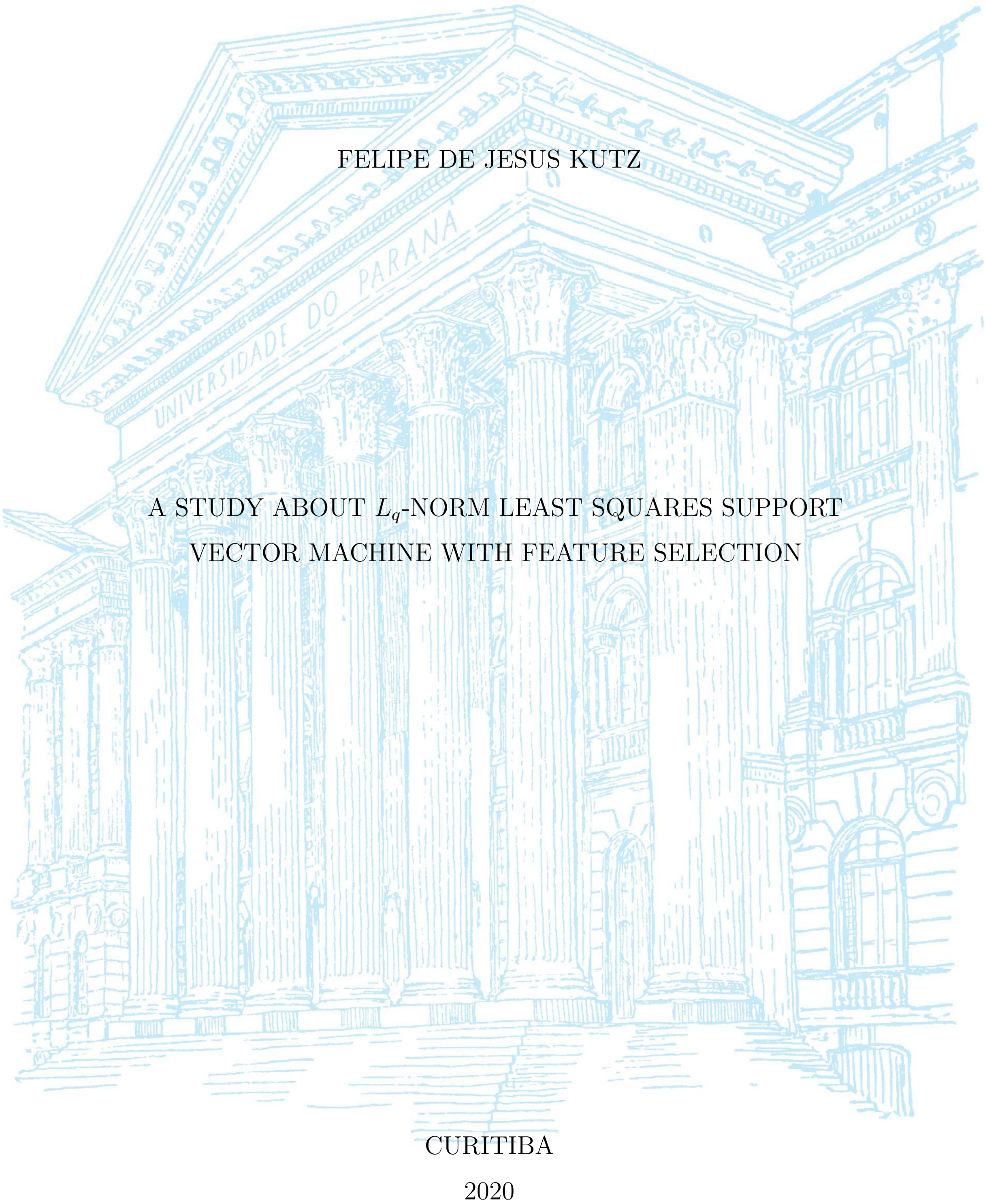
UNIVERSIDADE FEDERAL DO PARANÁ

FELIPE DE JESUS KUTZ

A STUDY ABOUT L_q-NORM LEAST SQUARES SUPPORT VECTOR MACHINE WITH FEATURE SELECTION

CURITIBA

2020



FELIPE DE JESUS KUTZ

A STUDY ABOUT L_q-NORM LEAST SQUARES SUPPORT VECTOR MACHINE WITH FEATURE SELECTION

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Matemática, no Curso de Pós-Graduação em Matemática, Setor de Ciências Exatas, da Universidade Federal do Paraná.
Orientador: Prof. Dr. Jose Alberto Ramos Flor

CURITIBA
2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

K97s Kutz, Felipe de Jesus

A study about L_q-norm Least Squares Support Vector Machine with feature selection [recurso eletrônico] / Felipe de Jesus Kutz – Curitiba, 2020.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Matemática.

Orientador: Prof. Dr. José Alberto Ramos Flor

1. Aprendizagem supervisionada. 2. Lq-norm Least Squares Support Vector Machine. 3. Problemas de classificação. I. Universidade Federal do Paraná. II. Flor, José Alberto Ramos. III. Título.

CDD: 519.5

Bibliotecária: Roseny Rivelini Morciani CRB-9/1585

ATA Nº99

ATA DE SESSÃO PÚBLICA DE DEFESA DE MESTRADO PARA A OBTENÇÃO DO GRAU DE MESTRE EM MATEMÁTICA

No dia vinte e nove de outubro de dois mil e vinte às 10:00 horas, na sala virtual, https://meet.google.com/qiy-cqoe-etp, remoto, foram instaladas as atividades pertinentes ao rito de defesa de dissertação do mestrando **FELIPE DE JESUS KUTZ**, intitulada**: A STUDY ABOUT Lq-NORM LEAST SQUARES SUPPORT VECTOR MACHINE WITH FEATURE SELECTION**, sob orientação do Prof. Dr. JOSÉ ALBERTO RAMOS FLOR. A Banca Examinadora, designada pelo Colegiado do Programa de Pós-Graduação em MATEMÁTICA da Universidade Federal do Paraná, foi constituída pelos seguintes Membros: JOSÉ ALBERTO RAMOS FLOR (UNIVERSIDADE FEDERAL DO PARANÁ), LEONARDO DELARMELINA SECCHIN (UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO), DAVID MENOTTI GOMES (UNIVERSIDADE FEDERAL DO PARANÁ), PAULO J. SILVA E SILVA (UNIVERSIDADE ESTADUAL DE CAMPINAS). A presidência iniciou os ritos definidos pelo Colegiado do Programa e, após exarados os pareceres dos membros do comitê examinador e da respectiva contra argumentação, ocorreu a leitura do parecer final da banca examinadora, que decidiu pela APROVAÇÃO. Este resultado deverá ser homologado pelo Colegiado do programa, mediante o atendimento de todas as indicações e correções solicitadas pela banca dentro dos prazos regimentais definidos pelo programa. A outorga de título de mestre está condicionada ao atendimento de todos os requisitos e prazos determinados no regimento do Programa de Pós-Graduação. Nada mais havendo a tratar a presidência deu por encerrada a sessão, da qual eu, JOSÉ ALBERTO RAMOS FLOR, lavrei a presente ata, que vai assinada por mim e pelos demais membros da Comissão Examinadora.

CURITIBA, 29 de Outubro de 2020.

Assinatura Eletrônica
05/11/2020 09:14:17.0
JOSÉ ALBERTO RAMOS FLOR
Presidente da Banca Examinadora

Assinatura Eletrônica
29/10/2020 15:46:54.0
LEONARDO DELARMELINA SECCHIN
Avaliador Externo (UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO)

Assinatura Eletrônica
29/10/2020 16:25:50.0
DAVID MENOTTI GOMES
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
30/10/2020 09:10:47.0
PAULO J. SILVA E SILVA
Avaliador Externo (UNIVERSIDADE ESTADUAL DE CAMPINAS)

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em MATEMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **FELIPE DE JESUS KUTZ** intitulada: **A STUDY ABOUT Lq-NORM LEAST SQUARES SUPPORT VECTOR MACHINE WITH FEATURE SELECTION**, sob orientação do Prof. Dr. JOSÉ ALBERTO RAMOS FLOR, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.
A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 29 de Outubro de 2020.

Assinatura Eletrônica
05/11/2020 09:14:17.0
JOSÉ ALBERTO RAMOS FLOR
Presidente da Banca Examinadora

Assinatura Eletrônica
29/10/2020 15:46:54.0
LEONARDO DELARMELINA SECCHIN
Avaliador Externo (UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO)

Assinatura Eletrônica
29/10/2020 16:25:50.0
DAVID MENOTTI GOMES
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
30/10/2020 09:10:47.0
PAULO J. SILVA E SILVA
Avaliador Externo (UNIVERSIDADE ESTADUAL DE CAMPINAS)

Dedico essa dissertação aos meus pais Adelson(In Memoriam) e Terezinha também a minha esposa Maria Alice

Agradecimentos

Aos meus pais Adelson Kutz e Terezinha Clara de Jesus, pelo carinho e amor, por terem me proporcionado a condição de poder chegar aqui, por sempre acreditarem em mim.
À minha esposa Maria Alice dos Santos pelo seu companherismo, carinho e me apoiar em todos os meus desafios.
Ao meu orientador Professor José Alberto Ramos Flor, por todas as lições, conselhos e motivação para realizar este trabalho.
Aos professores David Menotti Gomes, Paulo J. Silva e Silva e Leonardo Delarmelina Secchin por terem aceitado participar da banca examinadora e pelas sugestões que me ajudaram a melhorar a versão final deste trabalho.
Aos meus amigos que me ajudaram nessa jornada.
À Coordenação de Aperfeicoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro.

"While the individual man is an insoluble puzzle, in the aggregate he becomes a mathematical certainty. You can, for example, never foretell what any one man will do, but you can say with precision what an average number will be up to. Individuals vary, but the percentages remain constant"

Arthur Conan Doyle

RESUMO

Neste trabalho são estudados os problemas de classificação binária, nos quais há um número m de observações em um espaço de n dimensões e $n \geq m$ ou $n \gg m$. No último caso o problema é denominado Amostras de Tamanho Pequeno (SSS, do inglês *Small Sample Size*). Nesse caso, o problema de classificação se torna mais difícil, pois os algoritimos podem sofrer de um fenômeno chamado de maldição da dimensionalidade. Tal fenômeno faz com que o classificador tenha um baixo número de acertos. Também pode ocorrer um problema chamado de sobreajuste, isto é, quando o classificador consegue classificar bem os dados que foram utilizados para construí-lo, no entanto, não apresenta uma boa taxa de acertos para novos dados. Uma técnica utilizada para espaços com muitas características é a seleção de características, que pode melhorar a taxa de acertos de um classificador, além de fornecer um melhor entendimento dos dados que estão sendo classificados. O modelo estudado neste trabalho é uma variante de *Least Squares Support Vector Machine* (LS-SVM) com o *Kernel* linear, chamado de L_q*-norm Least Squares Support Vector Machine* (L_q-norm LS-SVM), que constrói um classificador capaz de realizar a seleção de características e predição de dados simultaneamente, mesmo no caso SSS.

Palavras-chave: Aprendizagem supervisionada. Classificação. Lq-norm Least Squares Support Vector Machine. Seleção de Características

ABSTRACT

In this work we study the binary classification problems, in which we have a number m of data observations in a space with dimension n and we have that $n \geq m$ or $n \gg m$. In the last case the problem is called Small Sample Size (SSS). In this case, the classification problem becomes more difficult, since the algorithms can suffer from a phenomenon called the curse of dimensionality. This makes the classifier have a low number of correct answers, also can occur a problem called overfitting, that is, when the classifier is able to classify correctly the data that were used to build it, however, it does not present a good rate of correct predictions for new data. A technique used for spaces with many characteristics is known as feature selection, which can improve the ability of a classifier to predict correctly new data, avoid overfitting and also provide us a better understanding of the data. The model that we study in this work is a variant of *Least Squares Support Vector Machine* (LS-SVM) with linear Kernel, called *L_q-norm Least Squares Support Vector Machine* (L_q-norm LS-SVM), which builds a classifier able to perform feature selection and prediction, even in SSS case.

Keywords: Supervised learning. Classification. Lq-norm Least Squares Support Vector Machine. Feature selection.

LIST OF FIGURES

LIST OF TABLES

NOTATIONS

$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	n-dimensional euclidean space
e	column vector $(1, 1, \dots, 1)^T \in \mathbb{R}^n$
$(x_j)_{1 \leq j \leq n}$	column vector $(x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$
x^k	k-th input column vector
x_j	j-th component of $x \in \mathbb{R}^n$
$u^{(k)}$	k-th iterate
$int(S)$	Interior of a set S
$\|\cdot\|$	Euclidean norm
$\lvert\cdot\rvert$	The absolute value
$\|x\|_1$	$\sum_{i=1}^{n} \lvert x_i \rvert$, for $x \in \mathbb{R}^n$
$x^T y = \sum_{i=1}^{n} x_i y_i$	Inner product between $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$
$\lvert S \rvert$	Number of elements of set S
$d(x, S)$	Distance between a point $x \in R^n$ and a set $S \subset \mathbb{R}^n$
$\dfrac{\partial f}{\partial x}$	Partial derivative of the function f with respect to x
$\nabla f(x)$	Gradient of the function f at point x
I_n	Identity matrix of order n
$\{u^k\}_k$	A sequence in $\mathbb{R}^n$, where $u^k \in \mathbb{R}^n$ and k represents the sequence index
$\{u^{k_j}\}_j$	A subsequence in $\mathbb{R}^n$, where $u^{k_j} \in \mathbb{R}^n$ and j represents the subsequence index
$(x)_+$	$\max(x, 0)$ for $x \in \mathbb{R}^n$

CONTENTS

INTRODUCTION

In Machine Learning, there is a well known class of problems called *supervised learning problems* [3, 19, 25]. In these problems, given a data set $S = \{(x^k, y_k)\}_{k=1}^m$, called *training data*, in which $x^k \in X \subset \mathbb{R}^n$ is called *instance* or *input vector* and $y_k \in Y \subset \mathbb{R}$ is respective *label*, (also called *output value*), the goal is to determine the function $f : X \to Y$ which is able to predict correctly the label of a new instance. The set $X \subset \mathbb{R}^n$ is called *domain*, meanwhile $Y \subset \mathbb{R}$ is the *label set*, which is the set of possible labels, each pair (x^k, y_k) is called *observation*, each component $x_i^k \in \mathbb{R}$ of $x^k \in \mathbb{R}^n$ is called *feature*, and the function $f : X \to Y$ is called *predictor*.

The supervised learning problems have two main branches: Regression problems and Classification problems.

When y takes continuous values, we have a regression problem. The goal is to understand the data pattern to construct a predictor that we can use to predict a quantity. For example, we want to predict the price of a house in a neighborhood based on house area and number of the rooms. Note that the domain set X is in $\mathbb{R}^2$ and the label set $Y \subset \mathbb{R}$.

When y takes only discrete values, we have a classification problem. The aim is to define a predictor which we use to predict a label/class. For example, suppose that we have a data set composed by clinical measurements (e.g. weight, blood pressure, height, age, family history of disease) for a number of patients, as well as information about whether each patient has diabetes. Based on this data, we can train/build a predictor to predict the risk of diabetes that can be low, moderate or high. The domain set here is X, a subset of $\mathbb{R}^5$ and the label set Y is a categorical set given by $\{low, moderate, high\}$, we can replace for the discrete set $\{1, 2, 3\}$. In general, in multiclass classification problems the set Y is defined as $Y = \{1, \ldots, p\}$ where $p \in \mathbb{N}$, is the number of classes, however, in binary classification problems for convenience we consider $Y = \{-1, +1\}$.

There are several techniques to determine a classifier such as, Logistic Regression, Linear Discriminant Analysis, Tree Based Methods for classification, etc [3, 7, 10]. The models that we study in this dissertation generate classifiers based on hyperplanes, due this fact they are called *Linear Classifiers*, (see [7]).

In some applications the number of observations m is much smaller than the number of feature n, i.e., $m \ll n$. We mention for example the data collected from *gene-expression microarrays* [18] that consists of thousands of genes expressions that constitute features with a limited number of observations; Satellite Imagery [24] captures high quality hyperspectral images used for natural resources. In these situations many classification models do not work well [7, 10, 22, 30]. For example, the model can suffers of a phenomenon called the *Curse of dimensionality* [22, 30]. In practice, it implies that for every data set, there exists a number $N \in \mathbb{N}$, such that, if we use $K \in \mathbb{N}$ features of the data set to build a classifier the number of correct predictions will be small for all $K > N$. Also the classifier can suffers from *overfitting*, that is, when the model yields a classifier which has an excellent ability to predict correctly the training data, however, the model is not good to predict unseen data called *test data* or *validation data*.

In [22], we see that a common way to deal with classification problems in spaces with many features is to use a technique called *Feature Selection*, which aims to select a small subset of features which are relevant to the model. This technique improve the number of correct predictions for some models as mentioned in [22], reduces the computational complexity to generate a classifier. And lastly help us to understand better the data that we use to build the classifier, in other words, what features are really important to classification.

The main contributions of this dissertation is to present a detailed study about the main properties of L_q-norm Least Squares Support Vector Machine (L_q-norm LS-SVM) with feature selection, proposed in [26]. We reformulated some theorems and proofs in order to make them clearer and more understandable, correct some wrong affirmations and reproduce the numerical experiments present in original study.

Our work is organized as follows: In Chapter 1 We will study Support Vector Machine (SVM). Our main references for SVM are [3, 5, 13, 16, 23]. We will also cover some key definitions and theorems, and will lastly examine Least Squares Support Vector Machine (LS-SVM). Our references are [1, 29].

In Chapter 2, we will present the main topic of this work, that is, our study about L_q-norm Least Squares Support Vector Machine with feature selection proposed in [26]. We will start by explaining how to perform feature selection, then we will describe the L_q-norm LS-SVM algorithm and at last we will study its convergence results.

Finally, in Chapter 3, we will present our empirical study where we will describe our methodology to test the models that we will study in this dissertation, also we present the data sets which we divides into two groups Artificial and Real World data sets. At the end we will show our numerical experiments that consists in comparisons among SVM based models and next we show our L_q-norm LS-SVM numerical tests, with respect to accuracy, features selection, convergence and sparsity.

Chapter 1

SUPPORT VECTOR MACHINES AND LEAST SQUARES SUPPORT VECTOR MACHINES

In this chapter we will review some basic concepts about Support Vector Machine (SVM) and Least Square Support Vector Machine (LS-SVM).

Both models construct classifiers based on hyperplanes. In binary classification problem, by using the observed data, such models seek for a hyperplane (by some criteria that we will outline later) and with this hyperplane, we define a classifier. To make our approach clear, suppose that the method provides a hyperplane of the form

$$\mathcal{H} = \{x \in \mathbb{R}^n \mid w^T x + b = 0\}, \text{ for } w \neq 0$$

Note that the hyperplane $\mathcal{H}$ separates the space $\mathbb{R}^n$ into two sets, $\mathcal{H}_- = \{x \in \mathbb{R}^n \mid w^T x + b \leq 0\}$ and $\mathcal{H}_+ = \{x \in \mathbb{R}^n \mid w^T x + b \geq 0\}$. We will use this observation to determine a classifier.

Suppose for a moment that $\mathcal{H}$ separates the observed data correctly, that is, every point in the observed data belonging to the class $\{+1\}$ is in the interior of $\mathcal{H}_+$, and $\{-1\}$ is in the interior of $\mathcal{H}_-$. Thus, for a new input data $x \in \mathbb{R}^n$, we define a classifier $y(x)$ as follows

$$y(x) = sign(w^T x + b), \tag{1.1}$$

where *sign* is the signal function, $sign(a) = 1$ if $a > 0$ and $sign(a) = -1$ if $a < 0$.

Note that $y(x)$ classifies correctly the previously observed data, since by definition of $y(x)$, we see that $y(x) = 1$ for every $x \in int(\mathcal{H}_+)$ and $y(x) = -1$ for every $x \in int(\mathcal{H}_-)$.

When x belongs to the hyperplane $\mathcal{H}$, we cannot decide (by using $y(x)$) the class associated to the new input data x. The set in the input space, where we are not able to associate a corresponding class is called of *Decision Boundary.* In this case, note that the Decision Boundary is the hyperplane $\mathcal{H}$ which is defined by an affine transformation. This is the reason why we say that $y(x)$ is a linear classifier although $y(x)$ is not linear.

1.1 Support Vector Machines (SVMs)

Support Vector Machine (SVM) is a model used to generate a hyperplane in order to separate the data into two distinct classes. We separate our discussion about SVMs into two cases: SVM with hard margin and SVM with soft margin.

1.1.1 Support Vector Machines - Hard Margin

In this subsection we explain the SVM Hard Margin approach that generates a classifier for linearly separable data, but first we need some basic definitions.

