2013-01-01

A Convex Optimization Algorithm For Sparse Representation And Applications In Classification Problems

Reinaldo Sanchez Arias
University of Texas at El Paso, reinaldosanar@gmail.com

A CONVEX OPTIMIZATION ALGORITHM FOR SPARSE REPRESENTATION AND APPLICATIONS IN CLASSIFICATION PROBLEMS

REINALDO SANCHEZ ARIAS

Computational Science Program

APPROVED:

Miguel Argáez, Ph.D., Chair

Rodrigo Romero, Ph.D.

Martine Ceberio, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School

A mi amada madre Alid,

mi padre Reinaldo, y mi hermano Juan Camilo

que son la luz de mi vida.

A CONVEX OPTIMIZATION ALGORITHM FOR SPARSE REPRESENTATION AND APPLICATIONS IN CLASSIFICATION PROBLEMS

by

REINALDO SANCHEZ ARIAS

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science Program

THE UNIVERSITY OF TEXAS AT EL PASO

May 2013

Abstract

In pattern recognition and machine learning, a classification problem refers to finding an algorithm for assigning a given input data into one of several categories. Many natural signals are sparse or compressible in the sense that they have short representations when expressed in a suitable basis. Motivated by the recent successful development of algorithms for sparse signal recovery, we apply the selective nature of sparse representation to perform classification. Any test sample is represented in an overcomplete dictionary with the training sample as base elements. A given test sample can be expressed as a linear combination of only those training samples belonging to the same class, therefore providing a naturally sparse representation. Finding the correct coefficients in a given basis or training dataset, allows us to identify the correct category or class of any given input that needs to be categorized. In order to find such sparse linear representation, we implement an ℓ_1-minimization algorithm. This methodology overcomes the lack of robustness with respect to outliers, and in contrast to other classification algorithms, no model selection dependence is involved in the optimization method. The minimization algorithm is a convex relaxation-like algorithm that has been proven to efficiently recover sparse signals. To study its performance, the proposed method is applied to several test datasets with different number of features and samples. A dimensionality reduction technique is also proposed and implemented as part of the classification process.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many areas of engineering and applied mathematics benefit greatly from the results and methods developed in numerical optimization. Applications in signal processing, machine learning, economics, geophysics, biosciences, physical phenomena simulations, among others are enhanced and improved with the different achievements in numerical optimization.

Several engineering and science applications involve solving linear inverse problems that are usually ill-conditioned and for which the use of regularization techniques is required to be able to propose useful solutions. Recently, regularization via *sparsity* constraints has become very popular, where we look for an approximate solution to a linear system of equations, with the requirement that it has as few nonzero components as possible. This kind of problems can be found in several applications in machine learning, image and signal processing, and coding and information theory among others. Many natural signals are sparse or compressible in the sense that they have short representations when expressed in a suitable basis. Moreover, it has been proven that sparse signals can effectively approximate compressible signals [6], [9],[14].

The theory of *compressed sensing* (compressive sampling) has been a "hot" research topic of interest for many applied mathematics and engineering researchers in recent years. The work in this area was initiated in late 2004 by Emmanuel Cands, Justin Romberg, and Terence Tao [9], and independently by David Donoho [19]. The general theme aims to answer the question of how much information is necessary to accurately reconstruct a signal. Several encouraging numerical results followed by theoretical conditions and characterizations, showed that one can reconstruct sparse or compressible signals accurately from a very limited number of measurements. This motivated the rise of new techniques imaging sciences and

signal processing, based on compressed sensing and sparse representation methods, with a broad set of applications in engineering and sciences.

In machine learning and pattern recognition, the term "classification" refers to the result of an algorithm/technique for assigning a given set of input data into one of a given number of categories. An example would be assigning a given email into "spam" or "non-spam" classes, or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). An algorithm that implements classification is referred to as a classifier. Inspired by the recent successful development of algorithms for sparse signal recovery [17, 23, 28, 35], we apply the *selective nature* of sparse representation to perform classification. Any test sample is represented in an overcomplete dictionary with the training sample as base elements. In such way, test samples can be expressed as a linear combination of only those training samples belonging to the same class, therefore providing a naturally sparse representation. In order to find the sparsest linear representation we propose an optimization algorithm based on ℓ_1-minimization that allows us to overcome the lack of robustness to outliers [8]. Convex relaxation techniques based on ℓ_1 optimization have shown great results in compressed sensing and sparse representation problems. The use and development of ℓ_1 optimization approaches present several advantages over nonconvex optimization and greedy pursuit methodologies also proposed to attack the sparse signal reconstruction problem.

Sparse representations of signals have received a great deal of attention in recent years. The sparse representation problem consists in searching for the most compact representation of a signal in terms of a linear combination of *atoms* in an overcomplete *dictionary*.

Research has focused on *pursuit methods* for solving the optimization problem, such as matching pursuit [37], orthogonal matching pursuit [40], basis pursuit [17], and also on the *applications* of a sparse representation for different tasks, such as signal separation, denoising, and coding.

This dissertation is organized as follows:

Chapter 2 presents the mathematical background in the theory of compressed sensing that

gave rise to the development of efficient optimization algorithms for sparse signal recovery. We explain the formulation for the sparse representation problem, the ideas guaranteeing the recovery of sparse signals via ℓ_1 minimization, and some of the strategies to solve this problem.

Chapter 3 includes a description of the novel ℓ_1-minimization algorithm we propose to use for solving the classification problem. Convergence results for our algorithm and comparisons with other state-of-the-art solvers are also presented.

Chapter 4 explains the classification problem we aim to solve using a sparse representation approach. We motivate the use of the sparse representation in classifications tasks, and show how to use the ℓ_1 minimization algorithm developed in this work for solving classification problems. A description of the mathematical formulation and the strategies used are presented.

Chapter 5 presents numerical results of the classification technique we propose in this work for different datasets. We explain the experiment design, describe the datasets used, and present a comparison of our results with commonly used algorithms for classification.

Chapter 6 includes a discussion of techniques to enhance the classification algorithm. A novel dimensionality reduction approach is also proposed.

Chapter 7 includes the conclusions of our work and the future research directions we have in mind to improve our technique. We describe some strategies that can be used to enhance our methods and discuss their viability.

Chapter 2

Sparse Solution of Linear Inverse Problems

The problem of sparse representation consists in representing a given signal as a linear combination of as few "base" elements as possible from a fixed collection. That is, we aim to identify a sparse vector $x \in \mathbb{R}^n$ such that the target signal $b \in \mathbb{R}^m$ can be represented by $Ax \approx b$, where A is a known $m \times n$ matrix. In this chapter we formulate the problem that must be solved in order to obtain approximate sparse solutions to linear systems of equations, and discuss the strategies that have been proposed in recent years.

2.1 Problem Formulation

Consider a real matrix $A \in \mathbb{R}^{m \times n}$ whose columns a_j have unit Euclidean norm, that is $\|a_j\|_2 = 1$, for $j = 1, ..., n$. We will often refer to this type of matrix as the *dictionary*. We say that a vector (signal) $x \in \mathbb{R}^n$ is *k-sparse* if $\|x\|_0 \leq k$, where the counting function $\|\cdot\|_0 \colon \mathbb{R}^n \to \mathbb{R}$, known as the ℓ_0 "norm" [22], gives the number of nonzero elements in its argument. In other words,

$$\|x\|_0 = \operatorname{card}\{i \colon x_i \neq 0\}. \tag{2.1}$$

Even though we call it the ℓ_0-norm, one can easily verify that it does not satisfy the positive homogeneity (positive scalability) property in the definition of a norm. Namely we have that $\|\lambda x\|_0 \neq |\lambda| \|x\|_0$, for any given nonzero scalar λ.

A signal x is said to be *nearly sparse* if the rearranged entries of x decay exponentially

when sorted in decreasing order of magnitude [9]. Since compressible signals are well approximated by sparse ones, the framework of sparse approximation applies to this class too.

Given that we are looking for the sparsest vector x satisfying the linear system of equations $Ax = b$, we are interested in solving the following optimization problem:

$$\begin{aligned} \min \quad & \|x\|_0 \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{2.2}$$

assuming that the matrix $A \in \mathbb{R}^{m \times n}$ is short and wide, that is $m \ll n$. Unfortunately, Problem (2.2) is a combinatorial minimization problem and NP-hard (non-deterministic polynomial-time) [39]. Therefore any algorithm that is intended to solve (2.2) given the matrix A and the vector b will be computationally intractable. Thus, strategies to overcome this difficulty had to be developed, which gave rise to different algorithmic approaches with remarkable results in different applications.

2.2 Algorithmic Approaches

During the last decade, several strategies have been proposed to find approximate solutions to problem (2.2). These different approaches include:

Convex Relaxation. In this case, the objective function in Problem (2.2) is replaced by a convex function (as the ℓ_1 norm), overcoming the combinatorial nature of the problem [17].

Nonconvex Optimization. The idea consists in relaxing the ℓ_0 norm with a related nonconvex function, and attack the problem by identifying the corresponding stationary points. The use of ℓ_q quasi-norms ($0 < q < 1$) has been studied in [2, 14, 38].

Greedy Pursuit. Iterative refinement of a sparse solution is proposed, by successively identifying those entries in the vector producing the greatest improvement [37].

In this work, we focus on developing a ***Convex Relaxation*** technique for finding an approximate solution to the sparse representation problem. The strategy uses an ℓ_1 relaxation of the ℓ_0 norm, through which successful recovering of sparse signals has been shown. We solve the ℓ_1 optimization problem by iteratively solving a sequence of convex subproblems that depend on a regularization parameter. The methodology falls in a path-following framework, and convergence results along with comparison with other methodologies show its efficiency in finding sparse solutions of linear systems of equations.

2.3 ℓ_1-minimization problem

A practical alternative to Problem (2.2) is the ℓ_1 minimization approach, which consists in finding the solution to the problem

$$\begin{aligned} \min \quad & \|x\|_1 \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{2.3}$$

where $\|x\|_1 = \sum_{i=1}^{n} |x_i|$. We now have an optimization problem whose objective function is convex, unlike the ℓ_0-norm in Problem (2.2). However, we must have special conditions on the matrix A and on the sparsity of x in order to guarantee that the solution of Problem (2.3) will lead us to find the solution of the original problem.

The motivation for this approach comes from studying the theory of ***Compressed Sensing*** (compressive sampling) which has been a research topic of interest in the last years. The work in this area initiated in late 2004 by Emmanuel Candès, Justin Romberg and Terence Tao [9], and independently by David Donoho [19]. The general theme aims to answer the question: *How much information is necessary to accurately reconstruct a signal?*. It turns out that one can reconstruct *sparse* or *compressible* signals accurately from a very limited number of measurements. We wish to recover an object $x \in \mathbb{R}^n$, using information from a collection of m linear measurements $b_i = \langle a_i, x \rangle$ for $i = 1, ..., m$. In matrix notation, we can

write this as $b = Ax$, where $A \in \mathbb{R}^{m \times n}$ has the vectors a_i as rows. We will assume that $m < n$ and the *measurement matrix* A has full rank [22].

Restricted Isometry Property

We will say that a matrix A satisfies the *restricted isometry property* (RIP) with parameters (r, δ) if (see [10]):

$$(1 - \delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \delta)\|x\|_2^2, \quad \text{for all } r \text{ sparse vectors } x. \tag{2.4}$$

The restricted isometry constant δ_r of a matrix A is the smallest number satisfying (2.4). This property essentially requires that every set of columns with cardinality less than r approximately behaves like an orthonormal system [9]. In the following, let us present a mathematical result that shows the effectiveness of the ℓ_1 optimization approach when finding sparse solutions to linear systems of equations.

The Null Space Property

A matrix $A \in \mathbb{R}^{m \times n}$ satisfies the *null space property* (NSP) of order r with constant $\gamma \in (0, 1)$ if (see [15])

$$\|v_S\|_1 \leq \gamma \|v_{S^c}\|_1, \tag{2.5}$$

for all sets $S \subset \{1, ..., n\}$ with $\#S \leq r$, and $v \in \ker(A)$. Here S^c is the complement of S in the set $\{1, ..., n\}$. It can be shown that if a matrix A satisfies the *restricted isometry property* (2.4) then it also satisfies the *null space property* (see [15]).

Sparse Recovery Result

Let $A \in \mathbb{R}^{m\times n}$ be a matrix satisfying the NSP of order r with constant $\gamma \in (0,1)$. Let x^* be the solution of the ℓ_1-minimization Problem (2.3). If $x \in \mathbb{R}^n$ and $Ax = b$, then

$$\|x - x^*\|_1 \leq \tfrac{2(1+\gamma)}{(1-\gamma)}\sigma_x, \tag{2.6}$$

where σ_x is a quantity that depends on the sparsity of x. If the vector x is r-sparse then $x = x^*$.

Proof Since $Ax = Ax^* = b$, then the vector $v = x - x^*$ is in $\ker(A)$. Also, since x^* solves (2.3), then $\|x^*\|_1 \leq \|x\|_1$. Let S be the set of the r largest components of x in absolute value. We have

$$\|x_S^*\|_1 + \|x_{S^c}^*\|_1 \leq \|x_S\|_1 + \|x_{S^c}\|_1.$$

Notice also that (use triangle inequality)

$$\|x_S\|_1 - \|v_S\|_1 + \|v_{S^c}\|_1 - \|x_{S^c}\|_1 \leq \|x_S\|_1 + \|x_{S^c}\|_1,$$

so we get

$$\begin{aligned}\|v_{S^c}\|_1 &\leq \|v_S\|_1 + 2\|x_{S^c}\|_1,\\ &\leq \gamma\|v_{S^c}\|_1 + 2\sigma_x.\end{aligned}$$

Therefore, $\|v_{S^c}\|_1 \leq \frac{2}{(1-\gamma)}\sigma_x$. Since $v = x - x^*$, then

$$\begin{aligned}\|x - x^*\| &= \|v_S\| + \|v_{S^c}\|\\ &\leq (\gamma+1)\|v_{S^c}\|\\ &\leq \tfrac{2(1+\gamma)}{(1-\gamma)}\sigma_x.\end{aligned}$$

In case the vector x is r sparse, then $\|x_{S^c}\|_1 = \sigma_x = 0$, so we get $x = x^*$.

We have shown with the results presented above, that the notion of ℓ_1 minimization is indeed an effective technique for finding the sparsest solution x^* of a linear system of equations $Ax = b$.

2.4 Convex Relaxation Strategies

The ℓ_1 convex relaxation approach has been proven to sucessfully find sparse solutions to linear system of equations [22]. In the following, we briefly describe some state-of-the-art algorithms developed for finding approximate solutions of the sparse representation problem based on ℓ_1 optimization techniques.

Donoho, Saunders et al. - Basis Pursuit (BP)

In their work [17], Donoho et al. proposed to reformulate Problem (2.3) as a linear programming problem of the form

$$\begin{aligned} \min_x \quad & \sum_{i=1}^{n} u_i \\ \text{s.t} \quad & -u_i \leq x_i \leq u_i, \\ & Ax = b. \end{aligned} \tag{2.7}$$

They were able to solve linear programs of size 8192 by $212,992$. In their work, reasonable success with a primal-dual logarithmic barrier method and a conjugate gradient solver was obtained. It is easy to check that Problem (2.3) is equivalent to

$$\begin{aligned} \min \quad & c^T z \\ \text{subject to} \quad & \Phi z = f,\ z \geq 0, \end{aligned} \tag{2.8}$$

by letting $\Phi = [A, -A]$, $f = b$, $c = (\mathbb{1}; \mathbb{1})$, $z = (u, v)$ and $x = u - v$. Here, $\mathbb{1} \in \mathbb{R}^n$ is a vector with all components equal to 1.

Even though the approach provides strong guarantees and stability, it relies on *linear programming*, whose methods do not yet have strong polynomially bounded runtimes. It is worthwhile to mention, that the work by the authors of [17] was done several years before the results Candès and Tao proved on the recovery of sparse signals via the ℓ_1 minimization approach. In [11], Candès and Tao characterized the conditions that must be satisfied for finding the actual solution to the original problem (2.2), when using the ℓ_1-minimization alternative.

A natural variation to the basis pursuit Problem (2.3) consists in relaxing the linear constraint in order to consider an error tolerance, say $\epsilon \geq 0$, for the situation when the signal is contaminated with some additive noise. More specifically, the following problem is considered:

$$\begin{aligned} \min \quad & \|x\|_1 \\ \text{subject to} \quad & \|Ax - b\|_2 \leq \epsilon. \end{aligned} \tag{2.9}$$

The work in [9] claims that the convex relaxation approach (2.9) is also effective in finding an approximated solution of the sparse problem (2.2) whenever the observations are contaminated with a bounded additive noise.

Boyd, Lustig et al.

Boyd and his research group [35] proposed to solve a generalized version of (2.3) that allows certain degree of noise, given by the unconstrained minimization problem

$$\min_x \quad \lambda\|x\|_1 + \|Ax - b\|_2^2. \tag{2.10}$$

where the parameter $\lambda > 0$ is used as a penalization parameter balancing the tradeoff between error and sparsity.

First, Problem (2.10) is posed as the following constrained optimization problem

$$\begin{aligned} \min \quad & \lambda \sum_{i=1}^{n} u_i + \|Ax - b\|_2^2 \\ \text{s.t} \quad & -u_i \leq x_i \leq u_i. \end{aligned} \tag{2.11}$$

Secondly, using the notions of interior-point method (log-barrier method) they designed an algorithm to find a solution of the dual problem of (2.11). Their method makes use of a preconditioned conjugate gradient (PCG) to accelerate convergence and stabilize the algorithm. They also showed the application of their algorithm on a magnetic resonance imaging (MRI) data set. One drawback of their approach is that each step would require the solution of a Newton system of the form $H\Delta x = g$, where $H \in \mathbb{R}^{2n \times 2n}$ is the Hessian matrix and g is the gradient at the current iterate. To overcome this difficulty, they compute a search direction of an approximate Newton system using a PCG. This alternative is commonly known as the Truncated Newton Method. The truncation rule for the PCG provides the condition for terminating the algorithm. The total number of PCG iterations required by the truncated Newton interior-point method depends on the value of the regularization parameter λ and a given relative tolerance ϵ. An implementation of their algorithm is available at `http://www.stanford.edu/~boyd/l1_ls/`.

Figueiredo, Wright et al.

Figueiredo et al. [23] studied the unconstrained problem

$$\min_{x} \quad \lambda \|x\|_1 + \tfrac{1}{2}\|b - Ax\|_2^2, \tag{2.12}$$

as an alternative to find the sparsest solution x of the system $Ax = b$. They posed (2.12) as a quadratic programming problem of the form

$$\begin{aligned} \min \quad & z^T B z + c^T z \\ \text{s.t} \quad & z \geq 0, \end{aligned} \tag{2.13}$$

and their algorithm follows the *gradient projection* methodology.

The Gradient Projection for Sparse Reconstruction (GPSR) algorithm is based on the well-known projected gradient step technique

$$v^{(k+1)} = v^{k-1} - \alpha_k \nabla F(v^k),$$

where F is the function to be minimized. In this case

$$F(v) = \lambda \mathbb{1}^T v + \tfrac{1}{2}\|b - [A, -A]\mathbf{v}\|_2^2,$$

with $\mathbb{1}$ the vector of ones, and $\mathbf{v} = [v_1, v_2]$ with $\mathbf{v}_i \geq 0$ for all i. The step-length α_k is chosen following a backtracking technique.

Notice that

$$\|x\|_1 = \mathbb{1}^T \begin{bmatrix} v_1 \\ v_2 \end{bmatrix},$$

if we let $(v_1)_i = (x_i)_+$ and $(v_2)_i = (-x_i)_+$ where $(\cdot)_+$ denotes the positive part, $(x)_+ = \max\{0,\, x\}$. Therefore we can formulate the quadratic programming problem:

$$\begin{aligned} \min \quad & \mathbf{c}^T v + \tfrac{1}{2} v^T \mathbf{B} v \\ \text{s.t} \quad & v \geq 0, \end{aligned} \tag{2.14}$$

where $x = v_1 - v_2$ and

$$\mathbf{b} = A^T b, \quad \mathbf{c} = \lambda \mathbb{1} + \begin{bmatrix} -\mathbf{b} \\ \mathbf{b} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{bmatrix}.$$

An implementation of the algorithm is available at `http://www.lx.it.pt/~mtf/GPSR/`. One of the issues of the GPSR algorithm is that the formulation of the problem in (2.13) doubles the dimension of the variables involved in the original problem (2.12). Any matrix operation involving the matrix B must then take special care of its structure with respect to the matrices A and A^T.

Zhang et al.

The group from Rice University leaded by Y. Zhang, developed an algorithm to solve the problem

$$\min_x \quad \|x\|_1 + \tfrac{\mu}{2}\|Ax - b\|_2^2, \tag{2.15}$$

using a Fixed Point Continuation (FPC) method [28]. The main idea described in [28] consists on deducing a fixed point equation of the form $w = F(w)$ which holds at the solution, making use of subgradient optimality conditions. To guarantee convergence, appropriate parameters are chosen so that F is a contraction, and therefore $x_{k+1} = F(x_k)$ converges. Taking into acount that the *shrinkage* operator

$$\text{shrink}\left(c, \frac{\tau}{\mu}\right) \quad = \text{sgn}(c) \circ \max\left\{|c| - \tfrac{\tau}{\mu}, 0\right\}, \tag{2.16}$$

(for a fixed constant τ),is the explicit solution of

$$\min_x \quad \|x\|_1 + \tfrac{\mu}{2\tau}\|x - c\|_2^2, \tag{2.17}$$

the authors in [28] proved that the fixed-point iterations

$$x^{k+1} \quad = \operatorname{sgn}\left(x^k - \tau g(x^k)\right) \circ \max\left\{|x^k - \tau g(x^k)| - \tfrac{\tau}{\mu}, 0\right\} \tag{2.18}$$

where $g(x) = (A^T(Ax - b))$, converge to a solution of (2.15) as long as $0 < \tau < 2$. The convergence rate is accelerated by letting μ be small, in which case $\frac{\tau}{\mu}$ is large producing a solution x^* very sparse.

