M} \mu_i \mathbf{k}_i \\ & \text{trace}(\mathbf{K}) = c \end{aligned}$$

Here, the constraints $\boldsymbol{\mu} \geq \mathbf{0}$ and $\text{trace}(\mathbf{K}) = c$ are added to promote positive semidefiniteness and reduce the search space.

The hierarchical nature of the multi-level problem encourages solving the inner problem in the same way 2.3.5 was solved to obtain the SDP dual formulation [39]. The following optimization problem is obtained:

[39] Notice that when creating the Lagrangian, it is not necessary to compute the derivative with respect to $\boldsymbol{\mu}$ since it is not part of the inner problem

$$\begin{aligned} \min_{\mu,\eta,\delta,\lambda,t} \quad & t \\ s.t. \quad & \begin{bmatrix} \left(\mathbf{Y} \circ \sum_{i=1}^{N} \mu_i \mathbf{k}_i\right) & [\mathbf{1} + \lambda \mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}]^\top \\ [\mathbf{1} + \lambda \mathbf{y} - \boldsymbol{\eta} + \boldsymbol{\delta}] & t - 2C\boldsymbol{\eta}^\top \mathbf{1} \end{bmatrix} \succeq 0 \\ & \boldsymbol{\eta}, \boldsymbol{\delta}, \boldsymbol{\delta}^* \geq 0 \\ & \text{trace}(\mathbf{K}) = c \end{aligned}$$

As explained in 2.2.5, the problem derived in this section is less efficient to solve using current interior-point methods compared to the equivalent QCQP problem in 2.2.5. Therefore, it should be reduced to

a QCQP when possible. The purpose of presenting this problem is to provide the necessary background for the propositions in 3.1, which demonstrate a computational advantage over this problem.

2.3.10 SVC SDP L2 Multikernel formulation

To derive the L2 formulation, the same logic is followed:

$$\begin{aligned}
\min_{\mu}\max_{\alpha} \quad & -\frac{1}{2}\boldsymbol{\alpha}^\top\left[\mathbf{Y}\circ\left(\mathbf{K}+\frac{1}{c}\mathbf{I}\right)\right]\boldsymbol{\alpha}+\boldsymbol{\alpha}^\top\mathbf{1} \\
s.t. \quad & \boldsymbol{\alpha}^\top\mathbf{y}=0 \\
& \alpha_k \geq 0 \quad k=1,\cdots,N \\
& \mu_i \geq 0 \quad i=1,\dots M \\
& \mathbf{K}=\sum_{i=1}^{M}\mu_i\mathbf{k}_i \\
& \text{trace}(\mathbf{K})=c
\end{aligned}$$

Computing the dual with respect to $\boldsymbol{\alpha}$ and formatting as an SDP

$$\begin{aligned}
\min_{\mu,\eta,\lambda} \quad & t \\
s.t. \quad & \begin{bmatrix} \mathbf{Y}\circ\left(\sum_{i=1}^{M}\mu_i\mathbf{k}_i+\frac{1}{C}\mathbf{I}\right) & [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}]^\top \\ [\mathbf{1}+\lambda\mathbf{y}+\boldsymbol{\eta}] & t \end{bmatrix} \succeq 0 \\
& \boldsymbol{\eta},\boldsymbol{\delta}\geq 0 \\
& \boldsymbol{\mu}\geq\mathbf{0} \\
& \text{trace}(\mathbf{K})=c
\end{aligned}$$

2.3.11 SVR SDP L1 Multikernel formulation

By exploiting the linearity of the Linear Matrix Inequality constraint, the SVR SDP multikernel will be introduced following the process expressed in 2.3.9

Suppose that the performance measure Ω is given by:

$$\begin{aligned}
\Omega(\mathbf{K})=\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\sum_{i=1}^{M}\mu_i\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& -\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\
& 0\leq\boldsymbol{\alpha},\boldsymbol{\alpha}^*\leq C, \quad k=1,\cdots,N
\end{aligned}$$

Then the multi-level problem would be written as follows:

$$\min_{\boldsymbol{\mu}} \Omega(\mathbf{K}) \; ; \text{s.t } \boldsymbol{\mu} \geq \mathbf{0}, \text{trace}(\mathbf{K}) \leq \tau$$

Where $\boldsymbol{\mu} \geq 0$ is used to ensure $\mathbf{K} \succeq 0$ and $\tau \geq 0$. From here, for the inner problem, it's possible to follow the derivation described in section (2.3.7), which results in,

Compared to section 2.3.9, a relaxed formulation for the trace($\mathbf{K}$) constraint is presented to simplify computation.

$$\begin{aligned}
\min_{\boldsymbol{\mu}} \min_{\boldsymbol{\eta},\boldsymbol{\delta},\boldsymbol{\delta}^*,\lambda} \quad & t \\
\text{s.t.} \quad & \begin{bmatrix} \sum_{i=1}^{N} \mu_i \mathbf{k}_i & (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) & t - 2C(\boldsymbol{\delta} + \boldsymbol{\delta}^*)^\top \end{bmatrix} \succeq 0 \\
& \boldsymbol{\eta} - (\boldsymbol{\delta}^* - \boldsymbol{\delta}) \geq 2\epsilon\mathbf{1} \\
& \delta_k, \delta_k^*, \eta_k \geq 0 \quad k = 1, \cdots, N \\
\\
& \mu_i \geq 0 \quad i = 1, \cdots, M \\
& \text{trace}\left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i\right) \leq \tau
\end{aligned}$$

The inner and outer problems are merged into a single problem, considering that both aim to minimize,

$$\begin{aligned}
\min_{\boldsymbol{\mu},\boldsymbol{\eta},\boldsymbol{\delta},\boldsymbol{\delta}^*,\lambda} \quad & t \\
\text{s.t.} \quad & \begin{bmatrix} \sum_{i=1}^{M} \mu_i \mathbf{k}_i & (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) & t - 2C(\boldsymbol{\delta} + \boldsymbol{\delta}^*)^\top \end{bmatrix} \succeq 0 \\
& \boldsymbol{\eta} - (\boldsymbol{\delta}^* - \boldsymbol{\delta}) \geq 2\epsilon\mathbf{1} \\
& \delta_k, \delta_k^*, \eta_k \geq 0 \quad k = 1, \cdots, N \\
& \mu_i \geq 0 \quad i = 1, \cdots, M \\
& \text{trace}\left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i\right) \leq \tau
\end{aligned}$$

Where the search space is constrained by the sum of traces of the resulting kernel.

Since $(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)$ is retrieved from (2.94), the resulting matrix $\mathbf{K}$ must be invertible.

2.3.12 *SVR SDP L2 Multikernel formulation*

The method outlined above allows us to obtain the following new formulation

Considering the following performance measure ω:

$$
\begin{aligned}
\omega(\mu) = \max_{\alpha,\alpha^*} \quad & -\frac{1}{2}\left(\alpha-\alpha^*\right)^\top \left(\sum_{i=1}^{M} \mu_i k_i + I\frac{1}{C}\right)\left(\alpha-\alpha^*\right) \\
& -\epsilon \mathbf{1}^\top(\alpha+\alpha^*) + y^\top(\alpha-\alpha^*) \\
\text{s.t.} \quad & \mathbf{1}^\top(\alpha-\alpha^*) = 0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N
\end{aligned}
$$

It follows that,

$$
\begin{aligned}
\min_{\mu,\eta,\lambda} \quad & t \\
& \begin{bmatrix} -\left(K + I\frac{1}{C}\right) & \left(-\epsilon\mathbf{1} + y + \lambda\mathbf{1} + \eta\right)^\top \\ \left(-\epsilon\mathbf{1} + y + \lambda\mathbf{1} + \eta\right) & t \end{bmatrix} \succeq 0 \\
\text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\eta} \leq 2\epsilon\mathbf{1} \\
& \mu_i \geq 0, \quad i = 1, \cdots, M \\
& \text{trace}\left(\sum_{i=1}^{M} \mu_i k_i\right) \leq \tau
\end{aligned}
$$

3 Main Results

Contents

3.1 SDP formulations

As discussed in section 2.3, although SDP formulations have computational disadvantages compared to QCQP, they are essential for proving transitional properties between single and multikernel methods.

This module presents the theoretical findings related to Multikernel SDP problems, including variants of the formulations discussed in Sections 2.3.11 and 2.3.12. These formulations aim to enhance computational stability, in terms of feasibility, by regularizing the μ_i terms without relying on the kernel set (3.1.1, 3.1.2) and by adjusting the problem to regularize the μ term rather than imposing it as a constraint (3.1.3, 3.1.4, 3.1.5). Additionally, we'll introduce an alternative approach to the filtration process for selecting support vectors (3.1.6, 3.1.7). It will be demonstrated in the next chapter that this method has numerical

precision advantages, as it avoids depending on the inverse of the derived kernel, which may not exist for real data.

3.1.1 SVR L1 SDP Multikernel explicit μ constraint

Section 2.3.11 presents a problem in which the search space is constrained by the sum of the traces of the resulting kernels. In this case, this sum depends on the kernel set election [1] and the number of data points in the problem [2], which can be numerically fluctuant and not easy to visualize as a first approximation.

[1] For example, for a non-scaled linear kernel the values ranges between 1 and -1, while in a polynomial kernel, the values are influenced by polynomial's degree and can become substantially large.

[2] The trace increases with respect to the number of values.

Motivated by the previously expressed concern, we will swap the trace constraint for a minimization of the sum of μ as follows :

$$\min_{\boldsymbol{\mu}} \Omega(\mathbf{K}) \text{ ; s.t, } \boldsymbol{\mu} \geq \mathbf{0}, \boldsymbol{\mu}^\top \mathbf{1} \leq \tau$$

The idea is to get an L1 penalization term for μ, which will be crucial for the kernel design apparatus

Upon completion of the procedure presented in (2.3.7), the following formulation is obtained:

$$\begin{aligned}
\min_{\boldsymbol{\mu},\boldsymbol{\eta},\boldsymbol{\delta},\boldsymbol{\delta}^*,\lambda} \quad & t \\
\text{s.t.} \quad & \begin{bmatrix} \sum_{i=1}^{M} \mu_i k_i & (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) & t - 2C(\boldsymbol{\delta} + \boldsymbol{\delta}^*)^\top \end{bmatrix} \succeq 0 \\
& \boldsymbol{\eta} - (\boldsymbol{\delta}^* - \boldsymbol{\delta}) \geq 2\epsilon\mathbf{1} \\
& \delta_k, \delta_k^*, \eta_k \geq 0 \quad k = 1, \cdots, N \\
& \mu_i \geq 0 \quad i = 1, \cdots, M \\
& \boldsymbol{\mu}^\top \mathbf{1} \leq \tau
\end{aligned}$$

The linearity of the last two constraints over μ give rise to a L1 like penalization, which makes possible to reduce the set of resulting kernels found after computation, by choosing $\mu_i > 0$.