Definition 1.1. *We say that two sets X_1, $X_2 \subset \mathbb{R}^n$ are linearly separable when there exists $w \in \mathbb{R}^n \setminus \{0\}$ and $b \in \mathbb{R}$, such that, $w^T x + b > 0$ for every $x \in X_1$ and $w^T x + b < 0$ for every $x \in X_2$. The hyperplane $\mathcal{H} = \{x \,|\, w^T x + b = 0\}$ is called separating hyperplane.*

Observe that every separating hyperplane defines a classifier as follows

$$y(x) = \begin{cases} +1, & \text{if } w^T x + b > 0. \\ -1, & \text{if } w^T x + b < 0, \end{cases}$$

which is equivalent to

$$y(x) = sign(w^T x + b).$$

Now the task is how to determine a separating hyperplane since it might exist an infinity quantity of hyperplanes, as the next figure shows.

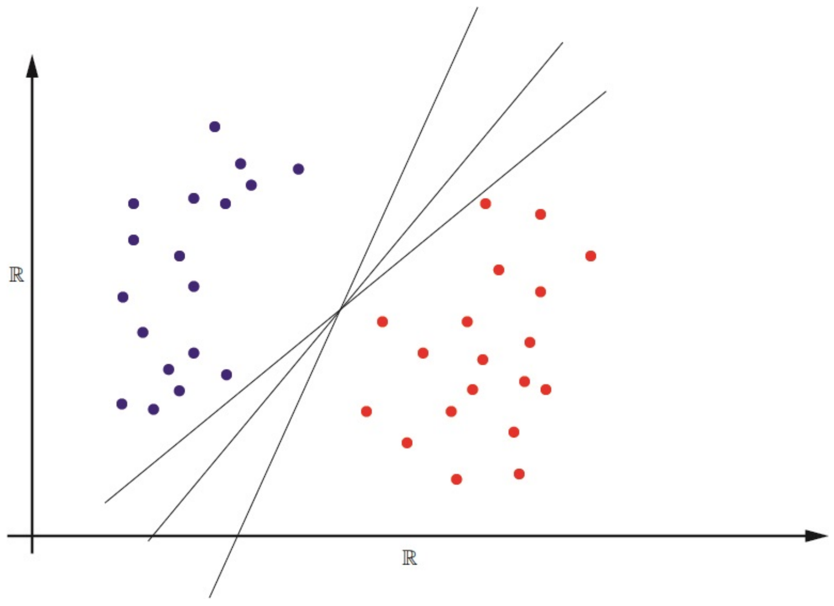


Figure 1.1: Two linearly separable sets in $\mathbb{R}^2$. We observe that might exists an infinity quantity of separating hyperplanes.

From Figure 1.1, we note the existence of infinite hyperplanes that separate correctly the data. Thus, we need some criterion to choose an adequate hyperplane. Here, we consider a geometric approach based on the idea of *margin*.

Definition 1.2. *Given two linearly separable finite sets $X_1 , X_2 \subset \mathbb{R}^n$, the margin $\mathcal{M}$ of a separating hyperplane $\mathcal{H} = \{x \in \mathbb{R}^n \,|\, w^T x + b = 0\}$ is the value defined by*

$$\mathcal{M} := \min d(z, \mathcal{H}) \text{ where } z \in X_1 \cup X_2 \subset \mathbb{R}^n.$$

We say that a point $x \in \mathbb{R}^n$ is on margin if $\mathcal{M} = d(x, \mathcal{H})$.

Proposition 1.1. *Let X_1, $X_2 \subset \mathbb{R}^n$ two linearly separable finite sets defined as in Definition 1.1, such that, $\mathcal{H} = \{x \,|\, w^T x + b = 0\}$ is the separating hyperplane. Then there exists $\tilde{w}$, $\tilde{b}$ such that $\mathcal{H} = \{x \,|\, \tilde{w}^T x + \tilde{b} = 0\}$, $\tilde{w}^T x + \tilde{b} \geq 1$ for $x \in X_1$, and $\tilde{w}^T x + \tilde{b} \leq -1$ for $x \in X_2$.*

In this case, we say that $\tilde{w}$ and $\tilde{b}$ normalize $\mathcal{H}$ or $\mathcal{H}$ is normalized by $(\tilde{w}, \tilde{b})$.

Proof: Since $X_1 \cup X_2$ is a finite set we can define $\zeta := \min\limits_{x \in X_1 \cup X_2} |w^T x + b|$, then we have that, for $x \in X_1 \cup X_2$, $\zeta \leq |w^T x + b|$ and since X is linearly separable we get $\zeta > 0$, otherwise $\mathcal{H}$ would not be the separating hyperplane. Thus, $\dfrac{|w^T x + b|}{\zeta} \geq 1$, therefore, for $x \in X_1$ we obtain

$$\frac{w^T x + b}{\zeta} = \frac{|w^T x + b|}{\zeta} \geq 1$$

and for $x \in X_2$ we get

$$-\frac{w^T x + b}{\zeta} = \frac{|w^T x + b|}{\zeta} \geq 1.$$

Now if we take $\tilde{w} = \dfrac{w}{\zeta}$ and $\tilde{b} = \dfrac{b}{\zeta}$, then we have equality $\mathcal{H} = \{x \,|\, w^T x + b = 0\} = \{x \,|\, \tilde{w}^T x + \tilde{b} = 0\}$. □

It is not difficult to see that the distance of a point x in $\mathbb{R}^n$ to the hyperplane $\mathcal{H}$ is given by the expression(see [14]).

$$d(x, \mathcal{H}) = \frac{|w^T x + b|}{\|w\|},$$

Now by Proposition 1.1, assuming that $\mathcal{H} = \{x \in \mathbb{R}^n \,|\, w^T x + b = 0\}$ is normalized, then

$$d(x, \mathcal{H}) = \frac{|w^T x + b|}{\|w\|} = \frac{1}{\|w\|}. \tag{1.2}$$

for every x on the margin.

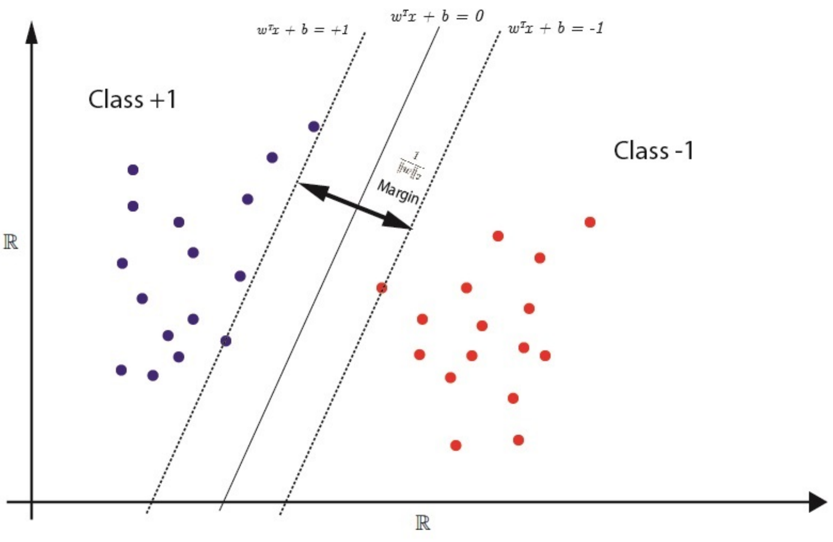


Figure 1.2: Geometrical interpretation of a linearly separable set and its optimal separating hyperplane $\mathcal{H} = \{x \in \mathbb{R}^2 \,|\, w^T x + b = 0\}$.

From Definition 1.2 and the equation (1.2), the separating hyperplane margin is given by $\dfrac{1}{\|w\|}$. The SVM model looks for the separating hyperplane $\mathcal{H}$ which has the maximal margin, that is, we want to maximize $1/\|w\|$ over the all vectors w such that there exists $b > 0$ with (w, b) defining a separating hyperplane as the statement of Proposition 1.1. Note that we can write the inequalities $w^T x + b \geq 1$ for $x \in X_1$, and $w^T x + b \leq -1$ for $x \in X_2$, in a compact form,

$$y(w^T x + b) \geq 1,$$

where $y = sign(w^T x + b)$. Therefore the problem to find the optimal hyperplane is given by

$$\begin{array}{ll} \min\limits_{w,b} & \dfrac{1}{2}\|w\|^2 \\ \text{s.t} & y_k(w^T x^k + b) \geq 1, \quad k = 1, \dots, m \end{array} \tag{1.3}$$

in which $w \in \mathbb{R}^n$, $b \in \mathbb{R}$ and y_k is the known output for x^k.

The associated Lagrangian function is defined as

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{k=1}^{m} \alpha_k (y_k(w^T x^k + b) - 1).$$

Now we will determine the dual optimization problem associated to (1.3) which can be useful to determine the classifier. The dual form of (1.3) is given the following problem

$$\begin{array}{ll} \max\limits_{\alpha} & \inf\limits_{w,b} \mathcal{L}(w, b, \alpha) \\ \text{s.t} & \alpha \geq 0. \end{array}$$

For more information about the dual problem, see the books [2, 21].

To evaluate $\inf_{w,b} \mathcal{L}(w,b,\alpha)$ we consider two cases: If $\sum \alpha_k y_k = 0$ and if $\sum \alpha_k y_k \neq 0$. Note that, if $\sum\limits_{k=1}^{m} \alpha_k y_k = 0$ take $\bar{w} = \sum\limits_{k=1}^{m} \alpha_k y_k x^k$ and $\bar{b} \in \mathbb{R}$, then by (1.5) we have

$$\begin{aligned} \inf_{w,b} \mathcal{L}(w,b,\alpha) &= \mathcal{L}(\bar{w},\bar{b},\alpha) \\ &= \frac{1}{2}\sum_{k,j=1}^{m} \alpha_k \alpha_j y_k y_j \left(x^k\right)^T x^j - \sum_{k=1}^{m} \alpha_k y_k \left(\sum_{j=1}^{m} \alpha_j y^j x^j\right)^T x^k - \sum_{k=1}^{m} \alpha_k y_k \bar{b} + \sum_{k=1}^{m} \alpha_k \\ &= -\frac{1}{2}\sum_{k,j=1}^{m} \alpha_k \alpha_j y_k y_j \left(x^k\right)^T x^j + \sum_{k=1}^{m} \alpha_k. \end{aligned} \tag{1.4}$$

Now observe that for α fixed, the function $\mathcal{L}(w,b,\alpha)$ is convex, then the minimizers $(\bar{w},\bar{b})$ of $\mathcal{L}(w,b,\alpha)$ with respect to w and b satisfies

$$\nabla_{(w,b)}\mathcal{L}(\bar{w},\bar{b},\alpha) = 0,$$

which is equivalent to

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow \bar{w} = \displaystyle\sum_{k=1}^{m} \alpha_k y_k x^k \\ \dfrac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \displaystyle\sum_{k=1}^{m} \alpha_k y_k = 0 \end{cases}. \tag{1.5}$$

Lets analyze the case when $\sum\limits_{k=1}^{m} \alpha_k y_k \neq 0$, take $b_i = i\left(\sum\limits_{k=1}^{m} \alpha_k y_k\right)$, $i \in \mathbb{N}$, we have

$$\mathcal{L}\left(0, b_i, \alpha\right) = \sum_{k=1}^{m} \left(\alpha_k - \alpha_k y_k b_i\right) = \sum_{k=1}^{m} \alpha_k - i\left(\sum_{k=1}^{m} \alpha_k y_k\right)^2,$$

then, we get

$$\inf_{w,b} \mathcal{L}(w,b,\alpha) = -\infty. \tag{1.6}$$

From equations (1.4) and (1.6), we have that the dual problem is given by

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\sum_{k,j=1}^{m} \alpha_k \alpha_j y_i y_j \left(x^k\right)^T x^j + \sum_{k=1}^{m} \alpha_k \\ \text{s.t} \quad & \sum_{k=1}^{m} \alpha_k y_k = 0, \\ & \alpha_k \geq 0, \quad k = 1,\ldots,m. \end{aligned} \tag{1.7}$$

Now, observe that function in (1.3) is convex and the constraints are linear, then the Karush-Kuhn-Tucker (KKT) conditions hold for the problem (1.3). In order to determine the classifier (1.1), we can solve (1.7). In fact, if α^* a solution of (1.7), we can obtain w^* solution of (1.3) from system (1.5), that is, $w^* = \sum\limits_{k=1}^{m} \alpha_k^* y_k x^k$ and in order to obtain b^* we can use KKT complementarity condition

$$\alpha_k^*[y_k((w^*)^T x^k + b^*) - 1] = 0. \tag{1.8}$$

Indeed, consider $S = \{k \,|\, \alpha_k^* > 0\}$, then for each $k \in S$, after multiply (1.8) by y_k (note that y_k^2), we see that

$$b^* = y_k - (w^*)^T x^k = y_k - \sum_{j \in S} \alpha_j^* y_j (x^j)^T x^k.$$

Thus, the linear classifier (1.1) is given by

$$y(x) = \sum_{k \in S} \alpha_k^* y_k x^k + b^*.$$

Observe that only positive components of solution $\alpha^* \in \mathbb{R}^m$ are relevant to define the classifier.

Definition 1.3. *A vector $x^k \in X \subset \mathbb{R}^n$ is called a Support Vector when $\alpha_k^* > 0$, where $\alpha^* \in \mathbb{R}^m$ is a solution of dual problem* (1.7).

Remark 1.1. Observe that if x^k is a support vector, then $y_k((w)^T x^k + b) = 1$, thus we have that, $x^k \in \mathcal{H}_1 = \{x \in \mathbb{R}^n \,|\, w^T x + b = +1\}$ or $x^k \in \mathcal{H}_2 = \{x \in \mathbb{R}^n \,|\, w^T x + b = -1\}$. The Figure 1.2 gives us the intuition that a small number of points will belong to $\mathcal{H}_1$ and $\mathcal{H}_2$. Then, we should expect a sparse solution α^* of problem (1.7).

1.1.2 Support Vector Machines - Soft Margin

In many real applications, the data is not linearly separable. For example, consider the following image recognition problem. Given a set of handwriting letters and we want to define a classifier that predicts if a new handwriting letter is the character “A” or not. The training data was constructed by a person who saw every letter and then labeled it as letter “A” (positive class $+1$) and not “A” (negative class -1). In this case probably there will be mislabeled data, due bad calligraphy or inattention. Therefore, the data set will probably not be linearly separable, since there will be some characters “A” classified as not ”A” and vice versa. Therefore we cannot apply the SVM hard margin theory to determine the classifier.

The next figure illustrates two sets in $\mathbb{R}^2$ which are not linearly separable.

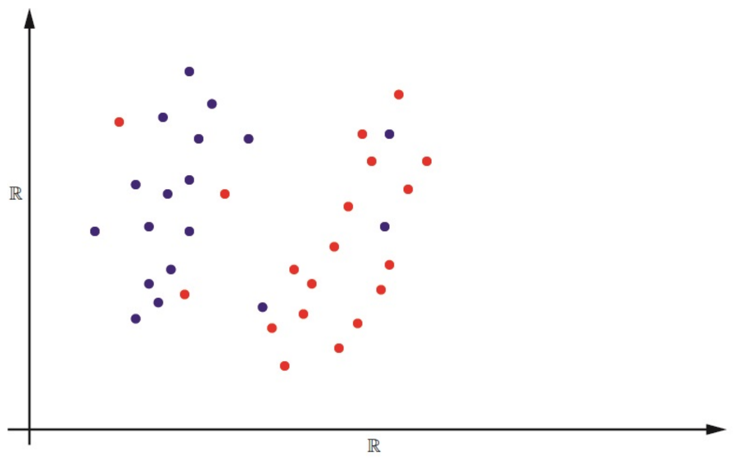


Figure 1.3: Two data sets in $\mathbb{R}^2$ which are not linearly separable.

Now we study the SVM Soft Margin, that generates a classifier by relaxing the constraints in (1.3). Consider the following optimization problem

$$\begin{array}{ll} \min\limits_{w,b,\xi} & \frac{1}{2}\|w\|^2 + C\sum\limits_{k=1}^{m}\xi_k \\ \text{s.t} & y_k(w^T x^k + b) \geq 1 - \xi_k, \quad k = 1, \dots, m \\ & \xi_k \geq 0, \quad k = 1, \dots, m \end{array}, \tag{1.9}$$

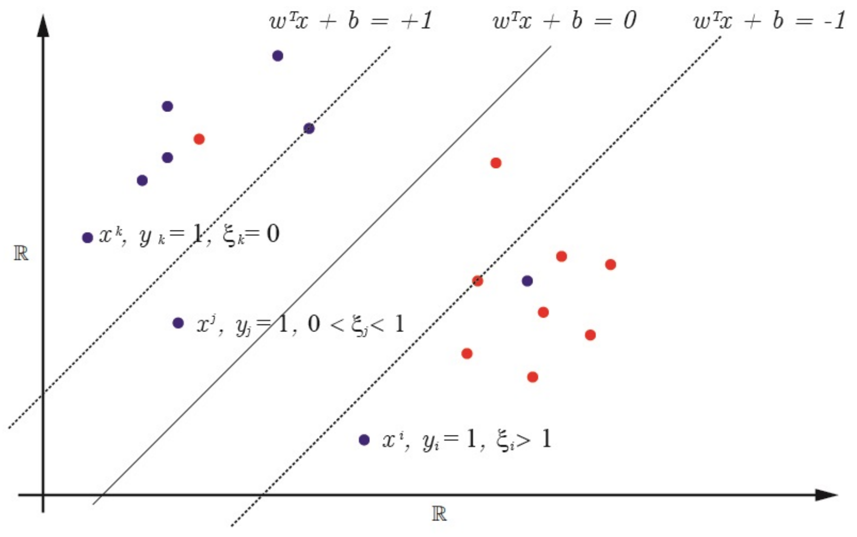
where $w \in \mathbb{R}^n$, ξ_k is a slack variable introduced in order to allow some missclassifications and $C > 0$ real positive parameter, predetermined by the user. Note that when the parameter C increases we expect that the sum $\sum\limits_{k=1}^{m}\xi_k$ decreases since we are minimizing the objective function which depends on w, b and ξ. Then, we tolerate less missclassfications for large values of C, however, when C is small the sum $\sum\limits_{k=1}^{m}\xi_k$ can be a large value, and thus, potentially, we tolerate more missclassifications.

Remark 1.2. In contrast to SVM Hard Margin, the constraints in (1.9) are always satisfied independently if the data is linearly separable or not. Indeed, for every $k \in \{1, \dots, m\}$, consider the slack variable ξ_k, associated to x^k. For any $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$

$$\xi_k = \begin{cases} \max\left\{0, 1 - w^T x^k - b\right\}, & \text{if } y_k = +1, \\ \max\left\{0, 1 + w^T x^k + b\right\}, & \text{if } y_k = -1. \end{cases}$$

□

In next figure, we take x^k, x^j and x^i in the positive class, that is, y_k, y_j , y_i $= +1$, our goal is to illustrate when x^k is correctly classified then ($\xi_k = 0$), when x^j is between the margin we have ($\xi_j \in (0,1)$), and when x^i is missclassified we have ($\xi_i > 1$).



$w^Tx + b = +1$
$w^Tx + b = 0$
$w^Tx + b = -1$
$x^k,\ y_k = 1,\ \xi_k = 0$
$x^j,\ y_j = 1,\ 0 < \xi_j < 1$
$x^i,\ y_i = 1,\ \xi_i > 1$
$\mathbb{R}$
$\mathbb{R}$

Figure 1.4: Geometrical interpretation of slack variables role in SVM Soft Margin.

The associated Lagrangian to (1.9) is given by

$$\mathcal{L}(w,b,\xi,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{k=1}^{m}\xi_k - \sum_{k=1}^{m}\alpha_k(y_i(w^Tx^k+b)-1+\xi_k) - \sum_{k=1}^{m}\beta_k\xi_k.$$

In order to determine the dual form we need solve the following problem

$$\begin{array}{cl} \max\limits_{\alpha,\beta} & \inf\limits_{w,b}\mathcal{L}(w,b,\xi,\alpha,\beta) \\ s.t & \alpha \geq 0, \\ & \beta \geq 0. \end{array}$$

Now we proceed to solve the minimization problem $\inf\limits_{w,b}\mathcal{L}(w,b,\xi,\alpha,\beta)$. Observe that for fixed α,β the function $\mathcal{L}(w,b,\xi,\alpha,\beta)$ is convex, then the minimizer $(\bar{w},\bar{b},\bar{\xi})$ with respect to w, b and ξ must satisfies

$$\nabla_{(w,b,\xi)}\mathcal{L}(\bar{w},\bar{b},\bar{\xi}) = 0,$$

which is equivalent to

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial w} = 0 \Rightarrow \bar{w} = \sum\limits_{k=1}^{m}\alpha_k y_k x^k \\ \dfrac{\partial\mathcal{L}}{\partial b} = 0 \Rightarrow \sum\limits_{k=1}^{m}\alpha_k y_k = 0 \\ \dfrac{\partial\mathcal{L}}{\partial \xi_k} = 0 \Rightarrow \alpha_k + \beta_k = C, \quad k = 1,\dots,m \end{cases},$$

by a similar analysis of Subsection 1.1.1, we obtain the dual problem

$$\begin{array}{cl} \max\limits_{\alpha} & -\dfrac{1}{2}\sum\limits_{k,l=1}^{m} y_k y_l \left(x^k\right)^T x^l\alpha_k\alpha_l + \sum\limits_{k=1}^{m}\alpha_k \\ \text{s.t} & \sum\limits_{k=1}^{m}\alpha_k y_k = 0 \\ & 0 \leq \alpha_k \leq C, \quad k = 1,\dots,m. \end{array} \tag{1.10}$$

Our goal is to determine the classifier (1.1), for this purpose we need to find the solution (w^*, b^*) of (1.9), we can obtain this solution by solving the the dual (1.10). In fact, let $\alpha^* \in \mathbb{R}^m$ be a solution of problem (1.10). Since the objective function in (1.9) is convex and the constraints are linear, the KKT conditions hold for the problem (1.9), that is, there exists (w^*, b^*, ξ^*) such that

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial w} = 0 \Rightarrow w = \displaystyle\sum_{k=1}^{m} \alpha_k y_k x^k \\ \dfrac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \displaystyle\sum_{k=1}^{m} \alpha_k y_k = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \xi_k} = 0 \Rightarrow \alpha_k + \beta_k = C \quad k = 1, \dots, m \\ \alpha_k \left[y_k \left(w^T x^k + b \right) - 1 + \xi_k \right] = 0, \quad k = 1, \dots, m \\ \beta_k \xi_k = 0, \quad k = 1, \dots, m \\ \alpha_k \geq 0, \quad k = 1, \dots, m \\ \beta_k \geq 0, \quad k = 1, \dots, m \end{cases} \tag{1.11}$$

Immediately, $w^* = \sum_{k=1}^{m} \alpha_k^* y_k x^k$, in order to obtain b^* observe that, for $0 < \alpha_k^* < C$, we have, $\beta_k = C - \alpha_k > 0$ and then $\xi_k = 0$. Therefore we can determine b^* by the equation $y_k \left((w^*)^T x^k + b^* \right) - 1 = 0$.