The FPC algorithm can be found at `http://www.caam.rice.edu/~optimization/L1/fpc/`

M. Argáez et al.

The research group led by M. Argáez has been working on the basis pursuit ℓ_1-minimization problem (2.3). In [3] we propose to find a solution to (2.3) by solving a sequence of problems of the form

$$\begin{aligned} \min_x \quad & \sum_{i=1}^{n} \left(x_i^2 + \mu\right)^{1/2} \\ \text{s.t} \quad & Ax = b, \end{aligned} \tag{2.19}$$

as the parameter μ tends to 0. This approach leads to a path-following method to find the solution x of the ℓ_1-minimization problem, by solving a sequence of linear equality constrained multiquadric problems that depend on a regularization parameter that converges to zero. The solution of subproblem k is used as initial point for the the next subproblem $k+1$, and a path-following framework is proposed. We have developed a homotopic principle for solving large-scale ℓ_1 underdetermined problems. Numerical experimentation has shown that our algorithm is capable of recovering sparse signals, with results comparing favorably - in both accuracy and CPU running time - with the state-of-the-art algorithms mentioned before, as reported in [3]. The MATLAB implementation of the path-following algorithm can be found at `http://www.math.utep.edu/Student/rsanchez/`

Becker et al.

Becker et al. used a Nesterov's smoothing technique for minimizing non-smooth functions, and solved the problem as formulated in (2.9). Their method [7] produces a decreasing sequence of iterates converging to the solution, where the ℓ_1 norm is approximated by

$$\|x\|_1 \approx \max_{v \in Q_d} v^T x - \frac{\mu}{2}\|v\|_2^2$$

with $\mu \to 0$ and $Q_d = w : \|w\|_\infty \leq 1$.

Their algorithm, named NESTA (Nesterov's algorithm) can be freely downloaded from `http://www-stat.stanford.edu/~candes/nesta/`.

Chapter 3

An ℓ_1 minimization algorithm for sparse representation

In this chapter we describe a novel path-following method for solving the ℓ_1 optimization problem that arises when searching for the sparsest solution to linear systems of equations. The problem formulation, the methodology used, and convergence results of the algorithm proposed are presented. We also present a numerical comparison of our algorithm with other state-of-the-art solvers dedicated to ℓ_1 optimization problems.

3.1 Formulation of the Problem

Consider a matrix $A \in \mathbb{R}^{m\times n}$, where $m < n$, and a vector $b \in \mathbb{R}^m$. We aim to find the sparsest vector $x \in \mathbb{R}^n$, such that $Ax = b$. In order to do this, and as it was discussed in the previous chapter, we can consider solving the following optimization problem

$$\begin{aligned} &\min_x \ \|x\|_1 \\ &\text{subject to } \|Ax - b\|_2 \leq \epsilon, \end{aligned}$$

for $\epsilon > 0$. This *denoising* problem is equivalent to

$$\min_x \ \lambda\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{3.1}$$

where the parameter $\lambda > 0$ is used as a penalization parameter balancing the tradeoff between error and sparsity. Our strategy consists in relaxing the ℓ_1 norm by a smooth function, and reformulating problem (3.1) in a constrained form. Then we find the solution to the optimization problem by solving a sequence of linear equality constrained multiquadric problems that depends on a regularization parameter μ that converges to zero. The procedure generates a central path that converges to a point on the solution set of the ℓ_1-minimization problem (3.1).

Let us first define the approximation we used for the ℓ_1 norm. Consider the scalar function

$$f_\mu(x) = \sum_{i=1}^{n} \left(x_i^2 + \mu\right)^{1/2}, \tag{3.2}$$

for a regularization parameter $\mu > 0$. Notice that the function $f_\mu(\cdot)$ is continuously differentiable for every value of $\mu > 0$. In fact we have the following properties:

1. $f_\mu(x) \to \|x\|_1$ as $\mu \to 0$.

2. $f_\mu(x) \in \mathcal{C}^2(\mathbb{R}^n)$, that is the function is twice continuously differentiable.

3. $\left(\nabla f_\mu(x)\right)_j = \frac{x_j}{\left(x_j^2+\mu\right)^{1/2}}$. We write the gradient of $f_\mu(x)$ as

$$\nabla f_\mu(x) = D^{-1/2}(x)\; x,$$

   where the matrix $D(x)$ is a diagonal matrix whose entries are given by $(x_i^2 + \mu)$. That is, $D(x) = \mathrm{diag}(x^2 + \mu)$.

In order to reformulate problem (3.1) as a constrained problem, we introduce an auxiliary variable $e \in \mathbb{R}^m$, defined as $e = Ax - b$. Using this new variable and replacing the ℓ_1 norm

by the smooth approximation f_μ, we obtain the following constrained optimization problem

$$(P_\mu) \qquad \min_{x,e} \ \lambda f_\mu(x) + \frac{1}{2} e^T e$$
$$\text{subject to } Ax - e = b, \tag{3.3}$$

where the $(\cdot)^T$ operator denotes the transposition. Problem (3.3), that we call problem (P_μ), is a convex optimization problem with unique solution for each value of μ. We plan to solve a sequence of problems of the form (P_μ) for a given sequence of regularization parameters μ that tends to 0, in order to obtain the solution of the ℓ_1 optimization problem. Notice that the formulation for problem (P_μ) can be thought as a *multiobjective optimization* problem where we try to minimize both the function $f_\mu(x)$ (our approximation for the ℓ_1 norm) and the residual $\|e\|_2^2$, balancing the tradeoff between error and sparsity.

3.2 Algorithmic Approach

We next characterize the optimality conditions for problem (P_μ). The Lagrangian associated to problem (P_μ)

$$\mathcal{L}(x, e, y) = \lambda f_\mu(x) + \frac{1}{2} e^T e + y^T (Ax - e - b), \tag{3.4}$$

where the vector $y \in \mathbb{R}^m$ is the Lagrange multiplier associated to the equality constraint in (3.3). The corresponding KKT (Karush-Kuhn-Tucker) optimality conditions are given by

$$\begin{aligned} \lambda D^{-1/2}(x)\ x + A^T y &= 0, \\ e - y &= 0, \\ Ax - e - b &= 0, \end{aligned} \tag{3.5}$$

where $D(x) = \text{diag}(x^2 + \mu)$. From the second relation in (3.5) we see that our auxiliary variable, the *error* vector e, coincides with the Lagrange multiplier y associated to the equality constraint. Therefore, we can rewrite the KKT conditions as

$$\begin{aligned} \lambda D^{-1/2}(x)\ x + A^T e &= 0, \\ Ax - e - b &= 0. \end{aligned} \tag{3.6}$$

Now, multiplying first by $D^{1/2}(x)$ and then by A the first equation in (3.6) we get

$$\lambda Ax + AD^{1/2}(x)A^T e = 0,$$

and since $Ax - e = b$, then we end up with

$$\left(AD^{1/2}(x)A^T + \lambda I\right) e = -\lambda b, \tag{3.7}$$

where I is the $m \times m$ identity matrix.

Our algorithmic approach consists in solving (3.7) for $D(x) = D_{\mu_k}(x_k)$, where x_k is the optimal solution of the previous subproblem k of the form (3.3) for $\mu = \mu_k$. If that is the case, then we can compute a new approximation for e, denoted by e_{k+1}, as the solution of the $m \times m$ linear system of equations:

$$\left(AD^{1/2}(x_k)A^T + \lambda I\right) e_{k+1} = -\lambda b, \tag{3.8}$$

and then update the approximation of x by

$$x_{k+1} = -\lambda^{-1} D^{1/2}_{\mu_k}(x_k) A^T e_{k+1}. \tag{3.9}$$

For a decreasing sequence of regularization parameters μ approaching to 0, $\mu_k < \mu_{k-1}$, we

form a sequence of approximations $\{(x_k,\ e_k)\}_k$ such that

$$x_k \to x^* \ \text{ and } e_k \to e^*,$$

where x^* is the sparse solution of the ℓ_1 optimization problem (3.1).

The algorithm, which we call *Iterative Smooth Convex Relaxation* (ISCR), is described as follows.

Algorithm 1 Iterative Smooth Convex Relaxation (ISCR)

Goal: Find an approximate solution x to problem

$$\min_{x,e} \lambda\|x\|_1 + \frac{1}{2}e^T e \quad \text{subject to } Ax - e = b.$$

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$.
Output: x^* solution of the ℓ_1 optimization problem.

Step 1: Initialization: set μ_0, tolerance ϵ, penalization λ,
maximum number of iterations $k_{\max}$

Step 2: Initial approximate solution: $x_0 = A^T b$, $e_0 = Ax_0 - b$.

Step 3: **For** $k = 0, ...,\ k_{\max}$

Step 4: Solve for e_{k+1} : $\left(AD^{1/2}_{\mu_k}(x_k)A^T + \lambda I\right) e_{k+1} = -\lambda b$,
where D is a diagonal matrix with $[D_{\mu_k}(x_k)]_{jj} = (x_k)_j^2 + \mu_k$.

Step 5: Update x_{k+1} as $x_{k+1} = -\lambda^{-1} D^{1/2}_{\mu_k}(x_k)A^T e_{k+1}$.

Step 6: Check convergence: if $\|x_{k+1} - x_k\|_2 \le \epsilon$,
$x = x_{k+1}$, $e = e_{k+1}$,
stop

Step 7: Reduce regularization parameter: $\mu_{k+1} < \mu_k$.

Step 8: **Go to** step 3.

Notice in Step 4 of Algorithm 1, that we always solve a $m \times m$ system of linear equations,

and in our formulation we have $m < n$, which means that our method solves at each step a small linear system compared with the dimension of the problem. Such linear system can be solved using a Conjugate Gradient (CG) algorithm, in order to involve only matrix vector multiplications in our computations (a specialized CG algorithm tuned for our methodology is presented later in this chapter).

3.3 Fixed-Point Framework

Notice that in Algorithm 1, for the stopping criteria in Step 6, we check the closeness of consecutive approximations x_k and x_{k+1}. The reason for this comes from the fact that we can actually pose our iterative procedure as a fixed point iteration. In order to see this, let us recall the simplified KKT conditions listed in (3.6), replacing the approximation for $D(x)$ we use in the algorithm and the variable x for the current iteration x_{k+1}. Then we obtain the following

$$\begin{aligned} \lambda D_{\mu_k}^{-1/2}(x_k)\ x_{k+1} + A^T e_{k+1} &= 0, \\ Ax_{k+1} - e_{k+1} - b &= 0, \end{aligned}$$

which can be rewritten in matrix form as

$$\begin{bmatrix} \lambda D_{\mu_k}^{-1/2}(x_k) & A^T \\ A & -I \end{bmatrix} \begin{bmatrix} x_{k+1} \\ e_{k+1} \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}. \tag{3.10}$$

The block matrix in (3.10) is invertible, and we then have the relation

$$\begin{aligned}
\begin{bmatrix} x_{k+1} \\ e_{k+1} \end{bmatrix} &= \begin{bmatrix} \lambda D_{\mu_k}^{-1/2}(x_k) & A^T \\ A & -I \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ b \end{bmatrix}, \\
&= \begin{bmatrix} I & \mathbf{0} \\ A & I \end{bmatrix} \begin{bmatrix} \left(\lambda D_{\mu_k}^{-1/2}(x_k) + A^T A\right)^{-1} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} I & A^T \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} 0 \\ b \end{bmatrix}, \\
&= \mathcal{P} \begin{bmatrix} \left(\lambda D_{\mu_k}^{-1/2}(x_k) + A^T A\right)^{-1} & \mathbf{0} \\ \mathbf{0} & I \end{bmatrix} \mathcal{P}^T \begin{bmatrix} 0 \\ b \end{bmatrix}, \\
&= \mathcal{S}\left(\begin{bmatrix} x_k \\ e_k \end{bmatrix} \right), \qquad (3.11)
\end{aligned}$$

where $\mathcal{P} = \begin{bmatrix} I & \mathbf{0} \\ A & I \end{bmatrix}$ is a block matrix whose inverse is given by $\mathcal{P}^{-1} = \begin{bmatrix} I & \mathbf{0} \\ -A & I \end{bmatrix}$. We have defined a functional $\mathcal{S} : \mathbb{R}^{m+n} \to \mathbb{R}^{m+n}$ to characterize the fixed point iteration, since everything on the right hand side of the relation in (3.11) depends only on the previous step k and is used to compute the new approximation $k+1$. Thus, our iterative procedure can indeed be posed as a fixed point iteration that finds the solution of the ℓ_1 optimization problem.

We are now in position of studying the convergence of our proposed methodology and present some error estimates.

3.4 Convergence results for the proposed method

Let us now study the convergence of the methodology described in Algorithm 1 for finding the sparsest solution of a linear system of equations. For this, consider two consecutive

subproblems

$$(P_{\mu_k}) \qquad \min_{x,e} \ \lambda f_{\mu_k}(x) + \frac{1}{2}e^T e \qquad \text{subject to } Ax - e = b,$$

$$(P_{\mu_{k+1}}) \qquad \min_{x,e} \ \lambda f_{\mu_{k+1}}(x) + \frac{1}{2}e^T e \qquad \text{subject to } Ax - e = b,$$

where $\mu_{k+1} < \mu_k$. Let x_k and x_{k+1} be the optimal solutions of subproblems (P_{μ_k}) and $(P_{\mu_{k+1}})$ respectively, found using the procedure described in Algorithm 1.

We want to analyze the difference between both optimal values, that is we are interested in the quantity

$$\begin{aligned} q &= \left(\lambda f_{\mu_k}(x_k) + \frac{1}{2}e_k^T e_k\right) - \left(\lambda f_{\mu_{k+1}}(x_{k+1}) + \frac{1}{2}e_{k+1}^T e_{k+1}\right) \\ &= \lambda\left(f_{\mu_k}(x_k) - f_{\mu_{k+1}}(x_{k+1})\right) + \frac{1}{2}\left(\|e_k\|_2^2 - \|e_{k+1}\|_2^2\right). \end{aligned} \tag{3.12}$$

Notice that

$$\frac{1}{2}\left(\|e_k\|_2^2 - \|e_{k+1}\|_2^2\right) = \frac{1}{2}\|Ax_k - Ax_{k+1}\|_2^2 + (e_{k+1})^T(Ax_k - Ax_{k+1}), \tag{3.13}$$

since

$$\frac{1}{2}(Ax_k - Ax_{k+1})^T(Ax_k - Ax_{k+1}) = \frac{1}{2}\left(x_k^T(A^TA)x_k - 2x_k^T(A^TA)x_{k+1} + x_{k+1}^T(A^TA)x_{k+1}\right),$$

and using the optimality conditions we have

$$\begin{aligned} (e_{k+1})^T(Ax_k - Ax_{k+1}) &= e_{k+1}^T Ax_k - e_{k+1}^T Ax_{k+1} \\ &= x_{k+1}^T(A^TA)x_k - b^TAx_k - + x_{k+1}^T(A^TA)x_{k+1} + b^TAx_{k+1}. \end{aligned}$$

Therefore we can express the difference q by

$$q = \lambda \left(f_{\mu_k}(x_k) - f_{\mu_{k+1}}(x_{k+1}) \right) + \frac{1}{2} \|Ax_k - Ax_{k+1}\|_2^2 + (e_{k+1})^T (Ax_k - Ax_{k+1}).$$

Also, rewriting the product $(e_{k+1})^T (Ax_k - Ax_{k+1})$ we get

$$x_k^T A^T e_{k+1} - x_{k+1}^T A^T e_{k+1} = (x_k - x_{k+1})^T A^T e_{k+1}.$$

Using the first relation in the KKT conditions (3.6) we have that

$$A^T e_{k+1} = -\lambda D_{\mu_k}^{-1/2}(x_k) \cdot x_{k+1}, \tag{3.14}$$

which allows us to further simplify the difference q as

$$q = \lambda \left(f_{\mu_k}(x_k) - f_{\mu_{k+1}}(x_{k+1}) \right) + \frac{1}{2} \|Ax_k - Ax_{k+1}\|_2^2 - \lambda x_{k+1}^T D_{\mu_k}^{-1/2}(x_k) \left[x_k - x_{k+1} \right]. \tag{3.15}$$

If we substitute the expressions for $f_\mu(\cdot)$ and $D_\mu(\cdot)$, we get

$$\begin{aligned} q &= \lambda \left[\sum_{i=1}^{n} \left((x_k)_i^2 + \mu_k \right)^{1/2} - \sum_{i=1}^{n} \left((x_{k+1})_i^2 + \mu_{k+1} \right)^{1/2} \right] \\ &\quad - \lambda \sum_{i=1}^{n} \frac{(x_{k+1})_i \left[(x_k)_i - (x_{k+1})_i \right]}{\left((x_k)_i^2 + \mu_k \right)^{1/2}} + \frac{1}{2} \|Ax_k - Ax_{k+1}\|_2^2 \\ &= \lambda \mathcal{H} + \frac{1}{2} \|Ax_k - Ax_{k+1}\|_2^2, \end{aligned} \tag{3.16}$$

where

$$\mathcal{H} = \sum_{i=1}^{n} \left[\left((x_k)_i^2 + \mu_k \right)^{1/2} - \left((x_{k+1})_i^2 + \mu_{k+1} \right)^{1/2} - \frac{(x_{k+1})_i \left[(x_k)_i - (x_{k+1})_i \right]}{\left((x_k)_i^2 + \mu_k \right)^{1/2}} \right]. \tag{3.17}$$

We now use the following lemma to prove that the quantity $\mathcal{H}$ is greater than a positive value Δ.

Proposition 1 *For $\mu_{k+1} < \mu_k$, and $w, z \in \mathbb{R}$, the following inequality holds:*

$$\left(w^2+\mu_k\right)^{1/2} - \left(z^2+\mu_{k+1}\right)^{1/2} - \frac{z(w-z)}{\left(w^2+\mu_k\right)^{1/2}} > \Delta, \tag{3.18}$$

for some $\Delta > 0$.

Proof *Notice that*

$$\left(w^2+\mu_k\right)^{1/2} - \left(z^2+\mu_{k+1}\right)^{1/2} - \frac{z(w-z)}{\left(w^2+\mu_k\right)^{1/2}} \tag{3.19}$$

$$= \frac{\left(w^2+\mu_k\right) - \left(z^2+\mu_{k+1}\right)^{1/2}\left(w^2+\mu_k\right)^{1/2} - z(w-z)}{\left(w^2+\mu_k\right)^{1/2}} \tag{3.20}$$

Recall now the arithmetic and geometric inequality that states

$$\left(\frac{1}{n}\sum_{i=1}^{n} a_i\right) \geq \left(\prod_{i=1}^{n} a_i\right)^{1/n}, \quad a_i > 0, \ \forall i. \tag{3.21}$$

Therefore we get bound_proof2

$$\begin{aligned} \frac{1}{2}\left[\left(z^2+\mu_{k+1}\right) + \left(w^2+\mu_k\right)\right] &\geq \left[\left(z^2+\mu_{k+1}\right)\cdot\left(w^2+\mu_k\right)\right]^{1/2}, \\ &\Downarrow \\ -\left(z^2+\mu_{k+1}\right)^{1/2}\left(w^2+\mu_k\right)^{1/2} &\geq -\frac{1}{2}\left(z^2+\mu_{k+1}\right) - \frac{1}{2}\left(w^2+\mu_k\right). \end{aligned} \tag{3.22}$$

Thus, using the relation in (3.19) we have

$$\begin{aligned}
&\frac{(w^2+\mu_k)-(z^2+\mu_{k+1})^{1/2}(w^2+\mu_k)^{1/2}-z(w-z)}{(w^2+\mu_k)^{1/2}} \\
&\geq \frac{(w^2+\mu_k)-\frac{1}{2}(z^2+\mu_{k+1})-\frac{1}{2}(w^2+\mu_k)-z(w-z)}{(w^2+\mu_k)^{1/2}} \\
&= \frac{\frac{1}{2}(w^2+\mu_k)-\frac{1}{2}(z^2+\mu_{k+1})-zw+z^2}{(w^2+\mu_k)^{1/2}} \\
&= \frac{\frac{1}{2}(w^2+\mu_k+z^2-\mu_{k+1}-2zw)}{(w^2+\mu_k)^{1/2}} \\
&= \frac{(\mu_k-\mu_{k+1})+(w-z)^2}{2(w^2+\mu_k)^{1/2}}.
\end{aligned}$$

Therefore, since $\mu_{k+1} < \mu_k$, we get

$$\left(w^2+\mu_k\right)^{1/2}-\left(z^2+\mu_{k+1}\right)^{1/2}-\frac{z(w-z)}{(w^2+\mu_k)^{1/2}} \geq \frac{(w-z)^2}{2(w^2+\mu_k)^{1/2}} = \Delta > 0. \tag{3.23}$$

Using the above lemma we can say that $\mathcal{H} \geq \Delta$. Thus, the difference q between optimal values that we are interested in satisfies

$$\begin{aligned}
q &= \lambda\mathcal{H} + \frac{1}{2}\|Ax_k - Ax_{k+1}\|_2^2 \\
&> \lambda\Delta + \frac{1}{2}\|Ax_k - Ax_{k+1}\|_2^2 \\
&> \frac{1}{2}\|Ax_k - Ax_{k+1}\|_2^2.
\end{aligned} \tag{3.24}$$

Therefore

$$\begin{aligned}
\|Ax_k - Ax_{k+1}\|_2^2 &\leq 2q \\
&\leq 2\left(\lambda f_{\mu_k}(x_k) - \lambda f_{\mu_{k+1}}(x_{k+1}) + \frac{1}{2}e_k^T e_k - \frac{1}{2}e_{k+1}^T e_{k+1}\right).
\end{aligned} \tag{3.25}$$

The relationship (3.25) also means that

$$\lambda f_{\mu_{k+1}}(x_{k+1}) + \frac{1}{2}e_{k+1}^T e_{k+1} \leq \lambda f_{\mu_k}(x_k) + \frac{1}{2}e_k^T e_k$$
$$g(x_{k+1}, e_{k+1}, \mu_{k+1}) \leq g(x_k, e_k, \mu_k), \tag{3.26}$$

where we have used the function $g(\cdot)$ to denote the objective function of problem (P_μ). From (3.26) shows that $g(x_k, e_k, \mu_k)$ is monotonically decreasing. This in turn means that

$$g(x_k, e_k, \mu_k) \leq g(x_0, e_0, \mu_0) = M. \tag{3.27}$$

Using this result, we have the following bound on the sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by Algorithm 1.

Proposition 2 *The sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by Algorithm 1 satisfies*

$$\|x_k\|_1 \leq \frac{M}{\lambda}, \tag{3.28}$$

where $M = g(x_0, e_0, \mu_0) = \lambda f_{\mu_0}(x_0) + \frac{1}{2}e_0^T e_0$.