Due to numerical considerations, during computation, μ_i values below a certain threshold are set to zero (e.g., 1×10^{-5})

3.1.2 SVR L2 SDP Multikernel explicit μ constraint

For the L2 formulation, it follows that,

$$\begin{aligned}
\min_{\boldsymbol{\mu},\boldsymbol{\eta},\lambda} \quad & t \\
& \begin{bmatrix} -\left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right) & (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \mathbf{J}) & t \end{bmatrix} \succeq 0 \\
\text{s.t.} \quad & 0 \leq \eta_k \leq 2\epsilon \quad k = 1, \cdots, N \\
& \mu_i \geq 0 \quad i = 1, \cdots, M \\
& \boldsymbol{\mu}^\top \mathbf{1} \leq \tau
\end{aligned}$$

3.1.3 SVR L1 SDP Multikernel with μ dependent regularization term

As explained in section 2.3.9 we will consider the original dual problem (2.92) as a performance measure that will be minimized with respect to μ, this step will prevent the computation of the gradient with respect to μ during the Wolfe dual derivation, simplifying the overall process.

Let us now suppose that the performance measure is given by:

$$\begin{aligned}\Omega(\boldsymbol{K}) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{N} \mu_i \boldsymbol{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ & -\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - \tau g(\boldsymbol{\mu}) \qquad (3.1)\\ \text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \le \alpha_k, \alpha_k^* \le C, \quad k = 1, \cdots, N\end{aligned}$$

Where $g(\boldsymbol{\mu})$ is a linear function on $\boldsymbol{\mu}$, that penalizes the set of μ_i and $\tau \ge 0$. Since the inner problem or the performance measure does not depend on the $\boldsymbol{\mu}$ parameter, the computation of the Wolf dual is straightforward. By following the procedure detailed in (2.3.7),

(3.1) remains convex because it is created by adding a positive linear function of $\boldsymbol{\mu}$, and the sum of two convex functions is also convex .

Laurent El Ghaoui. Lecture 3: Convex sets and functions, 2012. URL https://people.eecs.berkeley.edu/~elghaoui/Teaching/EE227A/lecture3.pdf. Accessed on July 14, 2025

$$\begin{aligned}\min_{\boldsymbol{\mu},\boldsymbol{\eta},\boldsymbol{\delta},\boldsymbol{\delta}^*,\lambda} \quad & t \\ \text{s.t.} \quad & \begin{bmatrix} \sum_{i=1}^{N} \mu_i \boldsymbol{k}_i & (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) & t - 2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top - \tau g(\boldsymbol{\mu}) \end{bmatrix} \succeq 0 \\ & \boldsymbol{\eta} - (\boldsymbol{\delta}^* - \boldsymbol{\delta}) \ge 2\epsilon\mathbf{1} \\ & \delta_k, \delta_k^*, \eta_k \ge 0 \quad k = 1, \cdots, N \\ & \mu_i \ge 0 \quad i = 1, \cdots, M\end{aligned}$$

For $g(\mu) = \text{trace}\left(\sum_{i=1}^{N} \mu_i k_i\right)$

$$\begin{aligned}\min_{\boldsymbol{\mu},\boldsymbol{\eta},\boldsymbol{\delta},\boldsymbol{\delta}^*,\lambda} \quad & t \\ \text{s.t.} \quad & \begin{bmatrix} \sum_{i=1}^{N} \mu_i \mathbf{k}_i & (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + y + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) & t - 2C(\boldsymbol{\delta}+\boldsymbol{\delta}^*)^\top - \tau\text{trace}\left(\sum_{i=1}^{N} \mu_i \mathbf{k}_i\right) \end{bmatrix} \succeq 0 \\ & \boldsymbol{\eta} - (\boldsymbol{\delta}^* - \boldsymbol{\delta}) \ge 2\epsilon\mathbf{1} \\ & \delta_k, \delta_k^*, \eta_k \ge 0 \quad k = 1, \cdots, N \\ & \mu_i \ge 0 \quad i = 1, \cdots, M\end{aligned}$$

And for $g(\boldsymbol{\mu}) = \boldsymbol{\mu}^\tau \mathbf{1}$

$$
\begin{aligned}
\min_{\mu,\eta,\delta,\delta^*,\lambda} \quad & t \\
\text{s.t.} \quad & \begin{bmatrix} \sum_{i=1}^{N} \mu_i \mathbf{k}_i & (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \boldsymbol{\eta} - \boldsymbol{\delta} + \lambda\mathbf{1}) & t - 2C(\boldsymbol{\delta} + \boldsymbol{\delta}^*)^\top - \tau\boldsymbol{\mu}^\top\mathbf{1} \end{bmatrix} \succeq 0 \\
& \boldsymbol{\eta} - (\boldsymbol{\delta}^* - \boldsymbol{\delta}) \geq 2\epsilon\mathbf{1} \\
& \delta_k, \delta_k^*, \eta_k \geq 0 \quad k = 1, \cdots, N \\
& \mu_i \geq 0 \quad i = 1, \cdots, M
\end{aligned}
\tag{3.2}
$$

Unlike the problem in 2.3.11—where a smaller τ acts as a tighter upper limit on the parameter $\boldsymbol{\mu}$—this method increases penalization by raising the value of τ. This occurs because the term $\tau g(^-)$ is part of the objective function and is therefore regularized.

3.1.4 *SVR L2 SDP Multikernel with μ dependent regularization term*

Similarly, considering (2.103) to build the following performance measure:

$$
\begin{aligned}
\Omega(\boldsymbol{\mu}) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -\frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^\top \left(\sum_{i=1}^{N} \mu_i \mathbf{k}_i + I\frac{1}{C} \right) (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \\
& -\epsilon\mathbf{1}^\top(\boldsymbol{\alpha} + \boldsymbol{\alpha}^*) + y^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) - g(\boldsymbol{\mu}) \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) = 0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N
\end{aligned}
$$

We can derive the following two expressions:
With $g(\boldsymbol{\mu}) = \text{trace}\left(\sum_{i=1}^{N} \mu_i \mathbf{k}_i\right)$

$$
\begin{aligned}
\min_{\mu,\eta,\lambda} \quad & t \\
& \begin{bmatrix} -\left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right) & (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) & t - \tau\text{trace}\left(\sum_{i=1}^{N} \mu_i \mathbf{k}_i\right) \end{bmatrix} \succeq 0 \\
\text{s.t.} \quad & 0 \leq \eta_k \leq 2\epsilon, \quad k = 1, \cdots, N \\
& \mu_i \geq 0, \quad i = 1, \cdots, M
\end{aligned}
$$

$g(\boldsymbol{\mu}) = \boldsymbol{\mu}^\tau\mathbf{1}$

$$
\begin{aligned}
\min_{\boldsymbol{\mu},eta,\lambda} \quad & t \\
& \begin{bmatrix} -\left(\mathbf{K} + \mathbf{I}\frac{1}{C}\right) & (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \lambda\mathbf{1} + \boldsymbol{\eta}) & t - \tau\boldsymbol{\mu}^\top\mathbf{1} \end{bmatrix} \succeq 0 \\
\text{s.t.} \quad & 0 \leq \eta_k \leq 2\epsilon, \quad k = 1, \cdots, N \\
& \mu_i \geq 0, \quad i = 1, \cdots, M
\end{aligned}
$$

3.1.5 SVR L1 SDP with Kronecker kernel and μ regularization term

To enhance the numerical stability of (3.3), a Kronecker kernel (see 2.1.3) with variable weight can be added to the problem to ensure positive semidefiniteness [3] of the resulting kernel, when it is not assured by the sum of the other kernels.

[3] This stability enhancement phenomena is further explained in section 2.2.2

Considering:

$$\begin{aligned}\min_{\mu}\max_{\alpha,\alpha^*} \quad & -(\alpha-\alpha^*)^\top \mu_0 \mathbf{I} + \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\alpha-\alpha^*) \\ & -2\epsilon \mathbf{1}^\top(\alpha+\alpha^*) + 2y^\top(\alpha-\alpha^*) + \tau\sum_{i=0}^{M}\mu_i \\ \text{s.t.} \quad & \mathbf{1}^\top(\alpha-\alpha^*) = 0 \\ & 0 \le \alpha_k, \alpha_k^* \le C, \quad k = 1,\cdots,N \\ & \mu_i \ge 0, \quad i = 1,\cdots,M\end{aligned}$$

It's possible to obtain the following problem:

$$\begin{aligned}\min_{\mu,\eta,\delta,\delta^*,\lambda} \quad & t \\ \text{s.t.} \quad & \begin{bmatrix} \mu_0\mathbf{I} + \sum_{i=1}^{N}\mu_i\mathbf{k}_i & (-\epsilon\mathbf{1} + \mathbf{y} + \eta - \delta + \lambda\mathbf{1})^\top \\ (-\epsilon\mathbf{1} + \mathbf{y} + \eta - \delta + \lambda\mathbf{1}) & t - 2C(\delta+\delta^*)^\top - \tau\mu^\top\mathbf{1} \end{bmatrix} \succeq 0 \\ & \eta - (\delta^* - \delta) \ge 2\epsilon\mathbf{1} \\ & \delta_k, \delta_k^*, \eta_k \ge 0 \quad k = 1,\cdots,N \\ & \mu_i \ge 0 \quad i = 1,\cdots,M\end{aligned} \tag{3.3}$$

Where strong convexity is guaranteed, it allows for finding a solution even when $\mathbf{K}$ is not positive semidefinite.

3.1.6 SVR L1 SDP KKT Conditions

Section 2.3.7 showed that obtaining the vector $(\alpha - \alpha^*)$ theoretically requires inverting the resulting kernel. A significant challenge arises with real datasets, as the kernel matrix is frequently singular and non-invertible. Although pseudo-inverse techniques are available, their accuracy can be unreliable for singular matrices.

Moreover, the calculation of the bias b relies on accurately identifying support vectors and subsequently filtering unbounded ones. Because this process is sensitive to the solution's accuracy, the filtering approach described in Section 3.1.6 can introduce inaccuracies in the final predictions.