Now we are able to determine the classifier

$$y(x) = \operatorname{sign} \left(\sum_{k=1}^{m} \alpha_k^* y_k (x^k)^T x + b^* \right).$$

where α^*, b^* follow from the linear system (1.11).

1.2 Least Squares Support Vector Machines (LS-SVMs)

The Least Squares Support Vector Machine (LS-SVM) model [1, 29], arises from two modifications in the optimization problem of SVM soft margin (1.1)

$$\begin{array}{ll} \displaystyle\min_{w,b,\xi} & \dfrac{1}{2}\|w\|^2 + \dfrac{\gamma}{2} \sum_{k=1}^{m} \xi_k^2 \\ \text{s.t} & y_k (w^T x^k + b) = 1 - \xi_k, \quad k = 1, \dots, m, \\ & \xi_k \in \mathbb{R}, \quad k = 1, \dots, m. \end{array} \tag{1.12}$$

where $\gamma > 0$ is a parameter and ξ_k is an error variable which plays a similar role as slack variable in SVM but it is not necessarily positive.

Observe that the first modification lies on the objective function, where we add the L_2-norm of ξ instead of the L_1-norm. The second one is on the constraints, instead inequalities constraints we have an equality one. These two modifications will simplify the way that we solve the optimization problem. Indeed, different to SVM, we have to solve a linear system instead of a linear optimization problem. Regrettably, we lost some geometrical intuition but still maintain some nice interpretation (see Remark 1.4).

Now, we proceed by showing why solve LS-SVM is equivalent to solve linear system. Indeed, consider the Lagrangian function associated to LS-SVM, defined as

$$\mathcal{L}(w,b,\xi,\alpha)=\frac{1}{2}\|w\|^2+\sum_{k=1}^{m}\frac{\gamma}{2}\xi_k^2-\alpha_k(y_k(w^Tx^k+b)-1+\xi_k).$$

Since the constraints are linear, the KKT conditions hold, that is, the minimizer (w^*,b^*,ξ^*) satisfies the relations:

$$\begin{cases}\dfrac{\partial\mathcal{L}}{\partial w}=0 & \Longleftrightarrow \quad w^*=\sum\limits_{k=1}^{m}\alpha_ky_kx^k\\ \dfrac{\partial\mathcal{L}}{\partial b}=0 & \Longleftrightarrow \quad \sum\limits_{k=1}^{m}\alpha_ky_k=0\\ \dfrac{\partial\mathcal{L}}{\partial \xi}=0 & \Longleftrightarrow \quad \alpha_k=\gamma\xi_k^* \quad k=1,\ldots,m,\\ \dfrac{\partial\mathcal{L}}{\partial \alpha}=0 & \Longleftrightarrow \quad y_k((w^*)^Tx^k+b^*)-1+\xi_k^*=0, \quad k=1,\ldots,m\end{cases}, \tag{1.13}$$

Furthermore, since the involved functions are convex, every solution of (1.13) is also a global minimizer of LS-SVM, see [1]. Thus, we will simplify (1.13) by using some adequate matrices. In fact, it is not difficult to see that (1.13) is equivalent to the linear system:

$$\left[\begin{array}{ccc|c} I_n & 0 & 0 & -Z^T\\ 0 & 0 & 0 & -y^T\\ 0 & 0 & \gamma I_m & -I_m\\ \hline Z & y & I_m & 0\end{array}\right]\left[\begin{array}{c} w^*\\ b^*\\ \xi^*\\ \hline \alpha\end{array}\right]=\left[\begin{array}{c}0\\0\\0\\ \hline e\end{array}\right], \tag{1.14}$$

where $Z^T=\big(x^1y_1,\ldots,x^my_m\big)\in\mathbb{R}^{n\times m}$, $I_k\in\mathbb{R}^{k\times k}$ is the identity matrix, and y, e, ξ^*, α are column vectors in $\mathbb{R}^m$ defined as:

$$y=(y_1,\ldots,y_m)^T,\quad e=(1,\ldots,1)^T,\quad \xi^*=(\xi_1^*,\ldots,\xi_m^*)^T \text{ and } \alpha=(\alpha_1,\ldots,\alpha_m)^T\in\mathbb{R}^m.$$

Moreover, since w^* and ξ^* depend on α and b^* (see (1.13)), we can simplify (1.14) and obtain a more compact linear system:

$$\left[\begin{array}{c|c} y^T & 0\\ \hline \Omega+\frac{1}{\gamma}I & y\end{array}\right]\left[\begin{array}{c}\alpha\\ \hline b^*\end{array}\right]=\left[\begin{array}{c}0\\ \hline e\end{array}\right], \tag{1.15}$$

where $\Omega=ZZ^T$. Note that if the matrix in (1.15) is full rank then, we have an unique solution (α^*,b^*), see [1].

From equations (1.13), we have $w^*=\sum\limits_{k=1}^{m}\alpha_k^*y_kx^k$, thus the linear classifier (1.1) takes the form

$$y(x) = \operatorname{sign}\left[\sum_{k=1}^{m} y_k \alpha_k^* (x^k)^T x + b^*\right]. \tag{1.16}$$

Remark 1.3. From equation (1.16) we that the classifier will depends on solution α^* and from system (1.13) we have

$$\alpha_k^* = \gamma \xi_k, \quad k = 1, \ldots, m.$$

Thus, $\alpha_k^* = 0$ only when $\xi_k = 0$ which implies that $w^T x^k + b = y_k$, that is, x^k is on the margin, because from the system (1.13), we have that, $y_k(w^T x^k + b) - 1 + \xi_k = 0$. This can be a disadvantage, because $\alpha^* = 0$ only when the input vector x is on the margin. If we consider the example in Figure 1.4 just two input vectors are on the margin. Therefore, when m is a large number we expect that $\alpha_k^* \neq 0$ for many k, thus the classifier will have to perform many calculations to predict a new input vector. $\square$

Remark 1.4. As we just described, to find a solution of the optimization problem associated to LS-SVM, we just need to solve a linear system. Thus, from the numerical point of view, solutions associated to LS-SVM are more easy to compute, and hence we avoid the use of more complex numerical methods for solving constrained optimization problems as Interior Point Methods(IPMs), Sequential Minimal Optimization (SMO), etc [5, 23]. On the other hand, we lose the nice geometrical interpretation of the classical SVM, where the hyperplane is chosen such that it maximizes the margins related to the data. Beside this observation, we can associate to the solutions of LS-SVM, a different interpretation, roughly, LS-SVM try to find a hyperplane that maximizes the margin, meanwhile the least square error $\sum_{i=1}^{m}((w^T x^k + b) - y_k)^2$ is the less possible. To make the statement clear, observe that the optimization problem (1.12) can be written as:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\sum_{i=1}^{m}((w^T x^k + b) - y_k)^2. \tag{1.17}$$

In fact, since $y_k^2 = 1$ and by (1.1), we see that

$$\begin{aligned}\sum_{k=1}^{m} \xi_k^2 &= \sum_{k=1}^{m}(y_k(w^T x^k + b) - 1)^2 = \sum_{k=1}^{m}(y_k(w^T x^k + b) - y_k^2)^2 = \sum_{k=1}^{m}(y_k((w^T x^k + b) - y_k))^2 \\ &= \sum_{k=1}^{m}((w^T x^k + b) - y_k)^2.\end{aligned}$$

$\square$

Remark 1.5. Observe that solving (1.15), when $m \ll n$, it is more advantageous to solve than (1.17). The reason of this is that we will look for a solution in $\mathbb{R}^{m+1}$ instead of a solution in $\mathbb{R}^{n+1}$ which is the case of (1.17). On the other hand, $m \gg n$, it is more advantageous to solve (1.17). However, in the next chapter we will study the case that $m \ll n$ and we look for solutions of (1.17). The reason for this approach will be explained at beginning of the next chapter. $\square$

Chapter 2

L_q-NORM LEAST SQUARES SUPPORT VECTOR MACHINE

In this chapter we will study a model called L_q-norm Least Squares Support Vector Machine (L_q-norm LS-SVM) proposed in [26]. The idea is to find a sparse solution for the problem (2.1), to perform feature selection and classification simultaneously. For convenience sake, we will organize our training data set $\{(x^k, y_k)\}_{k=1}^{m}$, $x^k \in \mathbb{R}^n$ and $y_k \in \{-1, 1\}$ as follows. Let $X \subset \mathbb{R}^{m\times n}$ be a matrix, where the k-th row is the input vector x^k, and $Y \subset \mathbb{R}^{m\times m}$ a diagonal matrix which element $Y_{k,k}$ corresponds to y_k, thus we can rewrite (1.12) as the following

$$\begin{array}{ll} \min\limits_{w,b,\xi} & f(w,b) = \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\xi^T\xi \\ \text{s.t} & Y(Xw + eb) + \xi = e \\ & \xi \in \mathbb{R}^m \end{array} \tag{2.1}$$

where $e = (1, \dots, 1)^T \in \mathbb{R}^m$.

As we mentioned in Remark 1.3, LS-SVM model has a drawback. When we choose to solve the dual problem (1.15), the dual solution α^* suffers of lack of sparseness when the classifier is given by (1.16) which can increases the computational complexity to calculate the classifier, furthermore the classifier can have a bad performance with respect to the number of correct predictions.

An approach to handling this LS-SVM issue is to solve the primal problem (2.1) and perform feature selection on the data set. An effective way is to conduct feature selection and classification simultaneously, and we do this in LS-SVM by finding sparse solution $(w^*, b^*) \in \mathbb{R}^{n+1}$ of (2.1). The next figure illustrates how classification and features selection can be conduct at same time.

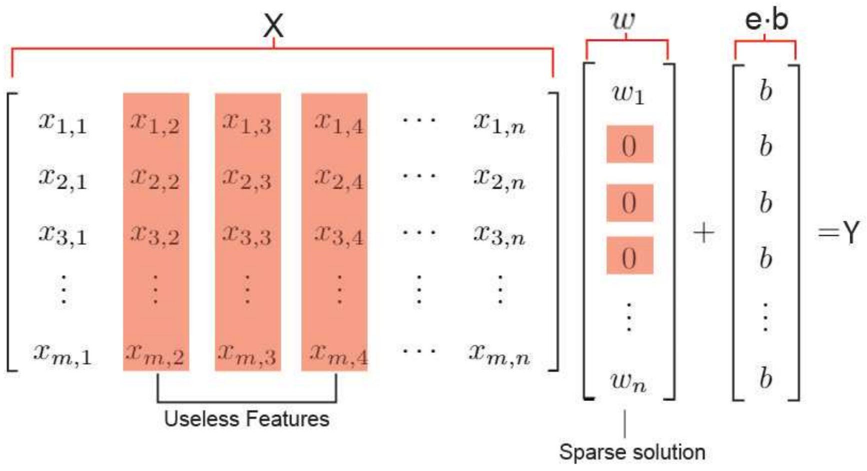

X
w
e·b
$x_{1,1}$ $x_{1,2}$ $x_{1,3}$ $x_{1,4}$ $\cdots$ $x_{1,n}$
$x_{2,1}$ $x_{2,2}$ $x_{2,3}$ $x_{2,4}$ $\cdots$ $x_{2,n}$
$x_{3,1}$ $x_{3,2}$ $x_{3,3}$ $x_{3,4}$ $\cdots$ $x_{3,n}$
$x_{m,1}$ $x_{m,2}$ $x_{m,3}$ $x_{m,4}$ $\cdots$ $x_{m,n}$
w_1 0 0 0 w_n
b
+
=Y
Useless Features
Sparse solution

Figure 2.1: Indeed features associated to null components of w can be eliminated without affecting the performance of the classifier

We note that in the bibliography, there are studies that aim to solve the lack of sparseness of LS-SVM, for example [11, 15, 28]. All of these works are interested in solve (1.15), specifically, they are looking for a sparser solution $(\alpha^*, b^*) \in \mathbb{R}^{m+1}$ of (1.15), and then it is not possible to implement feature selection and classification simultaneously, because a sparse solution α^* cannot perform feature selection. The Figure 2.2 illustrates this fact.

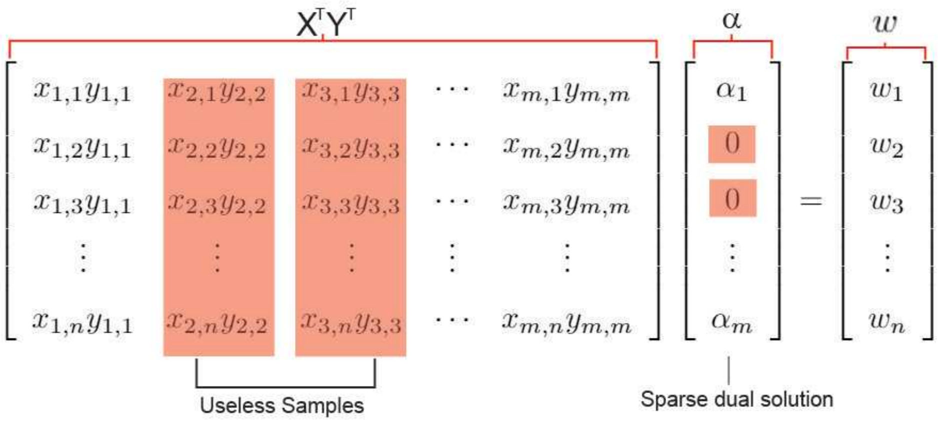

X^TY^T
α
w
$x_{1,1}y_{1,1}$ $x_{2,1}y_{2,2}$ $x_{3,1}y_{3,3}$ $\cdots$ $x_{m,1}y_{m,m}$
$x_{1,2}y_{1,1}$ $x_{2,2}y_{2,2}$ $x_{3,2}y_{3,3}$ $\cdots$ $x_{m,2}y_{m,m}$
$x_{1,3}y_{1,1}$ $x_{2,3}y_{2,2}$ $x_{3,3}y_{3,3}$ $\cdots$ $x_{m,3}y_{m,m}$
$x_{1,n}y_{1,1}$ $x_{2,n}y_{2,2}$ $x_{3,n}y_{3,3}$ $\cdots$ $x_{m,n}y_{m,m}$
α_1 0 0 α_m
=
w_1 w_2 w_3 w_n
Useless Samples
Sparse dual solution

Figure 2.2: A sparse solution α decreases the computational complexity however, it cannot perform feature selection, therefore a sparse α does not help us to understand better the problem.

2.1 Sparse Approximation of LS-SVM with L_q-norm

Observe that (2.1) is equivalent to

$$\min_{w,b} f(w,b) = \frac{1}{2}\|w\|^2 + \frac{\gamma}{2}\|Y(Xw+eb)-e\|^2.$$

This is a convex problem, therefore we will search for stationary points by setting gradient of f equals to zero,

$$\begin{cases} \dfrac{\partial f}{\partial w}(w,b) = 0 \Leftrightarrow w + \gamma X^T Y^T (Y(Xw+eb)-e) = 0, \\ \\ \dfrac{\partial f}{\partial b}(w,b) = 0 \Leftrightarrow \gamma e^T Y^T (Y(Xw+eb)-e) = 0. \end{cases} \tag{2.2}$$

Arranging the system (2.2) in a matrix form, we have

$$\begin{bmatrix} X^TX + \frac{1}{\gamma}I & X^Te \\ e^TX & e^Te \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} X^T \\ e^T \end{bmatrix} Ye. \tag{2.3}$$

Let us define

$$H := H(\gamma) = \begin{bmatrix} X^TX + \frac{1}{\gamma}I & X^Te \\ e^TX & e^Te \end{bmatrix}, \tag{2.4}$$

and

$$d = \begin{bmatrix} X^T \\ e^T \end{bmatrix} Y, \quad u = \begin{bmatrix} w \\ b \end{bmatrix} \tag{2.5}$$

observe that $H = H(\gamma) \in \mathbb{R}^{(n+1)\times(n+1)}$ is a symmetric matrix, $u \in \mathbb{R}^{n+1}$ and $d \in \mathbb{R}^{n+1}$. Therefore (2.3) can be written as the linear system

$$Hu = d. \tag{2.6}$$

Remark 2.1. When the number of input data m is much smaller than the number of attributes/features n, i.e., $m \ll n$ the solution of (2.6) may not be unique. To illustrate this fact, since $m \ll n$, we can suppose that the rows of the data matrix $X \in \mathbb{R}^{m\times n}$ are orthonormal

$$\begin{bmatrix} X^TX + \frac{1}{\gamma}I & X^Te \\ e^TX & e^Te \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

then

$$\begin{cases} X^TXw + \dfrac{1}{\gamma}Iw + X^Teb = 0 \\ e^TXw + bm = 0 \Rightarrow b = \dfrac{-e^TXw}{m} \end{cases},$$

substituting b into first equation, as result, we get the following equation

$$X^TXw + \frac{1}{\gamma}Iw - \frac{X^Tee^TXw}{m} = 0.$$

Now, by multiplying X and since $XX^T = I$, we see that

$$Xw + \frac{1}{\gamma}Xw - \frac{ee^T Xw}{m} = 0,$$

and putting Xw on evidence, it implies that

$$\left(I + \frac{1}{\gamma}I - \frac{ee^T}{m}\right) Xw = 0. \tag{2.7}$$

Note that the matrix $\left(1+\frac{1}{\gamma}\right) I - \frac{ee^T}{m}$ is definite positive. In fact, by Cauchy's inequality, given a, b in R^m, we have

$$\left(\sum_{i=1}^{m} a_i b_i\right)^2 \leq \left(\sum_{i=1}^{m} a_i^2\right)\left(\sum_{i=1}^{m} b_i^2\right).$$

For Cauchy's inequality with $a \in \mathbb{R}^m$ and $b = e$ we get

$$\left(\sum_{i=1}^{m} a_i\right)^2 \leq \left(\sum_{i=1}^{m} a_i^2\right) m.$$

Thus for $a = x$,

$$x^T\left[\left(1+\frac{1}{\gamma}\right) I - \frac{ee^T}{m}\right] x = (1+\frac{1}{\gamma})\|x\|^2 - \frac{(x_1 + \ldots + x_m)^2}{m} \geq \frac{1}{\gamma}\|x\|^2 > 0$$

therefore we conclude $\left(1+\frac{1}{\gamma}\right) I - \frac{ee^T}{m}$ is definite positive, as consequence, we have that the equation (2.7) is equivalent to $Xw = 0$, which has nontrivial kernel since $m \ll n$. Thus the solution of (2.7) is not unique. $\square$

Since the problem (2.6) might have multiple solutions and we are interested to perform feature selection and classification simultaneously using a linear classifier (1.1) with a sparse w, thus we look for a sparse solution u of (2.6) and consequently a sparse vector w. One strategy to find a sparse acceptable solution for the problem is by introducing the L_0-norm, then the new problem is given by

$$\begin{array}{ll} \min\limits_{u} & \|u\|_0 \\ \text{s.t.} & \|Hu - d\|^2 \leq \delta \end{array} \tag{2.8}$$

where $\delta > 0$ is a tolerance measure, and

$$\|u\|_0 = |\{i \mid u_i \neq 0\}|,$$

that is, $\|u\|_0$ is the number of nonzero components of u.

In order to get a simpler problem, we consider a penalised version of (2.8), see [2, 21], that is

$$\min_{u} \|u\|_0 + \frac{1}{2\rho}\|Hu - d\|^2 \tag{2.9}$$

where $\rho > 0$ is a parameter.