Proof *Notice that*

$$\begin{aligned}\lambda\|x_k\|_1 &\leq \lambda \sum_{i=1}^{n} \left((x_i)_k^2 + \mu_k\right)^{1/2} \\ &\leq \lambda \sum_{i=1}^{n} \left((x_i)_k^2 + \mu_k\right)^{1/2} + \frac{1}{2}e_k^T e_k \\ &\leq M,\end{aligned}$$

where we first used the fact that the approximation $f_\mu(x)$ is an upper bound for the ℓ_1 norm of x, and the second inequality follows from the bound on $g(x_k, e_k, \mu_k)$ described in (3.27). Therefore, the sequence $\{x_k\}_{k\in\mathbb{N}}$ is bounded by $\frac{M}{\lambda}$, and there exists a subsequence $\left\{x_{k_j}\right\}_{k_j\in\mathbb{N}}$ converging to some point x^*.

Theorem 1 *x^* the limit point of the converging subsequence $\left\{x_{k_j}\right\}_{k_j\in\mathbb{N}}$ is the solution of the ℓ_1 problem*

$$\min_{x,e} \lambda\|x\|_1 + \frac{1}{2}e^T e \quad \textit{subject to } Ax - e = b.$$

Now, given two consecutive elements of the sequence $\{x_k\}_{k\in\mathbb{N}}$ let us find bounds on $\|Ax_k - Ax_{k+1}\|$ and $\|x_k - x_{k+1}\|$. Recall the relation in (3.24) and the value we found for Δ in (3.23). The bound for $\|Ax_k - Ax_{k+1}\|$ is given in (3.25). Now, from

$$q > \frac{1}{2}\left(\sum_{i=1}^{n}\left(\frac{\left[(x_k)_i - (x_{k+1})_i\right]^2}{\left[(x_k)_i^2 + \mu_k\right]^{1/2}}\right) + \|Ax_k - Ax_{k+1}\|_2^2\right), \tag{3.29}$$

and since

$$\sum_{i=1}^{n}\left[(x_k)_i - (x_{k+1})_i\right]^2 = \|x_k - x_{k+1}\|_2^2, \tag{3.30}$$

we can find a bound for the difference of two consecutive elements of the sequence generated by Algorithm 1. First we used the bound in (3.28),

$$\|x_k\|_2 \leq \|x_k\|_1 \leq \frac{M}{\lambda},$$

and $\mu_0 > \mu_k$, to obtain

$$\frac{1}{\left[(x_k)_i^2 + \mu_k\right]^{1/2}} \geq \frac{1}{\left[\left(\frac{M}{\lambda}\right)^2 + \mu_0\right]^{1/2}} = \beta.$$

Then, from (3.29)

$$\begin{aligned} q &\geq \frac{1}{2\left[\left(\frac{M}{\lambda}\right)^2+\mu_0\right]^{1/2}} \sum_{i=1}^{n} \left[(x_k)_i - (x_{k+1})_i\right]^2 \\ &= \frac{1}{2\beta}\|x_k - x_{k+1}\|_2^2. \end{aligned}$$

Therefore,

$$\|x_k - x_{k+1}\|_2^2 \leq 2\beta q. \tag{3.31}$$

This means we have bounded the difference between two consecutive approximations for the solution of the ℓ_1 optimization problem, in terms of the difference of the objective function values.

Proposition 3 *The sequence $\{x_k\}_{k\in\mathbb{N}}$ generated by Algorithm 1 is bounded and satisfies the following*

$$\|Ax_k - Ax_{k+1}\|_2^2 \leq 2q,$$

$$\|x_k - x_{k+1}\|_2^2 \leq 2\beta q.$$

where $q = \left(\lambda f_{\mu_k}(x_k) - \lambda f_{\mu_{k+1}}(x_{k+1}) + \frac{1}{2}e_k^T e_k - \frac{1}{2}e_{k+1}^T e_{k+1}\right)$ and $\beta = \frac{1}{\left[\left(\frac{M}{\lambda}\right)^2+\mu_0\right]^{1/2}}$ and the constant M depends only on the initial approximation.

Proposition 4 *The sequence $\{(x_k,\ e_k)\}_{k\in\mathbb{N}}$ of solutions of subproblems of the form*

$$(P_\mu) \qquad \min_{x,e}\ \lambda f_\mu(x) + \frac{1}{2}e^T e, \quad \mu > 0,$$
$$\text{subject to } Ax - e = b,$$

obtained using Algorithm 1 is bounded, and there exists a subsequence $\left\{(x_{k_j},\ e_{k_j})\right\}_{k_j\in\mathbb{N}}$ that

converges to (x^*, e^*) *the solution of the* ℓ_1 *optimization problem*

$$\min_{x,e} \lambda\|x\|_1 + \frac{1}{2}e^T e,$$
$$\textit{subject to } Ax - e = b,$$

as the parameter μ *approaches* 0.

3.5 Numerical Implementation

We now present a numerical experimentation for a MATLAB implementation of our Iterative Smooth Convex Relaxation algorithm described before. First, notice that in Algorithm 1, the main step consists of solving the linear system of equations in Step 4 given by

$$\left(AD^{1/2}_{\mu_k}(x_k)A^T + \lambda I\right) e_{k+1} = -\lambda b, \tag{3.32}$$

which is a $m \times m$ system of equations with a positive definite system matrix. This is easily seen by checking the following relation

$$\begin{aligned} w^T \left(AD^{1/2}_{\mu_k}(x_k)A^T + \lambda I\right) w &= w^T AD^{1/2}_{\mu_k}(x_k)A^T w + w^T w \\ &= w^T \left(AD^{1/4}_{\mu_k}(x_k)\right)\left(AD^{1/4}_{\mu_k}(x_k)\right)^T w + w^T w \\ &= \| \left(AD^{1/4}_{\mu_k}(x_k)\right)^T w\|_2^2 + \|w\|_2^2 \\ &> 0, \end{aligned}$$

for any nonzero vector $w \in \mathbb{R}^m$.

We then propose the following Conjugate Gradient (CG) algorithm for solving the system (3.32), which only requires matrix vector multiplications with the matrices A and D_μ (notice that multiplication by the matrix D_μ can be easily computed given its diagonal nature). The fact that in our methodology we solve a $m \times m$ linear systems of equations, even though the

solution vector we are looking for belongs to $\mathbb{R}^n$, is definitely an advantage of the formulation in (3.3) we proposed for finding the solution of the ℓ_1 optimization problem.

Algorithm 2 Conjugate Gradient (CG) algorithm

Goal: Find an approximate solution e_{k+1} to

$$\left(AD^{1/2}_{\mu_k}(x_k)A^T + \lambda I\right) e_{k+1} = -\lambda b,$$

Input: $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $D_{\mu_k}(x_k)$.

Output: e_{k+1} solution of the $m \times m$ system of equations. It also returns $A^T e_{k+1}$.

Step 1: Initialization: set tolerance ϵ_{CG}, penalization λ,
maximum number of iterations k_{CG}

Step 2: Initial approximate solution: $e_{k+1} = \mathbf{0} \in \mathbb{R}^m$,
First residual: $r = -\lambda b - \left(AD^{1/2}_{\mu_k}(x_k)A^T + \lambda I\right) e_k$,
First direction: $d = r$, and set $\beta_N = r^T r$.

Step 3: **For** $j = 0, ..., k_{CG}$

Step 4: Steplength:
a. Set $q_1 = A^T d$, $q_2 = D^{1/2}_{\mu_k}(x_k)q_1$, and $q_3 = Aq_2 + \lambda d$.
b. Set $\alpha_D = d^T q_3$, and let $\alpha = \frac{\beta_N}{\alpha_D}$.

Step 5: Updates:
a. $A^T e_{k+1} = A^T e_k + \alpha q_1$.
b. $e_{k+1} = e_{k+1} + \alpha d$.

Step 6 Residual: $r = r - \alpha q_3$.

Step 6: Check convergence: if $\|r\|_2 \leq \epsilon_{CG}$, **stop**

Step 7: Update direction:
a. Set $\beta_D = \beta_N$, $\beta_N = r^T r$, and $\beta = \frac{\beta_N}{\beta_D}$.
b. Set $d = r + \beta d$.

Step 8: **Go to** step 3.

Notice that in Algorithm 2 we also automatically find an approximation for $A^T e_{k+1}$. The motivation for this comes from the fact that the values of $A^T e_{k+1}$ characterize the optimality set, as we can see from the relation in (3.14), since we compute $x_{k+1} = -\lambda^{-1} D^{1/2}_{\mu_k}(x_k)A^T e_{k+1}$. Thus, our conjugate gradient algorithm formulation finds an approximation of $A^T e_{k+1}$ at

each CG step. The MATLAB codes of our algorithms can be found in the Appendix.

3.6 Experimentation and Comparison

3.6.1 Sparse Signal Recovery Example

In order to test the efficiency of our algorithm when finding the sparsest solution of a linear system of equations, we design a compressed sensing type of problem where $x \in \mathbb{R}^n$, with $n = 4096$ is a sparse signal with only $T = 160$ nonzero elements, and the measurement vector b has $m = 1024$ entries. The matrix A is a partial DCT matrix whose m rows are chosen randomly from the $n \times n$ discrete cosine transform, without having access to it in explicit form, but using A as a linear operator on $\mathbb{R}^n$. The same for the matrix A^T . Partial DCT matrices are fast transforms for which matrix-vector multiplications cost just $\mathcal{O}(n \log n)$ flops, and storage is not required. This case is common for compressed sensing [Zhang], [Boyd]. The noise vector e is set according to a Gaussian distribution with mean 0 and standard deviation 0.01, such that $\|e\| = 0.03343$.

The original and reconstructed signal are shown in Figure 3.1 . The ISCR algorithm recovered the original signal successfully and the approximation e^* was such that $\|e^*\| = 0.0357063$. Moreover, we run the algorithm successfully one hundred times with an average CPU time of 0.7020 seconds, and average relative 2-norm error of 0.0005.

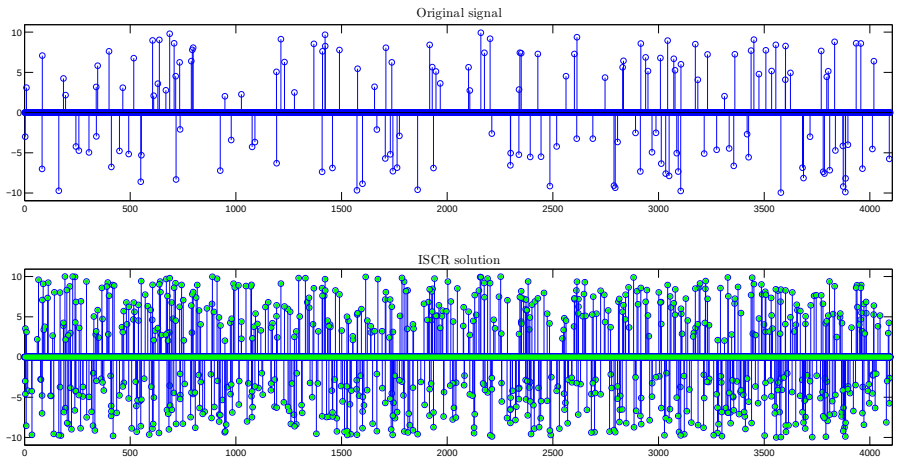


Figure 3.1: (Top) Original sparse signal x. (Bottom) Approximate solution x^* obtained using ISCR.

In Figure 3.2, we show the behavior of the vector $\frac{1}{\lambda}A^T e^*$ at the solution x^*, that according to the optimality condition (3.14) must coincide with either 1, 0, or -1 depending on the sign of the x_i^* component. This is easily confirmed since for a large iteration value j, and the regularization parameter μapproaching 0, the j-th nonzero entry of the vector x_{k+1} is related to the j-th component of $A^T e_{k+1}$by

$$\begin{aligned}\left[A^T e_{k+1}\right]_j &= -\lambda \left[D_{\mu_k}^{-1/2}(x_k)\cdot x_{k+1}\right]_j \\ &= -\lambda \frac{\left[x_{k+1}\right]_j}{\left[(x_k)_j^2+\mu_k\right]^{1/2}} \\ &\approx -\lambda \frac{(x_{k+1})_j}{\left|(x_{k+1})_j\right|} = -\lambda\cdot \operatorname{sgn}\left((x_{k+1})_j\right),\end{aligned}$$

where sgn$(\cdot)$ denotes the *sign* function (1 for positive arguments, -1 for negative arguments, 0 if the argument is equal to 0).

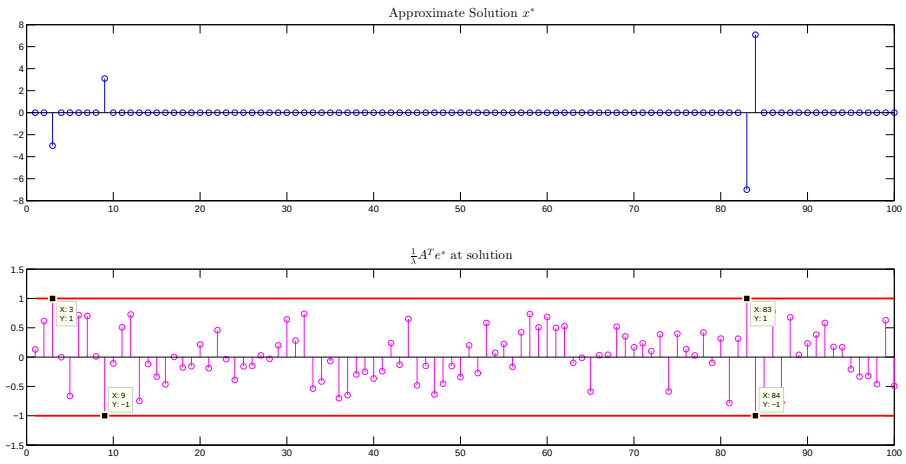


Figure 3.2: (Top) Segment of the recovered solution x^*. (Bottom) Corresponding A^Te^* values at the solution.

3.6.2 Scalability and Numerical Comparison

As described in the previous chapter, there are several ℓ_1 optimization solvers which we are interested in comparing our algorithm with. We created a set of problems similar to the one presented before, where we control the sparsity T of the signal $x \in \mathbb{R}^n$ and recover an approximation of the sparse signal based on a vector b consisting of m measurements. The matrix A is again a fast DCT transformation.

We compared the efficiency of the proposed method in Algorithm 1, with the `l1_ls` solver (Boyd et al.), the `FPC_AS` solver (Zhang et al.), the `NESTA` algorithm (Becker et al.), and the `GPSR` method (Wright et al.). Running time for finding the sparse solution of the system of linear equations, relative error of the approximation with respect to the true solution, and final residuals were recorded in every experiment.

For $n = 2^{12}$ to $n = 2^{23}$, the sparsity of the signal $x \in \mathbb{R}^n$ is controlled so that the total number of nonzero entries is $T = \frac{5}{2^7}n$. Given the number of nonzero elements T, the original sparse signal x^* is generated by randomly selecting the location of these nonzero entries. The matrix A is set as in the Sparse Signal Recovery Example presented in the previous

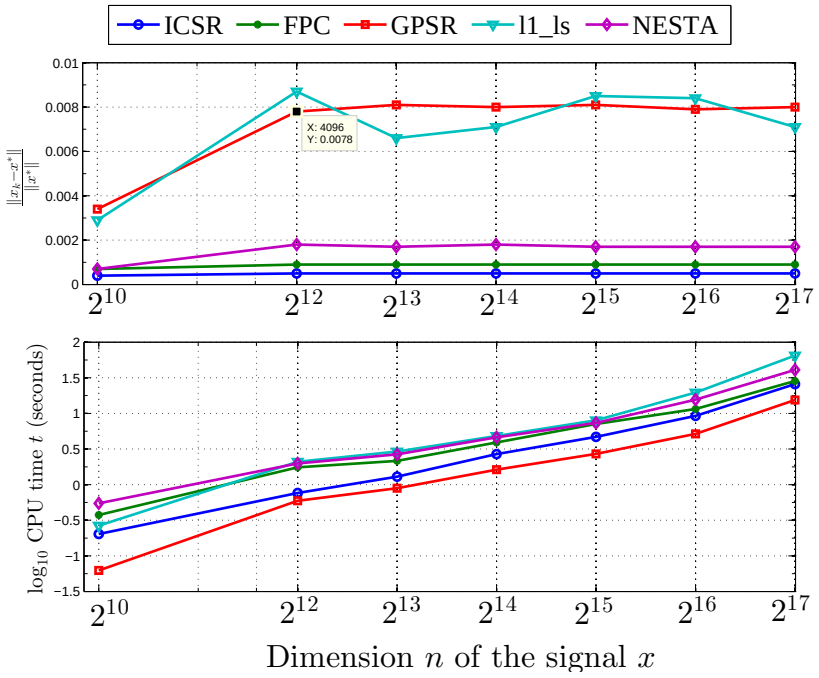


Figure 3.3: Scalability for $n = 2^{10}$ to $n = 2^{17}$ (noisy case)

subsection, that is, a partial DCT matrix whose m rows are chosen randomly from the $n \times n$ discrete cosine transform. The level of noise is 1% with Gaussian distribution $\mathcal{N}(0, 0.01^2 I)$.

In Figure 3.3 and Figure 3.4 we show the scalability of our proposed algorithm in both CPU running time and relative error (noisy case). We compared our methodology with the other four algorithms mentioned before. The scalability study shows that the `ICSR` method we proposed in this chapter, has always the best accuracy in terms of the relative error $\frac{\|x_k - x^*\|}{\|x^*\|}$ where x_k is the approximation found and x^* the true solution of the problem. In terms of CPU time, our algorithm is second best after the `GPSR` solver proposed in [23], but whose relative errors are significantly worse than the other solvers. Tables 3.1 and 3.2 show more details of the experimentation.

We also performed the same set of experiments for the noiseless case, that is when $\|e\| = 0$. The results are similar to the one presented for the noisy set of problems, and the trend remains for both CPU running time of each algorithm and the relative error reached.

Table 3.1: Scalability comparison for $n = 2^{10}$ to $n = 2^{16}$ (noisy case)

Solver	m	n	T	$\|e\|_2$	$\frac{\|x_k - x^*\|}{\|x^*\|}$	Time (s)
l1ls	512	1024	51	0.0241	0.0029	0.2652
NESTA	512	1024	51	0.0241	0.0007	0.546
GPSR	512	1024	51	0.0241	0.0034	**0.0624**
FPC	512	1024	51	0.0241	0.0007	0.3744
ISCR	512	1024	51	0.0241	**0.0004**	0.2028
l1ls	1024	4096	160	0.03188	0.0087	2.0904
NESTA	1024	4096	160	0.03188	0.0018	1.9812
GPSR	1024	4096	160	0.03188	0.0078	**0.5928**
FPC	1024	4096	160	0.03188	0.0009	1.7472
ISCR	1024	4096	160	0.03188	**0.0005**	0.7644
l1ls	2048	4096	205	0.04555	0.0027	0.702
NESTA	2048	4096	205	0.04555	0.0006	1.2324
GPSR	2048	4096	205	0.04555	0.0033	**0.2652**
FPC	2048	4096	205	0.04555	0.0007	1.0296
ISCR	2048	4096	205	0.04555	**0.0004**	0.858
l1ls	2048	8192	320	0.04578	0.0066	2.9172
NESTA	2048	8192	320	0.04578	0.0017	2.6676
GPSR	2048	8192	320	0.04578	0.0081	**0.8892**
FPC	2048	8192	320	0.04578	0.0009	2.1528
ISCR	2048	8192	320	0.04578	**0.0005**	1.2948
l1ls	4096	16384	640	0.06451	0.0071	4.8204
NESTA	4096	16384	640	0.06451	0.0018	4.6176
GPSR	4096	16384	640	0.06451	0.008	**1.6224**
FPC	4096	16384	640	0.06451	0.0009	3.9156
ISCR	4096	16384	640	0.06451	**0.0005**	2.6832
l1ls	8192	32768	1280	0.09084	0.0085	7.9873
NESTA	8192	32768	1280	0.09084	0.0017	7.3632
GPSR	8192	32768	1280	0.09084	0.0081	**2.6988**
FPC	8192	32768	1280	0.09084	0.0009	7.098
ISCR	8192	32768	1280	0.09084	**0.0005**	4.68
l1ls	16384	32768	1638	0.12868	0.0037	3.9624
NESTA	16384	32768	1638	0.12868	0.0007	5.9436
GPSR	16384	32768	1638	0.12868	0.0035	**1.17**
FPC	16384	32768	1638	0.12868	0.0008	5.0388
ISCR	16384	32768	1638	0.12868	**0.0004**	3.9936
l1ls	16384	65536	2560	0.12778	0.0084	19.5937
NESTA	16384	65536	2560	0.12778	0.0017	15.6157
GPSR	16384	65536	2560	0.12778	0.0079	**5.1636**
FPC	16384	65536	2560	0.12778	0.0009	11.5441
ISCR	16384	65536	2560	0.12778	**0.0005**	9.2197

Table 3.2: Scalability comparison for $n = 2^{17}$ to $n = 2^{23}$ (noisy case)

Solver	m	n	T	$\|e\|_2$	$\frac{\|x_k - x^*\|}{\|x^*\|}$	Time (s)
l1ls	32768	131072	5120	0.18103	0.0071	64.9744
NESTA	32768	131072	5120	0.18103	0.0017	40.7475
GPSR	32768	131072	5120	0.18103	0.008	**15.4441**
FPC	32768	131072	5120	0.18103	0.0009	28.4858
ISCR	32768	131072	5120	0.18103	**0.0005**	25.787
l1ls	65536	262144	10240	0.25581	0.0058	166.4219
NESTA	65536	262144	10240	0.25581	0.0017	96.5178
GPSR	65536	262144	10240	0.25581	0.0079	**32.9942**
FPC	65536	262144	10240	0.25581	0.0009	65.8792
ISCR	65536	262144	10240	0.25581	**0.0005**	50.8095
l1ls	131072	524288	20480	0.36299	0.0076	419.7207
NESTA	131072	524288	20480	0.36299	0.0017	197.1229
GPSR	131072	524288	20480	0.36299	0.008	**66.5344**
FPC	131072	524288	20480	0.36299	0.0009	130.2764
ISCR	131072	524288	20480	0.36299	**0.0005**	117.6716
l1ls	262144	1048576	40960	0.51329	0.006	1224.2334
NESTA	262144	1048576	40960	0.51329	0.0016	421.7955
GPSR	262144	1048576	40960	0.51329	0.0079	**165.9227**
FPC	262144	1048576	40960	0.51329	0.0009	308.0396
ISCR	262144	1048576	40960	0.51329	**0.0005**	245.452
l1ls	524288	1048576	52429	0.72438	0.0023	573.4285
NESTA	524288	1048576	52429	0.72438	0.0006	290.9107
GPSR	524288	1048576	52429	0.72438	0.0034	**59.3272**
FPC	524288	1048576	52429	0.72438	0.0007	202.0525
ISCR	524288	1048576	52429	0.72438	**0.0004**	196.9045
l1ls	524288	2097152	81920	0.72374	0.0063	3035.8887
NESTA	524288	2097152	81920	0.72374	0.0017	911.0926
GPSR	524288	2097152	81920	0.72374	0.0079	**261.0833**
FPC	524288	2097152	81920	0.72374	0.0009	631.726
ISCR	524288	2097152	81920	0.72374	**0.0005**	560.2776
l1ls	1048576	4194304	163840	1.02412	0.0081	4989.3332
NESTA	1048576	4194304	163840	1.02412	0.0016	1963.3974
GPSR	1048576	4194304	163840	1.02412	0.0079	**568.5612**
FPC	1048576	4194304	163840	1.02412	0.0009	1454.6313
ISCR	1048576	4194304	163840	1.02412	**0.0005**	1134.0961
l1ls	2097152	8388608	327680	1.44764	0.0069	13307.6809
NESTA	2097152	8388608	327680	1.44764	0.0017	3527.7598
GPSR	2097152	8388608	327680	1.44764	0.0079	**1182.082**
FPC	2097152	8388608	327680	1.44764	0.0009	2440.6356
ISCR	2097152	8388608	327680	1.44764	**0.0005**	2091.443