Here, we will derive and analyze the KKT conditions for the given problem. This conditions can be used to filter the set of bounded and

unbounded conditions, without the need to compute $(\alpha - \alpha^*)$ in the first place.

Case 1: $\alpha_k = 0$ **and** $\alpha_k^* = 0$

Using (2.96), the KKT conditions in 2.3.7 and 2.3.7, and recalling that the primal variables are active, when the duals are inactive and vice-versa:

$$\begin{aligned}
&\eta_k \alpha_k = 0 \rightarrow \eta_k > 0 \\
&\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* > 0 \\
&\delta_k (C - \alpha_k) = 0 \rightarrow \delta_k C = 0 \rightarrow \delta_k = 0 \\
&\delta_k^* (C - \alpha_k^*) = 0 \rightarrow \delta_k^* C = 0 \rightarrow \delta_k^* = 0 \\
&(\eta_k - \delta_k) + (\eta_k^* - \delta_k^*) = 2\epsilon \rightarrow \boxed{\eta_k + \eta_k^* = 2\epsilon}
\end{aligned}$$

From section , $\alpha_k = 0$ and $\alpha_k^* = 0$ implies (y_k, x_k) inside the margin.

Case 2: $0 < \alpha_k < C$ **and** $\alpha_k^* = 0$

$$\begin{aligned}
&\eta_k \alpha_k = 0 \rightarrow \eta_k = 0 \\
&\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* > 0 \\
&\underbrace{\delta_k}_{=0} \underbrace{(C - \alpha_k)}_{>0} = 0 \rightarrow \delta_k = 0 \\
&\underbrace{\delta_k^*}_{=0} \underbrace{(C - \alpha_k^*)}_{>0} = 0 \rightarrow \delta_k^* = 0 \\
&(\eta_k - \delta_k) + (\eta_k^* - \delta_k^*) = 2\epsilon \rightarrow \boxed{\eta_k^* = 2\epsilon}
\end{aligned}$$

Which implies (y_k, x_k) on one side of the margin

Case 3: $\alpha_k = 0$ **and** $0 < \alpha_k^* < C$

$$\begin{aligned}
&\eta_k \alpha_k = 0 \rightarrow \eta_k > 0 \\
&\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* = 0 \\
&\underbrace{\delta_k}_{=0} \underbrace{(C - \alpha_k)}_{>0} = 0 \rightarrow \delta_k = 0 \\
&\underbrace{\delta_k^*}_{=0} \underbrace{(C - \alpha_k^*)}_{>0} = 0 \rightarrow \delta_k^* = 0 \\
&(\eta_k - \delta_k) + (\eta_k^* - \delta_k^*) = 2\epsilon \rightarrow \boxed{\eta_k = 2\epsilon}
\end{aligned}$$

Which implies (y_k, x_k) on the other side of the margin

Case 4: $\alpha_k = C$ **and** $\alpha_k^* = 0$

$$\eta_k \alpha_k = 0 \rightarrow \eta_k = 0$$
$$\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* > 0$$
$$\underbrace{\delta_k}_{\geq 0} \underbrace{(C - \alpha_k)}_{=0} = 0 \rightarrow \delta_k \geq 0$$
$$\underbrace{\delta_k^*}_{=0} \underbrace{(C - \alpha_k^*)}_{>0} = 0 \rightarrow \delta_k^* = 0$$
$$(\eta_k - \delta_k) + (\eta_k^* - \delta_k^*) = 2\epsilon \rightarrow \boxed{\eta_k^* - \delta_k = 2\epsilon}$$

Corresponding to support vectors outside of one side of the margin.

Case 5: $\alpha_k = 0$ **and** $\alpha_k^* = C$

$$\eta_k \alpha_k = 0 \rightarrow \eta_k > 0$$
$$\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* = 0$$
$$\underbrace{\delta_k}_{=0} \underbrace{(C - \alpha_k)}_{\geq 0} = 0 \rightarrow \delta_k = 0$$
$$\underbrace{\delta_k^*}_{\geq 0} \underbrace{(C - \alpha_k^*)}_{=0} = 0 \rightarrow \delta_k^* \geq 0$$
$$(\eta_k - \delta_k) + (\eta_k^* - \delta_k^*) = 2\epsilon \rightarrow \boxed{\eta_k - \delta_k^* = 2\epsilon}$$

Corresponding to support vectors outside of the other side of the margin.

3.1.7 *SVR L2 SDP KKT Conditions*

Recalling (2.106) and the following KKT conditions, we can obtain the following relations

Case 1: $\alpha_k = 0$ **and** $\alpha_k^* = 0$

$$\eta_k \alpha_k = 0 \rightarrow \eta_k > 0$$
$$\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* > 0$$
$$\eta_k + \eta_k^* = 2\epsilon \rightarrow \boxed{\eta_k = -\eta_k^* + 2\epsilon}$$

From section , $\alpha_k = 0$ and $\alpha_k^* = 0$ implies (y_k, x_k) inside the margin.

Case 2: $\alpha_k > 0$ **and** $\alpha_k^* = 0$

$$\eta_k \alpha_k = 0 \rightarrow \boxed{\eta_k = 0}$$
$$\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* > 0$$
$$\eta_k^* + \eta_k = 2\epsilon \rightarrow \eta_k^* = 2\epsilon$$

Corresponding to support vectors on or outside of one side of the margin.

Case 3: $\alpha_k = 0$ **and** $\alpha_k^* > 0$

$$\eta_k \alpha_k = 0 \rightarrow \eta_k > 0$$
$$\eta_k^* \alpha_k^* = 0 \rightarrow \eta_k^* > 0$$
$$\eta_k^* + \eta_k = 2\epsilon \rightarrow \boxed{\eta_k = 2\epsilon}$$

Corresponding to support vectors on or outside of the other side of the margin.

For the case of the L2 formulation, we will focus on the value of η, since η^* is not considered for the optimization problem.

3.2 *QCQP formulations*

In this module, we introduce QCQP problems, which differ from those in 2.2.7 and 2.2.8. By using a performance metric with regularization of μ, as shown in the previous section for SDP formats. Instead of employing the epigraph formulation, we apply the Objective-to-Constraint transformation detailed in Section 2.1.5. This approach prevents quadratic constraints from being normalized by their corresponding kernel trace, which increases feasibility instability, as demonstrated in the next chapter through an applied example.

3.2.1 *SVR QCQP L1 Multikernel (μ regularization term)*

Considering the SVR QCQP explored in section 2.2.7, we will present a different way to constrain the search space of $\boldsymbol{\mu}$ by specifying a penalization term to the problem's objective function.

Consider the following performance metric,

$$\begin{aligned}
\Omega(\mathbf{K}) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& -2\epsilon \mathbf{1}^\top (\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2y^\top (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \tau g(\boldsymbol{\mu}) \\
\text{s.t.} \quad & \mathbf{1}^\top (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N
\end{aligned}$$

Where $g(\boldsymbol{\mu})$ is a penalization term for the set of μ_i parameters. The complete problem now appears as,

$$\begin{aligned}
\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \sum_{i=1}^{M} \mu_i \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& -2\epsilon \mathbf{1}^\top (\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2y^\top (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \tau g(\boldsymbol{\mu}) \\
\text{s.t.} \quad & \mathbf{1}^\top (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N \\
& \mu_i \geq 0, \quad i = 1, \cdots, M
\end{aligned}$$

Rearranging the $\boldsymbol{\mu}$ depend terms

$$\begin{aligned}
&\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\
&\qquad+\min_{\boldsymbol{\mu}}\left[-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\sum_{i=1}^{M}\mu_i\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+\tau g(\boldsymbol{\mu})\right]\\
&\text{s.t.}\quad \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0\\
&\qquad 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N\\
&\qquad \mu_i\geq 0,\quad i=1,\cdots,M
\end{aligned}$$

lets select $g(\boldsymbol{\mu})=\boldsymbol{\mu}^\top\mathbf{1}=\sum_{i=1}^{M}\mu_i$

$$\begin{aligned}
&\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\
&\qquad+\min_{\boldsymbol{\mu}}\left[-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\sum_{i=1}^{M}\mu_i\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+\tau\sum_{i=1}^{M}\mu_i\right]\\
&\text{s.t.}\quad \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0\\
&\qquad 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N\\
&\qquad \mu_i\geq 0,\quad i=1,\cdots,M
\end{aligned}\tag{3.4}$$

Factoring μ_i

$$\begin{aligned}
&\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\
&\qquad+\min_{\boldsymbol{\mu}}\sum_{i=1}^{N}\mu_i\left[-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+\tau\right]\\
&\text{s.t.}\quad \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0\\
&\qquad 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N\\
&\qquad \mu_i\geq 0,\quad i=1,\cdots,M
\end{aligned}\tag{3.5}$$

Due to the emerging Lagrangian-like structure of the functional and the final constraint in (3.5), the problem can be reformulated as follows

$$\begin{aligned}
&\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\
&\text{s.t.}\quad -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+\tau\geq 0,\quad i=1\dots M\\
&\qquad \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0\\
&\qquad 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N
\end{aligned}\tag{3.6}$$

And modifying the order of the terms on the first set of constraints in (3.6),

$$\begin{aligned}
&\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} -2\epsilon\mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\\
&\text{s.t.}\quad (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top\mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)\leq\tau,\quad i=1\dots M\\
&\qquad \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0\\
&\qquad 0\leq\alpha_k,\alpha_k^*\leq C,\quad k=1,\cdots,N
\end{aligned}\tag{3.7}$$

To understand the previous transformation, see Objective-to-Constraint Transformation (2.1.5).