From the theory of the classical penalty method, the sequence of solutions of (2.9) converges to solution of (2.8) as $\rho \to 0_+$. It was proved in [20] this problem is NP-Hard [1].

There are studies as pointed in [17, 26] that show the effectiveness of L_1-norm to find sparse solutions, that is the problem (2.9) is replaced by

$$\min_u \|u\|_1 + \frac{1}{2\rho}\|Hu - d\|^2. \tag{2.10}$$

However in this work we study another approach to find sparse solutions of (2.6) which consists in to replace L_0-norm by L_q-norm with $0 < q < 1$, that is,

$$\min_u \|u\|_q^q + \frac{1}{2\rho}\|Hu - d\|^2 \tag{2.11}$$

where $\|u\|_q^q = \sum_{i=1}^{n+1} |u_i|^q$. As mentioned in [17] the motivation of this approach is the fact

$$\lim_{q \to 0_+} \|u\|_q^q = \|u\|_0, \quad \forall u \in \mathbb{R}^{n+1}.$$

and the property illustrated in the next figure [26]. The next figure illustrates the L_q norm ability to find sparse solutions of $Hu = d$ (2.6).

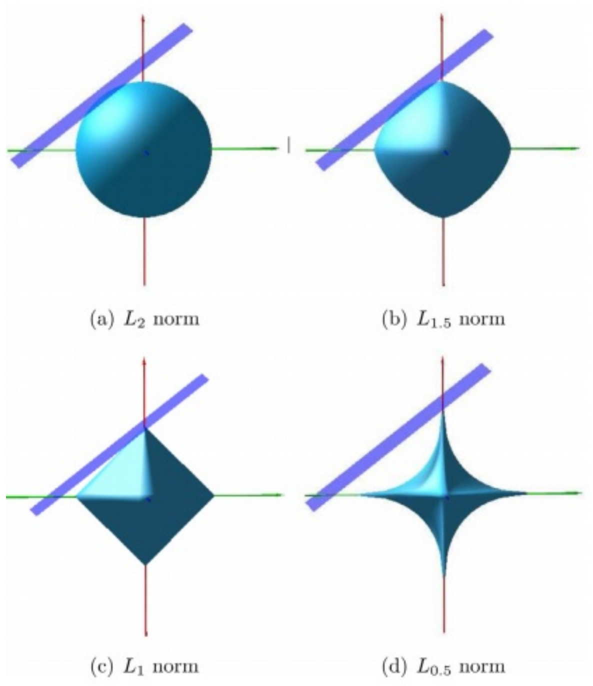


Figure 2.3: Intersection between L_q-balls and the subset of $Hu = d$. For the $L_{0.5}$-norm we can see that solutions will be sparser, since the intersection is over the axis. However, we observe that L_1-norm will find a sparse solution.

[1]There is no known method that solves the problem in a polynomial time, under the $P \neq NP$ hypothese. See [9].

However, it was proved in [9] that (2.11) is also a NP-Hard problem when $0 < q < 1$. In order to avoid nondifferentiable of the problem (2.11), it is introduced a new parameter $\epsilon > 0$ and thus we have a smoothed version of (2.11) which is given by

$$\min_u f_q(\epsilon, u) = \|u\|_{q,\epsilon}^q + \frac{1}{2\rho}\|Hu - d\|^2 \tag{2.12}$$

where $\|u\|_{q,\epsilon}^q = \sum_{j=1}^{n+1} \left(\epsilon + u_j^2\right)^{q/2}$, H, d and u are defined in (2.4) and (2.5) respectively.

Proposition 2.1. *The problem* (2.12) *admits a solution.*

Proof: As consequence of Weierstrass's Theorem (see [2]), the problem (2.12) will admit a solution, if the objective function of (2.12) is coercive. Hence, our goal is to show that $f_q(\epsilon, x) = ||x||_{q,\epsilon}^q + \frac{1}{2\rho}\|Hx - b\|_2^2$ is coercive. We recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is called coercive when $\lim_{\|x\|\to\infty} f(x) = \infty$. In fact, let $||x|| \to \infty$ then $||x||^2 = \sum_{j=1}^{n+1} x_j^2 \to \infty$, we can see that there exists at least one $j \in \{1, \dots, n\}$ such that $x_j^2 \to \infty$ otherwise $||x||^2$ is bounded, then $x_j^2 + \epsilon \to \infty$, thus $(x_j^2 + \epsilon)^{q/2} \to \infty$ as $x_j \to \infty$, therefore $||x||_{q,\epsilon}^q \to \infty$ as $||x|| \to \infty$ and we have our result because $\|Hx - b\|_2^2 \geq 0$ for every $x \in \mathbb{R}^{n+1}$.$\square$

The following figure illustrates effect of parameter ϵ on $\|\cdot\|_{q,\epsilon}^q$.

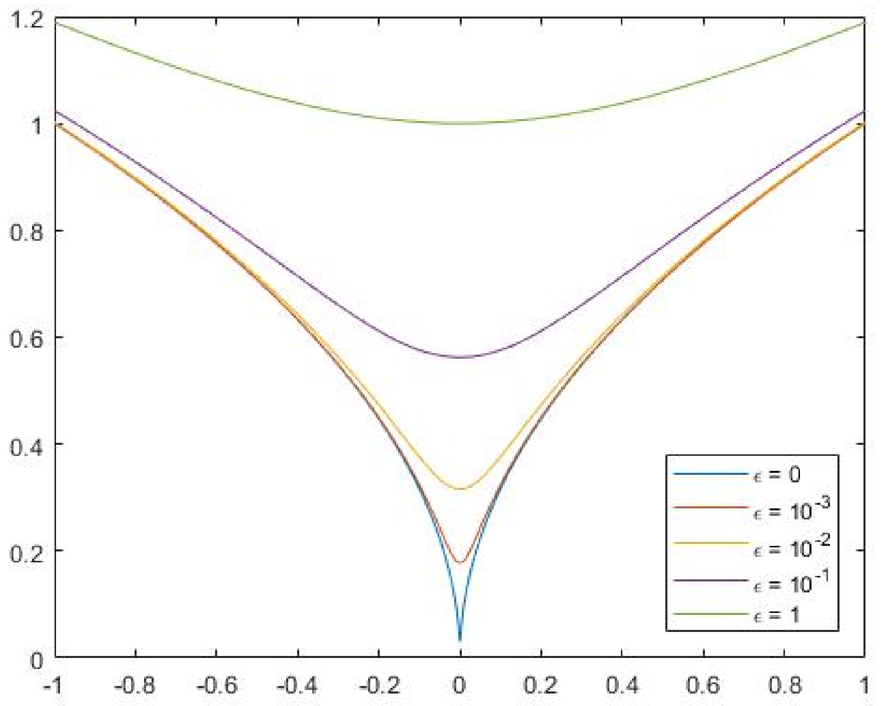


Figure 2.4: The behavior of $\|x\|_{1/2,\epsilon}^{1/2} = (x^2 + \epsilon)^{\frac{1}{4}}$, $x \in [-1, 1]$, for different values of ϵ. The function $\|x\|_{1/2,\epsilon}^{1/2}$ is a smooth approximation of $\sqrt{|x|}$.

2.2 L_q-norm LS-SVM algorithm and its convergence analysis

In this subsection we present the algorithm proposed in [26] to find a solution of (2.12), also we show its convergence and the relation among the problems (2.12), (2.11), (2.9) and (2.8).

2.2.1 L_q-norm LS-SVM Algorithm

A critical point of the problem (2.12) satisfies the following equation

$$\nabla_u f_q(\epsilon, u^{\epsilon,q}) = 0,$$

that is, $u^{\epsilon,q}$ satisfies the equation:

$$\left(\frac{q u_j^{\epsilon,q}}{\left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{1-q/2}} \right)_{1\leq j\leq n+1} + \frac{1}{\rho} H^T \left(H u^{\epsilon,q} - d\right) = 0. \tag{2.13}$$

This is a necessary condition for minimizers. Note that equation (2.13) is nonlinear then can be hard to solve it. In order to overcome this difficulty an iterative method is proposed in [17, 26]. The method consists in starting with initial $u^{(0)} \in \mathbb{R}^{n+1}$, for a given $u^{(k)}$ we solve the following linear system for $u^{(k+1)}$

$$\left(\frac{q u_j^{(k+1)}}{\left(\epsilon + \left(u_j^{(k)}\right)^2\right)^{1-q/2}} \right)_{1\leq j\leq n+1} + \frac{1}{\rho} H^T \left(H u^{(k+1)} - d\right) = 0, \tag{2.14}$$

or equivalently

$$\left[\operatorname{diag} \left(\frac{q\rho}{\left(\epsilon + \left(u_j^{(k)}\right)^2\right)^{1-q/2}} \right)_{1\leq j\leq n+1} + H^T H \right] u^{(k+1)} = H^T d, \tag{2.15}$$

where $u_j^{(k)}$ is the j-th component of $u^{(k)}$ and $u_j^{(k+1)}$ is the j-th component of $u^{(k+1)}$.

The method stops when $u^{(k+1)} = u^{(k)}$. We can see that the matrix at right side of (2.15) is positive-definite for every $u^{(k)}$ and thus invertible, therefore, the method is well defined. Now we outline L_q-norm LS-SVM algorithm

Algorithm 1 L_q-norm LS-SVM

```
1: Input: The training input matrix $X \in \mathbb{R}^{m\times n}$, the output matrix $Y \in \mathbb{R}^{m\times m}$ and parameters $\gamma$, $\rho$,
   $\epsilon$ and $q$
2: Process:
3: Compute matrix $H$ and vector $d$ as in (2.4) and (2.5) respectively
4: Take any $u^{(0)} \in \mathbb{R}^{n+1}$ as initial point and set $k = 0$
5: For $k = 0, 1, 2, \ldots$
6: Find $u^{(k+1)}$ by solving (2.15) using $u^{(k)}$;
7: Stop when $u^{(k+1)} = u^{(k)}$;
8: Output: $u^{(k+1)}$
9: end process
```

More details about implementation of Algorithm 1 is in the Chapter 3, and in Appendix A we present our Matlab implementation.

2.2.2 L_q-norm LS-SVM convergence analysis for ϵ, q and γ fixed

Observe that for each choice of parameters $\epsilon > 0$, $q \in (0, 1)$, $\rho > 0$ and $\gamma > 0$, we get a different optimization problem of the form (2.12). In this subsection, for each result (Lemmas/Theorems) we first choose the parameters and fixed them, except for Theorem 3 where ρ will depends on the others parameters.

The goal of this subsection is to show that the sequence $\{u^{(k)}\}_k$ generated by Algorithm 1 has subsequence $\{u^{(k_i)}\}_i$ which converges to the solution of (2.13). For this purpose, we will need some auxiliary results.

Proposition 2.2. *For $\epsilon > 0$, $q \in (0, 1)$ and arbitrary α, $\beta \in \mathbb{R}$ the following inequality holds*

$$\left(\epsilon + \alpha^2\right)^{q/2} - \left(\epsilon + \beta^2\right)^{q/2} - \frac{q\beta(\alpha-\beta)}{\left(\epsilon + \alpha^2\right)^{1-q/2}} \geq 0. \tag{2.16}$$

Proof: In order to obtain (2.16), we will show that

$$\left(\epsilon + \alpha^2\right) - \left(\epsilon + \alpha^2\right)^{1-q/2} \left(\epsilon + \beta^2\right)^{q/2} - q\beta(\alpha-\beta) \geq 0. \tag{2.17}$$

We will use Young's inequality which says that for a, $b > 0$ and u, $v > 0$ such that $\frac{1}{u} + \frac{1}{v} = 1$, the following inequality holds

$$ab \leq \frac{1}{u}a^u + \frac{1}{v}b^v. \tag{2.18}$$

Take $a = \left(\epsilon+\alpha^2\right)^{1-q/2}$, $b = \left(\epsilon+\beta^2\right)^{q/2}$ and $u = \frac{1}{1-q/2}$, $v = \frac{1}{q/2}$. Note that $\dfrac{1}{u}+\dfrac{1}{v} = 1-\dfrac{q}{2}+\dfrac{q}{2} = 1$, then by (2.18)

$$\begin{aligned}\left(\epsilon+\alpha^2\right)^{1-q/2}\left(\epsilon+\beta^2\right)^{q/2} \leq& (1-q/2)\left[\left(\epsilon+\alpha^2\right)^{1-q/2}\right]^{\frac{1}{(1-q/2)}} + (q/2)\left[\left(\epsilon+\beta^2\right)^{q/2}\right]^{2/q} \\ =& \left(\epsilon+\alpha^2\right) + q/2\left(\left(\epsilon+\beta^2\right)-\left(\epsilon+\alpha^2\right)\right) = \left(\epsilon+\alpha^2\right)+q/2\left(\beta^2-\alpha^2\right).\end{aligned} \tag{2.19}$$

Using the previous inequality (2.19), we get

$$\left(\epsilon+\alpha^2\right)-\left(\epsilon+\alpha^2\right)^{1-q/2}\left(\epsilon+\beta^2\right)^{q/2}+q/2\left(\beta^2-\alpha^2\right) \geq 0.$$

Now, observe that $\beta^2-2\alpha\beta+\alpha^2 \geq 0$, and then $\beta^2-2\alpha\beta \geq -\alpha^2$, thus $q(\beta^2-2\alpha\beta) \geq (q/2)(\beta^2-\alpha^2)$ for $q>0$, hence we have

$$\begin{aligned}0 \leq& \left(\epsilon+\alpha^2\right)-\left(\epsilon+\alpha^2\right)^{1-q/2}\left(\epsilon+\beta^2\right)^{q/2}+q/2\left(\beta^2-\alpha^2\right) \leq \\ \leq& \left(\epsilon+\alpha^2\right)-\left(\epsilon+\alpha^2\right)^{1-q/2}\left(\epsilon+\beta^2\right)^{q/2}-q\beta(\alpha-\beta).\end{aligned}$$

Therefore, we proved (2.17) and consequently we get (2.16). □

Lemma 2.1. *Let $\epsilon>0$ and $\{u^{(k)}\}_k$ be the sequence generated by Algorithm 1, then*

$$f_q\left(\epsilon, u^{(k)}\right)-f_q\left(\epsilon, u^{(k+1)}\right) \geq \frac{1}{2\rho}\left\|Hu^{(k)}-Hu^{(k+1)}\right\|^2, \quad \forall k\in\mathbb{N} \tag{2.20}$$

Proof: Computing the right side of (2.20),

$$\begin{aligned}f_q(\epsilon,u^{(k)})-f_q(\epsilon,u^{(k+1)}) =& \sum_{j=1}^{n+1}\left(\epsilon+(u_j^{(k)})^2\right)^{q/2}-\sum_{j=1}^{n+1}\left(\epsilon+(u_j^{(k+1)})^2\right)^{q/2} \\ &+\frac{1}{2\rho}\left(\left\|Hu^{(k)}-d\right\|^2-\left\|Hu^{(k+1)}-d\right\|^2\right),\end{aligned}$$

and, then

$$\begin{aligned}f_q(\epsilon,u^{(k)})-f_q(\epsilon,u^{(k+1)}) =& \sum_{j=1}^{n+1}\left(\epsilon+\left(u_j^{(k)}\right)^2\right)^{q/2}-\sum_{j=1}^{n+1}\left(\epsilon+\left(u_j^{(k+1)}\right)^2\right)^{q/2} \\ +\frac{1}{2\rho}\left(\left\|Hu^{(k)}-Hu^{(k+1)}\right\|^2\right)&+\frac{1}{\rho}\left(Hu^{(k+1)}-d\right)^T\left(Hu^{(k)}-Hu^{(k+1)}\right).\end{aligned} \tag{2.21}$$

By (2.14), we get

$$H^THu^{(k+1)} = H^Td-\left(\frac{q\rho u_j^{(k+1)}}{\left(\epsilon+\left(u_j^{(k)}\right)^2\right)^{1-q/2}}\right)_{1\leq j\leq n+1}.$$

The last term of (2.21) can be computed as

$$\begin{aligned}\frac{1}{\rho}\left(Hu^{(k+1)}-d\right)^T\left(Hu^{(k)}-Hu^{(k+1)}\right)=&\frac{1}{\rho}\left[\left(u^{(k+1)}\right)^T H^T Hu^{(k)}-\left(u^{(k+1)}\right)^T H^T Hu^{(k+1)}\right]\\&-\frac{1}{\rho}\left[d^T Hu^{(k)}-d^T Hu^{(k+1)}\right]\\=&\frac{1}{\rho}\left[d^T Hu^{(k)}-\left(\frac{q\rho u_j^{(k+1)}}{\left(\epsilon+\left(u_j^{(k)}\right)^2\right)^{1-q/2}}\right)^T_{1\le j\le n+1}u^{(k)}\right]\\&-\frac{1}{\rho}\left[d^T Hu^{(k+1)}-\left(\frac{q\rho u_j^{(k+1)}}{\left(\epsilon+\left(u_j^{(k)}\right)^2\right)^{1-q/2}}\right)^T_{1\le j\le n+1}u^{(k+1)}\right]\\&-\frac{1}{\rho}\left[d^T Hu^{(k)}-d^T Hu^{(k+1)}\right].\end{aligned}$$

Thus,

$$\frac{1}{\rho}\left(Hu^{(k+1)}-d\right)^T\left(Hu^{(k)}-Hu^{(k+1)}\right)=-\sum_{j=1}^{n}\frac{qu_j^{(k+1)}\left(u_j^{(k)}-u_j^{(k+1)}\right)}{\left(\epsilon+\left(u_j^{(k)}\right)^2\right)^{1-q/2}}.$$

Hence, by inequality (2.16) of Proposition 2.2, and the equations (2.21), (2.2.2), we have

$$\begin{aligned}f_q(\epsilon,u^{(k)})-f_q(\epsilon,u^{(k+1)})=&\sum_{j=1}^{n+1}\left[\left(\epsilon+\left|u_j^{(k)}\right|^2\right)^{q/2}-\left(\epsilon+\left|u_j^{(k+1)}\right|^2\right)^{q/2}-\frac{qu_j^{(k+1)}\left(u_j^{(k)}-u_j^{(k+1)}\right)}{\left(\epsilon+\left(u_j^{(k)}\right)^2\right)^{1-q/2}}\right]\\&+\frac{1}{2\rho}\left\|Hu^{(k)}-Hu^{(k+1)}\right\|^2\ge\frac{1}{2\rho}\left\|Hu^{(k)}-Hu^{(k+1)}\right\|^2\ge 0\end{aligned}\tag{2.22}$$

□

The next theorem show us that every limit point of the sequence of vectors generated by Algorithm 1 is a stationary point of (2.12) and this a necessary optmality condition.

Theorem 1. *Let $\{u^{(k)}\}_k$ a sequence generated by the Algorithm 1. Then every limit point $u^{\epsilon,q}$ of $\{u^{(k)}\}_k$, is a stationary point of* (2.12) *and the sequence is bounded.*

Proof: We begin by proving that $\{u^{(k)}\}_k$ has a convergent subsequence $\{u^{(k_i)}\}_i$ such that

$$u^{(k_i)}\to u=(u_1,\dots,u_{n+1})^T,\; u^{(k_i+1)}\to v=(v_1,\dots,v_{n+1})^T.$$

To prove this, we will show that $\{u^{(k)}\}_k$ is bounded. In fact, by Lemma 2.1, we have

$$f_q\left(\epsilon,u^{(k)}\right)\ge f_q\left(\epsilon,u^{(k+1)}\right)\tag{2.23}$$

that is, $\{f_q\left(\epsilon, u^{(k)}\right)\}_k$ is decreasing. Also this sequence is bounded, because $f_q\left(\epsilon, u^{(k)}\right) \geq 0$, then there exists $M \geq 0$ such that $\lim\limits_{k\to\infty} f_q(\epsilon, u^{(k)}) = M$. We can see that for k large enough the following inequality holds

$$\left\|u^{(k)}\right\|_q^q \leq \left\|u^{(k)}\right\|_{q,\epsilon}^q \leq f_q\left(\epsilon, u^{(k)}\right) \leq M+1.$$

Then $\left\|u^{(k)}\right\|_q^q$ is bounded and consequently the sequence $\{u^{(k)}\}_k$ is bounded, thus it has a convergent subsequence $\{u^{(k_i)}\}_i$ lets say $u^{(k_i)} \to u \in \mathbb{R}^{n+1}$. Also by equation (2.15) we have that the subsequence $\{u^{(k_i+1)}\}_i$ is convergent, i.e., $u^{(k_i+1)} \to v$ for some $v \in \mathbb{R}^{n+1}$.