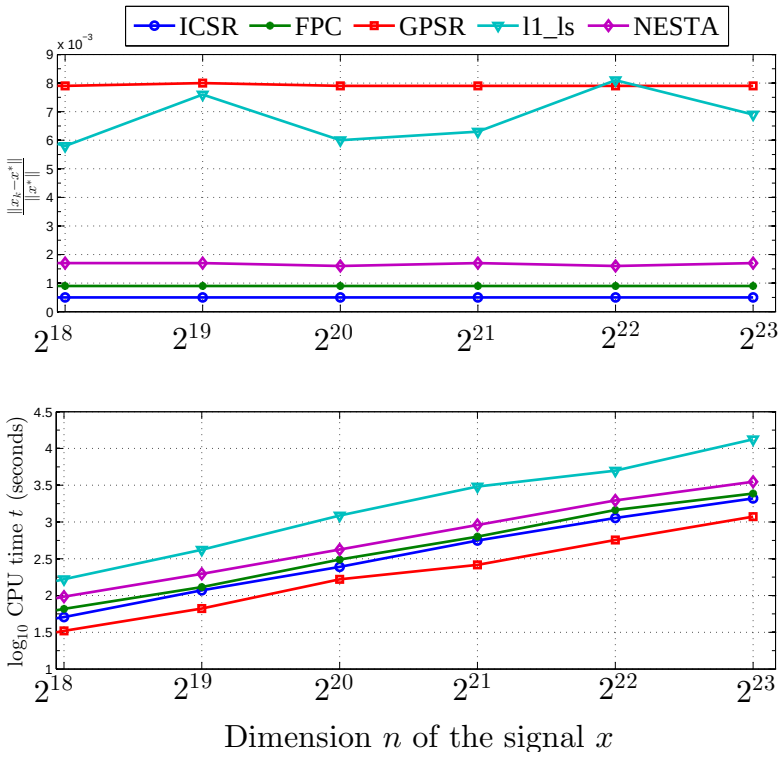


Figure 3.4: Scalability for $n = 2^{18}$ to $n = 2^{23}$ (noisy case)

Tables with the reported results for this case can be found as part of the Appendix.

The tolerance used in the CG method (Algorithm 2) for solving the $m \times m$ system of linear equations of the form (3.32) was $\epsilon_{CG} = 5 \times 10^{-4}$ with an allowed maximum number of iterations of $k_{CG} = 25$, which allowed us to get inexact directions sufficiently good for the convergence of our method. The maximum number of iterations $k_{\max}$ of Algorithm 1 was set to 70 (100 for the noiseless case) with a tolerance of $\epsilon = 6 \times 10^{-4}$ for checking proximity between two consecutive approximations for the solution of the ℓ_1 problem. The initial parameter μ was set as $\mu_0 = 1$, and the reduction at each step was computed as $\mu_{k+1} = \mu_k/5$.

For all experiments the penalization parameter was chosen as $\lambda = \frac{0.02}{\sqrt{T}} \left(= \frac{0.02}{\sqrt{\|x^*\|_0}} \right)$. The reason for this is explained in the following. Consider a sufficiently good approximation $\hat{x}$

for the solution x^* of the ℓ_1 optimization problem. Let h be the function defined by

$$h(x) = \lambda\|x\|_1 + \frac{1}{2}\|Ax - b\|_2^2, \tag{3.33}$$

that is, h is the objective function of problem (3.1). A vector x^* is a critical point of $h(x)$ if

$$\mathbf{0} \in \lambda\partial\left(\|x^*\|_1\right) + A^T(Ax^* - b), \tag{3.34}$$

where $\partial\left(\|x^*\|_1\right)$ denotes the subdifferential of the ℓ_1 norm [42], which satisfies

$$\partial\left(|x|\right) = \begin{cases} \{1\} & \text{if } x > 0, \\ [-1, 1] & \text{if } x = 0, \\ \{-1\} & \text{if } x < 0. \end{cases} \tag{3.35}$$

From the relation in (3.34) we can see that

$$\|A^T(Ax^* - b)\|_\infty^2 = \lambda^2. \tag{3.36}$$

Now, for the experiments we implemented with partial DCT matrices (also applies for orthogonalized Gaussian matrices), the relation $AA^T = I$, where I is the identity matrix, always holds. Therefore,

$$\begin{aligned} \|A\hat{x} - b\|_2^2 = \|A^T(A\hat{x} - b)\|_2^2 &\lessapprox \mathcal{O}(\|x^*\|_0)\|A^T(A\hat{x} - b)\|_\infty^2 \\ &\leq \mathcal{O}(\|x^*\|_0) \cdot \lambda^2, \end{aligned} \tag{3.37}$$

where in the first approximate inequality we have used an improved estimate of the relation $\|\cdot\|_2 \leq \sqrt{n}\|\cdot\|_\infty$ (equivalence of norms), and the second inequality follows from the optimality

of x^* (as (3.36) suggests). Therefore, a good estimate for the value of λ is given by

$$\lambda = \frac{\delta}{\sqrt{\|x^*\|_0}}, \quad \text{for some } \delta > 0, \tag{3.38}$$

and our experimentation showed that choosing $\delta = 0.02$ gives sufficiently accurate results. In [35] is shown that the parameter λ must be in the interval $\left(0, \|A^T b\|_\infty\right]$.

Our code was written in MATLAB and was run on a Windows 7 machine with a 2.20 GHz Intel i7 processor and 8 GB of memory. The MATLAB version was 7.14 (R2012a).

Chapter 4

Classification Problem

In pattern recognition and machine learning, a *classification problem* refers to finding an algorithm for assigning a given input data into one of several categories. Since many natural signals are sparse or compressible, in the sense that they have short representations when expressed in an appropriate basis, we propose to apply the *selective nature* of sparse representation to perform classification. As studied in the previous chapter, ℓ_1-minimization techniques provide a satisfactory methodology to solve sparse representation problems. We propose a classifier based on the solution of an ℓ_1-minimization problem for classification.

4.1 Description

Machine Learning is a research area concerned with the design of systems that can *learn* from provided input. Usually, such systems are designed to use learned knowledge to handle similar input in the future. For instance, an email spam-detecting system, where a given set of emails are marked as spam or not-spam, learns the common features of spam emails to be able to identify future email messages as either spam or not-spam. Assigning a diagnosis to a given patient as described by observed characteristics of the patient is another example. This technique is known as supervised statistical classification. Supervised because the system is first trained using already classified training data. A supervised learning system performing classification is commonly called a *classifier*.

Formally, given an input dataset, $\mathbf{W} = \{\mathbf{w}_1, ..., \mathbf{w}_n\}$, a set of labels/classes $\mathbf{T} = \{t_1, ..., t_n\}$, and a training dataset $\mathbf{D} = \{(\mathbf{x}_i, t_i) : i = 1, ..., n\}$ such that t_i is the label/class oassociated to the sample $\mathbf{x}_i$, a classifier is a mapping $\mathcal{F}$ from $\mathbf{W}$ to $\mathbf{T}$, assigning the correct label t to

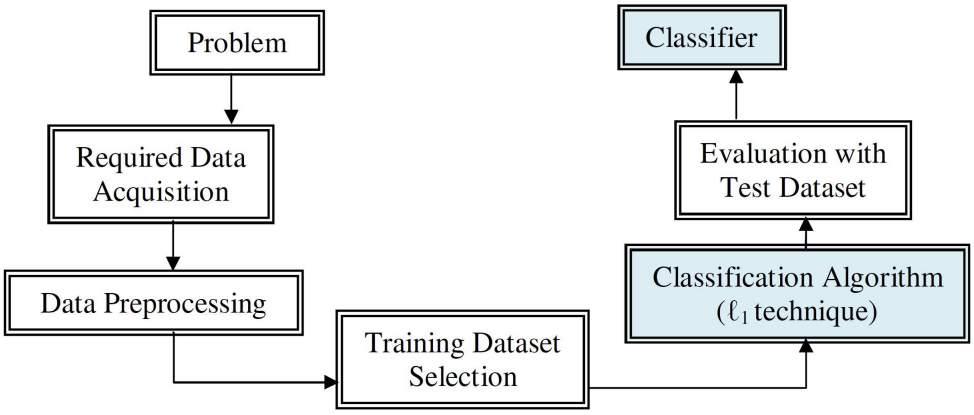


Figure 4.1: The process of classification

a given input $\mathbf{w}$, that is, $\mathcal{F}(\mathbf{w}, \mathbf{D}) = t$.

Classification problems arise in many different areas such as computer vision, medical imaging, video tracking, drug discovery and development, geostatistics, speech recognition, handwriting recognition, biological classification, document classification, internet search engines, among others.

4.2 Sparse Representation Problem: Mathematical Formulation

Let us consider a *training data set* $\{(\mathbf{x}_i, t_i) : i = 1, ..., n\},\ \mathbf{x}_i \in \mathbb{R}^d,\ t_i \in \{1, 2, ..., N\}$, where n is the number of samples and N the number of classes. The vector $\mathbf{x}_i \in \mathbb{R}^d$, represents the ith sample (for instance containing "gene expression" values), and t_i denotes its corresponding label. Assume that $d < n$, that is, the length of each sample is less than the number of elements in the training dataset.

The sparse representation problem is formulated as follows: For a testing sample $\mathbf{y} \in \mathbb{R}^d$, find the sparsest vector $\mathbf{c} = [c_1, c_2, ..., c_n]^T$ such that

$$\mathbf{y} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + ... + c_n\mathbf{x}_n. \tag{4.1}$$

Equation (4.1) states that we want to express the vector y as a linear combination of the collection $\{\mathbf{x}_1,\ \mathbf{x}_2,\ ...,\ \mathbf{x}_n\}$. Using matrix algebra notation, equation (4.1) can be posed as the underdetermined linear system of equations

$$\mathbf{y} = A\mathbf{c}, \tag{4.2}$$

where the matrix $A \in \mathbb{R}^{n\times d}$ is constructed such that column j corresponds to the sample $\mathbf{x}_j$, and the vector $\mathbf{c} = (c_1, ..., c_n)^T$. Since we look for a sparse vector $\mathbf{c}$, relation (4.1) states that the test sample $\mathbf{y}$ is a linear combination of only few training samples. The underdetermined linear system of equations in (4.2) has infinitely many solutions, from which we are interested in the sparsest one. This leads us to formulate the sparse representation problem

$$\begin{aligned} &\min \|\mathbf{c}\|_0 \\ &\text{subject to} \quad A\mathbf{c} = \mathbf{y}, \end{aligned} \tag{4.3}$$

where, as discussed in the previous chapter, $\|\cdot\|_0$ denotes the ℓ_0-"norm" and serves as measure of sparcity of the vector $\mathbf{c} \in \mathbb{R}^d$.

We show that indeed a valid test sample can be represented using only the training samples from the same class, therefore inducing a natural sparse representation. Let us rearrange the given n_i training samples from the same i-th class as the columns of a submatrix $A_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,n_i}] \in \mathbb{R}^{d\times n_i}$. That is, we group all of those samples with the *same label* into a matrix A_i. In case we have sufficient training samples of the i-th class, any test sample $\mathbf{y}$ from the same class will be represented as a linear combination of the training samples associated with class i :

$$\mathbf{y} = c_{i,1}\mathbf{x}_{i,1} + c_{i,2}\mathbf{x}_{i,2} + ... + c_{i,n_i}\mathbf{x}_{i,n_i}, \tag{4.4}$$

for some values of $c_{i,j} \in \mathbb{R},\ j = 1, ..., n_i$. Now, making use of the whole training dataset, we define a $d \times n$ matrix A by concatenating all of the n training samples of the different N

classes, that is $A = [A_1, A_2, ..., A_N]$. Then again, the linear representation of the test sample $\mathbf{y}$ that belongs to class i is written by:

$$\mathbf{y} = A\mathbf{c}, \tag{4.5}$$

where $\mathbf{c} = [0, ..., 0, c_{i,1}, c_{i,2}, ..., c_{i,n_i}, 0, ...0]^T \in \mathbb{R}^n$. Therefore, the test sample $\mathbf{y}$ is expressed by a sparse linear representation, and we find such representation by solving a problem of the form (4.3).

Consider now the case $d > n$, that is, the lenght d of each sample $\mathbf{x}_i$ is larger than the number n of available samples in our training dataset. In such case, a relation of the form (4.2) produces an overdetermined linear system of equations, since we would have more equations than unknowns. Therefore, in this work our formulation for this case considers a corruption vector $\mathbf{r}$ associated to the problem, in such way that any sample is written as:

$$\mathbf{y} = A\mathbf{c} + \mathbf{r}. \tag{4.6}$$

This can be expressed as $\mathbf{y} = B\mathbf{d}$, with

$$B = [A\ I]\,,\ \mathbf{d} = [\mathbf{c},\ \mathbf{r}]^T, \tag{4.7}$$

where I represents a $d \times d$ identity matrix, and $B \in \mathbb{R}^{d\times(d+n)}$, $\mathbf{d} \in \mathbb{R}^{n+d}$. Now, the system in (4.6) is always underdetermined, and equation (4.6) has infinitely many solutions from which we are interested in the sparsest one. Then, the sparse linear representation for the test sample $\mathbf{y}$ can be obtained by solving the following minimization problem

$$\begin{aligned} &\min \|\mathbf{d}\|_0 \\ &\text{subject to} \quad B\mathbf{d} = \mathbf{y}. \end{aligned} \tag{4.8}$$

To obtain a sparse linear representation for the test sample $\mathbf{y}$, we propose to use a

convex relaxation technique via ℓ_1 minimization as discussed in the previous chapter. The methodology described in the previous chapter can be perfectly implemented in these type of problems, an the algorithm we proposed can be used to find such sparse representation. Since real data is noisy, we might not be able to represent the test sample with exactly a sparse combination of the training samples. Therefore, we must consider a noise with bounded energy in our representation, and solve a problem of the form

$$\begin{aligned} &\min_{\mathbf{c}} \|\mathbf{c}\|_1 \\ &\text{subject to } \|A\mathbf{c} - \mathbf{y}\| \leq \epsilon, \end{aligned} \tag{4.9}$$

for some tolerance $\epsilon > 0$. Alternatively, as discussed in Chapter 3, problem (4.9) can in turn be formulated as:

$$\begin{aligned} &\min_{\mathbf{c},\mathbf{e}} \lambda\|\mathbf{c}\|_1 + \frac{1}{2}\mathbf{e}^T\mathbf{e} \\ &\text{subject to } A\mathbf{c} - \mathbf{e} = \mathbf{y}, \end{aligned} \tag{4.10}$$

and now we can use our proposed ISCR algorithm for ℓ_1-minimization problems. The formulation above applies for the case where $d < n$. For the case $d > n$, we obtain a similar formulation by simply letting $B = [A\ I]$, and $\mathbf{d} = [\mathbf{c},\ \mathbf{r}]^T$, play the role of A and $\mathbf{c}$ respectively in problem (4.10).

One of the advantages of our formulation is that lack of robustness with respect to noise, missing data and outliers can be overcome (a known property of the ℓ_1 norm as regularization of an inverse problem). Also, we do not need to care for model selection as in support vector machine approaches, since the selective nature of the sparse representation captures the level of membership of a given input to one of the different classes. In the following, we describe how to decide the class of a given input after obtaining its sparse representation.

4.3 Classification using Sparse Representation

Once the sparse representation vector $\mathbf{c}$ has been found as a solution to (4.3), we proceed to classify a testing sample $\mathbf{y}$ in one of the N classes. The approach consists in associating the nonzero components of $\mathbf{c}$ with the columns of A corresponding to those training samples that have the same class.

First, let Ω_k denote the set of indices given by

$$\Omega_k = \{j \ : \ \text{training sample } \mathbf{x}_j \text{ has label } t_j = k \ \}. \tag{4.11}$$

Therefore,

$$\Omega_1 \cup \Omega_2 \cup ... \cup \Omega_N = \{1, 2, ..., n\}, \tag{4.12}$$

that is, the collection of sets of indices $\{\Omega_i\}_{i=1}^{N}$ forms a partition of the set $\{1, 2, ..., n\}$, where n is the amount of samples available in the training dataset.

Then we define the *discriminant* functions by

$$g_k(\mathbf{y}) = \|\mathbf{y} - A\mathbf{c}_k\|_2, \qquad k = 1, ..., \ N, \tag{4.13}$$

where $A\mathbf{c}_k$ is defined by

$$A\mathbf{c}_k = \sum_{j \in \Omega_k} c_j \mathbf{x}_j, \tag{4.14}$$

Notice that the function g_k in (4.13), measures the error obtained when the testing sample $\mathbf{y}$ is represented with elements of the training set that have the same class k.

Finally, we classify $\mathbf{y}$ in the category with the smallest approximation error. That is, we

compute

$$g_s(\mathbf{y}) = \min\{g_1(\mathbf{y}),\ g_2(\mathbf{y}), ...,\ g_N(\mathbf{y})\},$$
$$\mathbf{t} = s, \tag{4.15}$$

and conclude that the testing sample $\mathbf{y}$ has label $\mathbf{t} = s$. In this manner, we identify the class of the test sample $\mathbf{y}$ based on how effectively the coefficients associated with the training samples of each class recreate $\mathbf{y}$.

Algorithm 3 Classification Algorithm based on Sparse Representation (SR)

Goal: Classify a given input $\mathbf{y}$ in one of several categories $j = 1, ...,\ N$

Input: training matrix $A = [\mathbf{x}_1|\ \mathbf{x}_2|...|\ \mathbf{x}_n]$ and corresponding labels.

Output: Label (class) $\hat{\mathbf{t}}$ of the given input sample $\mathbf{y}$.

Step 1: (optional) Normalize the columns of the matrix A to have unit norm.

Step 2: Use ISCR Algorithm 1 to solve the ℓ_1 minimization
$\min_{\mathbf{c},\mathbf{e}} \lambda\|\mathbf{c}\|_1 + \frac{1}{2}\mathbf{e}^T\mathbf{e}$ subject to $A\mathbf{c} - \mathbf{e} = \mathbf{y}$.

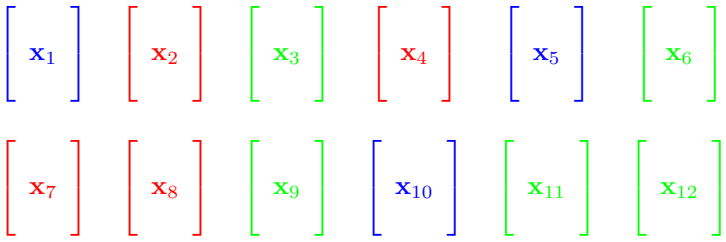
Step 3: Calculate the N discriminant functions $g_k(\mathbf{y})$ as in (4.13)

Step 4: Find minimum discriminant value
$g_s(\mathbf{y}) = \min\{g_1(\mathbf{y}),\ g_2(\mathbf{y}), ...,\ g_N(\mathbf{y})\}$.

Step 5: The input $\mathbf{y}$ has label $\mathbf{t} = s$

The Classification Algorithm 3 is written for the case when $d < n$. If we are in the case where $d > n$, we again simply let $B = [A\ I]$, and $\mathbf{d} = [\mathbf{c},\ \mathbf{r}]^T$, play the role of A and $\mathbf{c}$ respectively.