Using $g(\boldsymbol{\mu}) = \sum_{i=1}^{M} \mu_i \text{trace}(\mathbf{k}_i)$ we get:

$$\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \; & -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
\text{s.t.} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \leq \tau \; \text{trace}(\mathbf{k}_i), \quad i = 1 \dots M \\
& \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N
\end{aligned}$$

3.2.2 *SVR QCQP L2 Multikernel (μ regularization term)*

Consider the following performance metric,

$$\begin{aligned}
\Omega(\mathbf{K}) = \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i + I\frac{1}{C} \right) (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \tau g(\boldsymbol{\mu}) \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N
\end{aligned}$$

Where $g(\boldsymbol{\mu})$ is a penalization term for the set of μ_i parameters. The complete problem now appears as,

$$\begin{aligned}
\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \; & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \left(\sum_{i=1}^{M} \mu_i \mathbf{k}_i + I\frac{1}{C} \right) (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \tau g(\boldsymbol{\mu}) \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N \\
& \mu_i \geq 0, \quad i = 1, \cdots, M
\end{aligned}$$

Rearranging the $\boldsymbol{\mu}$ depend terms and selecting $g(\boldsymbol{\mu}) = \boldsymbol{\mu}^\top \mathbf{1} = \sum_{i=1}^{M} \mu_i$

$$\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) - \frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& + \min_{\boldsymbol{\mu}} \left[-\sum_{i=1}^{M} \mu_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \tau \sum_{i=1}^{M} \mu_i \right] \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k = 1, \cdots, N \\
& \mu_i \geq 0, \quad i = 1, \cdots, M
\end{aligned}$$

Factoring μ_i

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-\frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& + \min_{\boldsymbol{\mu}} \left[-(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \tau \sum_{i=1}^{M} \mu_i \right] \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k=1,\cdots,N
\end{aligned}
\tag{3.8}
$$

For the Lagrangian relaxation structure present in (3.8), we get:

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-\frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
\text{s.t.} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \leq \tau, \quad i=1\ldots M \\
& \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k=1,\cdots,N
\end{aligned}
$$

A more detailed explanation can be found in the previous section or 2.1.5.

Consider an alternative problem: let $g(\boldsymbol{\mu}) = \sum_{i=1}^{M} \mu_i \text{trace}(\mathbf{k}_i)$, then:

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -2\epsilon(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2\mathbf{y}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)-\frac{1}{C}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
\text{s.t.} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \leq \tau \ \text{trace}(\mathbf{k}_i), \quad i=1\ldots M \\
& \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\
& \alpha_k, \alpha_k^* \geq 0, \quad k=1,\cdots,N
\end{aligned}
$$

3.2.3 *SVR QCQP L1 with kronecker kernel (μ regularization term)*

Unlike the SDP equivalent problem (3.3), the derivation of (3.4) does not require strong convexity of the kernel $\mathbf{K}$ to find a solution [4]. However, a Kronecker kernel can be included to account for noise in time-series problems.

[4] Convexity is still needed to transition from the min-max to the max-min problem, as detailed in (2.65)

Considering,

$$
\begin{aligned}
\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & -(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mu_0 \mathbf{I} + \sum_{i=1}^{M} \mu_i \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
& -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*)+2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)+\tau\sum_{i=0}^{M}\mu_i \\
\text{s.t.} \quad & \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)=0 \\
& 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k=1,\cdots,N \\
& \mu_i \geq 0, \quad i=1,\cdots,M
\end{aligned}
$$

The following problem can be obtained with the processes expressed above:

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \; & -2\epsilon \mathbf{1}^\top(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) + 2y^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\
\text{s.t.} \quad & (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{I}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \leq \tau \\
& (\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)^\top \mathbf{k}_i(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \leq \tau, \quad i = 1 \ldots M \\
& \mathbf{1}^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\
& 0 \leq \alpha_k, \alpha_k^* \leq C, \quad k = 1, \cdots, N
\end{aligned}
$$

4 Case Study

Contents

4.1 Comparative analysis Mauna Loa CO_2 Dataset

The Mauna Loa CO_2 time series dataset will be used to test the multikernel design capabilities of the SVR QCQP formulations. A speed comparison will be presented against benchmark algorithms and the SDP formulation.

4.1.1 Data description

The problem discussed in this section is sourced from [1]. In section 5.4.3, they illustrated how to use a linear combination of kernels to fit the data, capturing various aspects such as long-term trends and minor

[1] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://gaussianprocess.org/gpml/chapters/RW.pdf

irregularities. This approach is subsequently outlined in the Scikit-learn documentation [2], using the kernel weightings that we will apply to this problem.

The dataset contains the monthly average CO_2 atmospheric concentrations recorded at the Mauna Loa Observatory in Hawaii from 1958 to 2001

[2] Scikit-Learn documentation. Gaussian process regression (gpr) with noise-level estimation. URL https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_co2.html

Date	CO_2 (ppm)
1958-03	316.10
1958-04	317.21
1958-05	317.46
1958-06	317.10
1958-07	315.86
⋮	⋮

Table 4.1: Monthly Average CO_2 Concentration (First 5 Months)

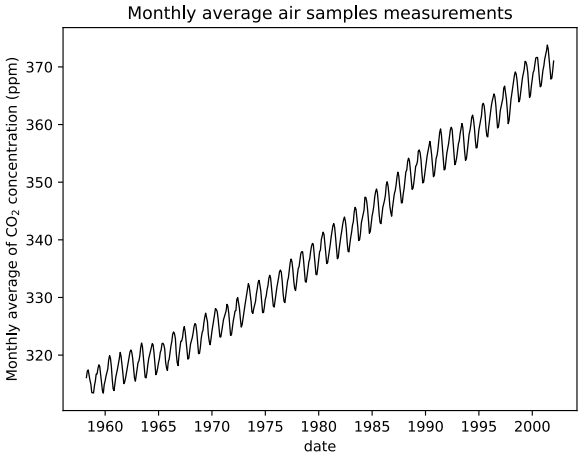

Figure 4.1: Data from Mauna Loa Observatory

Figure 4.1 illustrates the data grouped by month, excluding months where no information was recorded. Based on the series decomposition (Figure 4.2) using the Statsmodels library [3], the data showed two periodic regions characterized by their seasonality patterns. As part of the data cleaning process, the first 67 entries were removed, enhancing predictability by reducing complexity. This adjustment assumes that the seasonality will maintain the same amplitude, allowing the use of the same kernel weights obtained from the optimization process. The new starting point was set to 1964, resulting in a more stable seasonality pattern and a more linear trend.

[3] statsmodel documentation. statsmodels.tsa.seasonal.stl. URL https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.STL.html

4.1.2 *Training process*

The dataset was divided into training and testing sets, with the last 145 data points designated as the testing data, corresponding to 12 years.

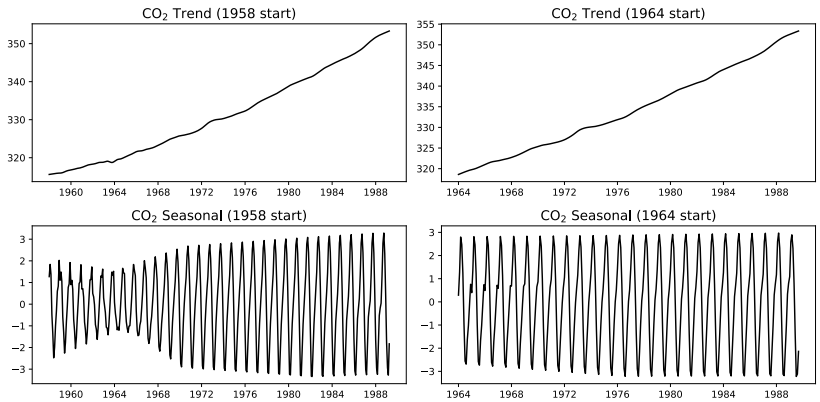

Figure 4.2: The figure displays the STL decomposition of CO2 data, showing the trend and seasonal components for two different start years (1958 and 1964).

To tune the model's hyperparameters, multiple t-folds were used to accommodate different window predictions, adjusting the number of training and validation data for each test size. The test size and t-fold values were chosen arbitrarily. An example of this method for a test size of 60 is shown in Figure 4.3.

# Predictions	test size (months)
3	60
15	12
30	6
40	3

Table 4.2: Predefined number of predictions per test set size for experiments with Mauna Loa CO_2 Dataset.

Table 4.2 shows the number of validation steps used for each test size value. The computational demand rises as the test size decreases.

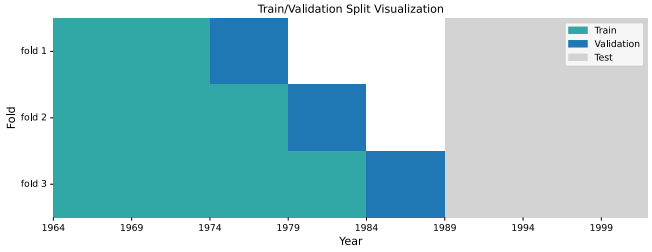

Figure 4.3: Illustration of T-Fold with a Test Size of 60.

For each hyperparameter combination, we employed the training procedure illustrated in Figure 4.3. This involved initially training the model on the first segment of data and evaluating its performance on the immediately following validation segment. Subsequently, this validation segment was added to the training set, the model was retrained, and then evaluated on the next validation segment. This process of incrementally expanding the training set and evaluating on the subsequent segment was repeated [4]. The final score for each

[4] This process is known as Walk-Forward validation

hyperparameter combination was calculated as the average of the scores obtained from all validation steps.

In this comparison, the QCQP SVR L1 formulation from 3.2.1 and the QCQP SVR L2 formulation were evaluated alongside the Scikit-Learn SVR L1 formulation [5] and the Scikit-Learn Gaussian Processes Regressor, with Mean Absolute Percentage Error as the metric.

As explained in [6], considering the following kernels

$$\text{long\_term\_kernel} = 50^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\cdot 50^2}\right)$$

This is the RBF kernel with a length-scale (σ) of 50 and 50^2 as scaler

$$\text{seasonal\_kernel} = 2^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\cdot 100^2}\right)\left[\exp\left(-\frac{2\sin^2(\pi\|\mathbf{x}-\mathbf{x}'\|/1)}{1}\right)\right]$$

This term comprises the product of an RBF kernel and a periodic kernel.

$$\text{irregular\_kernel} = 0.5^2\left(1+\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\cdot 1\cdot 1}\right)^{-1}$$

This irregular kernel includes the Rational Quadratic kernel with $\alpha, \ell = 1$

$$\text{noise\_kernel} = 0.1^2 \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\cdot 0.1^2}\right) + \sigma_n^2 I$$

This last kernel includes an RBF kernel with $\sigma = 0.1$ and a Noise kernel $\sigma_n^2 I$ to represent the normally-distributed noise in the time series; with standard deviation or noise level $\sigma_n^2 = 1$.