Now, we will prove that $u = v$. From the facts that $u^{(k_i)} \to u$, $u^{(k_i+1)} \to v$ and $\{f_q\left(\epsilon, u^{(k)}\right)\}_k$ is a convergent sequence, we get

$$f_q(\epsilon, u) = f_q(\epsilon, v). \tag{2.24}$$

By Lemma 2.1,

$$\|Hu - Hv\|^2 \leq 2\rho\left(f_q(\epsilon, u) - f_q(\epsilon, v)\right) = 0$$

which implies

$$Hu = Hv. \tag{2.25}$$

Therefore, by equations (2.24) and (2.25), we obtain

$$\|u\|_{q,\epsilon}^q = \|v\|_{q,\epsilon}^q.$$

Now, computing the inner product between $u - v$ and left side of (2.14), we obtain

$$\left[\left(\frac{q u_j^{(k+1)}}{\left(\epsilon + \left(u_j^{(k)}\right)^2\right)^{1-q/2}}\right)_{1\leq j\leq n+1} + \frac{1}{\rho} H^T\left(Hu^{(k+1)} - d\right)\right]^T (u - v) = 0,$$

or equivalently,

$$\sum_{j=1}^{n+1} \frac{q u_j^{(k+1)}(u_j - v_j)}{\left(\epsilon + \left(u_j^{(k)}\right)^2\right)^{1-q/2}} + \frac{1}{\rho}(Hu^{(k+1)} - d)^T H(u - v) = 0, \ \forall k \in \mathbb{N}.$$

Setting $k = k_i$ and letting $i \to +\infty$ we have

$$\sum_{j=1}^{n+1} \frac{q v_j(u_j - v_j)}{\left(\epsilon + (u_j)^2\right)^{1-q/2}} + \frac{1}{\rho}(Hu - Hv)^T(Hv - d) = 0.$$

Due to $Hu = Hv$, we get

$$\sum_{j=1}^{n+1} \frac{q v_j(u_j - v_j)}{\left(\epsilon + (u_j)^2\right)^{1-q/2}} = 0$$

and since $\|u\|_{q,\epsilon}^q = \|v\|_{q,\epsilon}^q$ we have the following expression:

$$\|u\|_{q,\epsilon}^q - \|v\|_{q,\epsilon}^q - \sum_{j=1}^{n+1} \frac{qv_j(u_j - v_j)}{\left(\epsilon + (u_j)^2\right)^{1-q/2}} = \sum_{j=1}^{n+1} \left(\left(\epsilon + u_j^2\right)^{q/2} - \left(\epsilon + v_j^2\right)^{q/2} - \frac{qv_j\left(u_j - v_j\right)}{\left(\epsilon + (u_j)^2\right)^{1-q/2}} \right) = 0. \tag{2.26}$$

By Proposition 2.2, we can see that each term of the sum at right side of equation (2.26) is equal to zero. Furthermore, each term can be written as

$$\begin{aligned}
&\left(\epsilon + u_j^2\right)^{q/2} - \left(\epsilon + v_j^2\right)^{q/2} - \frac{qv_j\left(u_j - v_j\right)}{\left(\epsilon + (u_j)^2\right)^{1-q/2}} = \\
&= \frac{2\left(\epsilon + (u_j)^2\right)^{1-q/2}\left(\epsilon + u_j^2\right)^{q/2} - 2\left(\epsilon + (u_j)^2\right)^{1-q/2}\left(\epsilon + v_j^2\right)^{q/2} - 2qv_j\left(u_j - v_j\right)}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}} \\
&= \frac{2\left(\epsilon + u_j^2\right) - 2\left(\epsilon + (u_j)^2\right)^{1-q/2}\left(\epsilon + v_j^2\right)^{q/2} - 2qv_j\left(u_j - v_j\right)}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}} \\
&= \frac{\left[2\epsilon + qu_j^2 + (2-q)u_j^2\right] - 2\left(\epsilon + (u_j)^2\right)^{1-q/2}\left(\epsilon + v_j^2\right)^{q/2} - \left[2qv_ju_j - 2qv_j^2\right]}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}},
\end{aligned} \tag{2.27}$$

then we have that (2.27) is equal to

$$\frac{qu_j^2 - 2qu_jv_j + qv_j^2}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}} + \frac{2\epsilon + (2-q)u_j^2 + qv_j^2 - 2\left(\epsilon + u_j^2\right)^{1-q/2}\left(\epsilon + v_j^2\right)^{q/2}}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}} = 0. \tag{2.28}$$

We can see that the first term of the sum at right part of equation (2.28) is nonnegative. In order to prove $u = v$, it remains to prove that second term of the sum is nonnegative. Note that $(1 - \frac{q}{2}) + \frac{q}{2} = 1$, then by Young's inequality we have

$$\begin{aligned}
&2\left(\epsilon + u_j^2\right)^{1-q/2}\left(\epsilon + v_j^2\right)^{q/2} \le 2\left[\frac{\left(\left(\epsilon + u_j^2\right)^{1-q/2}\right)^{2/2-q}}{1/(1-q/2)} + \frac{\left(\left(\epsilon + v_j^2\right)^{q/2}\right)^{2/q}}{2/q}\right] = \\
&= 2\left[(1-q/2)\left(\epsilon + u_j^2\right) + q/2\left(\epsilon + v_j^2\right)\right] = (2-q)\epsilon + (2-q)u_j^2 + qv_j^2 \le \\
&\le 2\epsilon + (2-q)u_j^2 + qv_j^2,
\end{aligned}$$

we can see that second term of the last equation (2.27) is nonnegative, then we get

$$\frac{q(u_j - v_j)^2}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}} = \frac{qu_j^2 - 2qu_jv_j + qv_j^2}{2\left(\epsilon + (u_j)^2\right)^{1-q/2}} = 0,$$

which implies $u_j = v_j$. Then we proved that $u_j = v_j$ for every $j \in \{1, 2, \ldots, n+1\}$ consequently $u = v$. We can conclude that u satisfies (2.13), therefore u is a critical point of (2.12). $\square$

Remark 2.2. In Theorem 1 we can also prove that the subsequences $\{u^{(k_i)}\}_i$, $\{u^{(k_i+1)}\}_i$, $\{u^{(k_i+2)}\}_i$, $\ldots$, $\{u^{(k_i+p)}\}_i$, $p \in \mathbb{N}$ of $\{u^{(k)}\}_k$, converges to the same point u, in order to show this we can repeat the same steps to prove that $\{u^{(k_i)}\}_i$, $\{u^{(k_i+1)}\}_i$ converges to the same point u, and repeat it for $\{u^{(k_i+1)}\}_i$, $\{u^{(k_i+2)}\}_i$, then for $\{u^{(k_i+2)}\}_i$, $\{u^{(k_i+3)}\}_i$ and so on. However, unlike stated in [26], this fact does not imply that all subsequences of $\{u^{(k)}\}_k$ converges to same point u.

Indeed, consider the next counter-example. Let $\{u^k\}_k \subset \mathbb{R}$ be a sequence defined as follows

$$u^k = \begin{cases} 1, \text{ if } k = 2^i - 1,\ i \in \mathbb{N} \\ 0, \text{ otherwise} \end{cases},$$

Clearly, the sequence $\{u^k\}_k$ does not converge. Let us show that there is a subsequence $\{u^{k_i}\}_i$ converging to some point, such that for every $p \in \mathbb{N}$, the subsequences $\{u^{(k_i)}\}_i$, $\{u^{(k_i+1)}\}_i$, $\{u^{(k_i+2)}\}_i$, $\ldots$, $\{u^{(k_i+p)}\}_i$ converges to the same limit point. For such purpose, take the subsequence $\{u^{(k_i)}\}_i$ with $k_i := 2^i$, $\forall i \in \mathbb{N}$. It is not difficult to see that $\lim\limits_{i\to\infty} u^{k_i} = 0$. Let $p \in \mathbb{N}$ be a fixed natural number and $i_0 \in \mathbb{N}$ be a scalar such that $p < 2^{i_0} - 1$. Note that $p < 2^{i_0} - 1 \leq 2^i - 1$ then $2^i + p < 2^{i+1} - 1$ for $i \geq i_0$, also observe that $2^i - 1 < 2^i + p$ then we have

$$2^i - 1 < k_i + p = 2^i + p < 2^{i+1} - 1, \text{ for all } i \geq i_0.$$

Thus, $u^{k_i+p} = 0$, $\forall i \geq i_0$ and hence $\lim\limits_{i\to\infty} u^{k_i+p} = 0$. □

Our goal now is to show that under sparsity assumptions on the critical point obtained from Algorithm 1. This critical point will be global minimizer of (2.12).

Lemma 2.2. *Let u, $y \in \mathbb{R}^{n+1}$. If both have sparsity $\|u\|_0 \leq n/2$, $\|y\|_0 \leq n/2$, then there exists a constant $C > 0$, which does not depend on u and y, such that*

$$\|u - y\| \leq \frac{1}{C}\|Hu - Hy\|,$$

where H is given by (2.4).

Proof: Let us define the matrix $G \in \mathbb{R}^{(2n+2)\times(2n+2)}$ as

$$H = \begin{pmatrix} H_0 & \alpha \\ \alpha^T & m \end{pmatrix}, \quad G = \begin{pmatrix} H & 0_{n+1} \\ I_{n+1} & I_{n+1} \end{pmatrix}$$

where H is given in (2.4), $H_0 = X^T X + \dfrac{1}{\rho} I \in \mathbb{R}^{n\times n}$, $\alpha = X^T e \in \mathbb{R}^{n\times 1}$, $m \in \mathbb{R}$, 0_{n+1} is the zero matrix in $\mathbb{R}^{(n+1)\times(n+1)}$ and $I_{n+1} \in \mathbb{R}^{(n+1)\times(n+1)}$ is the identity matrix.

We can see that $\|u - y\|_0 \leq n$. Without loss of generality, we can assume that $u - y = (\beta, 0)^T$, where $\beta \in \mathbb{R}^n$. Then we get

$$\|u - y\| = \|\beta\|\,. \tag{2.29}$$

Let $z \in \mathbb{R}^{2n+2}$ be the vector defined by

$$z = \left((u-y)^T, -(u-y)^T\right)^T = \left(\beta^T.0, -\beta^T, 0\right)^T$$

Hence, we obtain

$$Gz = \begin{pmatrix} H(u-y)^T \\ 0 \end{pmatrix} = \begin{pmatrix} H\left(\beta^T, 0\right)^T \\ 0 \end{pmatrix} = \begin{pmatrix} H_0\beta \\ \alpha^T\beta \\ 0 \end{pmatrix},$$

note that $H_0 = X^TX + \frac{1}{\rho}I$ is positive definite, then it admits an inverse matrix, then

$$\begin{pmatrix} H_0^{-1} & & \\ & 0 & \\ & & 0 \end{pmatrix} Gz = \begin{pmatrix} H_0^{-1} & & \\ & 0 & \\ & & 0 \end{pmatrix} \begin{pmatrix} H_0\beta \\ \alpha^T\beta \\ 0 \end{pmatrix} = \begin{pmatrix} \beta \\ 0 \\ 0 \end{pmatrix}. \tag{2.30}$$

Therefore by (2.29) and (2.30), we have

$$\|u-y\| = \|\beta\| \leq \left\|H_0^{-1}\right\| \cdot \|Gz\| = \left\|H_0^{-1}\right\| \cdot \|Hu - Hy\|.$$

Now, we can take $C = \left\|H_0^{-1}\right\|^{-1}$. □

Theorem 2. *Given $\epsilon > 0$, $q \in (0,1)$, define $\rho_{\epsilon,q} := \dfrac{C\epsilon^{1-q/2}}{q(2-q)}$ where C is given by Lemma 2.2. Then for every $\rho < \rho_{\epsilon,q}$, we have that*

$$f_q(\epsilon, y) - f_q(\epsilon, y) \geq \frac{C}{4\rho}\|y - u^{\epsilon,q}\|^2 \tag{2.31}$$

whenever $\|y\|_0, \|u^{\epsilon,q}\|_0 \leq n/2$.

Proof: We first calculate $f_q(\epsilon, y) - f_q\left(\epsilon, u^{\epsilon,q}\right)$ as the following

$$f_q(\epsilon, y) - f_q\left(\epsilon, u^{\epsilon,q}\right) = \|y\|_{q,\epsilon}^q + \frac{1}{2\rho}\|Hy - d\|^2 - \left[\|u^{\epsilon,q}\|_{q,\epsilon}^q + \frac{1}{2\rho}\left\|Hu^{\epsilon,q} - d\right\|^2\right],$$

then, we have that

$$\begin{aligned} f_q(\epsilon, y) - f_q\left(\epsilon, u^{\epsilon,q}\right) &= \left[\|y\|_{q,\epsilon}^q - \|u^{\epsilon,q}\|_{q,\epsilon}^q\right] + \left[\frac{1}{2\rho}\|Hy - d\|^2 - \frac{1}{2\rho}\left\|Hu^{\epsilon,q} - d\right\|^2\right] = \\ &= \sum_{j=1}^{n+1}\left[\left(\epsilon + y_j^2\right)^{q/2} - \left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{q/2}\right] + \frac{1}{\rho}\left(Hu^{\epsilon,q} - d\right)^T\left(Hy - Hu^{\epsilon,q}\right) + \frac{1}{2\rho}\left\|Hy - Hu^{\epsilon,q}\right\|^2. \end{aligned} \tag{2.32}$$

Now, computing the inner product between (2.13) and $(y - u^{\epsilon,q})$ we obtain

$$\frac{1}{\rho}\left(Hu^{\epsilon,q} - d\right)^T\left(Hy - Hu^{\epsilon,q}\right) = -\sum_{j=1}^{n+1}\frac{qu_j^{\epsilon,q}(y_j - u_j^{\epsilon,q})}{\left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{1-q/2}}, \tag{2.33}$$

substituting (2.33) into (2.32), and by inequality from Lemma 2.2, we have

$$
\begin{aligned}
&f_q(\epsilon, y) - f_q\left(\epsilon, u^{\epsilon,q}\right) = \\
&= \sum_{j=1}^{n+1}\left[\left(\epsilon + y_j^2\right)^{q/2} - \left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{q/2} - \frac{q u_j^{\epsilon,q}(y_j - u_j^{\epsilon,q})}{\left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{1-q/2}}\right] + \frac{1}{2\rho}\left\|Hy - Hu^{\epsilon,q}\right\|^2 \geq \\
&\geq \sum_{j=1}^{n+1}\left[\left(\epsilon + y_j^2\right)^{q/2} - \left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{q/2} - \frac{q u_j^{\epsilon,q}(y_j - u_j^{\epsilon,q})}{\left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{1-q/2}}\right] + \frac{C}{2\rho}\left\|y - u^{\epsilon,q}\right\|^2 \\
&= \sum_{j=1}^{n+1}\left[\left(\epsilon + y_j^2\right)^{q/2} - \left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{q/2} - \frac{q u_j^{\epsilon,q}(y_j - u_j^{\epsilon,q})}{\left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{1-q/2}} + \frac{C/2}{2\rho}\left(y_j - u_j^{\epsilon,q}\right)\right] + \frac{C/2}{2\rho}\left\|y - u^{\epsilon,q}\right\|^2.
\end{aligned}
\tag{2.34}
$$

Next, we will show that for a small enough ρ, each term in the sum of (2.34) is nonnegative. For this purpose, we define a function which corresponds to a term of the sum, and then we show that this function is nonnegative.

For a fixed $a \in \mathbb{R}$ define

$$
g(x) = \left(\epsilon + x^2\right)^{q/2} - (\epsilon + a)^{q/2} - \frac{qa(x-a)}{\left(\epsilon + a^2\right)^{1-q/2}} + \frac{C}{2\rho}(x-a)^2.
$$

We will prove that g has global minimizer at a and $g(a) = 0$. In fact, note that

$$
g'(x) = qx\left(\epsilon + x^2\right)^{q/2-1} - \frac{qa}{\left(\epsilon + a^2\right)^{1-q/2}} + \frac{C}{\rho}(x-a),
$$

hence

$$
g'(a) = qa\left(\epsilon + a^2\right)^{q/2-1} - \frac{qa}{\left(\epsilon + a^2\right)^{1-q/2}} + \frac{C}{\rho}(a-a) = 0.
$$

Also

$$
\begin{aligned}
g''(x) =& q\left(\epsilon + x^2\right)^{q/2-1} + 2qx^2\left(\frac{q}{2} - 1\right)\left(\epsilon + x^2\right)^{q/2-2} + \frac{C}{\rho} \\
=& q\left(\epsilon + x^2\right)^{q/2-1} + q(q-2)x^2\left(\epsilon + x^2\right)^{q/2-2} + \frac{C}{\rho}.
\end{aligned}
\tag{2.35}
$$

Observe that $q\left(\epsilon + x^2\right)^{q/2-1} \geq 0$ for every $x \in \mathbb{R}$ and

$$
0 \leq x^2\left(\epsilon + x^2\right)^{q/2-2} = \frac{x^2}{\left(\epsilon + x^2\right)^{2-q/2}} \leq \frac{x^2 + \epsilon}{\left(\epsilon + x^2\right)^{2-q/2}} = \frac{1}{\left(\epsilon + x^2\right)^{1-q/2}} \leq \frac{1}{\epsilon^{1-q/2}}
$$

then, the second term from the equation of (2.35) is bounded below, that is

$$
0 \geq q(q-2)x^2\left(\epsilon + x^2\right)^{q/2-2} \geq \frac{q(q-2)}{\epsilon^{1-q/2}},
$$

for every $x \in \mathbb{R}$. Thus, $g''(x) \geq -\dfrac{q(2-q)}{\epsilon^{1-q/2}} + \dfrac{C}{\rho}$, then $g''(x) > 0$ for every $x \in \mathbb{R}$ and $\rho > 0$ with $\rho < \dfrac{C\epsilon^{1-q/2}}{q(2-q)}$. Hence $g(x)$ is a convex function and since $g(a) = 0$, $g'(a) = 0$, then $a \in \mathbb{R}$ is a global minimizer of g. Therefore, we conclude $g(x) \geq 0$ for every $x \in \mathbb{R}$. □

Corolary 2.1. *Consider the parameter $\rho_{\epsilon,q}$ defined in Theorem 2. If $f_q(\epsilon,\cdot)$ admits a global solution v with $\|v\|_0 \leq \frac{n}{2}$ and $u^{\epsilon,q}$ is a limit point of a sequence generated by Algorithm 1, such that, $\|u^{\epsilon,q}\|_0 \leq \frac{n}{2}$. Then, $u^{\epsilon,q}$ is also a global minimizer of $f_q(\epsilon,x)$.*

Proof: In Theorem 2 we proved that for $\rho < \rho_{\epsilon,q}$ and y with sparsity $\|y\|_0 \leq n/2$, we have

$$f_q(\epsilon,y) - f_q\left(\epsilon,u^{\epsilon,q}\right) \geq \frac{C}{4\rho}\left\|y-u^{\epsilon,q}\right\|^2 \geq 0.$$

Take y equals to the global minimizer of $f_q(\epsilon,\cdot)$, i.e., $y=v$. By the previous inequality we get $f_q(\epsilon,v) \geq f_q\left(\epsilon,u^{\epsilon,q}\right)$, that is, $u^{\epsilon,q}=v$ is a global minimizer. $\square$

2.2.3 L_q-norm LS-SVM analysis when $\epsilon \to 0_+$ and $q \to 0_+$

Our goal in this subsection is to study the behavior of the problem (2.12) as $\epsilon \to 0_+$ and $q \to 0_+$. Also we present how the problems (2.12) and (2.9) are related.

For the purpose of this subsection we will need to use the concept of $\Gamma-$convergence [17].

Definition 2.1. *Let (X,d) be a metric space with metric d. We say that a sequence of functionals $E_k : X \to [-\infty,\infty]$ is $\Gamma-$convergent to a functional $E : X \to [-\infty,\infty]$ as $k \to \infty$, if for all $u \in X$, the following holds*

1. *For each sequence $\{u^k\}_k \subset X$ converging to u,*

$$E(u) \leq \liminf_k E_k\left(u^k\right).$$

2. *There exists a sequence $\{u^k\}_k \subset X$ converging to u such that*

$$E(u) \geq \limsup_k E_k\left(u^k\right).$$

Lemma 2.3. *If a sequence of functionals $\{E_k\}_k$ is $\Gamma-$convergent to a functional E on X as $k \to \infty$, then for any subsequence $\{E_{k_j}\}_j$ of $\{E_k\}_k$, we have*

$$\limsup_{k_j\to\infty} \inf_{u\in X} E_{k_j}(u) \leq \inf_{v\in X} E(v).$$

Proof: For any vector $v \in X$, by definition of $\Gamma-$ convergence, there exists $\{u^k\}_k$ converging to v such that

$$\limsup_{k\to\infty} E_k\left(u^k\right) \leq E(v).$$

Note that $\inf\limits_{u\in X} E_{k_j}(u) \leq E_{k_j}(u^{k_j})$, therefore

$$\limsup_{k_j\to\infty} \inf_{u\in X} E_{k_j}(u) \leq \limsup_{k_j\to\infty} E_{k_j}\left(u^{k_j}\right) \leq \limsup_{k\to\infty} E_k\left(u^k\right) \leq E(v).$$

Since v is arbitrary, we obtain that

$$\limsup_{k_j\to\infty} \inf_{u\in X} E_{k_j}(u) \leq \inf_{v\in X} E(v).$$

$\square$

An important consequence of a $\Gamma-$convergent of functionals is the following

Lemma 2.4. *Suppose that a sequence of functionals $\{E_k\}_k$ is $\Gamma-$convergent to a functional E on X as $k \to \infty$. Let $\{E_{k_j}\}_j$ be a subsequence of $\{E_k\}_k$ and u^{k_j} be a minimizer of E_{k_j}. If the sequence $\{u^{k_j}\}_j$ converges to u on X, then u is a minimizer of E.*

Proof: By definition of $\Gamma-$convergence

$$E(u) \leq \liminf_{k_j\to\infty} E_{k_j}\left(u^{k_j}\right) \leq \limsup_{k_j\to\infty} E_{k_j}\left(u^{k_j}\right) = \limsup_{k_j\to\infty} \inf_{v\in X} E_{k_j}(v) \leq \inf_{v\in X} E(v)$$

The first inequality follows from definition $\Gamma-$convergence and the last one follows from Lemma 2.3 $\square$

The next two lemmas will help us to prove that a minimizer of the problem (2.11) can be approximate by a sequence of critical points of (2.12) generated by Algorithm 1.