The basic idea of the Sparse Representation (SR) approach for classification problems is summarized in the following sentence: *"A valid test sample can be represented using only the training samples from the same class, therefore inducing a natural sparse representation"*



$$\mathbf{y} = c_1\mathbf{x}_1 + c_5\mathbf{x}_5 + c_{10}\mathbf{x}_{10}, \qquad \mathbf{c} = (c_1, 0, 0, 0, c_5, 0, 0, 0, 0, c_{10}, 0, 0)^T$$

$$\mathbf{y} = c_2\mathbf{x}_2 + c_4\mathbf{x}_4 + c_7\mathbf{x}_7 + c_8\mathbf{x}_8, \qquad \mathbf{c} = (0, c_2, 0, c_4, 0, 0, c_7, c_8, 0, 0, 0, 0)^T$$

$$\mathbf{y} = c_3\mathbf{x}_3 + c_6\mathbf{x}_6 + c_9\mathbf{x}_9 + c_{11}\mathbf{x}_{11} + c_{12}\mathbf{x}_{12}, \qquad \mathbf{c} = (0, 0, c_3, 0, 0, c_6, 0, 0, c_9, 0, c_{11}, c_{12})^T$$

Figure 4.2: (Top) Training samples for 3 different classes (blue, red, green) (Bottom) Representation for different inputs

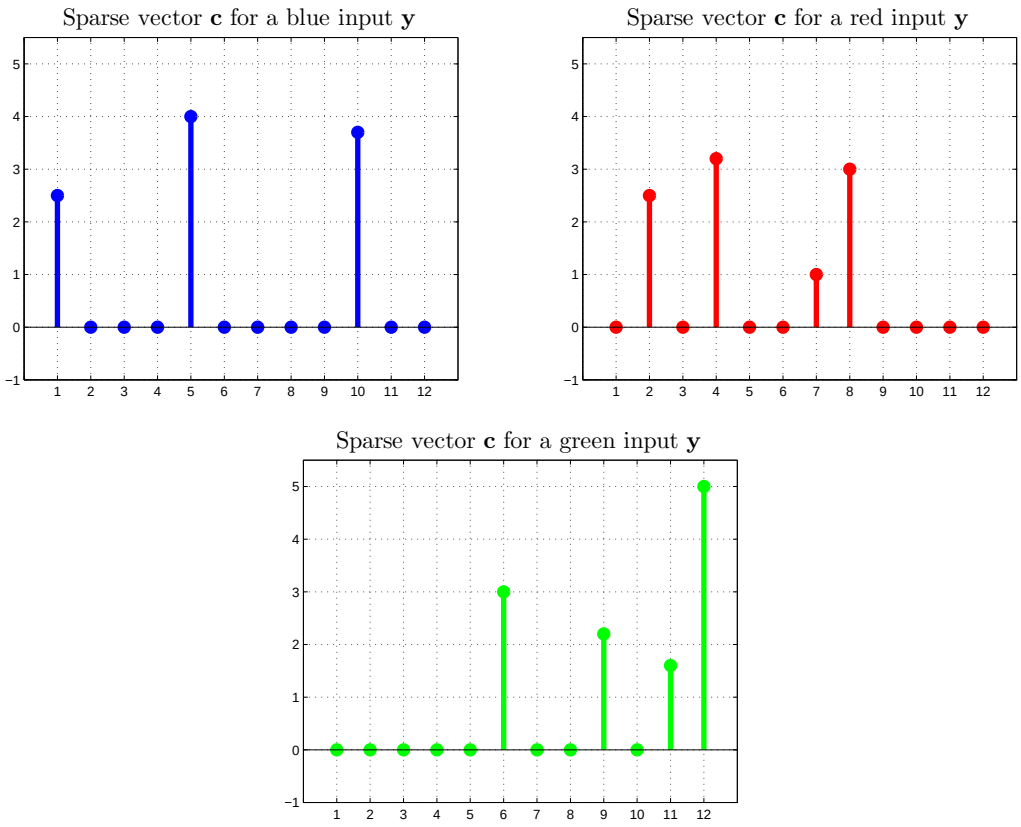


Figure 4.3: Sparse Vector Solutions for different inputs

Chapter 5

Numerical Experimentation

In this chapter we describe the methodology used for testing the effectiveness of our classification approach. As performance metric, we evaluate the accuracy of the proposed sparse representation technique for classification in a *10-fold stratified cross-validation* experiment.

5.1 Experiment Design

We test our method in two different type of datasets: one kind of datasets where we consider a large number of samples n with few features d; and the other one where we have a small number of samples n each with a large number of features d. In the former case, we test the performance of our method using the classic Fisher's Iris dataset [24]. This dataset, also known as the Iris flower dataset, consists of samples from each of the three classes of Iris flowers. The samples, which are included in the MATLAB Statistic Toolbox, can be easily accessed and used for classification purposes. For the kind of datasets where we have a large number of features with a few samples, we use six datasets available at the Gene Expression Model Selection (GEMS) library for cancer classification using gene expression data. The GEMS software includes a graphical user interface and can be freely downloaded at `http://www.gems-system.edu/`. This software was also used in [45] for studying the performance of multicategory classifiers on gene expression cancer diagnosis. Our results are compared with the Support Vector Machines (SVM) technique, which has been successfully applied in gene profile classification.

All experiments were performed on a PC with an Intel i7 processor 2.20 GHz processor, 8 GB of memory, and MATLAB version was 7.14 (R2012a) under Windows 7.

5.1.1 K-fold cross validation

Classifier performance is commonly measured by the classifier's error rate on the entire population. Cross Validation is a statistical method for evaluating machine learning algorithms in which the data is divided in two sets: one used for the training stage, and the second one used for testing (validation). These two training and testing sets should cross-over in consecutive rounds in such a way that each sample in the data set has a chance of being validated.

In the case of K-fold cross validation, a K-fold partition of the dataset is created by splitting the data into K equally (nearly equal) sized subsets (folds), and then for each of the K experiments, $K-1$ folds are used for *training* and the remaining one for *testing*. Therefore, each of the K subsamples is used exactly once as the validation data. One of the advantages of K-fold cross validation is that, eventually, all samples in the dataset are used for both testing and training.

If a large number of folds is used, the bias of the true error will be small though the method might be computationally expensive. A common choice for K-Fold Cross Validation is $K = 10$. The work in [36], compares several approaches for estimating accuracy, and recommends stratified 10-fold cross-validation as the best model selection method because it provides less biased estimation of the actual accuracy.

Given a dataset with n elements and a classifier algorithm, say $\mathcal{F}$, the averaged cross-validation accuracy of the classifier $\mathcal{F}$ on these n samples, can be considered as an estimate for the accuracy of $\mathcal{F}$ on *unseen* data when the classifier is trained with all the different samples.

5.1.2 Support Vector Machines (SVM)

We will be comparing the results of our proposed method for classification problems, with the well known Support Vector Machines (SVM) method, that has been commonly used in different pattern recognition and machine learning applications.

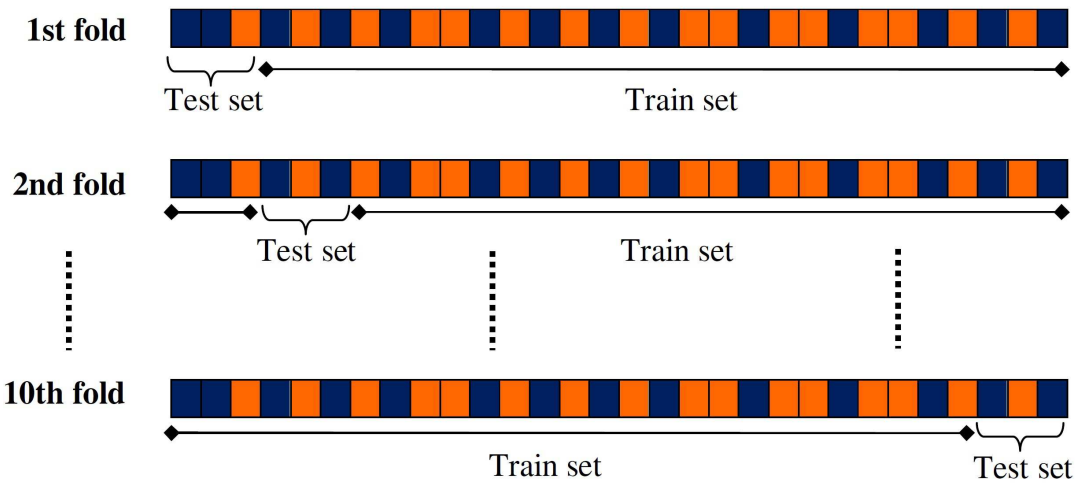


Figure 5.1: A 10-fold cross validation partition example

Support vector machines (SVMs) are a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The original SVM algorithm was proposed by Vladimir Vapnik and the current standard implementation was proposed by Corinna Cortes and Vladimir Vapnik [16]. Standard SVM takes a set of input data, and predicts, for each given input, which of two possible classes the input is a member of, which makes the SVM a non-probabilistic binary linear classifier.

Since an SVM is a classifier, then given a set of training examples, each marked as belonging to one of a set of specific categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. Intuitively, an SVM model is a representation of the samples as points in space, mapped so that the samples of the separate categories are divided by a clear gap that is as wide as possible.

In the original SVM approach, an N-class classification problem is converted into N two-class problems (binary classification), and in the j-th two-class problem, the optimal decision function that separates class j from the remaining classes is determined. If more than one decision function classify a data point into a definite class, the data point is not classifiable. Slow training is also a possible drawback of support vector machines approaches. This has to do with the fact that support vector machines are *trained* by solving quadratic programming

problems where the number of variables is equal to the number of samples in the training data set. When a large number of training data is available, the training process might turn slow. More information about the different strategies used in SVM for classification problems are described in [1, 49]. Techniques for accelerating the training process, specialized methods for multiclass problems, and nonlinear separation of class data, among other SVM related topics, are also discussed in [1]. A short mathematical formulation of the SVM method is included in the Appendix.

5.2 Large number of samples and few features $(d < n)$

We investigate the performance of the sparse representation approach for classification on the classic Fisher's Iris dataset [24]. Fisher developed a linear discriminant model to distinguish one species from another based on the combination of four different features. MATLAB has incorporated this dataset under the name of `fisheriris` as part of its Statistics Toolbox.

5.2.1 Dataset Description

This dataset was introduced in 1936 by Sir Ronald Aylmer Fisher and consists of 50 samples from each of the three classes of Iris flowers: *iris setosa*, *iris versicolor*, and *iris virginica*. The dataset contains 4 different feature measurements (in centimeters): the sepal length, sepal width, petal length, and petal width of 150 iris specimens.

We are interested in studying the effectiveness of the sparse representation approach on this dataset since it represents a dataset where we have more samples than features per sample. Specifically, we have $d = 4$, $n = 150$, and our formulation of the problem can be written as

$$\begin{aligned} \min_{\mathbf{c},\mathbf{e}} \quad & \lambda\|\mathbf{c}\|_1 + \tfrac{1}{2}\mathbf{e}^T\mathbf{e} \\ \text{subject to} \quad & A\mathbf{c} - \mathbf{e} = \mathbf{y}, \end{aligned} \tag{5.1}$$

giving rise to a highly underdetermined linear system where the matrix $A \in \mathbb{R}^{d \times n}$. In Figure 5.2 and 5.3 we show how the sepal and petal measurements differ from class to class. We use a scatter plot to show the values of width and length for both sepal and petal.

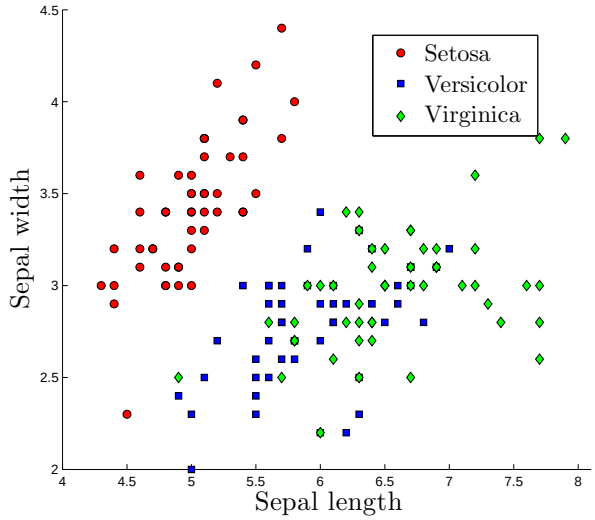


Figure 5.2: Sepal length and width

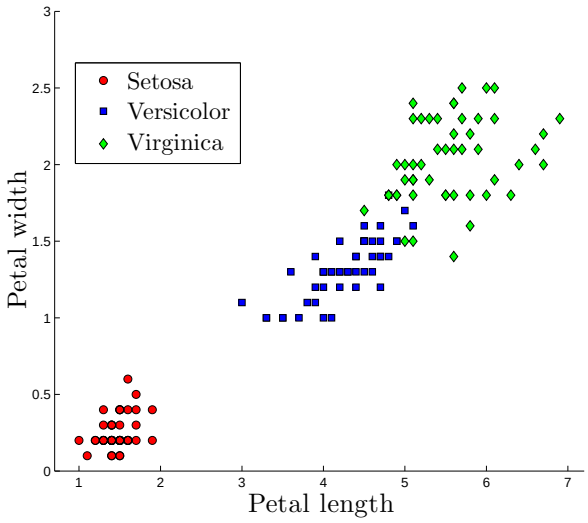


Figure 5.3: Petal length and width

5.2.2 Numerical Results

We set up a numerical experiment using a 10-fold cross-validation test. Our ℓ_1 sparse representation for classification technique was able to accurately predict the class of every test sample in the dataset with a 96.36% effective rate. In Figure 5.4 we show how an iris virginica test sample (class 3) is sparsely represented by only those samples in the training data set with the label for iris virginica class.

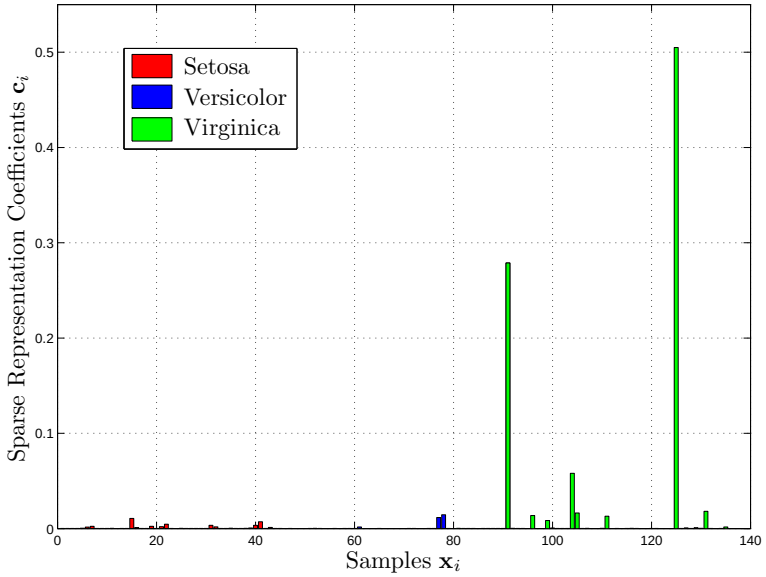


Figure 5.4: Sparse Representation of a virginica test sample $\mathbf{y}$ in the `fisheriris` dataset

Discussion The method proposed is very effective for classification in this case too, and the fact that the resulting linear system of equations for sparse representation is highly underdetermined helps to have classification results in a short period of time. We take advantage of the relation between measurements and features, namely $d \ll n$, inspired by the theory of Compressed Sensing, where our ISCR algorithm solves a system of size $d \times d$ at each iteration using a conjugate gradient method and boosting the performance of the classification procedure.

5.3 Large number of features and few samples ($d > n$)

In this numerical experimentation we use 6 different datasets from the GEMS library that are freely available in MATLAB `.mat` format at the webpage `http://www.gems-system.edu/`. We also compare our numerical results, with the best ones obtained using the Support Vector Machine (SVM) implementation available in the GEMS software.

5.3.1 Dataset Description

A short description of the datasets used follows:

`9_Tumor`. The dataset comes from a study of 9 human tumor types: NSCLC, colon, breast, ovary, leukemia, renal, melanoma, prostate, and CNS.

`11_Tumors`. Consists of gene expression data of 11 various human tumor types: ovary, bladder/ureter, breast, colorectal, gastro-esophagus, kidney, liver, prostate, pancreas, adeno lung, and squamous lung.

`Prostate_Tumor`. Binary dataset contains gene expression data of prostate tumor and normal tissues.

`Lung_Cancer`. Dataset of 4 lung cancer types and normal tissues.

`SRBCT`. Small, round blue cell tumors (SRBCT) of childhood.

`Brain_Tumor`. Dataset from a study of 5 human brain tumor types: medulloblastoma, malignant glioma, AT/RT, normal cerebellum, and PNET.

In the following table, the number of samples and genes for each dataset is described.

Table 5.1: GEMS dataset sizes

Dataset	# Samples	# Genes	# Classes
9_Tumors	60	5726	9
11_Tumors	174	12533	11
Prostate_Tumor	102	10509	2
Lung_Cancer	203	12600	5
SRBCT	83	2308	4
Brain_Tumor	90	5920	5

5.3.2 Numerical Results

We solve the classification problem as posed in (4.8), that is, for each test we look for a solution of the ℓ_1 minimization problem:

$$\begin{aligned} \min_{\mathbf{c},\mathbf{e}} \quad & \lambda\|\mathbf{d}\|_1 + \tfrac{1}{2}\mathbf{e}^T\mathbf{e} \\ \text{subject to} \quad & B\mathbf{d} - \mathbf{e} = \mathbf{y}, \end{aligned} \tag{5.2}$$

where B is an augmented matrix of the form $B = [A\ I]$, and $\mathbf{d} = [\mathbf{c},\ \mathbf{r}]^T$. The matrix A is just the matrix built using the dataset elements, and for our numerical experiment we normalize the columns of A in such a way that they all have unit norm, i.e. $\|e_i^T a_i\|_2 = 1$, with e_i the i-th canonical basis vector and a_i being the i-th column of A.

The ISCR algorithm (Algorithm 1) is applied to solve each of the problems of the form (5.2) that are needed at every iteration of a 10-fold cross-validation test. The ISCR algorithm and the complete validation experiment for each dataset are implemented in MATLAB. Notice that even though we use the augmented matrix $B = [A\ I]$ in our problem formulation, our proposed ISCR algorithm only requires matrix-vector multiplication operations as is confirmed also in the CG implementation shown in Algorithm 2. Thus, we do not need to

store the complete matrix B but only A since,

$$B\mathbf{d} = A\mathbf{c} + \mathbf{r}, \tag{5.3}$$

$$B^T\mathbf{y} = [A^T\mathbf{y}, \mathbf{y}]^T, \tag{5.4}$$

so we can implement in a fast way the matrix-vector multiplications required by our method.

We compare the results using the Sparse Representation (SR) approach proposed in this work, with the classification method of Support Vector Machines (SVM). In order to perform this comparison, we use the software GEMS which has implemented several multiclass SVMs including one-versus-rest (OVR), one-versus-one (OVO), and directed acyclic graph (DAG). Polynomial and Radial Basis Functions (RBF) kernels are used for SVMs. The results for the best possible setup of the SVM method is reported

In Table 5.2 we show the performance measure results for each of the datasets tested in this experimentation and we compare the classifier's error rate.

Table 5.2: Performance of classifier: sparse representation (SR) and (SVM)

Dataset	# Samples	# Genes	SR	SVM
`9_Tumors`	60	5726	66.67%	67.01%
`11_Tumors`	174	12533	97.55%	94.99%
`Prostate_Tumor`	102	10509	94.12%	93.27%
`Lung_Cancer`	203	12600	95.07%	96.05%
`SRBCT`	83	2308	100%	100%
`Brain_Tumor`	90	5920	91.11%	90.00%

In Figure 5.5, the sparse representation for the last cross-validation test on the binary dataset `Prostate_Tumor` is presented. One can notice the contrast between the large and small coefficients of the solution vector $\mathbf{c}$, suggesting that the given test sample belongs to exactly one of the two classes in this dataset. In this case, the last test sample corresponds to one of the prostate tumor samples which are represented in red color, while the normal

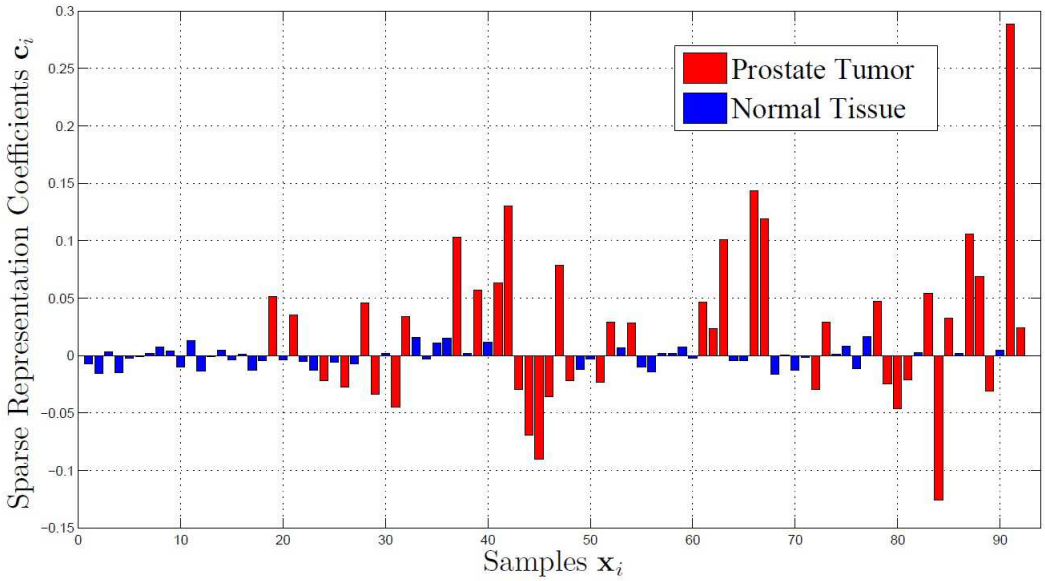


Figure 5.5: Sparse Representation of a test sample $\mathbf{y}$ in the `Prostate_Tumor` dataset

tissue samples are shown in blue. This fact confirms the idea of expressing any test sample as a linear combination of only those training samples belonging to the same class.

Discussion

The Sparse Representation (SR) results reported in Table 5.2 are encouraging when compared with the SVM approach for classification problems. We see that SR meets the best performance reached by SVMs method. In this work we reported the best performance we obtained from SVM methods using the GEMS software, which corresponds to using SVM one-versus-rest approach option available. Our SR technique in fact performs better than the SVM implementation for most of the datasets tested in this work, in terms of accuracy of the classifier according to the 10-fold cross-validation experiment studied for performance quantification.

From Table 5.2 we also see the low rate of accuracy for the dataset `9_Tumors`, which is probably related to the number of samples available for the training process: from a total of 60 samples only 2 are available for category 7 corresponding to the prostate tumor case. This contrasts with the 9 samples available for non-small cell lung cancer (NSCL); 8

samples for breast, renal and melanoma cases; 7 for colon and 6 for ovarian, leukemia, and central nervous system (CNS) cases. Therefore, in the situation when the only two samples available for category 7 happen to be in the testing dataset, generated by the random 10-fold cross validation stage, we will not have any samples of this category for training, i.e., these samples do not have any natural sparse linear representation using those elements in the training dataset.

There is also a noticeable difference in terms of running time of the experiments depending on the strategy used. For instance, for the dataset `11_Tumors`, the 10-fold cross-validation experiment took a total of 5397.93 seconds using the GEMS software, and reaching an accuracy percentage of 94.99%. On the other hand, this same experiment, using the Sparse Representation methodology, had a running time of 688.04 seconds and a final accuracy (classification rate) of 97.55%.

Definitely one of the advantages of the SR technique based on ℓ_1 optimization is that we do not need to care for *model selection* as with SVM. Also, robustness with respect to outliers and noise in the dataset is gained when using the ℓ_1 norm.

5.4 Face Recognition Example

We consider the database of human faces from the AT&T laboratories in Cambridge University Computer Laboratory, which consists of 400 images of 40 individuals each with ten different images. For some individuals, the images were taken at different times, lighting and facial expression. The size of each image is 128×128 pixels with 256 gray levels in a `.pgm` format.

In our setup, we reshape every image as a column vector with 16384 entries that represent also a column in our training data matrix .