The previous kernels were used to train the Multikernel algorithm, as a weighted sum, by finding the parameters for each kernel as follows

$$\begin{aligned}\text{multikernel} = (&\\ &\mu_1 \cdot \text{long\_term\_trend\_kernel}\\ &\mu_2 \cdot \text{seasonal\_kernel}\\ &\mu_3 \cdot \text{irregularities\_kernel}\\ &\mu_4 \cdot \text{noise\_kernel}\\ )&\end{aligned}$$

While the single kernel algorithm (Sickit-learn) used a linear combination of the previous without optimize weights as follows

[5] sklearn.svm.svr. URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[6] Scikit-Learn documentation. Gaussian process regression (gpr) with noise-level estimation. URL https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_co2.html

For details on individual kernels, refer to 2.1.3

$$\begin{aligned} \text{multikernel} = (& \\ & \text{long\_term\_trend\_kernel} \\ & \text{seasonal\_kernel} \\ & \text{irregularities\_kernel} \\ & \text{noise\_kernel} \\ ) & \end{aligned} \tag{4.1}$$

4.1.3 *Hyperparametes Analysis*

Hyperparameter optimization was conducted employing the default solver within the Optuna library [7]. The corresponding score metrics were recorded for subsequent analysis. This analysis indicated that, for the problem under consideration, the score metric was insensitive to variations in τ and C. In contrast, a correlation was observed between the score metric and changes in the ϵ parameter, which leaned towards small values (See Figure 6.3 and 6.4).

[7] Optuna. URL https://optuna.org/

The optimal hyperparameters were used to test the model via the walk-forward method. The standard deviation of the weight assigned to each kernel was computed across all training splits to assess weight stability. Figure 6.1 and 6.2 present these standard deviations. The figures indicate that a different subset of kernels was employed by each model, implying their relevance to the task.

4.1.4 *Kernel Weights Analysis*

Kernel weights resulting from the training process using optimal hyperparameters were found to be small in magnitude, attributed to a large τ value for all models. Furthermore, the walk-forward validation results exhibited low variance across the evaluation splits.

The following kernel weights were obtained after training:

Model	**Test size**	μ_1	μ_2	μ_3	μ_4
Multikernel L1	60	0.0699	0.1466	0.2425	0.1541
Multikernel L2	60	0.1151	0.2419	0.4040	0.1031
Multikernel L1	12	0.0860	0.1723	0.3217	0.5392
Multikernel L2	12	0.0509	0.0943	0.1564	0.0000
Multikernel L1	6	0.1195	0.2270	0.4001	2.5384
Multikernel L2	6	0.1350	0.2566	0.4553	2.7645
Multikernel L1	3	0.0745	0.1408	0.2757	0.0000
Multikernel L2	3	0.0503	0.0959	0.1717	0.9995

Table 4.3: Resulting kernel weights μ_i for SVR Multikernel algorithms (Mauna Loa CO_2 Dataset)

The kernel weights were obtained by calculating the mean of the resulting weights obtained through the Rolling Forecast method.

4.1.5 *Performance results*

To test the hyperparameters derived from optimization, we employed a Walk-Forward validation methodology applied to 100 synthetic time series. These series were constructed by first applying block bootstrap 2.1.10 to the noise component isolated from the original time series decomposition. Each resulting bootstrapped noise sequence was then added back to the deterministic components (e.g., trend, seasonality) of the original series. The results of this testing procedure are shown in Table 4.4.

Model	Test size	C	ϵ	τ	score
Scikit-Learn L1	60	313.0942	0.2123	–	0.3444
Multikernel L1	60	900.5077	0.4197	824.0626	0.3238
Multikernel L2	60	695.5491	1.8840	70.6037	0.3536
GPR	60	–	–	–	0.4377
Scikit-Learn L1	12	3.2966	0.3082	–	0.1584
Multikernel L1	12	221.9864	0.2762	340.1590	0.1584
Multikernel L2	12	367.6752	0.1261	573.1438	0.2596
GPR	12	–	–	–	0.2150
Scikit-Learn L1	6	0.9019	0.1281	–	0.1416
Multikernel L1	6	731.6232	0.0136	597.0183	0.1457
Multikernel L2	6	268.3914	0.6962	396.5969	0.2501
GPR	6	–	–	–	0.2103
Scikit-Learn L1	3	21.2718	0.0278	–	0.1585
Multikernel L1	3	7.1150	0.0406	454.3283	0.1319
Multikernel L2	3	801.1016	0.0020	944.8666	0.2297
GPR	3	–	–	–	0.2066

Table 4.4: Comparison of Scikit-Learn and Multikernel model performance (MAPE) with varying test sizes and hyperparameters. Values rounded to 4 decimal places.

Table 4.4 shows a lead on the metrics obtained through the Scikit-learn L1 and the Multikernel L1 models. It can be observed that, for the Scikit-Learn models, the MAPE metric decreases with test size, reaching a minimum at test size 6, and then increasing again. On the other hand, the score for the multikernel continues to decrease with increasing test size. This could indicate that the designed kernel (4.1) is best suited for predicting 6 steps, but may not be efficiently adjusted for other horizons by altering the hyperparameters.

Using the 100 results obtained from the bootstrapping process, multiple statistical tests (Table 4.5) were conducted for the two main models presented in 4.4. It is evident that for test sizes 60 and 12, the ranges overlap and the effect size is small (Refer to [8]), indicating no noticeable improvement by the Multikernel. However, for test size 6, there is a statistically significant mean difference between the two groups, with a medium effect size, favoring the performance of the

[8] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988

	Test size (Months)			
Metric	**60**	**12**	**6**	**3**
Scikit-learn SVR L1				
Mean MAPE	0.3444	0.1584	0.1416	0.1585
95% CI	(0.3271, 0.3617)	(0.1566, 0.1602)	(0.1403, 0.1429)	(0.1567, 0.1603)
Multikernel SVR L1				
Mean MAPE	0.3237	0.1584	0.1457	0.1319
95% CI	(0.3184, 0.3290)	(0.1567, 0.1601)	(0.1443, 0.1472)	(0.1305, 0.1333)
Paired t-test (Scikit-learn SVR L1 vs. Multikernel SVR L1)				
Cohen's d Effect	0.2268	0.0065	-0.6067	3.3153
p-value (2-sided)	0.0255	0.9486	<0.0001	<0.0001

Table 4.5: Comparison of Mean MAPE (95% CI), Effect Size and 2-sided p-value (H_a: means difference is not zero): Standard SVR vs. Multikernel SVR L1 Across Different Test Horizons. All differences were confirmed to be normally distributed by the Shapiro-Wilk test for normality.

Scikit-learn benchmark model. Conversely, test size 3 highlights a noticeable improvement for the Multikernel model over the Scikit-learn model, supporting the idea that the designed kernel lacks optimal adjustment when kernel weights are not specifically optimized for the problem.

4.1.6 *Alternative problem*

The previous problem was also addressed using the SVR QCQP L1 formulation with a different set of kernels.

The set of kernels was chosen to capture the visually identified characteristics of the data shape. The growing linear tendency was addressed by including the linear kernel. The high periodicity of the series motivated the use of the periodic kernel. To model the residual periodic noise and variations in the trend, the RBF kernel was added.

$$
\begin{aligned}
\text{multikernel} = (\\
&\mu_1 \cdot \underbrace{\mathbf{x}^T\mathbf{x}'}_{\text{linear}} + \\
&\mu_2 \cdot \underbrace{\exp\left(-1 \times 10^{-2}\|\mathbf{x}-\mathbf{x}'\|^2\right)}_{\text{RBF}(\gamma=1\times10^{-2})} + \\
&\mu_3 \cdot \underbrace{\exp\left(-\|\mathbf{x}-\mathbf{x}'\|^2\right)}_{\text{RBF}(\gamma=1)} + \\
&\mu_4 \cdot \underbrace{\exp\left(-1 \times 10^{2}\|\mathbf{x}-\mathbf{x}'\|^2\right)}_{\text{RBF}(\gamma=1\times10^{2})} + \\
&\mu_5 \cdot \underbrace{\exp\left(-2\sin^2(\pi\|\mathbf{x}-\mathbf{x}'\|/1)\right)}_{\text{periodic}(p=1)} + \\
)
\end{aligned}
$$

For the Scikit-Learn SVR L1 algorithm, the weights were set as one for each kernel. Following the process expressed in the previous section, the following metrics were obtained

Model	Test size	C	ϵ	τ	score
Multikernel L1	60	695.5491	1.8840	70.6037	0.3536
Scikit-Learn L1	60	26.4217	0.7758	–	2.4063
Multikernel L1	12	367.6752	0.1261	573.1438	0.2596
Scikit-Learn L1	12	402.9045	0.7823	–	44.5501
Multikernel L1	6	268.3914	0.6962	396.5969	0.2501
Scikit-Learn L1	6	619.9463	0.7674	–	73.9687
Multikernel L1	3	801.1016	0.0020	944.8666	0.2297
Scikit-Learn L1	3	673.4977	0.7619	–	93.9924

Table 4.6: Comparison of Scikit-Learn and Multikernel model performance (MAPE) with varying test sizes and hyperparameters. Values rounded to 4 decimal places (Alternative problem without predefined kernel weights).

4.1.7 *Speed comparison by kernel size*

The make regression dataset is composed of a group of random points to simulate a regression problem (A linear trend with noise).

To compare the speed of each algorithm, we utilized the make regression dataset with various sample sizes:

The Scikit-Learn algorithm demonstrates a significant advantage over the Multikernel versions, which can be mitigated by utilizing the optimized kernel resulting from hyperparameter tuning.

For the multikernel models, 12 kernels were employed: linear, polynomial of degree 2 and 3, and radial basis function (RBF) with gammas set to 1×10^{-2}, 1×10^{-1}, 1, 10, 1×10^{2}, 1×10^{3}, and 1×10^{4}. Additionally, periodic kernels with periodicities of 1 and 3 were used.