Lemma 2.5. *Let $E_l : \mathbb{R}^n \to \mathbb{R}$ defined by $E_l(u) := f_q(\epsilon_l, u) = \|u\|_q^{q,\epsilon_l} + \frac{1}{2\rho}\|Hu - d\|^2$ be a sequence of functionals. If $\epsilon_l \to 0$, then E_l is $\Gamma-$convergent to the functional E where $E := f_q(0, u)$.*

Proof: Let $u \in X$ and suppose that $\{u^l\}_l$ is a sequence converging to $u \in \mathbb{R}^{n+1}$. By definition

$$E(u) := f_q(0, u) = \frac{1}{2\rho}\|Hu - d\|^2 + \|u\|_q^q$$

and

$$E_l(u) := f_q(\epsilon_l, u) = \frac{1}{2\rho}\|Hu - d\|^2 + \|u\|_q^{q,\epsilon_l}$$

We can see that E_l is $\Gamma-$convergent to E as $l \to \infty$. Indeed,

$$f_q(\epsilon_l, u) \geq \frac{1}{2\rho}\|Hu - d\|^2 + \|u\|_q^q,$$

it follows that

$$\liminf_{l\to\infty} f_q(\epsilon_l, u^{(l)}) \geq f_q(0, u).$$

Thus, item 1 of the definition of $\Gamma-$convergence 2.1 is satisfied.

On other hand, for any u, take $u^{(l)} = u$ for $l \in \mathbb{N}$, then

$$\limsup_{l\to\infty} E_l(u^l) = \limsup_{l\to\infty} f_q\left(\epsilon_l, u^{(l)}\right) = \limsup_{l\to\infty} f_q\left(\epsilon_l, u\right) = f_q(0, u) = E(u)$$

and therefore item 2 of definition of $\Gamma-$convergence 2.1 holds and E_l is Γ-convergent to E. $\square$

Now consider the problem (2.11) and let $u^{\epsilon_l,q}$ be a critical point of (2.11), with $\|u^{\epsilon_l,q}\|_0 \leq n/2$, obtained through Algorithm 1, and a random initialization $u^{(0)}$, that is, $u^{(k_j)} \to u^{\epsilon_l,q}$, where $\{u^{(k_j)}\}_j$ is a subsequence of a sequence generated by Algorithm 1.

Lemma 2.6. *Consider $\{\epsilon_l\}_l$ a sequence with $\epsilon_l \to 0$. Suppose that for every ϵ_l, $l \in \mathbb{N}$, the problem* (2.12) *has a global minimizer v with sparsity $\|v\|_0 \leq n/2$. If $\{u^{\epsilon_l,q}\}_l$ is a sequence of critical points of* (2.11) *with $\|u^{\epsilon_l,q}\|_0 \leq n/2$, then $\{u^{\epsilon_l,q}\}_l$ is bounded and consequently has a convergent subsequence.*

Proof: To prove the result is enough to show that $\{u^{\epsilon_l,q}\}_l$ is bounded, with respect to ϵ_l. Take $z \in \mathbb{R}^{n+1}$ fixed. Since (2.12) has a global minimizer v with sparsity $\|v\|_0 \leq n/2$ for every ϵ_l and $\|u^{\epsilon_l,q}\|_0 \leq n/2$, then by Theorem 2, we get

$$\|u^{\epsilon_l,q}\|_q^q \leq \sum_{j=1}^{n+1} \left(\epsilon_l + \left(u_j^{\epsilon_1,q}\right)^2\right)^{q/2} \leq f_q\left(\epsilon_l, u^{\epsilon_l,q}\right) \leq f_q\left(\epsilon_l, z\right). \tag{2.36}$$

Hence we proved that the sequence $\{u^{\epsilon_l,q}\}_l$ is bounded, consequently there exists a convergent subsequence $\{u^{\epsilon_{l_j},q}\}_j \subset \{u^{\epsilon_l,q}\}_l$, therefore $\lim\limits_{j\to\infty} u^{\epsilon_{l_j},q} = u$ □

Theorem 3. *Let $\rho > 0$ and $q \in (0,1)$. Suppose that there exists a sequence $\{\epsilon_l\}_l$ with $\epsilon_l \to 0$, such that, for every ϵ_l:*

1. *The problem* (2.12) *admits a global minimizer $v \in \mathbb{R}^{n+1}$ with $\|v\|_0 \leq n/2$;*
2. *The Algorithm 1 generates a critical point $u^{\epsilon_l,q}$, with $\|u^{\epsilon_l,q}\|_0 \leq n/2$.*

Then, every limit point of $\{u^{\epsilon_l,q}\}_l$ is a minimizer of (2.11).

Proof: Since the hypothesis of Lemma 2.6 are satisfied, the sequence $\{u^{\epsilon_l,q}\}_l$ is bounded. Thus, this sequence has a limit point u^q. Now, we will show that u^q is a minimizer of (2.11). By Lemma 2.5, $f_q(\epsilon_l, u)$ is $\Gamma-$convergent to $f_q(0,u)$. By Lemma 2.4, u^q will be a minimizer of (2.11), if $u^{\epsilon_l,q}$ is a minimizer of $f_q(\epsilon_l, u)$, but this is a consequence of the sparsity of $u^{\epsilon_l,q}$ and Theorem 2. □

Now, let $\{u^q\}_q$ be a sequence of minimizers of (2.11) with $0 < q < 1$. We next show that every limit point of $\{u^q\}_q$ is a minimizer of (2.9).

Theorem 4. *For every $q \in (0,1)$, suppose that there exists a sequence $\{\epsilon_l^q\}_l$ with $\epsilon_l^q \to 0$, such that, for every ϵ_l^q, $l \in \mathbb{N}$, the problem* (2.12) *has a global minimizer v with sparsity $\|v\|_0 \leq n/2$. Denote by u^q a limit point of $\{u^{\epsilon_l^q,q}\}_l$. Then, $\{u^q\}_q$ is bounded and every limit point is a minimizer of the problem* (2.9).

Proof: Let $\epsilon_l^q \to 0$ in (2.36), as result we have that $\{u^q\}_q$ is bounded for $0 < q < 1$, hence $\{u^q\}_q$ has a convergent subsequence. Thus, in order to prove the result, by Lemma 2.4, we have to prove that $f_q(0,u)$, $0 < q < 1$ is $\Gamma-$convergent to $f_0(0,u)$ as $q \to 0_+$. Suppose that $\{u^q\}_q$ is the sequence that converges to $u^{(*)}$, then $\|Hu - d\|^2$ converges to $\left\|Hu^{(*)} - d\right\|^2$. Let $u^{(*)} = (u_1^{(*)}, u_2^{(*)}, \ldots, u_{n+1}^{(*)})$ and $\delta = \min\{|u_j^{(*)}| > 0\}$, then for a small enough q we get

$$f_q\left(0, u^q\right) \geq \sum_{\left|u_j^{(s)}\right|>0} |u_j^q|^q + \frac{1}{2\rho}\left\|Hu^q - d\right\|^2 \geq \sum_{\left|u_j^{(s)}\right|>0} \left|\frac{1}{2}\delta\right|^q + \frac{1}{2\rho}\left\|Hu^q - d\right\|^2$$

Since $u^q \to u^{(0)}$ as $q \to 0_+$, it follows

$$\liminf_{q\to 0_+} f_q\left(0, u^q\right) \geq \sum_{\left|u_j^{(*)}\right|>0} 1 + \frac{1}{2\rho}\|Hu^{(*)} - d\|^2 = f_q\left(0, u^{(*)}\right)$$

On other hand, for any $u^{(*)}$, we can take $u^q = u^{(*)}$ for $0 < q < 1$, hence

$$\limsup_{q\to 0_+} f_q\left(0, u^q\right) = f_0\left(0, u^{(*)}\right).$$

Therefore, we proved that $f_q(0, u)$ is $\Gamma-$convergent to $f_0(0, u)$, then by Lemma 2.4 we have the claimed result. □

The previously results show us that sparsity is a key point, e.g., in Theorem 3 requires that the vector $u^{\epsilon_l,q}$ must satisfy $\|u^{\epsilon_l,q}\|_0 \leq n/2$. Hence, it is important to outline a practical way to set an attribute of the critical point $u^{\epsilon,q}$ as being equals to zero or not. The following theorem shows how ϵ may be used to determine nonzero elements of $u^{\epsilon,q}$.

Theorem 5. *Let us define*

$$Q = \max_{1\leq j\leq n+1} 2\|h_j\|_2^2 f_q\left(1, u^{(0)}\right), \tag{2.37}$$

where $H = (h_1, h_2, \ldots, h_{n+1})$ is defined in (2.4), (h_j, $j \in \{1, \ldots, n+1\}$ are the columns of H) and $u^{(0)} \in \mathbb{R}^{n+1}$ is the initial vector in Algorithm 1. Also let

$$B = \left(\frac{2^{2-q}Q}{q}\right)^{\frac{1}{1-q}} > 0. \tag{2.38}$$

If $B\epsilon < 1$, then for $j = 1, 2, \ldots, n+1$ the j-th element of the critical point $u^{\epsilon,q} = (u_1^{\epsilon,q}, \ldots, u_{n+1}^{\epsilon,q})$ of problem (2.12) generated by Algorithm 1 satisfies

$$\left|u_j^{\epsilon,q}\right|^2 > \frac{1}{B} = \frac{1}{B\epsilon}\epsilon > \epsilon \quad \text{or} \quad \left|u_j^{\epsilon,q}\right|^2 < (B\epsilon)^{1-q}\epsilon < \epsilon.$$

Proof: Considering the j-th element of the vector in (2.13), we have

$$q\left(\frac{u_j^{\epsilon,q}}{\left(\epsilon + \left(u_j^{\epsilon,q}\right)^2\right)^{1-q/2}}\right)^2 \leq \frac{1}{\rho}\|h_j\|_2^2\|Hu^{\epsilon,q} - d\|_2^2 \leq 2\|h_j\|_2^2 f_q\left(\epsilon, u^{\epsilon,q}\right) \leq 2\|h_j\|_2^2 f_q\left(\epsilon, u^{(0)}\right) \leq Q \tag{2.39}$$

where $j = 1, 2, \ldots, n+1$. The last inequality (2.39) comes from (2.37). Now we will prove the conclusion by two cases.

Case 1: If $|u_j^{\epsilon,q}|^2 > \epsilon$, then by (2.39), we have

$$\frac{q}{2^{2-q}} \cdot \frac{1}{\left|u_j^{\epsilon,q}\right|^{2(1-q)}} = \frac{q\left|u_j^{\epsilon,q}\right|^2}{\left(2\left|u_j^{\epsilon,q}\right|^2\right)^{2-q}} < \frac{q\left|u_j^{\epsilon,q}\right|^2}{\left(\epsilon + \left|u_j^{\epsilon,q}\right|^2\right)^{2-q}} < Q$$

It follows by (2.38) that $|u_j^{\epsilon,q}|^2 > (q/2^{2-q}Q)^{\frac{1}{1-q}} = \dfrac{1}{B}$

Case 2: If $|u_j^{\epsilon,q}|^2 \leq \epsilon$, then by (2.39), we have

$$\frac{q\left|u_j^{\epsilon,q}\right|^2}{(2\epsilon)^{2-q}} \leq \frac{q\left|u_j^{\epsilon,q}\right|^2}{(\epsilon+\left|u_j^{\epsilon,q}\right|)^{(2-q)}} < Q.$$

It follows by (2.38) that $\left|u_j^{\epsilon,q}\right|^2 < \left(\frac{2^{2-q}Q}{q}\right)\epsilon^{2-q} = (B\epsilon)^{1-q}\epsilon$ □

Remark 2.3. From Theorem 5, we can see that parameter $\epsilon > 0$ can be used as estimator of nonzero entries of critical point $u^{\epsilon,q}$. Indeed suppose that $\epsilon > 0$ is small enough such that $1/B\epsilon$ is much larger than 1, we have two cases

1. From Theorem 5, we get that $\left|u_j^{\epsilon,q}\right|^2 > \dfrac{1}{B\epsilon}\epsilon \gg \epsilon$ since $\dfrac{1}{B\epsilon} \gg 1$, thus, $\left|u_j^{\epsilon,q}\right|^2$ is much larger than ϵ.

2. If $\left|u_j^{\epsilon,q}\right|^2 < \epsilon$, by Theorem 5, $\left|u_j^{\epsilon,q}\right|^2 < (B\epsilon)^{1-q}\epsilon$, since $B\epsilon \ll 1$. Thus $\left|u_j^{\epsilon,q}\right|^2 < (B\epsilon)^{1-q}\epsilon < (B\epsilon)\epsilon \ll \epsilon$ therefore $\left|u_j^{\epsilon,q}\right|^2$ is much smaller than ϵ.

We can conclude that there exists a gap between ϵ and $\dfrac{1}{B\epsilon}\epsilon$, that is, ϵ help us to determine if an entry is zero or nonzero. In our implementation, we consider $u_j^{\epsilon,q} = 0$, if $\left|u_j^{\epsilon,q}\right|^2 < \epsilon$. See Chapter 3, Section 3.4 (Numerical experiments).

Chapter 3

EMPIRICAL STUDY

In this chapter we will present our numerical experiments. We start describing our methodology and data sets that we use to perform numerical experiments, next we present an overview of comparisons models and finally we introduce our numerical experiments.

3.1 Methodology

Given a data set $S = \{(x^k, y_k)\}_{k=1}^n$, we define the *error rate* or *empirical error* of a classification model by

$$Err(S) = \frac{1}{n}\sum_{k=1}^{n} i(\hat{y}_k, y_k),$$

where $\hat{y_k}$ is the predicted label using the classifier at x^k and $i : \mathbb{R} \times \mathbb{R} \to \{0, 1\}$ is defined as follows

$$i(z,t) = \begin{cases} 0, & \text{if } z = t \\ 1, & \text{otherwise} \end{cases}, \text{ for } z, t \in \mathbb{R}.$$

Our goal is to test the L_q-norm LS-SVM feature selection ability and classification *accuracy*, which is defined as the percentage of correctly classified data, i.e.,

$$Accuracy = 1 - Err(S).$$

Now, observe that if the model has a small error rate, then it has a large accuracy, however, we only know the error rate on the training data and there is no guarantee that a model which has a small error rate on training data also has a small error rate on unseen data (*test data*). In fact, it is hard to estimate the error rate on test data (*test error rate*), however the following two techniques can help us to calculate a more accurate test error rate estimation as pointed in [7, 10, 12].

The *Holdout method* consists in divide the data into two subsets, a training set and a validation set. The accuracy is estimate as following: First, we determine the classifier using the training data, second the classifier is used to predict the labels of the validation set. The percentage of correctly classified validation data is taken to be the estimate accuracy on unseen data.

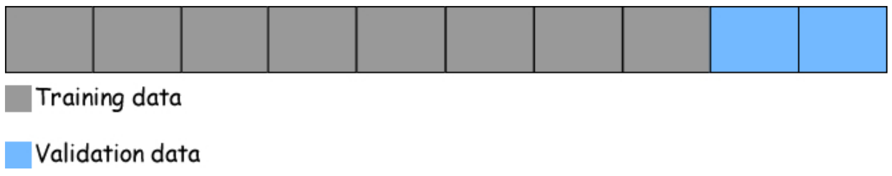


Figure 3.1: An illustration of Holdout method.

The *K-fold Cross-Validation* method, [10, 12], consists in randomly divide the data into $K \in \mathbb{N}$ groups or *folds* with approximately same size, then it takes the first fold to be the validation set and the remaining $K - 1$ folds to be training set, then it uses the validation set to compute and evaluate the accuracy as Holdout method. This procedure is repeated K times, thus it will generates K accuracies. The estimated accuracy of the classifier on unseen is estimated as being the average of K generated accuracies during the process.

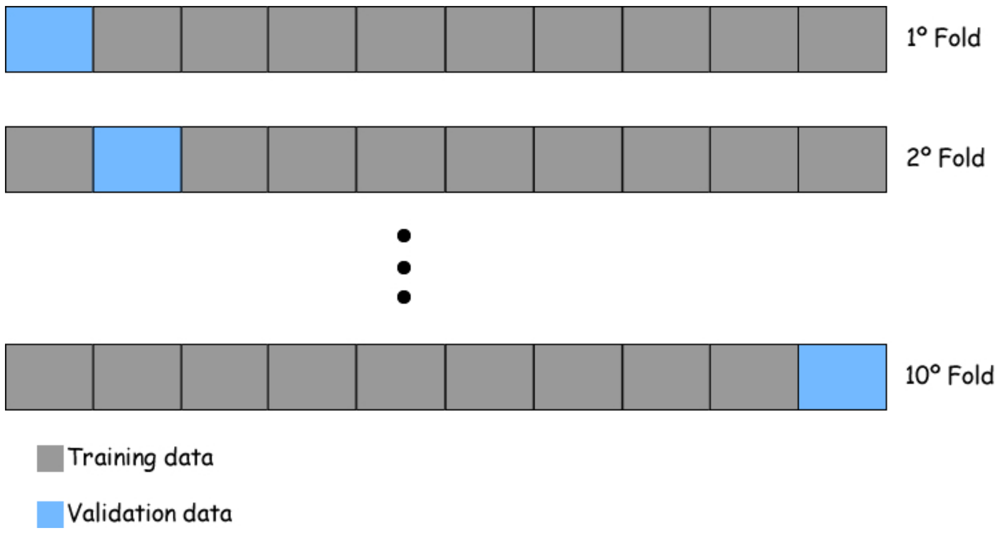


Figure 3.2: An illustration of 10-fold Cross-Validation method.

Observe that techniques (Holdout and K-fold Cross-Validation) do not depend on each other, that is, we can use only K-fold cross-validation as well as only Holdout method.

Now, we will outline our methodology, which was inspired in [1, 8, 10, 12, 26].

1. We separate the data set S randomly into two disjoint sets T and V where

$$T = \text{trainning data} \quad \text{and} \quad V = \text{validation data}.$$

   The set T contains 80% of the original data S, meanwhile V contains 20% of the original data S;

2. We set a discrete interval for each parameter, as described detailed in Section 3.4;

3. We build a grid where each node corresponds to a combination of parameters;

4. Now, we perform a *grid search*, that is, for each node of the grid, we perform 10-fold cross validation using the set T, next we take the mean of accuracies as being the model accuracy associated to each combination of parameters;

5. We select the node of the grid which correspond to a combination of parameters which yielded the highest 10-fold cross validation on the data set T;

6. We build the classifier using the selected parameters and the whole set T;

7. We test the model in validation data set V and then we take the resulting accuracy as being the model accuracy on the data set S.

3.2 Data Sets

In order to test and compare L_q-norm LS-SVM, we use artificial data sets which were generated following [26], describe in detail in Subsection 3.2.1, also for the same purpose we use real world data sets from [6]. In this section we will give a brief description of all data sets that we use in our numerical experiments.

3.2.1 Artificial Data Sets

We build 5 artificial data sets which are compound by 100 observations and the number of features are $100, 300, 500, 700, 1000$. All data sets are **balanced**, that is, we have 50 observations in positive class and 50 in negative class. For each data set, the first two features of positive class is uniformly distributed in the interval $[0, 1)$, for negative class the first two features are uniformly distributed in the interval $(1, 2]$, both we add 50% Gauss random noise as following: Consider the matrix $A \in \mathbb{R}^{m\times 2}$ where the first column contains the points of the positive class and second column contains the points of the negative class. Then, we generate[1] a matrix $G \in \mathbb{R}^{m\times 2}$, such that, the columns are normal random variables with mean 0 and matrix covariance equal to I. Next, we consider $\bar{A} = A + 0.5 \cdot G$ as being the new first two features of the data set. Observe that, if we consider just the first two features of positive and negative class, the data set is almost linear separable, see Figure 3.3. The remaining features are uniformly distributed in interval $[0, 2]$ with 50% Gauss random noise.