The database can be freely downloaded from the website

`http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

Using this dataset, we again performed a 10-fold cross validation experiment for testing

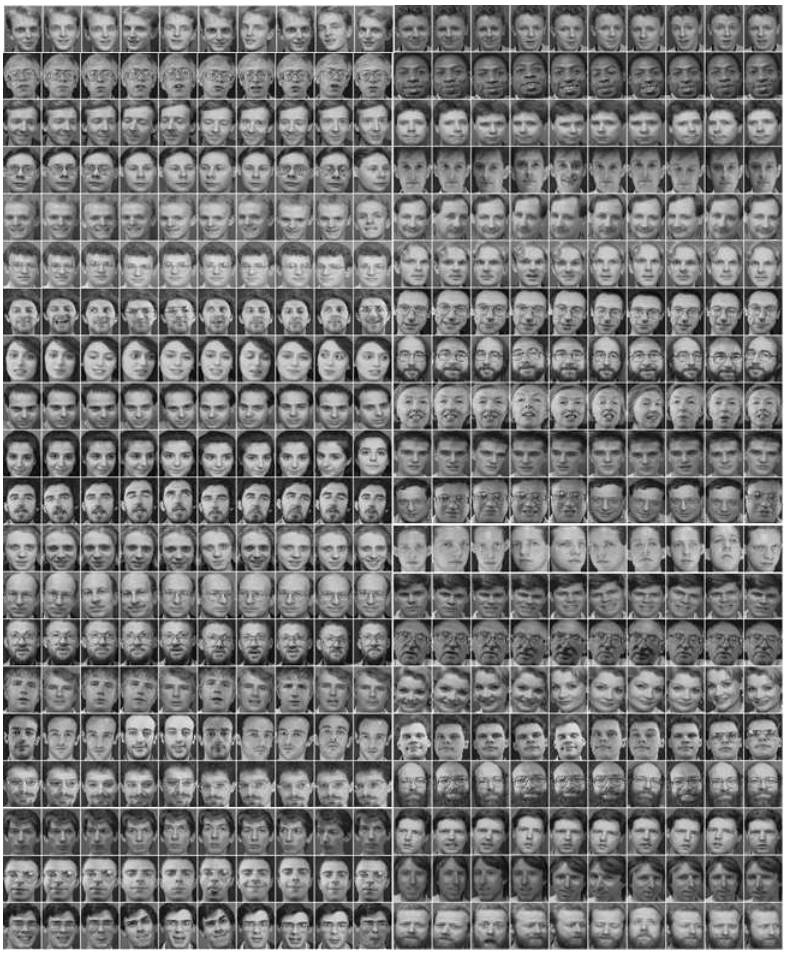


Figure 5.6: The AT&T database of human faces

the efficiency of our proposed algorithm. The results are shown in Table 5.3.

Table 5.3: Face recognition using sparse representation

Training dataset size	# Classes	Accuracy Rate	Average Time for each fold
16384×360	40	97.5%	119.689 secs

It is important to mention that we also used this dataset in a previous work for sparse representation, where the classification proccess was carried out after an inpainting process efficiently complete a dataset with missing information (missing pixels) [4]. Inpanting techniques using ℓ_1 optimization techniques have been proven to work successfully in image processong problems [41].

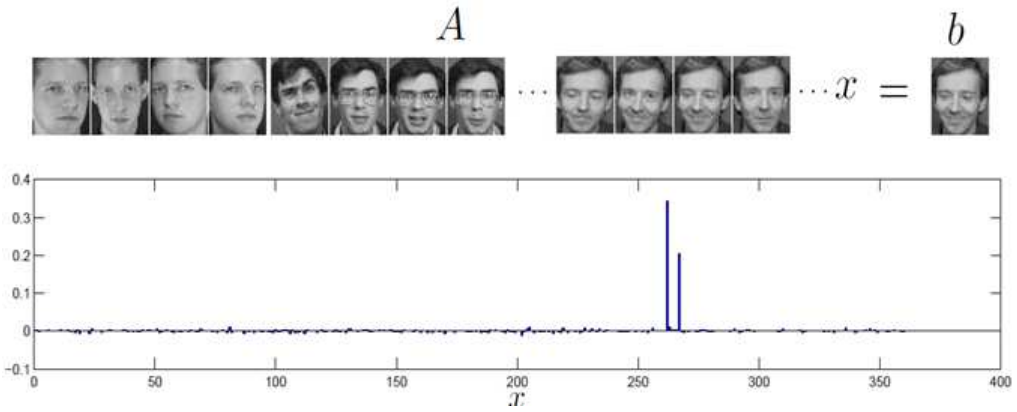


Figure 5.7: Illustration of face recognition problem using sparse representation

5.5 Handwritten Digits Example

We are also interested in testing the accuracy of our classification algorithm based on sparse representation, in a handwritten recognition problem. The dataset used in this case was the MNIST database of handwritten digits, used by Yann LeCun (Courant Institute, NYU) and Corinna Cortes (Google Labs, New York) for testing different pattern recognition algorithms, whose description can be found at `http://yann.lecun.com/exdb/mnist/index.html`

In our experimentation we used a subset of the MNIST dataset, consisting of 1000 samples by class, for a total of 10 classes (digits from 0 to 9). Each sample is a 28×28 image, that we reshape as a vector in $\mathbb{R}^{784}$.

The results of a 10-fold cross validation experiment using the MNIST database are shown in Table

Table 5.4: 10 fold cross validation results for MNIST

# Classes	# Samples by class	# Classification Rate
10	1000	95.27%

Our classification scheme based on sparse representation can be posed as the diagrama shown in Figure 5.9. Our input sample (in that case a handwritten 5 that we want to identify), enters a process where an ℓ_1 optimization problem (as the one studied in previous chapters, and for which the ISCR Algorithm 1 finds a solution) must be studied and whose

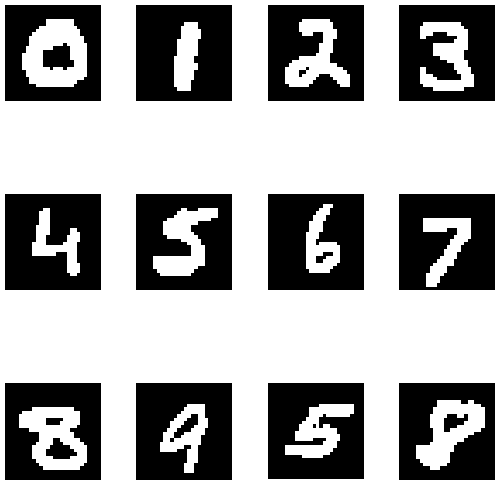


Figure 5.8: Examples of the training samples used from MNIST dataset

sparse solution is used for finding the correct label of the given input. Finally, the output of the process, expressed as a function depending on the input and the sparse solution found, is the correct label for the input we are interested in classifying.

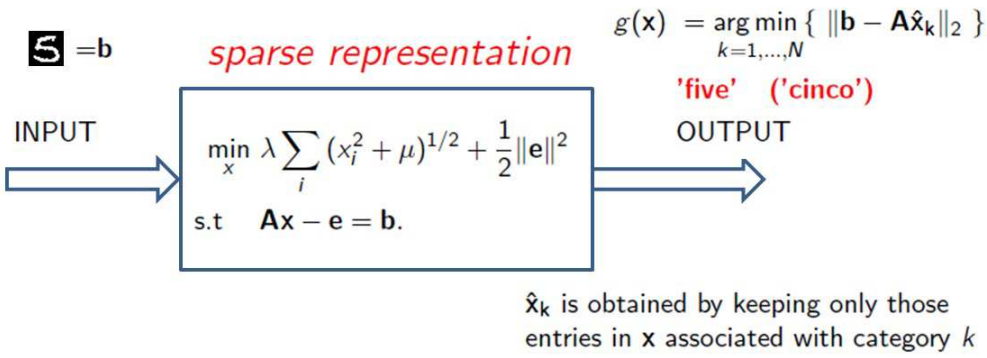


Figure 5.9: Scheme for classification using sparse representation

This dataset from MNIST is an an example for which normalization of each sample is not convenient for the process of classification. This normalization preprocessing, marked as optional on the first step of Algorithm 3, generates a difficulty for the classification process

since each sample is degraded as can be seen in Figure 5.10.

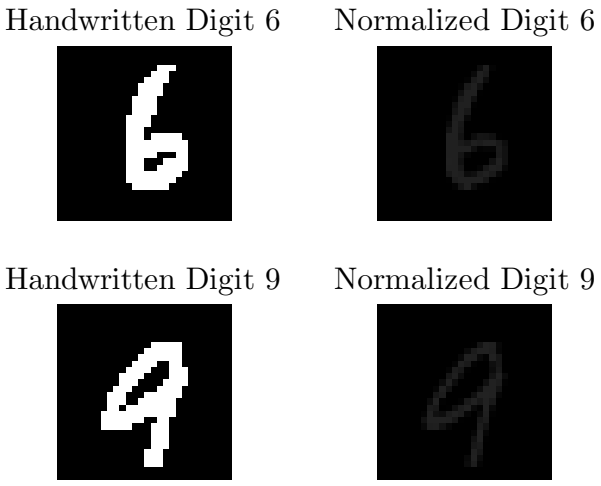


Figure 5.10: Normalized Digits

Chapter 6

Reduced-Order Classification

Since the process and techniques for acquiring and analyzing data advances every day at high rates, we are exposed to manage and study large amounts of data for many different scientific problems. High-dimensional data analysis definitely represents a key factor for classification problems, and the *"curses and blessings of dimensionality"* [21] can give us different ideas in order to improve our techniques. [1]

We focus in *dimensionality reduction* via *feature selection/extraction* in general. *Feature Selection* refers to approaches with the goal of finding optimal subsets of the original variables (attributes) [26]. A common strategy in these type of dimensionality reduction is *filtering*. For instance, in the case of gene expression data, several filtering techniques have already been studied. *Gene selection* is often applied to classification processes producing encouraging accuracy results [30], and reducing dramatically the number of genes used in classification.

6.1 Dimensionality Reduction via Feature Selection

We are interested in techniques for reducing the amount of features d on each sample data $\mathbf{x}_i \in \mathbb{R}^d$ to be considered, so the dimension of the problem can be decreased without losing important information affecting the classification rates. This will alleviate the large number of data that must be handled in some of the datasets, as well as produce quicker results when solving the sparse representation problems needed for classification.

[1] Donoho [21] uses the term "curse of dimensionality" to refer to the apparent intractability of systematically searching through a high-dimensional space. "Blessings of dimensionality" is the phenomenon where statements about high-dimensional settings could be made where moderate dimensions would be complicated.

Nowadays, there are several and effective techniques to collect data. This process accumulates data at high speed, and *preprocessing* is an important part of successful data mining and machine learning techniques. For instance, the number of genes responsible for a given type of disease may be small, so the original samples might be downsized in certain cases.

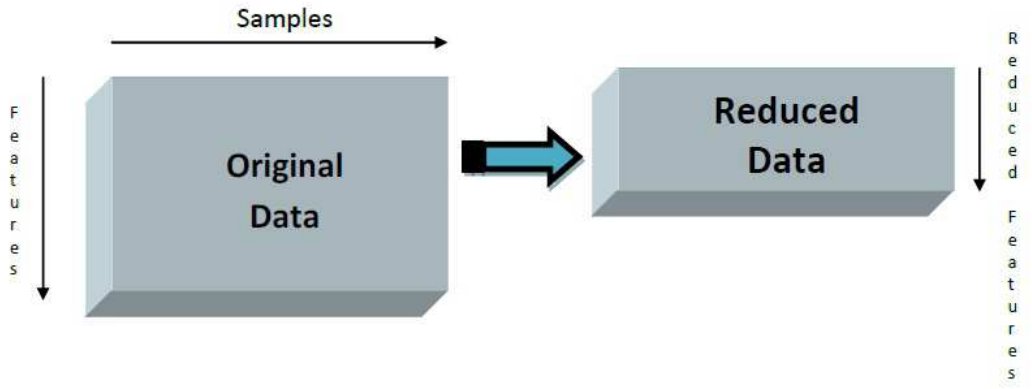


Figure 6.1: Dimensionality reduction of samples used in classification

In this work, for *feature selection* we study how some filtering techniques can improve our classification algorithm. The dimensionality reduction of the dataset can help us identify which features/attributes of the original samples actually influence in the sparse representation. Therefore irrelevant information can be discarded, and the selective nature of the sparsity technique recognizes those more important variables easily, faster, and with less error.

Classification of Subcortical Structures using Sparse Representation

The goal in this classification problem was to quantify changes in neural activity from subcortical structures. Micro Electrode recording (MER) signals were labeled by specialists in neurosurgery and neurophysiology. The acquisition and processing of this dataset is described in [25]. In this experimentation we use the dataset of $n = 52$ neural recordings divided in $N = 4$ classes: 13 signals from thalamus nucleus, 13 signals from subthalamic nucleus, 13 signals from substantia nigra, and 13 from zona incerta. Figure 6.2, taken from

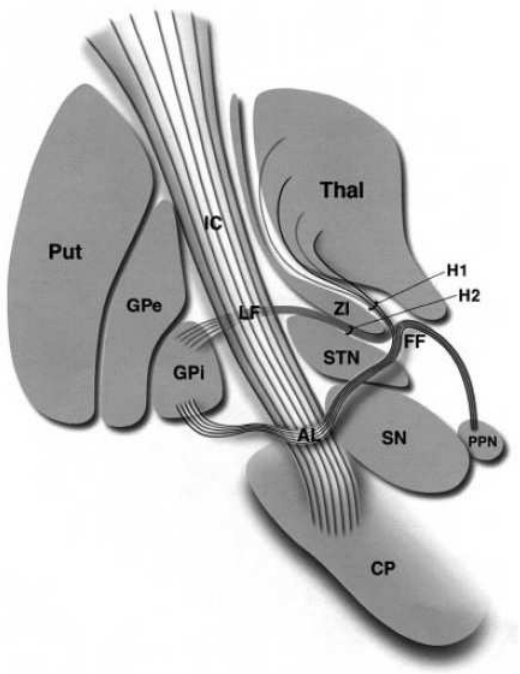


Figure 6.2: SN = substantia nigra; STN = subthalamic nucleus; Thal = thalamus; ZI = zona incerta.

[29], shows a representation of the major anatomical structures and fibre tracts associated with the subthalamic nucleus.

Raw Dataset

Each recording was acquired for 2 seconds with sampling frequency of 24 KHz, which leads to each recording having $d = 48,000$ features. The classes we are interested in identify are: Thalamus Nucleus, Zona incerta, Subthalamic Nucleus, and Substantia Nigra. For each class we have available 13 samples $\mathbf{x}_i$.

Reduced Dataset

For a given sample $\mathbf{x}_i \in \mathbb{R}^{48000}$, the final signal is divided into consecutive windows of 96 samples and for each of these windows we determine 6 statistical features, obtaining a total of $\tilde{d} = 500 \times 6 = 3,000$ features by reduced sample $\tilde{\mathbf{x}}_i \in \mathbb{R}^{\tilde{d}}$ [25]. Therefore, we consider a total

of 52 reduced samples $\tilde{\mathbf{x}}_i \in \mathbb{R}^{3000}$. The 6 statistical features considered are: Curve Length, Threshold, Peaks, Root mean square, Average of nonlinear energy, and Zero crossings. For a given vector $\mathbf{w} \in \mathbb{R}^p$ the definition of the features used in this case is the following:

Curve Length:

$$L = \sum_{i=1}^{p-1} |w_{i+1} - w_i|. \tag{6.1}$$

Threshold:

$$\gamma = \frac{3}{p-1}\sqrt{\sum_{i=1}^{p} \left(w_i - \bar{W}\right)^2}, \text{ where } \bar{W} \text{ is the mean.} \tag{6.2}$$

Peaks:

$$\kappa = \frac{1}{2}\sum_{i=1}^{p-2} \max\left\{0, |\mathrm{sgn}(w_{i+1} - w_i) - \mathrm{sgn}(w_{i+2} - w_{i+1})|\right\}. \tag{6.3}$$

Root mean square:

$$\delta = \sqrt{\frac{\sum_{i=1}^{p} w_i^2}{p}}. \tag{6.4}$$

Average of nonlinear energy:

$$\Psi = \frac{1}{p-2}\sum_{i=2}^{p-1} \left(w_i^2 - w_{i-1}w_{i+1}\right). \tag{6.5}$$

Zero crossings:

$$\beta = \frac{1}{2}\sum_{i=1}^{p} |\mathrm{sgn}(w_{i+1}) - \mathrm{sgn}(w_i)|. \tag{6.6}$$

We performed a classification experiment on this dataset using the ideas of sparse representation discussed throughout this document, and implemented a 10-fold cross-validation test on the dataset described before, for both the raw data and reduced data cases.

Results for the raw data case

In this case we have a total of 52 samples (13 for each of the 4 classes) $\mathbf{x}_i \in \mathbb{R}^{48000}$, and for each classification experiment we solve an optimization problem of the form

$$\begin{aligned} &\min_{\mathbf{c},\mathbf{e}} \lambda\|\mathbf{d}\|_1 + \frac{1}{2}\mathbf{e}^T\mathbf{e} \\ &\text{subject to } B\mathbf{d} - \mathbf{e} = \mathbf{y}, \end{aligned} \tag{6.7}$$

where $B = [A, I] \in \mathbb{R}^{48000\times 48052}$, and $\mathbf{d} = [\mathbf{c},\ \mathbf{r}]^T$,as explained in the previous chapter. The rate of effective classification in this case was 94.23%. The average running time need to find the sparse representation at each fold was 27.65 seconds.

Results for the reduced data case

When we consider the reduced dataset, a total of 52 samples (13 for each of the 4 classes) $\tilde{\mathbf{x}}_i \in \mathbb{R}^{3000}$ are formed using the 6 statistical features described above. The optimization problem that we must solve in this case is of the form

$$\begin{aligned} &\min_{\mathbf{c},\mathbf{e}} \lambda\|\mathbf{d}\|_1 + \frac{1}{2}\mathbf{e}^T\mathbf{e} \\ &\text{subject to } B\mathbf{d} - \mathbf{e} = \mathbf{y}, \end{aligned} \tag{6.8}$$

where $B = [A, I] \in \mathbb{R}^{3000\times 3052}$, with $A \in \mathbb{R}^{3000\times 52}$, and $\mathbf{d} = [\mathbf{c},\ \mathbf{r}]^T$. The percentage of correct classification in this case was 99.00%. In contrast with the raw dataset, the average running time needed to find the sparse representation at each fold, in this case was of 1.54 seconds. The classification rate obtain in this case is very competitive compared with the results reported in [25], where using a set of SVM methods the authors were able to reach a classification rate of 99.4%. In [25] the packages included in the software Weka 3.7 were used.

Face Recognition Dataset

The ideas discussed above can also be applied to different datasets, in order to alleviate computational load and overfitting problems, as well as remove irrelevant and redundant features that can potentially degrade the performance of the classifier.

In this example, we used the dataset from MNIST described in the previous chapter and shown in Figure 5.6. We used the same 6 features used in the previous classification problem of subcortical structures, since we consider each image as a vector after reshaping each sample in the dataset. Even though there are specialized image processing feature extraction tools (see [5] for example), we wanted to test the performance of the algorithm for the specific set of generic statistical features described in equations (6.1)-(6.6).

A total of 6 features were calculated for each vector (reshaped image), with a window-size of 128. Thus, we end up with a training data set where each training sample has 768 components (compared with the 16384 original size of each sample).

Again, we implemented an experiment with a 10-fold cross validation framework and recorded the accuracy rate of the classifier. The results are shown in Table 6.1

Table 6.1: Face recognition problem with feature extraction

Training dataset size	Number of Classes	Accuracy Rate	Average Time for each fold
16384×360	40	97.5%	119.689 secs
768×360	40	95.0%	9.1796 secs

As expected, the running time of the algorithm proposed for classification was reduced significantly, but at the same time the classification rate decreased. This shows how the classification process can be improved/enhanced depending of the application and the use of the correct feature extraction tools specially designed for each case. In our case, even though we are simply using the statistical features described above, we still get a very good classification rate for this specific pattern recognition problem.

6.2 Other Classification Strategy using ℓ_1-optimization

We propose an alternate method for classification problems also based on sparse representation. For a given dataset with N different classes, the approach consists in solving N different *binary classification* problems that are independent one from the other.

Binary Classification

In Binary Classification we aim to classify the elements of a given set into two different groups characterized by a certain property. Binary classification considers a training dataset $\{(\mathbf{x}_i, t_i) : i = 1, ..., n\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $t_i \in \{-1, 1\}$. We describe the two different classes using a linear model of the form

$$[\mathbf{x}^T, 1]\mathbf{w} = t, \tag{6.9}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a sample vector, and $\mathbf{w} \in \mathbb{R}^{d+1}$ characterizes the normal vector of the separating hyperplane [1]. Therefore, for each of the elements in the dataset, we have

$$[\mathbf{x}_i^T, 1]\mathbf{w} = t_i, \quad i = 1, ..., \ n. \tag{6.10}$$

Using matrix algebra notation, the above equation can be written as

$$\mathbf{X}\mathbf{w} = \mathbf{t}, \tag{6.11}$$

where $\mathbf{X} \in \mathbb{R}^{n\times(d+1)}$ is the matrix whose i-th row is given by $[\mathbf{x}_i^T, 1]$, and the vector $\mathbf{t} \in \mathbb{R}^n$ contains all the different labels $t_i \in \{-1, 1\}$ for each sample.

A sparse classifier aims to differentiate between classes, identifying a small number of relevant features. For a sparse separating hyperplane, this means that the weight vector $\mathbf{w}$ in (6.11) has few nonzero elements [1], and therefore the characterization of the separating hyperplane has a short form.

If we assume that $n < d + 1$, the system $\mathbf{Xw} = \mathbf{t}$, is underdetermined and we look for the solution of the ℓ_1-minimization problem

$$\begin{aligned} &\min \|\mathbf{w}\|_1 \\ &\text{s.t} \quad \mathbf{Xw} = \mathbf{t}. \end{aligned} \tag{6.12}$$

The sparse solution $\mathbf{w}^*$ of (6.12) is used to define the label of any other input sample $\mathbf{x}$ by computing

$$t = \operatorname{sgn}\left([\mathbf{x}^T, 1]\mathbf{w}^*\right), \tag{6.13}$$

where sgn denotes the sign function. This means that the sample $\mathbf{x}$ has the label 1 if the sign is positive, and $\mathbf{x}$ belongs to the class with label -1 otherwise.