4.1.8 *Speed comparison by kernel number*

Using the make regression dataset, with 1000 datapoints. The speed was measured with respect to the number of kernels. The order of the kernels was introduced in the following order :

For simplicity, the kernels are referenced by their name. Refer to 2.1.3 for more information on each kernel

```
1. linear()
2. polynomial(degree=2)
3. polynomial(degree=3)
4. rbf(gamma=1e-2)
5. rbf(gamma=1e-1)
6. rbf(gamma=1)
7. rbf(gamma=1e2)
8. rbf(gamma=1e3)
9. rbf(gamma=1e4)
10. periodic(periodicity=1)
11. periodic(periodicity=3)
```

The results are as follows:

4.2 *Comparative analysis Sunspot*

The Sunspot dataset will be used to compare the score obtained through SVR QCQP L1 against a benchmark model. For this example, the L1

Model	Number of Samples	Fit Time (seconds)
Multi QCQP L1	100	0.2231
Multi SDP L1	100	0.6975
Multi QCQP L2	100	0.2254
Multi SDP L2	100	0.5128
Scikit-Learn L1	100	0.0014
Multi QCQP L1	300	0.8828
Multi SDP L1	300	10.5152
Multi QCQP L2	300	0.9187
Multi SDP L2	300	9.5158
Scikit-Learn L1	300	0.0034
Multi QCQP L1	500	2.0260
Multi SDP L1	500	37.5062
Multi QCQP L2	500	2.2007
Multi SDP L2	500	29.4625
Scikit-Learn L1	500	0.0039
Multi QCQP L1	700	4.3650
Multi SDP L1	700	103.8042
Multi QCQP L2	700	4.4791
Multi SDP L2	700	72.8612
Scikit-Learn L1	700	0.0154
Multi QCQP L1	900	6.3076
Multi SDP L1	900	212.0914
Multi QCQP L2	900	6.9399
Multi SDP L2	900	131.7354
Scikit-Learn L1	900	0.0197
Multi QCQP L1	1100	8.9925
Multi SDP L1	1100	343.2903
Multi QCQP L2	1100	8.9369
Multi SDP L2	1100	236.1608
Scikit-Learn L1	1100	0.0166

Table 4.7: Convergence time by data sample size of multiple algorithms.

formulation is chosen for its speed and stability due to the L1 and L2-like penalization discussed in section 3.1.1

4.2.1 *Data description*

This data originates from the SILSO (Sunspot Index and Long-term Solar Observations) Data Center [9]. It seeks to maintain a standardized record of solar activity, represented by the number of sunspots, which in turn aids the development of scientific research and weather forecasting [10].

The monthly dataset includes the year-month and the corresponding mean of the daily total sunspot numbers over all the days of that specific

[9] Solar Influences Data Analysis Center (SIDC). Sunspot number, 2025. URL https://www.sidc.be/SILSO/datafiles. Accessed on 2025-05-04. Data series v2.0 implemented 2015-07-01

[10] NOAA / NWS Space Weather Prediction Center. Sunspots/solar cycle, May 2025. URL https://www.swpc.noaa.gov/phenomena/sunspotssolar-cycle. Accessed on 2025-05-05

Number of Kernels	Model	Fit Time (seconds)
1	Multi QCQP L1	0.5397
1	Multi QCQP L2	0.7781
2	Multi QCQP L1	0.7577
2	Multi QCQP L2	1.1263
3	Multi QCQP L1	1.1095
3	Multi QCQP L2	1.4004
4	Multi QCQP L1	1.4019
4	Multi QCQP L2	1.7413
5	Multi QCQP L1	1.7781
5	Multi QCQP L2	2.1225
6	Multi QCQP L1	2.1747
6	Multi QCQP L2	2.4947
7	Multi QCQP L1	2.5542
7	Multi QCQP L2	2.8841
8	Multi QCQP L1	3.1118
8	Multi QCQP L2	3.4514
9	Multi QCQP L1	4.4390
9	Multi QCQP L2	4.7643
10	Multi QCQP L1	6.1247
10	Multi QCQP L2	6.8536
11	Multi QCQP L1	6.5377
11	Multi QCQP L2	7.0301
12	Multi QCQP L1	6.8275
12	Multi QCQP L2	7.5304

Table 4.8: SVR QCQP algorithms convergence time by number of kernels.

month. It covers data from January 1749 to April 2025, comprising a total of 3,316 entries. Due to computational constraints for the QCQP algorithms, the dataset was trimmed to start from January 1900.

Year	Month	Monthly Mean Sunspot Number
1900	01	15.7
1900	02	22.8
1900	03	14.4
1900	04	26.8
1900	05	25.3

Table 4.9: First 5 Entries of SIDC Monthly Mean Sunspot Number Starting Jan 1900

From Figure 4.4, it can be visually observed that the time series exhibits a periodic pattern [11]. The sole assumption is that this pattern repeats every 11 years.

[11] NOAA / NWS Space Weather Prediction Center. Sunspots/solar cycle, May 2025. URL https://www.swpc.noaa.gov/phenomena/sunspotssolar-cycle. Accessed on 2025-05-05

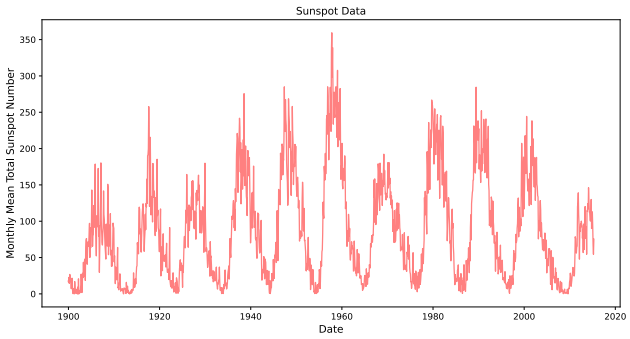

Figure 4.4: Sunspot Monthly Mean Sunspot Number dataset. Cropped from January 1900 to April 2025

4.2.2 *Problem description*

To model the data with limited prior information about the problem and thereby highlight the algorithm's automatic kernel design capabilities, a total of 49 kernels were employed. This quantity was chosen to provide diverse modeling capabilities while adhering to computational constraints.

The kernel selection process involved several considerations:

- Kernels 12, 13, and 14 were specifically included based on an assumed pattern spanning approximately 132 months (11 years). Additional periodic kernels were selected to account for potential yearly and monthly periodicities (e.g., 1, 3, 6 months, and sub-monthly intervals).

- For enhanced computational stability, a Kronecker kernel[12]. The remaining kernels were included without specific assumptions about the underlying data patterns.

1. `linear()`
2. `rbf(gamma=1e-2)`
3. `rbf(gamma=1e-1)`
4. `rbf(gamma=1)`
5. `rbf(gamma=1e2)`
6. `polynomial(degree=2)`
7. `polynomial(degree=3)`
8. `sigmoid()`
9. `laplacian(gamma=1e-1)`
10. `laplacian(gamma=1)`
11. `laplacian(gamma=1e1)`
12. `periodic(periodicity=132)`
13. `periodic(periodicity=132, length_scale=1e-1)`
14. `periodic(periodicity=132, length_scale=1e1)`
15. `periodic(periodicity=11)`
16. `periodic(periodicity=11, length_scale=1e-1)`

[12] The reference to the mathematical model that uses the Kronecker kernel is presented in 3.2.3 (kernel 49 in the list) was incorporated, functioning as an L2 penalization

For simplicity, kernels are referred to by name. Refer to 2.1.3 for more details on each kernel.

17. periodic(periodicity=11, length_scale=1e1)
18. periodic(periodicity=6)
19. periodic(periodicity=6, length_scale=1e-1)
20. periodic(periodicity=6, length_scale=1e1)
21. periodic(periodicity=3)
22. periodic(periodicity=3, length_scale=1e-1)
23. periodic(periodicity=3, length_scale=1e1)
24. periodic(periodicity=1)
25. periodic(periodicity=1, length_scale=1e-1)
26. periodic(periodicity=1, length_scale=1e1)
27. periodic(periodicity=0.25)
28. periodic(periodicity=0.25, length_scale=1e-1)
29. periodic(periodicity=0.25, length_scale=1e1)
30. periodic(periodicity=0.33)
31. periodic(periodicity=0.33, length_scale=1e-1)
32. periodic(periodicity=0.33, length_scale=1e1)
33. periodic(periodicity=0.5)
34. periodic(periodicity=0.5, length_scale=1e-1)
35. periodic(periodicity=0.5, length_scale=1e1)
36. rational quadratic()
37. rational quadratic(length_scale=1e-1)
38. rational quadratic(length_scale=1e1)
39. matern(nu=0.5)
40. matern(nu=0.5, length_scale=1e-1)
41. matern(nu=0.5, length_scale=1e1)
42. matern(nu=1.5)
43. matern(nu=1.5, length_scale=1e-1)
44. matern(nu=1.5, length_scale=1e1)
45. matern(nu=2.5)
46. matern(nu=2.5, length_scale=1e-1)
47. matern(nu=2.5, length_scale=1e1)
48. constantKernel()
49. KroneckerKernel()

The hyperparameters were tuned using the Walk-Forward method. For this study, models were trained using a step size of 12 and 10 iterations. These values were selected to support a one-year-ahead prediction scenario.

4.2.3 *Results*

The resulting hyperparameters were tested on the remaining 168 data points. Figure 4.5 presents the visual results of the QCQP L1 algorithm (3.7) compared to the classical SVR L1 algorithm with an RBF kernel optimized in the same manner.

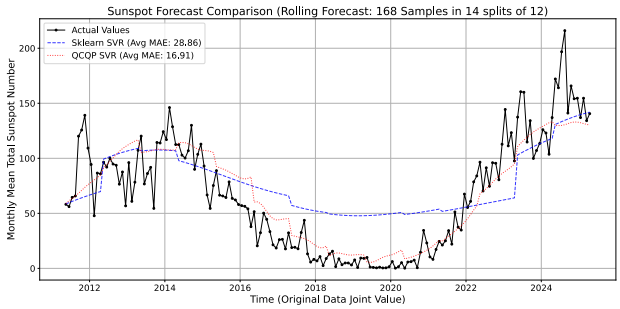

Figure 4.5: Sunspot dataset. Rolling Forecast 168 samples in 14 splits of 12

Model	C	ϵ	τ	γ	score
QCQP L1	2.3253	0.0396	6244.9230	–	16.9100
scikit-learn L1	1550.3079	47.7639	–	544.1141	28.8600

Table 4.10: Resulting Scores and Hyperparameters from the Optimization for the 12-Step Walk-Forward

The kernel obtained from the training process [13] is expressed as follows:

[13] This kernel was not used during the testing process shown in Figure 4.5; instead, the kernel weights were retrained for each step.

$$
\begin{aligned}
&\text{multikernel sunspot 12-Steps Walk-Forward} = (\\
&\quad 4.4417 \cdot \underbrace{\exp\left(\frac{-2\sin^2(\pi\|\mathbf{x}-\mathbf{x}'\|/132)}{0.1}\right)}_{\text{periodic}(p=132,\ell=0.1)} +\\
&\quad 0.66676 \cdot \underbrace{\exp\left(\frac{-2\sin^2(\pi\|\mathbf{x}-\mathbf{x}'\|/11)}{1}\right)}_{\text{periodic}(p=11,\ell=1)} +\\
&\quad 0.1709 \cdot \underbrace{\exp\left(\frac{-2\sin^2(\pi\|\mathbf{x}-\mathbf{x}'\|/11)}{1}\right)}_{\text{periodic}(p=11,\ell=0.1)} +\\
&\quad 0.0003 \cdot \underbrace{\exp\left(\frac{-2\sin^2(\pi\|\mathbf{x}-\mathbf{x}'\|/1)}{1}\right)}_{\text{periodic}(p=1,\ell=1)} +\\
&\quad 2.0990 \cdot \underbrace{\sigma_n^2\delta(\mathbf{x},\mathbf{x}')}_{\text{white kernel}} +\\
&)
\end{aligned}
$$

While this study focuses on kernel-based methods, other high-performing approaches have been reported in the literature. For instance, for long-horizon forecasting, [14] reported a MAE of 19.82 predicting 20 years in a single step using an ensemble of XGBoost and Deep Learning algorithms. For recursive one-step-ahead predictions, [15] achieved a MAE score of 1.186 with a Deep LSTM. These alternative methodologies and prediction windows fall outside the scope of the current work.