[1]We use the Matlab method *rand(m,n)*

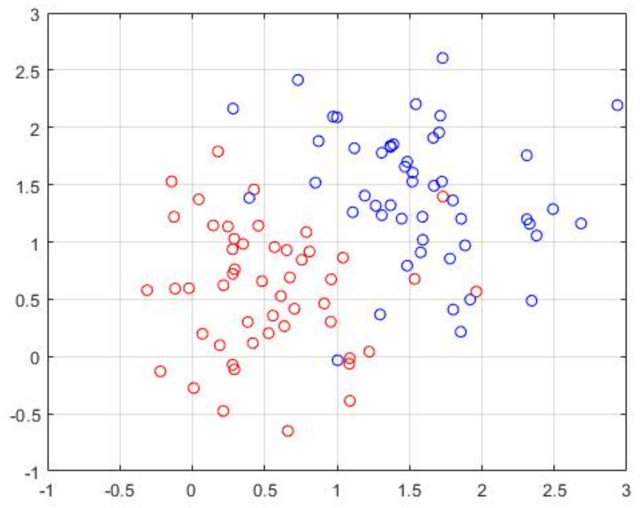


Figure 3.3: The artificial data set is almost separable when we consider just the first two features.

3.2.2 Real World Data Sets

- Cleveland Heart: It is a medical data set which contains information about patients, the goal is to determine the presence or absence of a predetermined heart disease. Thus we have a binary classification problem. The data set has 87 observations and 13 attributes.

- Ionosphere: It is a radar data collected by a system, the targets are free electrons found in the Ionosphere. It is a binary classification problem, the radar returns a presence or absence of some structure in Ionosphere . The data set has 351 observation and 34 attributes.

- LSVT Voice Rehabilitation: LSVT is a treatment for people with Parkinson's disease. This data set was collected from 14 participants. The goal is assess whether voice rehabilitation treatment lead to phonations considered "acceptable" or "unacceptable, that is, a binary class classification problem. The data set has 126 observations and 310 attributes.

More information about the data sets is available on UCI machine learning repository [6].

3.3 Overview of comparison models

In this section, we outline the models that we use to compare to the L_q-norm LS-SVM. Specifically, for each model we state the associated optimization problem. For all optimization problems we adopt the same notation introduced at beginning of Section 2.1

Remark 3.1. For SVMs models, a very well-known technique is the **Kernel Trick** [1, 3, 5, 23], which consists into map the set X to a space of larger dimension where the mapping set is linearly separable. However, we noticed that technique does not perform feature selection with a sparse vector w, since the classifier will not use the original features to classify.

3.3.1 Support Vector Machine (SVM)

The version that we use in our comparisons is that one studied at Chapter 1, Section 1.1. The SVM Soft Margin optimization problem (1.9) can be write as following

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{1}{2}\|w\|^2 + Ce^T\xi \\ \text{s.t} \quad & Y(Xw+eb) \geq e - \xi \\ & \xi \geq 0. \end{aligned}$$

For our comparisons, we use the implementation of [4].

3.3.2 Least Squares Support Vector Machine (LS-SVM)

The version that we use in our comparisons is that one studied at Chapter 1 Section 1.2. The LS-SVM optimization problem (1.9) can be write as following

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{1}{2}\|w\|^2 + \tfrac{\gamma}{2}\xi^T\xi \\ \text{s.t.} \quad & Y(Xw+eb) + \xi = e \\ & \xi \in \mathbb{R}^m. \end{aligned}$$

For our comparisons we use the implementation from [27].

3.3.3 Newton Method for Linear Programming Support Vector Machines (NLPSVM)

The Newton Method for Linear Programming Support Vector Machines (NLPSVM) introduced in [8], transform the quadratic optimization problem (1.9) into a linear minimization one. The motivation is that there exists an empirical evidence that this modification yields sparse classifiers than classical SVM as we study in Chapter 1.

We proceed by analyzing the main modification of (1.9). First, they replace the term $\|w\|_2$ by $\|w\|_1$, and in order to transform the problem into a linear one, it is employed a trick by replacing w and $\|w\|_1$ as follows

$$w = p - q$$

where the components $p_i,\, q_i \in \mathbb{R}$ of $p,\, q \in \mathbb{R}^n$ are defined as $p_i = \max(0, w_i)$, $q_i = |\min(0, w_i)|$, that is, p is the positive part of w and q is the negative one. Then, we have

$$\|w\|_1 = e^T(p+q)$$

where $p, q \geq 0$. Thus, we obtain the linear optimization problem

$$\begin{aligned} \min_{(p,q,b,\xi)} \quad & e^T(p+q) + Ce^T\xi \\ \text{s.t.} \quad & Y(X(p-q) - eb) + \xi \geq e, \\ & p, q, \xi \geq 0. \end{aligned}$$

where the dual optimization problem is given by

$$\begin{aligned} \max_{u \in R^m} \quad & e^T u \\ \text{s.t.} \quad & -e \leq X^T Y u \leq e \\ & -e^T Y u = 0 \\ & u \leq Ce \\ & u \geq 0. \end{aligned}$$

Instead of solving the dual/primal optimization problem, it is considered the next unconstrained optimization problem

$$\begin{aligned} \min_u f(u) = &- \rho e^T u + \frac{1}{2}\left\|\left(X^T Y u - e\right)_+\right\|^2 + \frac{1}{2}\left\|\left(-X^T Y u - e\right)_+\right\|^2 + \frac{1}{2}\left\|e^T Y u\right\|^2 \\ &+ \frac{1}{2}\left\|(u - Ce)_+\right\|^2 + \frac{\sigma}{2}\left\|(-u)_+\right\|^2, \end{aligned} \tag{3.1}$$

where ρ and σ are positive penalty parameters. The NLPSVM uses the Newton's method to solve (3.1). More details of the algorithm as well as Matlab's code can be found in [8].

3.4 Numerical Experiments

In this section we present our numerical experiments. We divided this section into two subsections. Results for artificial data sets and results for real world data sets. All the classification methods are implemented in Matlab 9.4 on a PC with Intel I7 processor (2.6 GHz) and with 8 GB RAM.

As we mentioned in Section 3.1, for each comparison model we have to determine a grid of parameters. In our experiments we use the following intervals

- For SVM, we consider the parameter $C \in \{10^i \,|\, i \in \{-8, -7, \ldots, 7, 8\}\}$;
- For LS-SVM we consider the parameter $\frac{\gamma}{2} \in \{10^i \,|\, i \in \{-8, -7, \ldots, 7, 8\}\}$;
- For NLPSVM, we follow [8] and consider the parameters $C \in \{2^i \,|\, i \in \{-12, -11, \ldots, 11, 12\}\}$, $\rho = 4 \cdot 10^{-4}$, $\sigma = 10^3$ and $\delta \in \{10^i \,|\, i \in \{-3, -2, \ldots, 2, 3\}\}$ where δ is used to calculate a modified Newton direction (See [8]);

- For L_q-norm LS-SVM, we follow [26] and consider the parameters $\gamma, \rho \in \{10^i \,|\, i \in \{-8, -7, \ldots, 7, 8\}\}$, $\epsilon \in \{10^i \,|\, i \in \{-6, -5, -4, -3, -2, -1\}\}$ and $q \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

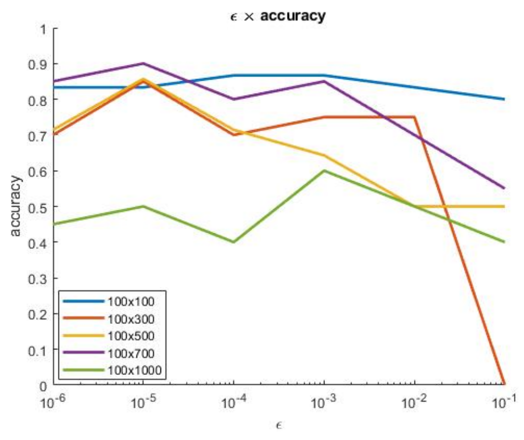
3.4.1 Results in Artificial Data Sets

	SVM	LS-SVM	NLPSVM	L_q-norm LS-SVM
Data sets	Accuracy(%)	Accuracy(%)	Accuracy(%)	Accuracy(%)
$m \times n$	Features	Features	Features	Features
	Train Time(sec)	Train Time(sec)	Train Time(sec)	Train Time(sec)
	80	75	90	93
100×100	100	100	9	7
	0.01	0.04	0.01	0.02
	75	65	90	85
100×300	300	300	18	36
	0.02	0.02	0.02	0.02
	80	75	82	85
100×500	500	500	17	36
	0.04	0.02	0.02	0.9
	70	65	92	95
100×700	700	700	10	50
	0.06	0.02	0.06	1.1
	65	60	87	70
100×1000	1000	1000	20	754
	0.09	0.06	0.06	1.5

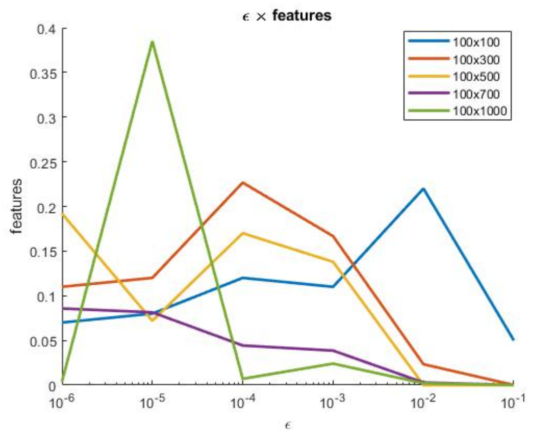
Table 3.1: Comparison among the models at same artificial sets. Accuracy is the percentage of the right classified data, Features is the number of selected features and train time is the CPU-time to determine the classifier.

In Table 3.1, we can see that L_q-norm LS-SVM has a good feature selection ability, as well as NLPSVM, it also reaches a good accuracy in most data sets, however the *train time* (the time to determine the classifier) increases fast as the number of features increases and all the others classfiers have a smaller train time than L_q-norm LS-SVM. The reason for this is that the system (2.15) depends on the number of features then it increases as number of features, increases and the complexity to solve the system also increases. The SVM and LS-SVM models select all the features, in sense that, the vector w does not have zero components, they have similar performance for accuracy and train time.

The NLPSVM shows a good feature selection ability and a high accuracy in all data sets, also it takes a small time to compute its classifier.

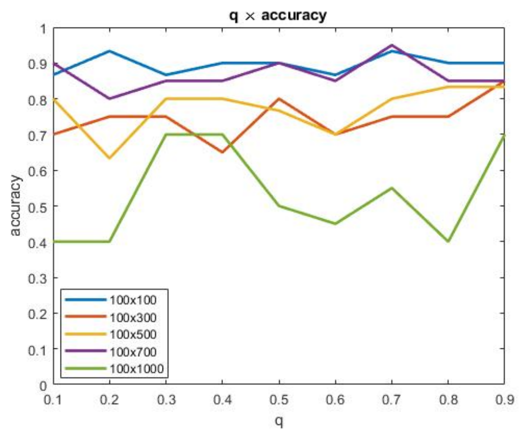


(a) The influence of parameter ϵ on accuracy.

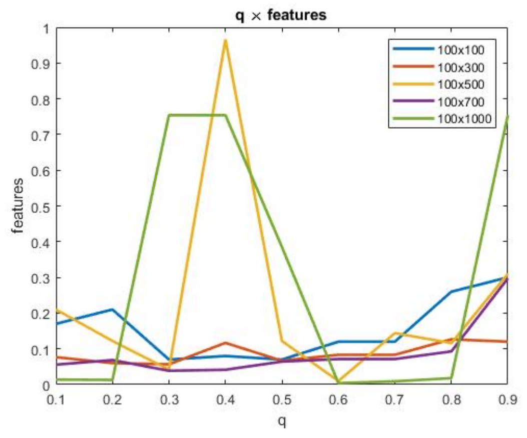


(b) The influence of parameter ϵ on selected features.

Figure 3.4: In both graphics we consider L_q-norm LS-SVM model. For a given parameter ϵ on the x-axis in the Figure (a) the accuracy associated to ϵ is the highest accuracy (y-axis) related to the parameters (ϵ, q, γ, ρ) in the grid. In Figure (b) the percentage of selected features, which correspond to the set of parameters (ϵ, q, γ, ρ) with highest accuracy.



(a) The influence of parameter q on accuracy.



(b) The influence of parameter q on selected features.

Figure 3.5: In both graphics we consider L_q-norm LS-SVM model. For a given parameter q on the x-axis in the Figure (a) the accuracy associated to q is the highest accuracy (y-axis) related to the parameters (ϵ, q, γ, ρ) in the grid. In Figure (b) the percentage of selected features, which correspond to the set of parameters (ϵ, q, γ, ρ) with highest accuracy.

In Figure 3.4, we can see as ϵ increases, for the most of data sets, the number of selected features decreases. This behavior is expected since in our implementation we set ϵ as a threshold for zero components of vector w, that is, we set the i-th component of w to be zero if $(w_i)^2 < \epsilon$, see Remark 2.3. Observe that when we have the number of selected features is small, the accuracy also is small. For example in Figure 3.4, when $\epsilon = 10^{-2}$ or $\epsilon = 10^{-1}$, the number of selected features is about 0, and in the most cases the accuracy is smaller than for other values of ϵ, that means that the model needs more features for a better classification.

In Figure 3.5, we expected that for a small q the number of selected features is small, and this number increases as q increases, this occurs for the data sets 100×300 and 100×700, also we note that for $q = 0.9$ the models select more features than others values of q, except for data sets 100×500 and 100×1000 that have some peaks. Observe that for a small number of selected features, the accuracy is not affected, this means that the model has selected useful features for classification.

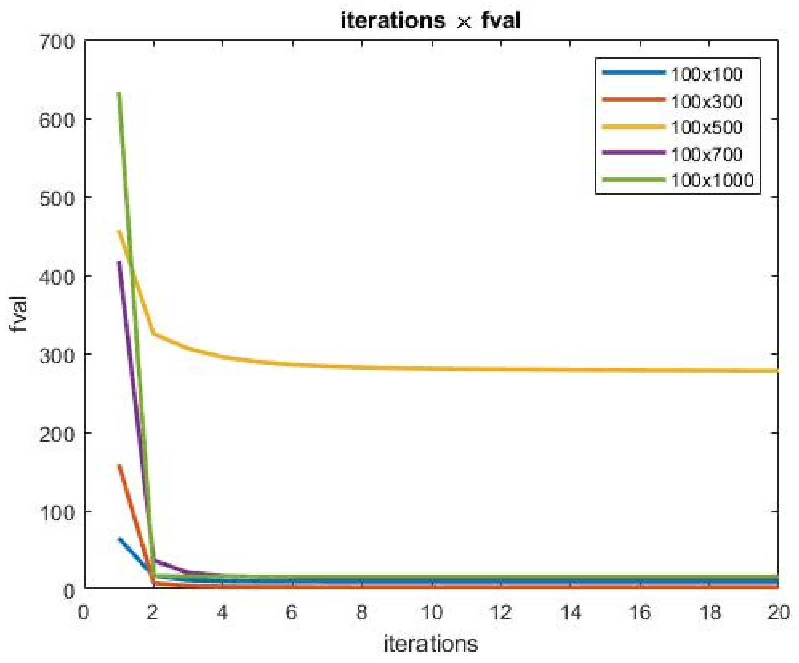


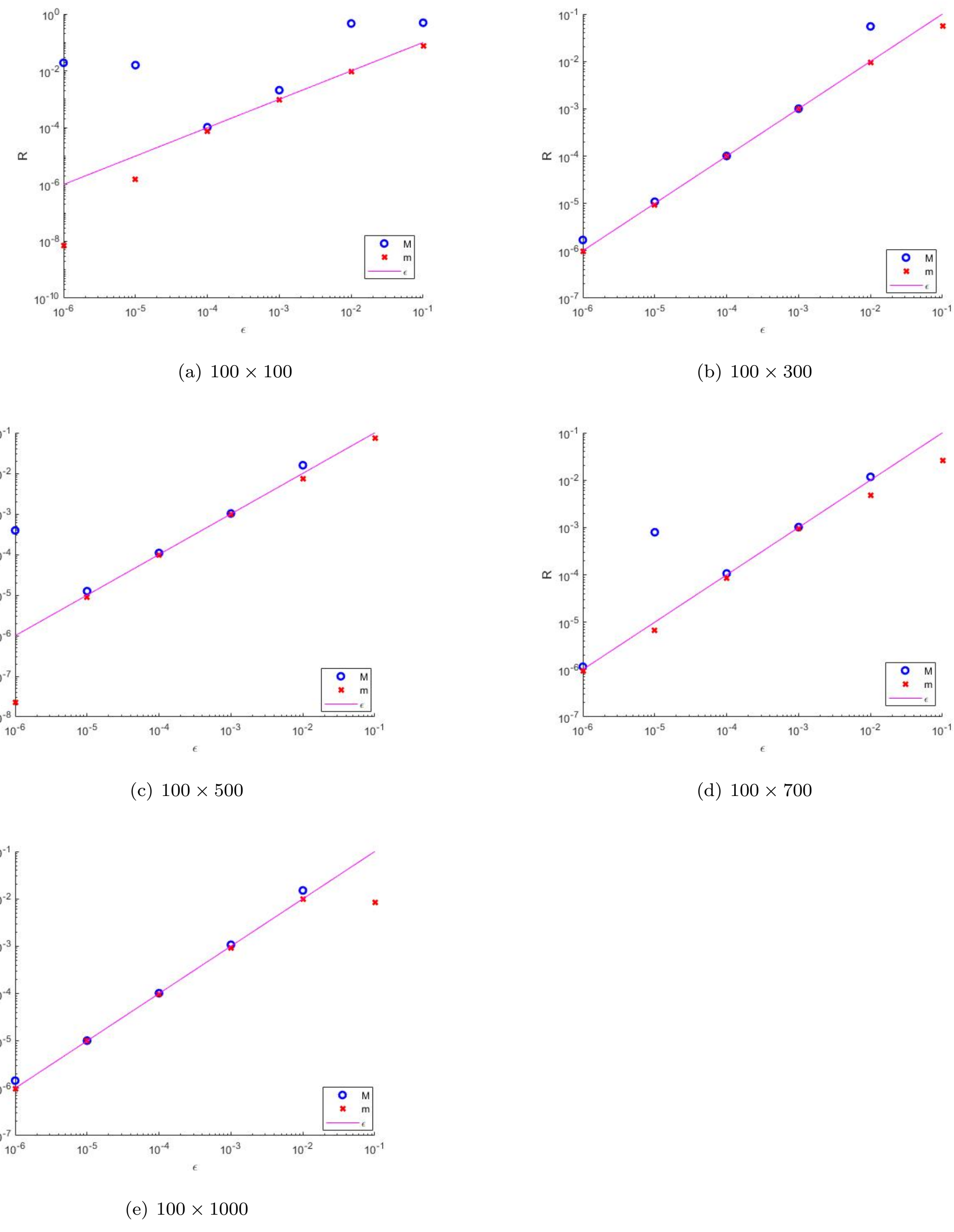
Figure 3.6: We consider the set of parameters $(\epsilon, q, \gamma, \rho)$ which yields the highest accuracy for each data set, then for $k \in \{0, 1 \ldots, 20\}$ we calculate $u^{(k)}$ using Algorithm 1, next we define fval $= \|u^{(k)}\|_{q,\epsilon}^q$.

In Figure 3.6, we consider the parameters combination which yields the highest accuracy for each data set. We can see as the increasing of iteration k the value of $\|u^{(k)}\|_{q,\epsilon}^q$ gradually converges, this illustrates the convergence of Theorem 1.

Now we illustrate the described gap in Theorem 5. Consider $u^{\epsilon,q} \in \mathbb{R}^{n+1}$ a critical point of

(2.12) obtained by Algorithm 1, and we recall from (2.5) that $w = (u_1^{\epsilon,q}, \dots, u_n^{\epsilon,q}) \in \mathbb{R}^n$. Let us define

$$M = \min_{j\in\{1,\dots,n\}} \{|w_j|^2 \, ; \, |w_j|^2 > \epsilon\} \text{ and } m = \max_{j\in\{1,\dots,n\}} \{|w_j|^2 \, ; \, |w_j|^2 \leq \epsilon\} \tag{3.2}$$



(a) 100 × 100

(b) 100 × 300

(c) 100 × 500

(d) 100 × 700

(e) 100 × 1000

Figure 3.7: The gap among m, M and ϵ as we pointed out in Remark 2.3, for each artificial data set.

In Figure 3.7 for each fixed parameter $\epsilon \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ we pick q, γ and ρ, such that, the classifier determined using the critical point $u^{\epsilon,q}$ obtained by Algorithm 1 has the highest accuracy for the fixed ϵ. As mentioned in Remark 2.3, under hypotheses of Theorem 5 the gaps among nonzero w components, the parameter ϵ and zero w components must be large, that occurs for some cases. In Figure 3.7 (a), for $\epsilon = 10^{-6}$ the nonzero component is about 10000 times larger than ϵ and the zero component is about 100 times smaller than ϵ. Observe that in Figures 3.7(b), (c), (d) and (e), for $\epsilon = 10^{-1}$ there is no a nonzero component, it occurs when no features were selected.