For the binary classification problem we tested one of the datasets used in the previous chapter from the GEMS library. We focused on the binary dataset `Prostate_Tumor` that has $n = 102$ samples $\mathbf{x}_i \in \mathbb{R}^d$ where $d = 10,509$ is the number of features of each sample. We again, performed a 10-fold cross validation experimentation, following the methodology described above for the binary classification case. The total CPU time used in the experimentation was 231.5 seconds. The results are presented in the following table

Table 6.2: Binary Classification of the GEMS dataset `Prostate_Tumor`

Samples n	Features d	Classification Rate	Average time by fold (seconds)
102	10,509	90.27%	22.9

Multicategory Classification

Now we describe a multicategory classification technique using the ideas behind binary classification, based on the one-versus-rest (OVR) approach for support vector machines and linear classifiers [1]. The idea consists in solving a series of binary classification subproblems, in order to obtain the correct label t for a given test sample $\mathbf{x}$.

Consider the multicategory dataset $\{(\mathbf{x}_i, t_i) : i = 1, ..., n\},\ \mathbf{x}_i \in \mathbb{R}^d,\ t_i \in \{1, 2, ..., N\}$, with $\mathbf{x}_i \in \mathbb{R}^d$ and N as the number of categories. For each category k, we determine a binary classifier, separating category k from the rest of the categories. Therefore we define N linear models of the form:

$$\mathbf{X}\mathbf{w}_k = \mathbf{t}_k, \quad k = 1, ...,\ N, \tag{6.14}$$

where the labels vector $\mathbf{t}_k$ is constructed by changing to 1 all of the labels of samples belonging to class k, and setting the rest to -1. Therefore, the matrix $\mathbf{X} \in \mathbb{R}^{n\times(d+1)}$ has $[\mathbf{x}_i^T, 1]^T$, as the i-th row as the i-th row. In the same way as in the binary case described previously, we can look for the solution of the underdetermined system of equations (6.14) by solving the N constrained minimization problems:

$$\begin{aligned} &\min \|\mathbf{w}_k\|_1 \\ &\text{s.t}\ \ \mathbf{X}\mathbf{w}_k = \mathbf{t}_k, \quad k = 1, ...,\ N. \end{aligned} \tag{6.15}$$

Once we have computed the N different solution vectors to problem (6.15), we determine the label of any given test sample $\mathbf{y}$ by computing

$$\begin{aligned} [\mathbf{y}^T, 1]\mathbf{w}_k &= \max\left\{[\mathbf{y}^T, 1]\mathbf{w}_1,\ [\mathbf{y}^T, 1]\mathbf{w}_2, ...,\ [\mathbf{y}^T, 1]\mathbf{w}_N\right\} \\ \hat{\mathbf{t}} &= k. \end{aligned} \tag{6.16}$$

If the product $[\mathbf{y}^T, 1]\mathbf{w}_k$ is positive for several classes, a larger value of $[\mathbf{y}^T, 1]\mathbf{w}_k$ means that $\mathbf{y}$ lies "further" into class k than into any other class j. If the product $[\mathbf{y}^T, 1]\mathbf{w}_k$ is negative for all k, the maximum means that we classify $\mathbf{y}$ according to the class represented by the closest hyperplane. In this manner, an original multicategory classification problem is solved by a series of binary classification subproblems.

Notice that the subproblems of the form (6.15) are independent, and therefore a parallel implementation of this approach can be studied. Different processors can look for the solution

of problem (6.15) in an independent manner, since all we will need for the classification process are the different N solutions $\mathbf{w}_k$.

We performed a numerical experimentation of the datasets in Table 5.1 using the multicategory classification framework described above. Cross validation results for the different datasets showing the obtained classification rate with this methodology are reported in Table 6.3. We used the same partitions for the training sets and testing sets, that were created for each dataset in the previous 10-fold cross validation experiment. The average CPU running time (in seconds) by fold is also reported in the table.

Table 6.3: Multicategory classification of GEMS datasets

Dataset	Samples n	Features d	Classes N	Class. Rate	Time
9_Tumors	60	5726	9	62.54%	1.61
11_Tumors	174	12533	11	92.15%	11.19
Prostate_Tumor	102	10509	2	90.18%	16.85
Lung_Cancer	203	12600	5	93.56%	22.46
SRBCT	83	2308	4	100%	1.13
Brain_Tumor	90	5920	5	91.20%	5.50

Discussion

Assume that we have a test sample $\mathbf{y} \in \mathbb{R}^d$, and that using the multicategory framework we just described above, we obtain the label $\mathbf{t}$ from equation (6.16). This means that for a given $s \in \{1, ..., N\}$ the following relations hold:

$$[\mathbf{y}^T, 1]\mathbf{w}_s = \max_{k=1,...,N} \left\{ [\mathbf{y}^T, 1]\mathbf{w}_k \right\}, \quad \text{and} \tag{6.17}$$

$$\mathbf{t} = s. \tag{6.18}$$

As discussed before, and as consequence of the formulation in (6.15), in this framework we look for the sparsest vector $\mathbf{w}$ that characterizes the separating hyperplane. This means that

$$\begin{aligned}[\mathbf{y}^T, 1]\mathbf{w}_s &= y_1 (\mathbf{w}_s)_1 + y_2 (\mathbf{w}_s)_2 + y_3 (\mathbf{w}_s)_3 + ... + y_d (\mathbf{w}_s)_d + (\mathbf{w}_s)_{d+1} \\ &= y_{k_1} (\mathbf{w}_s)_{k_1} + y_{k_2} (\mathbf{w}_s)_{k_2} + ... + y_{k_p} (\mathbf{w}_s)_{k_p}, \end{aligned} \tag{6.19}$$

where the set $\Gamma = \{k_1, k_2, ..., \ k_p\}$ indicates the nonzero entries of the vector $\mathbf{w}_s$. Therefore, the vector $\mathbf{w}_s$ that characterizes the separating hyperplane, can also be used to *select* the most important features of the sample $\mathbf{y}$, namely

$$\tilde{\mathbf{y}} = \left[y_{k_1}, y_{k_2}, ..., y_{k_p}\right]^T \in \mathbb{R}^p, \quad p < d. \tag{6.20}$$

This observation motivates us to present a reduced-order classification method with feature selection based on sparse representation.

6.3 Reduced-Order Classification with Feature Selection

In this section we present a methodology for solving classification problems using feature selection via sparse representation. We assume the number d of features of each sample is very large compared with the number n of samples we have available , that is $n \ll d$. Performing an efficient feature selection will allow us to reduce the dimensionality of the problem, focusing only on those most representative features of each sample. Once we are able to reduce the dimensionality of the training set, the classification process is executed in just a fraction of the time needed originally for the complete set of features.

The methodology has three main components both involving the use of the ℓ_1 norm. We call our methodology reduced-order $\ell_1 - \ell_1$ classification using sparse representation with feature selection. The main steps for this reduced-order technique are described as follows:

1. *Offline:* First, given a training dataset $\{(\mathbf{x}_i, t_i) : i = 1, ..., n\},\ \mathbf{x}_i \in \mathbb{R}^d,\ t_i \in \{1, 2, ..., N\}$, with $\mathbf{x}_i \in \mathbb{R}^d$ and N the number of different classes, we perform a multicategory classification following the framework explained in Section 6.2. That is, for N different classes, we obtain the N sparse vectors $\{\mathbf{w}_k\}_{k=1,..,N}$ that characterize the sparse separating hyperplanes, solving the ℓ_1-optimization subproblems of the form (6.15).

2. For any given test sample $\mathbf{y} \in \mathbb{R}^d$, we use the vectors $\{\mathbf{w}_k\}_{k=1,..,N}$ to perform *feature selection*. First, we find the vector $\mathbf{w}_s$ that satisfies

$$[\mathbf{y}^T, 1]\mathbf{w}_s = \max_{k=1,...,N} \left\{ [\mathbf{y}^T, 1]\mathbf{w}_k \right\}.$$

   That is, we identify the membership of $\mathbf{y}$ to the sparse separating hyperplane characterized by $\mathbf{w}_s$. Then, we find the nonzero components of the vector $\mathbf{w}_s$. Let those nonzero components be located in the positions $\Gamma = \{k_1, k_2, ...,\ k_p\}$ of $\mathbf{w}_s$, that is, the set $\Gamma = \{k_1, k_2, ...,\ k_p\}$ is the support of the sparse vector $\mathbf{w}_s$. We reduce the dimension of the sample $\mathbf{y} \in \mathbb{R}^d$ by considering its entries located at the positions described by Γ. Thus, our new test sample $\tilde{\mathbf{y}} \in \mathbb{R}^p$ is a vector in a smaller dimensional space ($p < d$), and captures the most significant features of the original test sample. Again using the set of indices listed in Γ, the dimension of each training sample $\mathbf{x}_i$ is reduced in the same fashion, by considering only the entries located in positions $\{k_1, k_2, ...,\ k_p\}$. Therefore, we now have a training dataset where feature selection has been used to capture the most salient features of each sample. The new training dataset of reduced dimension is denoted by $\{(\tilde{\mathbf{x}}_i, \tilde{t}_i) : i = 1, ..., n\},\ \tilde{\mathbf{x}}_i \in \mathbb{R}^p,\ t_i \in \{1, 2, ..., N\}$.

3. *Online:* Using the reduced-order sample and training set, we now perform classification as described in Chapter 4 and in Algorithm 3. That is, with $\tilde{\mathbf{y}} \in \mathbb{R}^p$ as test sample and the training dataset $\{(\tilde{\mathbf{x}}_i, \tilde{t}_i) : i = 1, ..., n\},\ \tilde{\mathbf{x}}_i \in \mathbb{R}^p,\ t_i \in \{1, 2, ..., N\}$, we aim to *represent* the $\tilde{\mathbf{y}}$ as a *sparse* linear combination of the elements in our training dataset.

Specifically, we solve an ℓ_1 optimization problem of the form:

$$\min_{\tilde{\mathbf{d}},\tilde{e}} \lambda\|\tilde{\mathbf{d}}\|_1 + \frac{1}{2}\tilde{\mathbf{e}}^T\tilde{\mathbf{e}} \qquad (6.21)$$
$$\text{subject to } \tilde{B}\tilde{\mathbf{d}} - \tilde{\mathbf{e}} = \tilde{\mathbf{y}},$$

where $\tilde{B}$ is the augmented matrix $\tilde{B} = \left[\tilde{A}\ I\right]$, and $\tilde{\mathbf{d}} = [\tilde{\mathbf{c}},\ \tilde{\mathbf{r}}]^T$. The matrix $\tilde{A}$ is just the matrix whose columns are the reduced training sample $\{\tilde{\mathbf{x}}_i\ :\ i = 1, ..., n\}$. The label $\mathbf{t}$ associated to sample $\mathbf{y}$ is then calculated following the same rule described in (4.15), that is

$$\text{If } \|\tilde{\mathbf{y}} - \tilde{A}\tilde{\mathbf{c}}_k\|_2 = \min\left\{\|\tilde{\mathbf{y}} - \tilde{A}\tilde{\mathbf{c}}_1\|_2,\ \|\tilde{\mathbf{y}} - \tilde{A}\tilde{\mathbf{c}}_2\|_2, ...,\ \|\tilde{\mathbf{y}} - \tilde{A}\tilde{\mathbf{c}}_N\|_2\right\},$$
$$\text{then } \mathbf{t} = k,$$

with $\tilde{\mathbf{c}}_k$ obtained by keeping only those entries in $\tilde{\mathbf{c}}$ associated with category k and assigning zeros to all the other entries. Problem (6.21) can now be solved using Algorithm 1 in just a fraction of the time needed to solve the classification problem with the complete set of features.

Numerical Experimentation

We tested the proposed reduced-order classification method in the datasets available at the Gene Expression Model Selection (GEMS) library for cancer classification using gene expression data. In this case, the feature selection that our methodology uses, might help to identify those biological markers associated with tumorigenesis and progression. We used the same datasets described in Table 5.1, and compared the running time and classification rate of the reduced-order version with the classification results when using the complete set of genes. The results we have for the different strategies are summarized in Table 6.4. We report the classification rate after a 10-fold cross validation experiment, and the average time needed to solve the classification problem at each fold. In Table 6.4, SR stands for Sparse

Representation, which is the technique described in Chapter 4; and SVM refers to Support Vector Machine, which is the methodology built-in in the software GEMS for classification. We also report the results for the separating hyperplanes idea presented in Section 6.2, and the results for the redcued-order $\ell_1 - \ell_1$ technique we just presented above.

Table 6.4: Reduced-order classification via sparse representation with feature selection

		9 Tumors	11 Tumors	Prostate Tumor	Lung Cancer	SRBCT	Brain Tumor
Features	d	5726	12533	10509	12600	2308	5920
Samples	n	60	174	102	203	83	90
Classes	N	9	11	2	5	4	5
Method							
SR	Time	**0.61**	**71.23**	**3.28**	**14.23**	**4.19**	**13.80**
	Rate	**66.67%**	**97.55%**	**94.12%**	**95.07%**	**100%**	**91.11%**
SVM	Time	34.35	539.79	99.79	365.21	19.29	50.06
	Rate	67.01%	94.99%	93.27%	96.05%	100%	90.00%
Sequence of binary	Time	1.61	11.19	16.85	22.46	1.13	5.50
	Rate	62.54%	92.15%	90.18%	93.56%	98.75%	91.20%
Reduced Order $\ell_1 - \ell_1$	p	95	307	234	345	180	156
	Time Offline	7.72	36.73	27.85	136.46	0.83	4.62
	Time Online	**0.24**	**0.44**	**0.97**	**2.67**	**0.15**	**0.20**
	Rate	**66.66%**	**93.64%**	**92.00%**	**93.56%**	**98.75%**	**90.13%**

We also tested the reduced-order classification methodology described in this section, with the dataset used for the face recognition example. In this case the average time needed

Table 6.5: Reduced-order classification face recognition example

Method	Number of Features	Accuracy Rate	Average Time for each fold
SR ℓ_1	16384	97.50%	119.689 seconds
$\ell_1 - \ell_1$	1916	96.66%	11.83 seconds

in the *offline* stage, that is the time needed to compute the sparse separating hyperplanes was 17.81 seconds. The final number of features used after the reduction was in average $p = 1916$, compared with the number $d = 16384$ of original features in each sample. The classification rate is still high when using the reduced-order classification technique proposed in this section. Results are reported in Table 6.5.

Discussion

The Reduced-Order classification using sparse representation methodology performs very well in this set of problems from the GEMS library. The number of genes p that were used using the proposed feature selection technique, is significantly small compared with the complete set of features d. This in turn translates into faster CPU running times. The classification rates results for the reduced-order $\ell_1 - \ell_1$ technique are very encouraging. Even though we used just a fraction of the genes, we still get high and acceptable classification rates, close to the results when working with the complete dataset in the Sparse Representation (SR) method. Similar encouraging results were obtained when working with the face recognition example, with a significant reduction in the number of features used and with high classification rates. We can conclude that the proposed feature selection technique using ℓ_1 optimization for sparse separating hyperplanes, indeed captures the most salient features of the samples used in the classification process.

Chapter 7

Conclusions and Future Work

The results reported for the numerical experimentation described in this work show the effectiveness of the sparse representation technique proposed for solving classification problems. We have also mentioned two of the advantages of using sparse representation via ℓ_1-optimization for classification: the first one being that no model selection dependence is involved since the selective nature of the sparse representation technique highlights the membership to a given particular class; and the second one recalling that the lack of robustness with respect to outliers and missing data can be overcome when using the ℓ_1 norm. Our results also show that the performance of the Sparse Representation (SR) approach can be as accurate and often higher than other classification techniques such as Support Vector Machines (SVM). We have shown the effectiveness of the proposed technique in different datasets from several applications in science and engineering. Also, a dimensionality reduction stage has been studied, showing that the SR methodology complements well with feature extraction techniques.

We have presented a novel ℓ_1 optimization algorithm based on smooth convex relaxation, which has been proven to successfully find sparse solutions to linear systems of equations. The algorithm performs very well when compared with others state-of-the-art methods for ℓ_1 optimization, as the comparison for compressed sensing and sparse representation problems suggests. We also showed convergence results for the proposed algorithm and described its MATLAB implementation.

Future Work

We are interested in formulating a classification algorithm under the framework of Reduced Order Modeling (ROM). The goal is to reduce the computational load needed to solve the classification problem by finding the optimal subspace where the solution actually resides. This will translate into decreasing the dimension of the problem, without compromising the classification rate of the method, and finding the correct labels in a fraction of the computational time needed in the full-order version (the original problem dimension). Recall the scheme shown in Figure 5.9, where the tested sample $\mathbf{y}$ is our *input*, and the corresponding label $\mathbf{t}$ is the *output*. The central box of that scheme (system) is the solution of the ℓ_1-optimizaton problem that provides the sparse representation needed in our methodology. A solution of such optimization problem is found, by looking for the vector that satisfies the optimality conditions (KKT conditions), given that we work with a convex optimization problem. We can express the KKT conditions as a *root-finding* problem (or as a fixed-point problem for our formulation as shown in Chapter 3). The key observation in our framework is the following: since we perform classification using the sparse representation obtained from an ℓ_1-optimization problem, then it is clear that the solution of our problem resides on a lower dimensional manifold, characterized by its *nonzero elements*. Encouraged by the work of C. Farhat and K. Carlberg and the results shown in [13] for reduced order modeling of complex dynamical engineering systems, we can formulate the reduced order modeling sparse classification methodology as follows. First, in order to capture the search subspace of interest, a set of solutions are computed *offline* for different inputs, providing a set of sparse representations for different possible scenarios. These solutions are called *snapshots*, and are used for learning from different inputs. Next, the collected snapshots are compressed into a reduced-order basis, using for example a Proper Orthogonal Decomposition (POD) [13]. Our model is then projected onto a subspace spanned by the resulting reduced-order basis. Finally, for any new sample we performed the classification process *online*, by searching for approximate solutions of the KKT conditions in the lower dimensional subspace spanned by the reduced-order basis. We plan to study these ideas for reduced order modeling (ROM)

classification in the future, and decide for which kind of classification problems the ROM sparse representation technique is accurate and can be efficiently implemented.

The classification method and ideas presented in this document can be applied in different areas of science and engineering. We plan to work with multidisciplinary research groups in order to motivate this alternative for pattern recognition problems arising in biological sciences and geological sciences problems. High performance computing can also be involved in the design and implementation of this classification methodology, and the migration of codes from MATLAB to another programming language is fairly straight-forward. Some experienced in this area was already gained in the work presented in [32].

Block sparsity can also be studied to enhance the classification process. Since in our classification formulation we can arrange the training samples in our matrix in our desired manner, we could take advantage of the block structure of the available samples, and use this knowledge as prior information in our formulation. This group/block sparsity framework can still be posed as an ℓ_1 optimization problem. The problem of clustering samples could also be treated in a similar framework and the study of effective formulations for this task is also a topic of future research.

Chapter 8

Appendix

8.1 Scalability Comparison Noiseless Case

We report the scalability comparison for the proposed ISCR algorithm with respect to the other four ℓ_1-optimization solvers `l1_ls` (Boyd et al.), `FPC_AS` (Zhang et al.), `NESTA` (Becker et al.), and `GPSR` (Wright et al.), in the same framework studied in Section 3.6.2. Table 8.1 and Table 8.2 show the results for this specific numerical experimentation.

8.2 Support Vector Machines (SVM)

In this section we present the mathematical formulation of the SVM method for classification problems.

The basic idea in this method is to find a hyperplane which separates the $d$dimensional data perfectly into its two classes. SVM methodologies also include the notion of *kernel induced feature space*, for cases when the data needs to be cast into a higher dimensional space where is separable. Typically, casting into such space can produce computational problems and overfitting, but some techniques have also been proposed to alleviate such problems on *unseen* data [47]. As mentioned before, the original SVM algorithm was invented by Vladimir Vapnik colleagues and the current standard implementation was proposed by Corinna Cortes and Vladimir Vapnik [16].

Suppose we are given l training samples $\{\mathbf{x}_i,\, y_i\}\,,\, i = 1, ..., l$, where each sample belongs to $\mathbb{R}^d$ and a class label with one of two values y_i is given for each training sample. For simplicity assume, $y_i \in \{-1, 1\}$. Notice that all hyperplanes $\mathcal{H}$ in $\mathbb{R}^d$ are parameterized by

a vector w and a constant b in the following way:

$$w^T x + b = 0, \tag{8.1}$$

where w is a vector orthogonal to the hyperplane $\mathcal{H}$. Given such hyperplane, that we denote by $\langle w, b \rangle$, that separates the data, the function

$$f(x) = \text{sgn}(w^T x + b), \tag{8.2}$$

correctly classifies the data. Notice though that a given hyperplane represented by $\langle w, b \rangle$, is equally expressed by all pairs $\{\lambda w, \lambda b\}$ for any $\lambda > 0$. Therefore, let us define the canonical hyperplane to be the one that separates the data from the hyperplane by a distance of at least 1. This means, the following relations must be satisfied:

$$\mathbf{x}_i^T w + b \geq 1 \quad \text{when } y_i = +1, \tag{8.3}$$

$$\mathbf{x}_i^T w + b \geq -1 \quad \text{when } y_i = -1. \tag{8.4}$$

These two relations can be compacted in the inequality

$$y_i \left(\mathbf{x}_i^T w + b \right) \geq 1, \quad \forall i = 1, .. \, l. \tag{8.5}$$

To obtain the geometric distance from the hyperplane $\mathcal{H}$ to a data point, we normalize by the magnitude of w, and get

$$d\left(\mathcal{H}, \mathbf{x}_i\right) = \frac{y_i \left(\mathbf{x}_i^T w + b \right)}{\|w\|} \geq \frac{1}{\|w\|}. \tag{8.6}$$

In the process of classifying objects, intuitively we look for the hyperplane $\mathcal{H}$ that maximizes the geometric distance to the closest data points. From the relations shown above, w can see that this is accomplished by minimizing $\|w\|$, subject to the distance constraints.