[14] Yuchen Dang and Ziqi Chen. A comparative study of non-deep learning, deep learning, and ensemble learning methods for sunspot number prediction, 2022

[15] S.B. Waykar, A.A. Rawool, S.K. Singh, and Vinay Farswan. Deep learning-based solar activity prediction using sunspot number and solar radio flux. *International Journal of Advanced Computer Science and Applications*, 13(10), 2022

4.2.4 μ - Support values analysis by τ value

It was observed that using the QCQP algorithm, predictions remained consistent across different values of τ. To explore this further, an experimental analysis was carried out. A model optimized to predict 12 data points was employed to examine the relationship between the kernel weights μ_i and the support values $(\alpha_k - \alpha_k^*) \in S$. For each τ value, 10 folds of 12 data points were predicted. From each fold, the median of μ and the maximum support value were determined. Then, the maximum of the previous μ vector and the mean of the obtained support values were calculated. The results are presented in Figure 4.6.

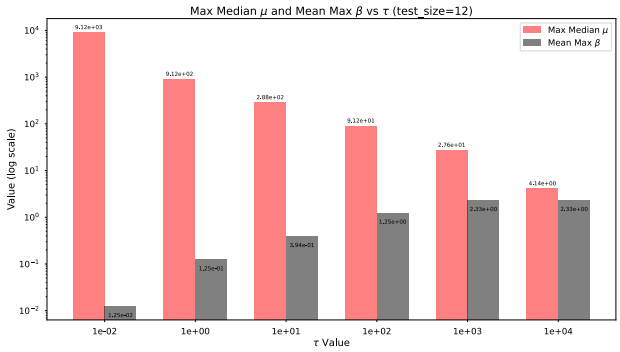

Figure 4.6: Log scaled maximum median μ and mean maximum Support Vector

The previous plot indicates that changes in the scale of μ are inversely related to those in the support values, leading to a *balancing effect*. As presented in Figure 6.3, this correlation might explain the minimal change in score as τ varies. Recall that new values are obtained through.

$$f(\mathbf{x}) = (\alpha - \alpha^*)^\top \sum_{i=0}^{M} \mu_i k_i(x, x) + b$$

4.3 Numerical Stability Comparison

In this section, we will examine the computational stability of two SVR QCQP algorithms to assess their computational stability in terms of multiple metrics. **trace** (3.7) corresponding to the precursory formulation where the performance metric is constrained by the trace of kernels, and **mu** 3.2.1 corresponding to the performance metric that is regularized by the sum of μ_i. For the SDP problem, we will also conduct a secondary analysis of the numerical stability concerning the filter of support values and unbounded support values.

4.3.1 QCQP formulations: Computational stability

Using the Sunspot dataset, as discussed in Section 4.2.1, a grid search was performed on two QCQP SVR L1 algorithms to assess computational stability between the previous algorithm and the newly derived one. The metrics included feasibility, numerical precision, and convergence time, with 540 iterations conducted for each algorithm.

Table 4.11 presents the performance results for both algorithms (**mu**, **trace**):

Metric	mu	trace
Min Target	40.1153	37.4741
Max Target	72.2176	32250.8159
Mean Target	55.3755	13386.3855
Total Non-Optimal Fits	0	33
Avg Comp Time (s)	232.7365	317.9482

Table 4.11: Comparison of computational stability between the SVR QCQP L1 algorithms, referred to here as **mu** (newly derived) and **trace** (former).

4.3.2 SDP formulations: Support Values analysis

Using a simulated dataset , we collected the indices of the support values (sv) and the unbounded support values (usv) after computing the optimal problem. The algorithm tested in this experiment was the SVR SDP multikernel, detailed in 3.1.1. The support values are filtered using the method described in section 2.3.7 (A), which represents the conventional approach, and the proposed method detailed in section 3.1.7 (B).

The simulated dataset was created with a linear trend and some random noise to generate the data points.

ϵ	C	SDP sv A	SDP sv B	QCQP sv	SDP usv A	SDP usv B	QCQP usv
0.1	1	92	95	95	100	95	95
0.1	10	92	95	95	100	95	95
0.1	100	92	95	95	100	95	95
1.0	1	50	51	53	100	51	53
1.0	10	50	51	53	100	51	53
1.0	100	50	51	53	100	51	53
10.0	1	12	3	3	100	3	3
10.0	10	31	3	3	100	3	3
10.0	100	21	3	3	100	3	3
100.0	1	11	2	2	100	2	2
100.0	10	9	2	2	100	2	2
100.0	100	11	2	2	100	2	2

Table 4.12: Comparison of the number of total support values and unbounded support values for the SDP problem. The QCQP is taken as a reference since it is computationally more stable than the SDP formulation.

5 Conclusions and future work

5.1 Conclusions

The results presented in Section 4.1 showed minimal improvement over the benchmark algorithms when adjusting the model specifically for horizon prediction in the Mauna Loa CO_2 dataset. This is likely due to the dataset containing sufficient information about the data's shape, allowing for effective modeling by existing benchmarks. It was also shown that there is a significant improvement when modeling using an initial state that contains some information about the data with varying weights, compared to the case where there are no varying weights, which is crucial for the process of automatic kernel design. Section 4.2 shows high sparsity on the automatic election of kernels and a significant modeling advantage over the single-kernel benchmark algorithm. The numerical stability comparison presented in Section 4.3 highlights two key findings. Firstly, the QCQP algorithms proposed in this work are not only computationally more stable than previously reported methods but also demonstrate improvements in speed, numerical precision, and solution feasibility. Secondly, the support value filtering method, when applied to SDP formulations, exhibits a significant numerical advantage, achieving a level of stability comparable to that of robust QCQP algorithms. The speed comparison reveals a notable advantage for the classical Vapnik's algorithm when compared to Multi-kernel algorithms in general. Furthermore, within the Multi-kernel approaches themselves, QCQP formulations are, as anticipated, significantly faster than SDP formulations (see [1]).
Overall, the newly derived algorithms provide more computationally stable variations compared to previous formulations. Combined with their ease of use, they constitute a valuable resource for researchers and practitioners in automatic regression multikernel design.

[1] Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004

5.2 Future work

Future work could extend this research in three main areas. First, the proposed algorithms should be benchmarked against modern methods,

such as those in [2] and [3]. Second, a deeper investigation into the trade-offs between the μ, τ, and C parameters, as discussed in section 4.2.4, would provide a better understanding of the SVR algorithms' properties. Finally, exploring alternative approaches, such as the conic formulation from section 2.1.7 or the simpleMKL algorithm [4], could significantly improve computational speed.

[2] Yuchen Dang and Ziqi Chen. A comparative study of non-deep learning, deep learning, and ensemble learning methods for sunspot number prediction, 2022

[3] S.B. Waykar, A.A. Rawool, S.K. Singh, and Vinay Farswan. Deep learning-based solar activity prediction using sunspot number and solar radio flux. *International Journal of Advanced Computer Science and Applications*, 13(10), 2022

[4] Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *J. Mach. Learn. Res.*, 9(Oct): 2491–2521, 2008. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1390681.1442775

6 Appendix

Contents

6.1 Code

The code used to test the results of this work can be found in https://github.com/Gegori1/Multikernel-SVR

6.2 Figures

6.2.1 Kernel weights analysis Mauna Loa CO_2

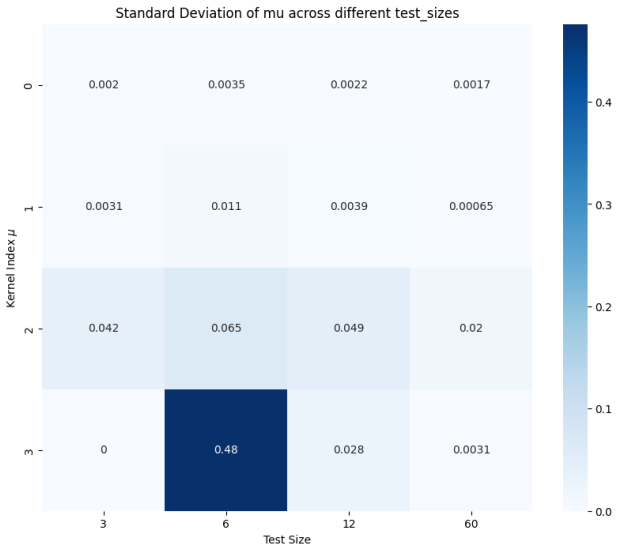

Figure 6.1: Kernel parameters standard deviation for optimal hyperparameters obtained from SVR L1 QCQP

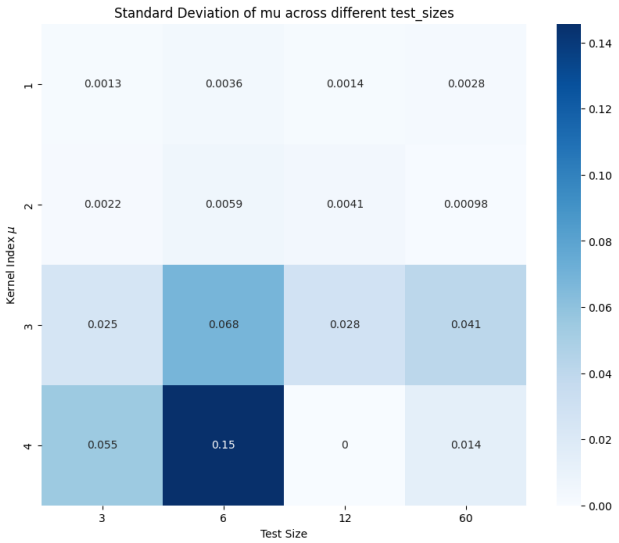

Figure 6.2: Kernel parameters standard deviation for optimal hyperparameters obtained from SVR L2 QCQP