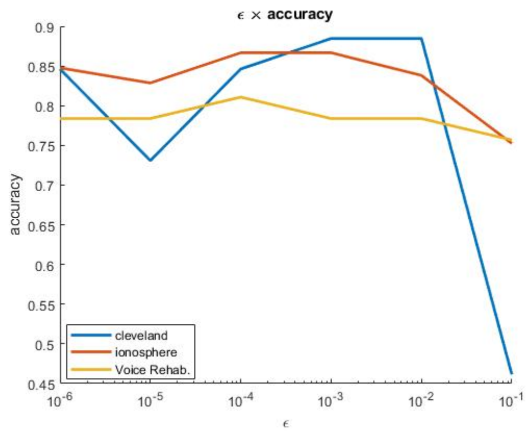
3.4.2 Results in Real World Data Sets

Data sets $m \times n$	SVM Accuracy(%) Features Train Time(sec)	LS-SVM Accuracy(%) Features Train Time(sec)	NLPSVM Accuracy(%) Features Train Time(sec)	L_q-norm LS-SVM Accuracy(%) Features Train Time(sec)
Cleveland Heart 87×13	88.23 13 0.01	87.5 13 0.02	88.24 8 0.01	88.46 7 0.01
Ionosphere 351×34	84.71 33 0.02	85.57 33 0.02	88.43 21 0.06	84.76 26 0.05
LSVT Voice Rehabilitation 126×310	85.6 310 0.4	85.67 310 0.3	86.80 23 0.02	81.08 23 0.3

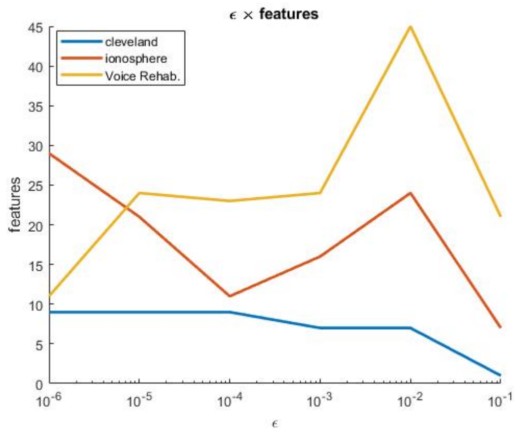
Table 3.2: Comparison among the models at same real data sets. Accuracy is the percentage of the right classified data, Features is the number of selected features and train time is the CPU-time to determine the classifier.

In Table 3.2, we see that L_q-norm LS-SVM perform well in real world data sets with respect to feature selection and accuracy. However, we cannot use these data sets with a huge number of features, for example *Arcene, Db World Mail* [6, 26], they have 10000 and 4702 features respectively, because our implementation (see Appendix A) took a long time to calculate a classifier (about 10 minutes), specifically it requires a large computation effort to evaluate $H^T H \in \mathbb{R}^{(n+1)\times(n+1)}$ that we need to solve the system (2.15), and then it would take a long time perform K-Fold cross-validation.

Remark 3.2. In our experiments we defined the Train Time used in Tables 3.1 and 3.2, as being the time to calculate the classifier, that is, we do not consider the time to find the best combination of parameters. We do this because we just want to compare the time that the algorithms take to compute the classifiers.

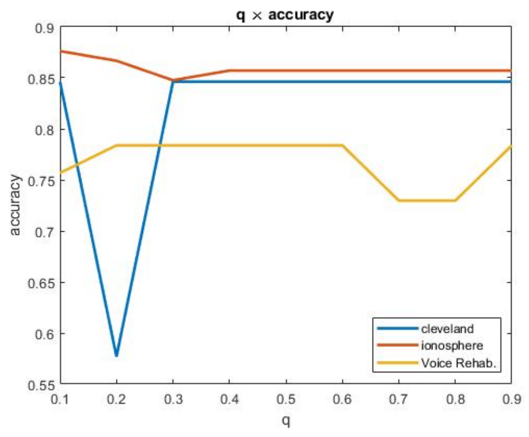


(a) The influence of parameter ϵ on accuracy

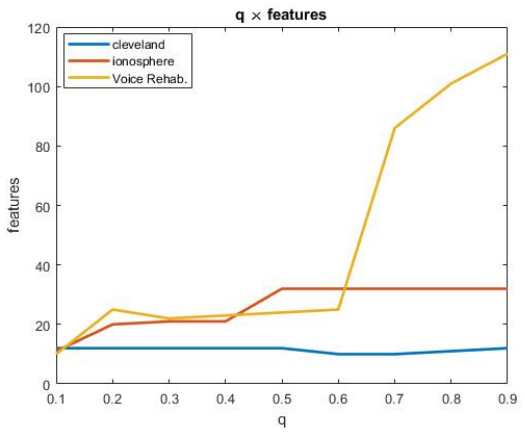


(b) The influence of parameter ϵ on selected features

Figure 3.8: In both graphics we consider L_q-norm LS-SVM model. For a given parameter ϵ on the x-axis in the Figure (a) the accuracy associated to ϵ is the highest accuracy (y-axis) related to the parameters (ϵ, q, γ, ρ) in the grid. In Figure (b) the percentage of selected features, which correspond to the set of parameters (ϵ, q, γ, ρ) with highest accuracy.



(a) The influence of parameter q on accuracy.



(b) The influence of parameter q on selected features.

Figure 3.9: In both graphics we consider L_q-norm LS-SVM model. For a given parameter q on the x-axis in the Figure (a) the accuracy associated to q is the highest accuracy (y-axis) related to the parameters (ϵ, q, γ, ρ) in the grid. In Figure (b) the percentage of selected features, which correspond to the set of parameters (ϵ, q, γ, ρ) with highest accuracy.

In Figure 3.8 (a) for all data sets we have the worst accuracy when $\epsilon = 10^{-1}$. In Figure 3.8

(b) for *Cleveland* data set we see that as ϵ increases the number of selected features decreases, also we see that when the number of selected features is small we do not have a good accuracy, it indicates that the number of selected features is not enough to describe the problem, that is, the classifier needs more features to produces a more accurate prediction. However we do not have that same pattern for *LSVT Voice Rehabilitation and Ionosphere* data sets.

In Figure 3.9 (a) we do not see a pattern that describes the accuracy. In Figure 3.9 (b) we can see clearly that the number of features increases as q increases for *LSVT Voice Rehabilitation and Ionosphere*, observe that when we have a small number of selected features we still have a good accuracy, for example in *LSVT Voice Rehabilitation* data set for $q = 0.2$ we have about 20 selected features and the accuracy about 80%, and for $q = 0.9$ we have about 100 selected features and the accuracy is about 80%, that means for $q = 0.2$ useful features were selected.

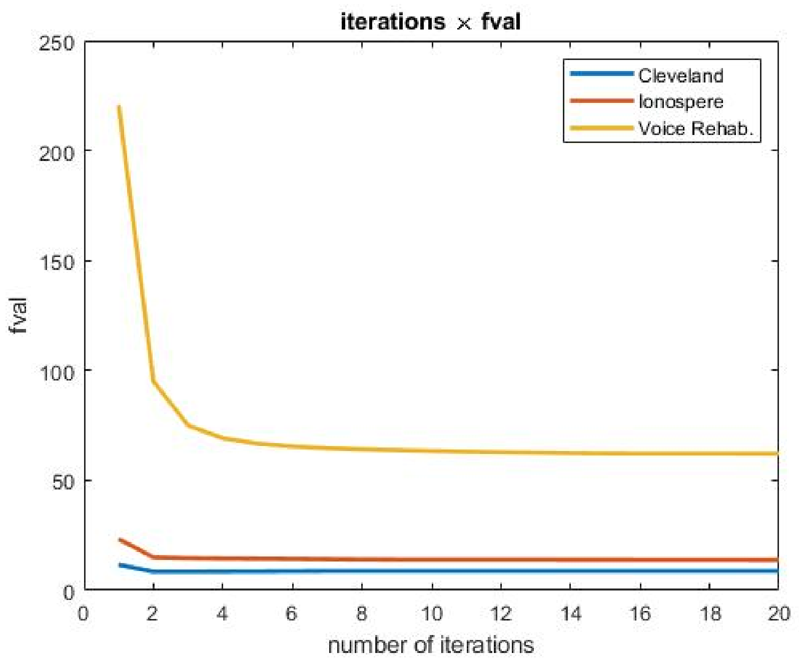
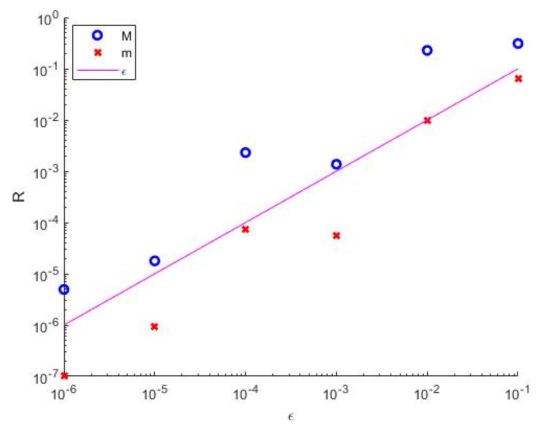


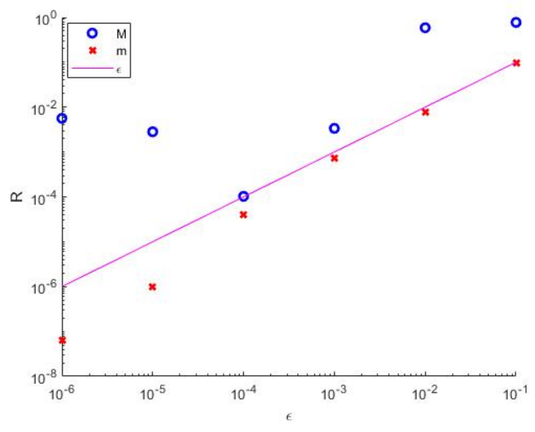
Figure 3.10: We consider the set of parameters $(\epsilon, q, \gamma, \rho)$ which yields the highest accuracy for each data set, then for $k \in \{0, 1 \ldots, 20\}$ we calculate $u^{(k)}$ using Algorithm 1, next we define fval $= \|u^{(k)}\|_{q,\epsilon}^q$.

In Figure 3.10, we illustrate the same idea than Figure 3.6, that is, for each real world data set we take the parameter combination which yield the highest accuracy. We can see that the value of $\|u^k\|_{q,\epsilon}^q$ decreases fast and get stable as well as for artificial data sets.

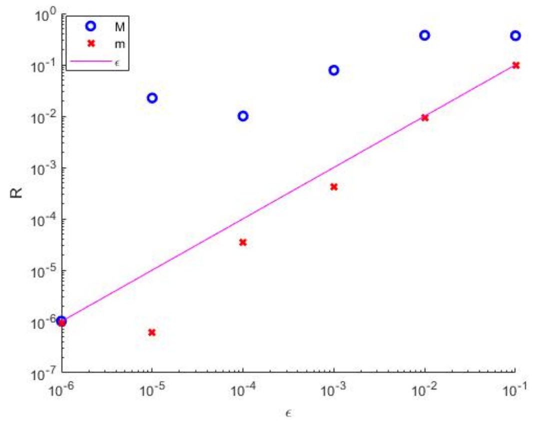
Consider M and m as defined in (3.2)



(a) Cleveland Heart



(b) Ionosphere



(c) LSVT Voice Rehabilitation

Figure 3.11: Gap among m, M and ϵ as we pointed out in Remark 2.3, for each real world data set

We took the parameters ϵ, q, ρ, γ as in Figure 3.7. We can see that for the most cases there is a clear gap between the smallest nonzero squared w component M and the greatest squared w component that we set to be zero m, that is, M, m are taken as defined in (3.2). For example, in Figure 3.11 (b) for $\epsilon = 10^{-6}$ we have that M is about 10^5 times greater than m. However, we cannot see a relationship between the value ϵ and the gaps size.

CONCLUSION

Our focus in this dissertation was to study the model L_q-norm Least Squares Support Vector Machine (L_q-norm LS-SVM) with feature selection [26] for classification problems, which is based on Least Squares Support Vector Machine (LS-SVM) [29], a variant of Support Vector Machine (SVM) [5].

In Chapter 1, we reviewed important concepts of SVM and LS-SVM theory. We started with the SVM simplest case: the SVM Hard Margin, which generates a linear classifier in a linearly separable data set, then we studied SVM Soft Margin, which generates a linear classifier for nonlinearly separable data sets. Lastly we studied LS-SVM which arose from some modifications of SVM Soft Margin optimization problem.

In Chapter 2, we presented our study over L_q-norm LS-SVM with feature selection. We showed that if we consider the linear classifier $y(x) = sign(w^T x + b)$, when w is a sparse vector, the classifier perform feature selection and classification simultaneously. Next, we studied the L_q-norm LS-SVM method which uses the L_q-norm with $0 < q < 1$ to find a sparse approximated solution of the LS-SVM optimization problem. We proved in Proposition 2.1 that a smooth version of L_q-norm optimization problem (2.12) admits solution. Then, we presented our study over the convergence results of L_q-norm LS-SVM method. In this part we made an important remark about Theorem 1.

In Chapter 3, we presented our empirical study. We started by describing our methodology that we used to test our implementation and compare to the others models as well as the data sets that we divide into two groups: Artificial and Real World data sets. Next, we briefly described the comparisons models and finally we presented our numerical experiments where we showed comparison table and some L_q-norm LS-SVM tests with respect to accuracy, features selection, convergence and sparsity.

We concluded that L_q-norm LS-SVM has a well structured theory, in sense that, an algorithm was proposed which searches an approximate sparse solution of LS-SVM also it was provided results with respect to method convergence. However, our numerical results were different from those presented in [26] with respect to training time, accuracy and the number of selected features. In most cases, the number of selected features of our algorithm were greater than the provided algorithm. The accuracy is not too good as presented in the article. The reason is because we have to use a different Matlab implementation than the one available in [26] and the reason of this is that the provided implementation does not follow the Algorithm 1. The matrix H used in the algorithm is not defined as in theory, also we used a different methodology because the one described in [26] was not clear for tests.

In our comparison, Tables 3.1 and 3.2, we noted that the model NLPSVM performs better than L_q-norm LS-SVM in almost all data sets, with respect to the number of selected features, accuracy and train time. However, this fact does not invalidate L_q-norm LS-SVM. Since we observed in bibliography that there is no a best classification model for every data set, then might there exists data sets where L_q-norm LS-SVM perform better than NLPSVM and others, with respect accuracy and number of selected features.

Finally, we observed that determining the parameters of a model through grid search can be an expensive task, especially, when the algorithm expend a long time to determine a classifier. Therefore, the study of a less expensive technique it is an interesting topic for a future work. Also, we noted that the weakness of the Algorithm 1 is to calculate the matrix multiplication $H^T H \in \mathbb{R}^{n+1 \times n+1}$ needed to solve system (2.15). Thus, we would consider an interesting topic for a future work the study of an approximated and less expensive computation technique for $H^T H \in \mathbb{R}^{n+1 \times n+1}$.

REFERENCES

[1] AK, S. J. et al. *Least squares support vector machines.* Singapore: World Scientific, 2002.

[2] BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear programming: theory and algorithms.* New Jersey: John Wiley & Sons, 2013.

[3] BISHOP, C. M. *Pattern recognition and machine learning.* New York: Springer, 2006.

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

[5] CRISTIANINI, N.; SHAWE-TAYLOR, J. et al. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge: Cambridge University Press, 2000.

[6] DUA, D.; GRAFF, C. UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, Last acess: 18/12/2019, 2017.

[7] FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. *The elements of statistical learning.* New York: Springer, 2001. v. 1.

[8] FUNG, G. M.; MANGASARIAN, O. L. A feature selection newton method for support vector machine classification. *Computational optimization and applications*, v. 28, n. 2, p. 185–202, 2004.

[9] GE, D.; JIANG, X.; YE, Y. A note on the complexity of Lp minimization. *Mathematical programming*, v. 129, n. 2, p. 285–299, 2011.

[10] JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. *An introduction to statistical learning.* New York: Springer, 2013. v. 112.

[11] JIAO, L.; BO, L.; WANG, L. Fast sparse approximation for least squares support vector machine. *IEEE Transactions on Neural Networks*, v. 18, n. 3, p. 685–697, 2007.

[12] KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Ijcai*, v. 14, n. 2, p. 1137–1145, 1995.

[13] KRULIKOVSKI, E. H. M. Análise teórica de máquinas de vetores suporte e aplicação a classificação de caracteres. *Dissertação de Mestrado, Universidade Federal do Paraná*, 2017.

[14] LIMA, E. L. *Geometria analítica e Álgebra linear.* Rio de Janeiro: IMPA, 2012. v. 2º Edição.

[15] LIU, J.; LI, J.; XU, W.; SHI, Y. A weighted lq adaptive least squares support vector machine classifiers–robust and sparse approximation. *Expert Systems with Applications*, v. 38, n. 3, p. 2253–2259, 2011.

[16] LORENA, A. C.; DE CARVALHO, A. C. Introduções máquinas de vetores suporte. *Sao Carlos-SP*, 2003.

[17] MING-JUN, L.; JINGYUE, W. An unconstrained ℓ_q minimization with $0 \leq q \leq 1$ for sparse solution of underdetermined linear systems. *SIAM Journal on Optimization*, v. 21, n. 1, p. 82–101, 2011.

[18] MOLLA, M.; WADDELL, M.; PAGE, D.; SHAVLIK, J. Using machine learning to design and interpret gene-expression microarrays. *AI Magazine*, v. 25, n. 1, p. 23–23, 2004.

[19] MURPHY, K. P. *Machine learning: a probabilistic perspective.* Massachusetts: MIT press, 2012.

[20] NATARAJAN, B. K. Sparse approximate solutions to linear systems. *SIAM journal on computing*, v. 24, n. 2, p. 227–234, 1995.

[21] NOCEDAL, J.; WRIGHT, S. *Numerical optimization.* New York: Springer, 2006.

[22] PAPPU, V.; PARDALOS, P. M. *High dimensional data classification.* New York: Springer, 2014.

[23] SCHOLKOPF, B.; SMOLA, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* Massachusetts: MIT press, 2001.

[24] SHAFAEY, M. A. et al. Deep learning for satellite image classification. *International Conference on Advanced Intelligent Systems and Informatics*, p. 383–391, 2018.

[25] SHALEV-SHWARTZ, SHAI; BEN-DAVID, S. *Understanding machine learning: From theory to algorithms.* Cambridge: Cambridge university press, 2014.

[26] SHAO, Y.-H. et al. Sparse Lq-norm least squares support vector machine with feature selection. *Pattern Recognition*, v. 78, n. 1, p. 167–181, 2018.

[27] SUYKENS, J. et al. Ls-svmlab toolbox user's guide: Version 1.8, 2011. *LS-SVMlab*, 2011. Available online: https://www. esat. kuleuven. be/sista/lssvmlab/ (accessed on 24 February 2020).

[28] SUYKENS, J. A.; LUKAS, L.; VANDEWALLE, J. Sparse least squares support vector machine classifiers. *ESANN*, p. 37–42, 2000.

[29] SUYKENS, J. A.; VANDEWALLE, J. Least squares support vector machine classifiers. *Neural processing letters*, v. 9, n. 3, p. 293–300, 1999.

[30] WAINWRIGHT, M. J. *High-dimensional statistics: A non-asymptotic viewpoint.* Cambridge: Cambridge university press, 2019.

Appendix A

LQ-NORM LS-SVM MATLAB IMPLEMENTATION

We present our Matlab implementation that we use in our comparisons and tests.

```
function [Predict_Y,w,b, wor] = mylqls(TestX,X,Y,FunPara)

% [Predict_Y,w,b]=mylqls(TestX,DataTrain,FunPara);
% Input:
%   TestX - Test data matrix which each row represents an input vector.
%   X - the input train data matrix.
%   Y - the train output/label vectors, the components should be +1 and -1.
%
% FunPara - Struct value in Matlab. The fields in options that can be set:
%   FunPara.epsilon:     small value the parameter in the Lq-norm LS-SVM.
%   FunPara.q:       (0,1)       the parameter in the Lq-norm LS-SVM.
%   FunPara.rho:     [0,inf)     the parameter in the Lq-norm LS-SVM.
%   FunPara.gamma:   [0,inf)     the parameter in the Lq-norm LS-SVM.
%
% Output:
%    Predict_Y - Predict value of the TestX.
%    w         - weight vector.
%    b         - bias.

[m,n] = size(X);
I= eye(n);
e = ones(m,1);
XTe = sum(X)'; % It is equivalent to  X^T*e
H=[X'*X+1/FunPara.gamma*I, XTe ; XTe', m];
d = [X' ;e']*Y;
H1 = sparse(H);
HTH = H1'*H1;
```

```
y = H'*d;
uk = rand(n+1,1);  % Initial guess u0
u = ones(n+1,1);
t = 1;
while t < 1000
    dn = (FunPara.epsilon + uk.^2).^(1 - FunPara.q/2);
    A = sparse(diag(nt./dn));
    M = A + sparse(HTH);
    u = uk;
    uk = M\ y;
    if norm(uk - u) < 1e-10 % tolerancy
         break
     end
    t = t+1;
 end
w = uk(1:n);
wor = w; % original w
b = uk(end);

% Remark at end of Chapter 2
 for i=1:n
     if abs(w(i))<sqrt(FunPara.epsilon)
        w(i)=0;
     end
end
Predict_Y = sign(TestX*w + b);
end
```