6.2.2 *Hyperparameters analysis Mauna Loa CO_2*

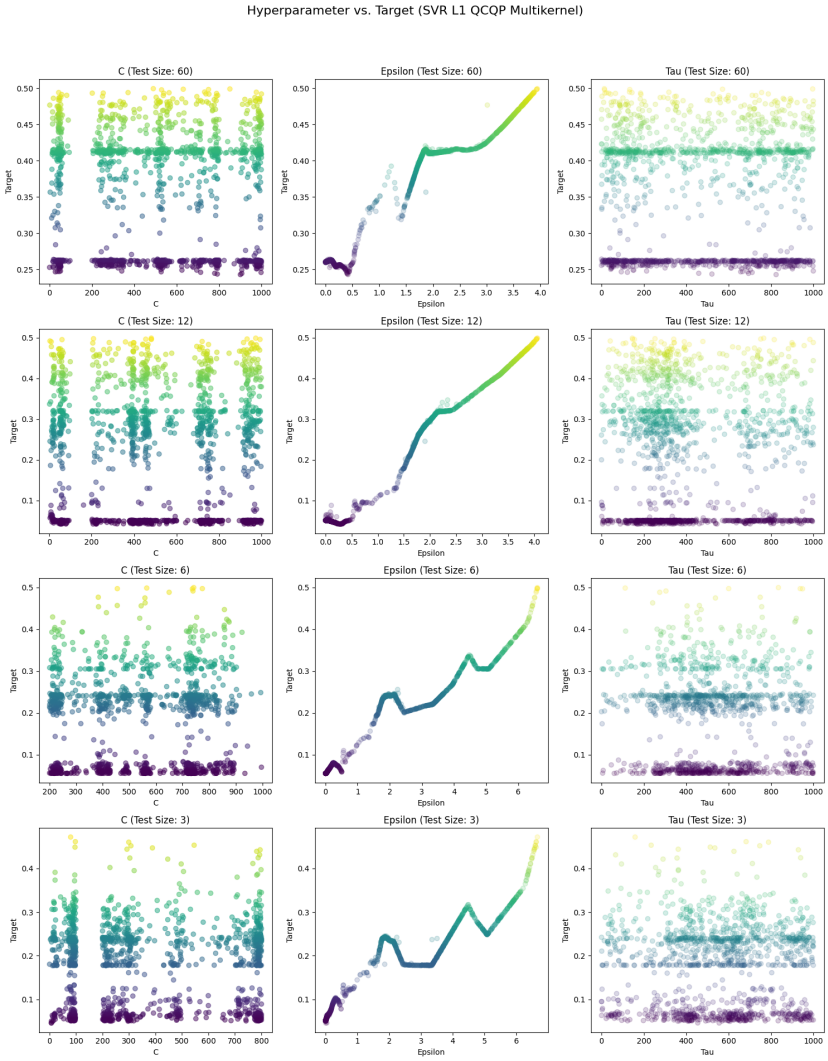

Figure 6.3: Resulting scores from hyperparameter optimization for SVR QCQP L1 at different test sizes (CO_2 dataset).

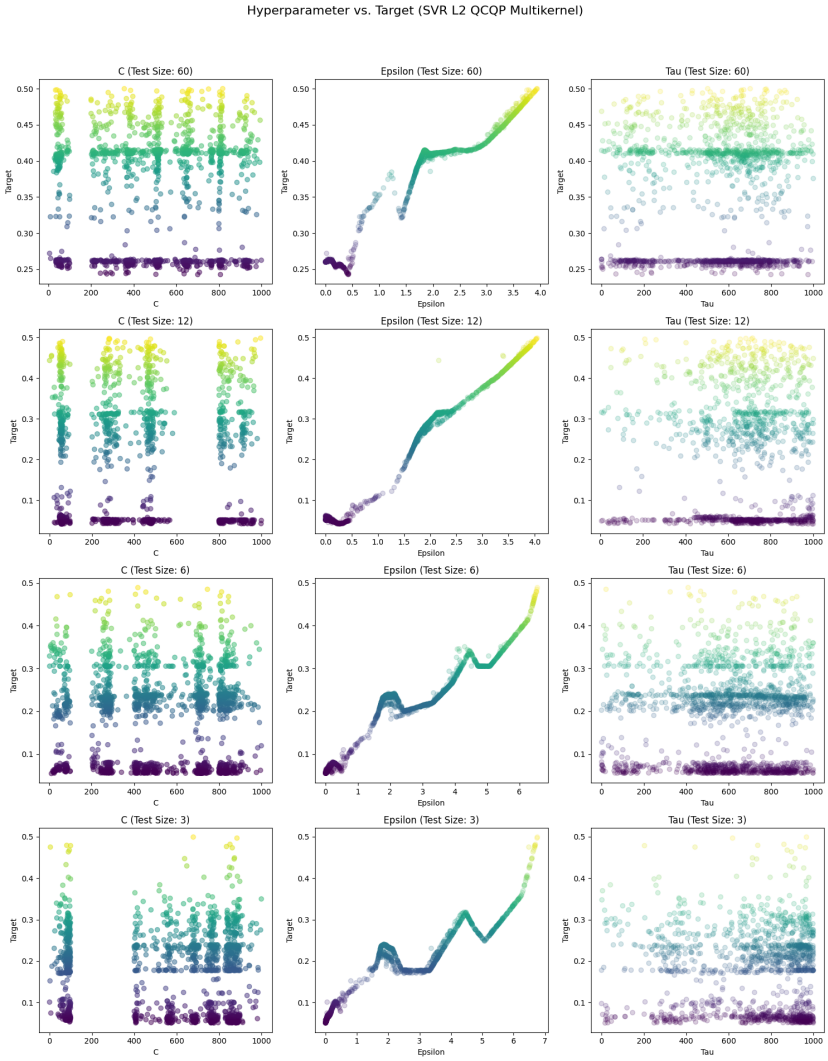

Figure 6.4: Resulting scores from hyperparameter optimization for SVR QCQP L2 at different test sizes (CO_2 dataset).

Bibliography

Optuna. URL https://optuna.org/.

sklearn.svm.svr. URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html.

Shigeo Abe. *Support Vector Machines for Pattern Classification*. Springer, 2nd edition, 2010. ISBN 978-1-84996-098-4.

Sakshi Aggarwal. Machine learning algorithms, perspectives, and real-world application: Empirical evidence from united states trade data. MPRA Paper 116579, Indian Institute of Foreign Trade, march 2023. URL https://mpra.ub.uni-muenchen.de/116579/. Posted 04 Mar 2023 09:21 UTC.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN 0-521-83378-7.

Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, Hillsdale, NJ, 2nd edition, 1988.

Yuchen Dang and Ziqi Chen. A comparative study of non-deep learning, deep learning, and ensemble learning methods for sunspot number prediction, 2022.

Hal Daumé III. From zero to reproducing kernel hilbert spaces in twelve pages or less. Technical report, UMIACS.

Scikit-Learn documentation. Gaussian process regression (gpr) with noise-level estimation. URL https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_co2.html.

Laurent El Ghaoui. Lecture 3: Convex sets and functions, 2012. URL https://people.eecs.berkeley.edu/~elghaoui/Teaching/EE227A/lecture3.pdf. Accessed on July 14, 2025.

Robert M. Freund. Introduction to semidefinite programming (sdp). Technical report, Massachusetts Institute of Technology, March 2004.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011. URL https://jmlr.csail.mit.edu/papers/volume12/gonen11a/gonen11a.pdf.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2nd edition, 2002. ISBN 0-89871-521-0.

Wolfgang Härdle, Joel L. Horowitz, and Jens-Peter Kreiss. Bootstrap methods for time series, August 2001. URL https://nbn-resolving.de/urn:nbn:de:kobv:11-10050152.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer Texts in Statistics. Springer New York, 2013. ISBN 978-1-4614-7137-0. DOI: 10.1007/978-1-4614-7138-7. URL http://doi.org/10.1007/978-1-4614-7138-7.

Gert R.G. Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5: 27–72, 2004.

Bernard Lemaire. Lagrangian relaxation. Technical report, ENS de Lyon, January 2012. URL https://www.ens-lyon.fr/DI/wp-content/uploads/2012/01/LagrangianRelax.pdf.

Guillermo Alexander Loayza-Delgado, Xiomara Luciana Tejada-Montalvo, María Fernanda Carnero-Quispe, and Christian Frederick Gárate-Rodríguez. Machine learning in industrialization: a bibliometric analysis. *DYNA*, 92(235):28–37, january-march 2025. ISSN 0012-7353.

James G. MacKinnon. Bootstrap Methods in Econometrics, February 2006. URL http://www.econ.queensu.ca/faculty/mackinnon/. Revised, June, 2006.

Mikhail Nedbai. 4.1 introduction 4.2 max cut local search algorithm. Lecture 4: Local Search, CS880: Approximation and Online Algorithms.

NOAA / NWS Space Weather Prediction Center. Sunspots/solar cycle, May 2025. URL https://www.swpc.noaa.gov/phenomena/sunspotssolar-cycle. Accessed on 2025-05-05.

Alain Rakotomamonjy, Francis R. Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *J. Mach. Learn. Res.*, 9(Oct):2491–2521, 2008. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1390681.1442775.

Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL https://gaussianprocess.org/gpml/chapters/RW.pdf.

Carsten Scherer and Siep Weiland. Linear matrix inequalities in control. Technical report, University of Stuttgart, 2015. URL https://www.imng.uni-stuttgart.de/mst/files/LectureNotes.pdf.

Solar Influences Data Analysis Center (SIDC). Sunspot number, 2025. URL https://www.sidc.be/SILSO/datafiles. Accessed on 2025-05-04. Data series v2.0 implemented 2015-07-01.

statsmodel documentation. statsmodels.tsa.seasonal.stl. URL https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.STL.html.

Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998. ISBN 0-471-03003-1.

S.B. Waykar, A.A. Rawool, S.K. Singh, and Vinay Farswan. Deep learning-based solar activity prediction using sunspot number and solar radio flux. *International Journal of Advanced Computer Science and Applications*, 13(10), 2022.