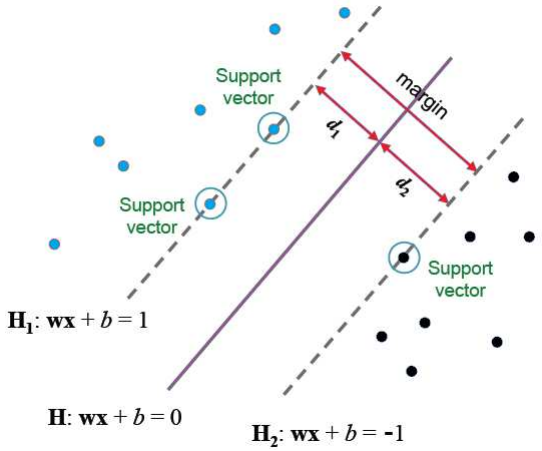


Figure 8.1: Hyperplanes separation SVM idea

We are now in position to formulate the optimization problem, whose primal formulation is given by

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 \\ &\text{subject to } \; y_i \left(\mathbf{x}_i^T w + b\right) \geq 1, \quad \forall i = 1, ..., l. \end{aligned} \tag{8.7}$$

Problem (8.7) can be solved using techniques from constrained optimization. The Lagrangian function associated to this problem is given by

$$\mathcal{L}(w, b) = \frac{1}{2} w^T w + \sum_{i=1}^{l} \alpha_i \left(1 - y_i \left(\mathbf{x}_i^T w + b\right)\right),$$

and setting the gradient of $\mathcal{L}$ with respect to w an b to zero, we have the equations

$$w = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i, \tag{8.8}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{8.9}$$

If we substitute $w = \sum \alpha_i y_i \mathbf{x}_i$ in the expression for $\mathcal{L}$ we get

$$\mathcal{L}(w, b) = -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + \sum_{i=1}^{l} \alpha_i, \tag{8.10}$$

which means we have expressed the Lagrangian function in terms of α_i only. If we know w we know all α_i, and if we know α_i we know w for all $i = 1, ..., l$.

Thus, we can formulate the dual problem as

$$\begin{aligned} \max\ W(\alpha) &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to}\ \ \alpha_i &\geq 0,\ \sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \tag{8.11}$$

Notice that problem (8.11) is a quadratic programming (QP) problem, and the global maximum of α_i can always be found using one of the different QP solvers developed in the latest years (sequential quadratic programming, sequential minimal optimization, etc) [8].

Some of the characteristics of the solution of (8.11) are that many of the α_i are zero, which means that w is a linear combination of a small number of data points. This can be thought as data compression. Those data points $\mathbf{x}_i$ where the corresponding α_i are called *support vectors* (SV).

Once a solution has been found, any new data point $\mathbf{u}$ is classified using the following two steps:

Compute

$$w^T \mathbf{u} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j}^T \mathbf{u}) + b, \tag{8.12}$$

Classify $\mathbf{u}$ as class 1 if the sum in (8.12) is positive, and class 2 otherwise.

When a dataset is not linearly separable, the SVM method can still identify different classes.

In some cases, it might be easier to separate the data using polynomial curves. However finding the optimal curve to fit the data is difficult, and this decision is part of the tuning stage of the SVM methodology. When the data cannot be separated by linear functions, the problem is transformed into one of finding a simple hyperplane, defining a mapping $\mathbf{z} = \phi(\mathbf{x})$ that transforms the d dimensional input vector $\mathbf{x}$ into a higher d' dimensional vector $\mathbf{z}$. The hope is that the new training data $\{\phi(\mathbf{x}_i), y_i\}$ is separable by a hyperplane. A similar optimization problem is again formulated where $\phi(\mathbf{x})$ plays the role of $\mathbf{x}$ in (8.11). Notice that we would have the following product in the objective function of the dual formulation of type (8.11)

$$K(\mathbf{x}_i, \mathbf{x}_j) := \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \tag{8.13}$$

where $K(\cdot, \cdot)$ is called a kernel. If we knew the formula for such such kernels expressing the dot product between two given samples, we would not need to deal with the mapping ϕ directly in the optimization process. Several kernels have been studied for SVM methodologies. More information about this can be found in [47].

8.3 MATLAB codes

In the following we show the MATLAB implementation for the sparse signal recovery algorithm ISCR proposed in this work, and described in Algorithm 1. We also show the MATLAB code for the Conjugate Gradiente (CG) method used in our methodology, and described in Algorithm 2.

Listing 8.1: ISCR algorithm MATLAB implementation.

```
% This solver finds the solution of the problem
%   min  lambda*||x||_1 + 0.5||Ax-b||^2
%   s.t.  Ax - e = b, where  A is mxn with m<n
% The resulting optimality conditions yield e = -y (Lagrange multiplier)
```

```
% Parameters:
%   mA : matrix A         mAt: transpose of matrix A
%   b  : observed signal  n  : length of sparse signal x
%   T  : estimated sparsity of the signal x
%
% Outputs:
%   x  : recovered signal     e: error vector
%   k  : total iterates       total_cg: Conjugate gradient iterations

function [x, e, k, total_cg ] = ISCR_rsa(mA, mAt,b,n,T)
% ------------------------- Initialization -------------------------------
% Make matrix A and its tranpose a handle function
if ~isa(mA,  function_handle )
    A  = @(x) mA*x;
    At = @(x) mAt*x;
else
    A = mA;
    At= mAt;
end            % from this point down, A and At are always function handles.
% Dimensions of right-hand side vector
m = length(b);
% Initial Point:
x   = At(b);
% Parameters
mu     = 1;               % initial regularization parameter
if exist( T , var )== 0  % check if variable T has been defined
    T = ceil(n/4);
end
lambda = 0.02/sqrt(T);   % noise penalty parameter lambda
% Counters:
k            = 0;
total_cg     = 0;
% Tolerance
k_max          = 70;      % maximum number of iterations
tol_x          = 6e-4;    % tolerance for two consecutive approximations
tol_Ate        = 1e-5;    % tolerance for Aty approximations
% First residual
Axb = A(x)-b;
% First approximation to e (first residual)(since y = e = Ax-b)
e   = Axb;
% First approximation to A *y
Ate = At(e);
```

```
% Right-hand side of the mxm system solved at each iteration
RHS   = -lambda*b;
Rei_e = e;
% Beginning of iterative process:
while(k < k_max)
    % Store previous approximation x for comparison
    x_prev = x;
    % Diagonal matrix
    invD   = (x.^2 + mu).^.5;
    % Approximation of A*D*A *y matrix
    DAte   = invD.*Ate;
    ADAte  = A(DAte);
    % CG method solves the problem (ADA +lambda*I)y = RHS
    [At_de, cg_iters, Rei_e] = CG_rsa(A, At, invD, RHS,m, lambda, 25, Rei_e, Ate, ADAte);
    total_cg = total_cg + cg_iters;

    % Stoping criteria
    if norm(Ate-At_de) < tol_Ate
        fprintf( Ate and Ate+ relative close after %d iterations \n ,k );
        break
    end
    Ate = At_de;                    % Approximation of A *e from CG method
    x   = -(1/lambda)*(invD.*(Ate));  % x = -(lambda)^(-1)*invD.*Aty;
                                    % x is the solution to the subproblem
    % Error criteria for consecutive approximations
    error_x = norm(x_prev-x)/norm(x_prev);
    % Stopping criteria
    if error_x < tol_x
        fprintf( Consecutive approximations sufficiently close \n  );
        break
    end
    % Decrease parameter mu
    mu  = mu/5;
    % Increase iteration counter
    k = k+1;
end    % end loop

end    % end-function
```

Listing 8.2: Conjugate Gradient MATLAB implementation.

```
function [At_y, i , Rei_y] = CG_rsa(A, At, D, b,m, lambda, max_iter, y, Aty, ADAty)
% Solve the system (ADA +lambda)y = b, using conjugate gradient
% D is a diagonal matrix, received as a vector with diagonal elements
% Tolerance of CG method
tol    = 5e-4;
% Initialization
Rei_y = zeros(m,1);
At_y  = Aty;
r     = b - (ADAty+lambda*y);
d     = r;
beta_n = r *r;
% err = norm(r);
for i=1:max_iter
    q_1    = At(d);
    q_2    = D.*q_1;
    q_3    = A(q_2) + lambda*d;
    alpha_d = d *q_3 ;
    alpha  = (beta_n)/(alpha_d);
    At_y   = At_y + alpha*q_1;
    % New approximation to solution
    Rei_y = Rei_y + alpha*d;
    % Residual
    r     = r - alpha*q_3;
    err   = norm(r);
    % Stopping criteria
    if (err < tol)
        break
    end
    % Updates
    beta_d = beta_n;
    beta_n = r *r;
    beta   = (beta_n)/(beta_d);
    d      = r + beta*d;
end    % end-for-loop

end    % end-function
```

Table 8.1: Scalability comparison for $n = 2^{10}$ to $n = 2^{15}$ (noiseless case)

Solver	m	n	k	$\frac{\|x_k - x^*\|}{\|x^*\|}$	Time (s)
l1ls	512	1024	51	0.0036	0.4836
NESTA	512	1024	51	0.0007	0.4368
GPSR	512	1024	51	0.0031	**0.078**
FPC	512	1024	51	**0**	0.1404
ISCR	512	1024	51	**0**	0.2028
l1ls	1024	4096	160	0.0068	2.0436
NESTA	1024	4096	160	0.0016	1.7472
GPSR	1024	4096	160	0.0082	0.8112
FPC	1024	4096	160	**0**	**0.4056**
ISCR	1024	4096	160	**0**	0.9204
l1ls	2048	4096	205	0.0027	0.8268
NESTA	2048	4096	205	0.0008	1.0764
GPSR	2048	4096	205	0.0033	0.2028
FPC	2048	4096	205	**0**	**0.1404**
ISCR	2048	4096	205	**0**	0.7488
l1ls	2048	8192	320	0.0083	2.9016
NESTA	2048	8192	320	0.0013	2.3868
GPSR	2048	8192	320	0.0077	**0.6864**
FPC	2048	8192	320	**0**	0.8112
ISCR	2048	8192	320	**0**	2.34
l1ls	4096	16384	640	0.0087	4.1184
NESTA	4096	16384	640	0.0013	5.382
GPSR	4096	16384	640	0.0078	1.56
FPC	4096	16384	640	**0**	**1.2324**
ISCR	4096	16384	640	**0**	3.3072
l1ls	8192	32768	1280	0.0084	7.02
NESTA	8192	32768	1280	0.0014	8.2837
GPSR	8192	32768	1280	0.0079	2.6832
FPC	8192	32768	1280	**0**	**2.34**
ISCR	8192	32768	1280	**0**	5.3352
l1ls	16384	32768	1638	0.0034	5.1792
NESTA	16384	32768	1638	0.0008	5.4288
GPSR	16384	32768	1638	0.0033	1.1076
FPC	16384	32768	1638	**0**	**0.8736**
ISCR	16384	32768	1638	**0**	3.2916

Table 8.2: Scalability comparison for $n = 2^{16}$ to $n = 2^{21}$ (noiseless case)

Solver	m	n	k	$\frac{\|x_k - x^*\|}{\|x^*\|}$	Time (s)
l1ls	16384	65536	2560	0.012	24.0242
NESTA	16384	65536	2560	0.0013	15.8965
GPSR	16384	65536	2560	0.0079	5.382
FPC	16384	65536	2560	**0**	**4.4928**
ISCR	16384	65536	2560	**0**	11.6689
l1ls	32768	131072	5120	0.0063	55.1308
NESTA	32768	131072	5120	0.0013	42.3855
GPSR	32768	131072	5120	0.008	15.0541
FPC	32768	131072	5120	**0**	**12.6049**
ISCR	32768	131072	5120	**0**	30.701
l1ls	65536	262144	10240	0.0095	152.6626
NESTA	65536	262144	10240	0.0013	84.0377
GPSR	65536	262144	10240	0.0079	27.3314
FPC	65536	262144	10240	**0**	**26.9102**
ISCR	65536	262144	10240	**0**	68.422
l1ls	131072	524288	20480	0.0066	435.43
NESTA	131072	524288	20480	0.0013	187.2948
GPSR	131072	524288	20480	0.0079	60.778
FPC	131072	524288	20480	**0**	**54.7876**
ISCR	131072	524288	20480	**0**	148.9342
l1ls	262144	1048576	40960	0.0055	1013.3201
NESTA	262144	1048576	40960	0.0013	456.5057
GPSR	262144	1048576	40960	0.0079	136.0329
FPC	262144	1048576	40960	**0**	**108.5611**
ISCR	262144	1048576	40960	**0**	292.7047
l1ls	524288	1048576	52429	0.0021	748.7112
NESTA	524288	1048576	52429	0.0008	284.7174
GPSR	524288	1048576	52429	0.0034	**58.9528**
FPC	524288	1048576	52429	**0**	71.1053
ISCR	524288	1048576	52429	**0**	193.6128
l1ls	524288	2097152	81920	0.0086	2413.0859
NESTA	524288	2097152	81920	0.0013	939.36
GPSR	524288	2097152	81920	0.0079	270.1001
FPC	524288	2097152	81920	**0**	**246.7936**
ISCR	524288	2097152	81920	**0**	647.5446

References

[1] S. Abe, "Support Vector Machines for Pattern Classification", *Springer Verlag London*, Advances in Pattern Recognition, 2005

[2] M. Argáez "Solving Overdetermined Systems in ℓ_q Quasi-Norms", *Special SCAN '08 Issue of Reliable Computing* Vol. 15, Issue 1, pp. 13-25, 2011.

[3] M. Argáez, C. Ramirez, R. Sanchez, "An ℓ_1-algorithm for underdetermined systems and applications", *IEEE proceedings of the North American Fuzzy Information Processing Society*, pp.1 - 6. 2011.

[4] M. Argáez, R. Sanchez, C. Ramirez "Face Recognition from Incomplete Measurements via ℓ_1-minimization", *American Journal of Computational Mathematics AJCM*, Vol. 2, No. 4, pp. 287-294. 2012.

[5] E. Bagherian, R. Rahmat "Facial feature extraction for face recognition: a review", *IEEE Proceedings of the International Symposium on Information Technology*, Vol.2, pp. 1-9. 2008

[6] R. Baraniuk "Compressive Sensing", *IEEE Signal Processing Magazine*, Vol 24, No. 4, pp. 118-121, 2007.

[7] S. Becker, J. Bobin, and E. J. Candès. "NESTA: a fast and accurate first-order method for sparse recovery", *SIAM J. on Imaging Sciences*, Vol. 4, No. 1, pp. 1-39. 2010.

[8] S. Boyd and L. Vandenberghe, Convex Optimization, *Cambridge University Press*, Cambridge, U.K., 2004.

[9] E. Candès, "Compressive sampling", *Proceedings of the International Congress of Mathematicians*, Madrid, Spain, 2006.

[10] E. Candès, "The restricted isometry property and its implications for compressed sensing", *Compte Rendus de l'Academie des Sciences*, Paris, Series I, 346, pp. 589-592, 2008.

[11] E. Candès, T. Tao, "Decoding by linear programming", *IEEE Trans. Inform. Theory 51* (12) (December 2005) 42034215.

[12] E. Candès, J. Romberg and T. Tao, "Robust Uncertainty Principles: Exact Signal Reconstruction from Highly Incomplete Frequency Information", *IEEE Transactions on Information Theory*, Vol. 52, pp. 489-509, 2006.

[13] K. Carlberg, C. Farhat. "A low-cost, goal-oriented compact proper orthogonal decomposition basis for model reduction of static systems", *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 3, p. 381402. 2011.

[14] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization", *IEEE Signal Processing Letters*, Vol. 14, pp. 707-710, 2007.

[15] A. Cohen, W. Dahmen and R. DeVore, "Compressed sensing and best k-term approximation", *J. Amer. Math. Soc. 22* pp. 211231, 2009.

[16] C. Cortes and V. Vapnik, "Support Vector Networks ", *Machine Learning*, Vol. 20, No. 3, pp. 273-297, 1995.

[17] S. Chen, D. Donoho, M. Saunders, "Atomic Decomposition by Basis Pursuit", *SIAM Review*, Vol. 43 No 1, pp. 129-159, 2001.

[18] D. Donoho, X. Huo, "Uncertainty principles and ideal atomic decomposition", *IEEE Trans. Inform. Theory* Vol 47, pp. 2845-2862, 2001.

[19] D. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, Vol. 52, No. 4, pp. 1289-1306, 2006.

[20] D. Donoho, "For most large underdetermined systems of linear equations, the minimal ℓ_1 solution is also the sparsest solution", *Communication on pure and applied Mathematics*, Vol. 59, No. 7, pp. 907934, 2006.

[21] D. Donoho, "High-dimensional Data Analysis: The Curses and Blessings of Dimensionality" , *International conference of mathematicians*, Paris, August 2000.

[22] M. Elad, "Sparse and redundant representations", Springer 2010.

[23] M. Figueiredo, R. Nowak, and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems", *IEEE Selected topics in signal processing*, Vol. 1, No. 4, pp. 586-597, 2007.

[24] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, Vol. 7, pp. 179-188, 1936.

[25] P. Guillén, F. Martínez-de-Pisón, R. Sanchez, M. Argaez, L. Velazquez, "Characterization of Subcortical Structures during Deep Brain Stimulation utilizing Support Vector Machines", *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 7949 - 7952, 2011.

[26] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection" *Journal of Machine Learning Research*, Vol. 3, pp. 1157-1182, 2003.

[27] I. Guyon, S. Gunn, M. Nikravesh and L. Zadeh, "Feature Extraction, Foundations and Applications" *Series Studies in Fuzziness and Soft Computing*, Physica-Verlag, Springer, 2006.

[28] E. Hale, W. Yin, and Y. Zhang, "A fixed-point continuation method for ℓ_1-regularized minimization with applications to compresses sensing", Technical Report TR07-07, Department of Computational and Applied Mathematics, Rice University, Houston, TX, USA, 2007.

[29] C. Hamani, J.A. Saint-Cyr, J. Fraser, M. Kaplitt, A. Lozano, "The subthalamic nucleus in the context of movement disorders", *BRAIN Journal of Neurology*, Vol. 127 No. 1, pp. 4-20, 2004.

[30] X. Hang, F. Wu, "Sparse Representation for Classification of Tumors Using Gene Expression Data", *Journal of Biomedicine and Biotechnology*, Vol. 2009, Article ID 403689.

[31] X. Hang, F. Wu, "ℓ_1 least square for cancer diagnosis using gene expression data", *Journal of Computer Science and System Biology*, Vol. 2009, pp. 167-173.

[32] M. Hernandez, J. Olaya, R. Sanchez, C. Ramirez, R. Romero, L. Velazquez, M. Argaez, "Performance Comparison of an HPC ℓ_1-optimization algorithm for compressed sensing", *IEEE proceedings of Department of Defense High Performance Computing Modernization Program Users Group Conference*, pp. 391 - 400. 2011.

[33] S. Jokar, M. Pfetsch, "Exact and approximate sparse solutions of underdetermined linear equations", *SIAM Journal on Scientific Computing*, Vol. 31, No. 1, pp. 23-44, 2008.

[34] I.T Jolliffe, "Principal Component Analysis", *Springer Series in Statistics*, 2nd ed., Springer, NY, 2002.

[35] S. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinvesky, "An interior-point method for large-scale ℓ_1-regularized least squares", *IEEE Selected topics in signal processing*, Vol. 1, No. 4, pp. 606-617, 2007.

[36] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", *Proceedings of International Joint Conference on AI* , pp 1137-1145, 1995.

[37] S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries", *IEEE Trans. on Signal Processing*, Vol. 41, pp. 33973415, 1993.

[38] C. Miosso, R. Von-Borries, M. Argáez, L. Velázquez, C. Quintero, C. Potes, "Compressed sensing reconstruction with prior information using penalized reweighted normal equations", *IEEE Transactions on Signal Processing*, Vol. 52, No. 4, pp. 1289-1306, 2009.

[39] B. K. Natarajan, "Sparse approximate solutions to linear systems", *SIAM Journal on computing*, Vol. 24, pp.227-234, 1995.

[40] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition", *27th Annual Asilomar Conference on Signals, Systems, and Computers*, 1993.

[41] Ramirez, C., Argaez, M., Jaimes, A., Tweedie C. E, "Image inpainting in micrometereological analysis", *IEEE proceedings of the IEEE International Conference in Image Processing ICIP2012*, pp. 1725-1728. 2012.

[42] R. T Rockafellar, "Convex Analysis", *Princeton. NJ : Princeton Univ. Press*, 1970.

[43] Y. Saeys, I. Inza, and P. Larranaga, "A review of feature selection techniques in bioinformatics", *Bioinformatics*, Vol.23, No. 19, pp. 25072517, 2007.

[44] Sanchez R., Argaez M., Guillen P. "Sparse Representation via ℓ_1-minimization for Underdetermined Systems in Classification of Tumors with Gene Expression Data" *33rd Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 3362 - 3366, 2011.

[45] A. Statnikov, C.F Aliferis, I. Tsamardinos, D. Hardin and S. Levy, "A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis", *Bioinformatics*, Vol. 21, No. 5, pp. 631-643, 2005.

[46] H. L. Taylor, S. C. Banks, J. F. McCoy. "Deconvolution with the ℓ_1 norm", *Geophysics*, Vol. 44, pp. 39-52, 1979.

[47] H. Trevor, R. Tibshiriani, J. Friedman, "The elements of statistical learning : data mining, inference, and prediction", *Springer Series in Statistics*, New York, Springer. 2011.

[48] J. Tropp, "Greed is good: Algorithmic results for sparse approximation", *IEEE Transactions on Information Theory*, Vol. 50, No. 10, pp. 22312242, 2004.

[49] L. Wang (Editor), "Support Vector Machines: Theory and Applications", *Springer-Verlag Berlin*, 2005.

[50] J. Wright, Y. Yang, A. Ganesh, S. S. Sastry and Y. Ma, "Robust Face Recognition via Sparse Representation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 61, No. 2, pp 210-227, 2009.

[51] W. Yin, Y. Zhang, "Extracting salient features from less data via ℓ_1-minimization", *SIAG/Optimization Views and News* Vol. 9, 2008.

Curriculum Vitae

Reinaldo Sanchez Arias was born in Cali, Colombia on February 17, 1986. He is the first son of Reinaldo Sanchez and Alid Arias. His interest in mathematics and science dates back to his early childhood, inspired by his father who was a mathematician, and the constant search for knowledge inculcated by his mother. Reinaldo is the brother of Juan Camilo Sanchez Arias, a current Medical School student. In Spring 2008 he earned his Bachelor of Science degree in Mathematics from Universidad del Valle in Cali, Colombia, under the direction of Dr. Jairo Duque. Later that year, he met Dr. Leticia Veláquez and Dr. Miguel Argáez during their visit to Colombia for attending a conference. Advised by them, he deciced to pursuit a doctoral degree in Computational Science in the United States. In Fall 2008 he arrived at El Paso, and started his doctoral studies at The University of Texas at El Paso. During his first semester he worked as a teaching assistant at the Mathematical Sciences Department at UTEP. Thereafter he became a research assistant under the direction of principal investigator Dr. Miguel Argez in a work with the Army High Performance Computing Research Center (AHPCRC), funded by an ARL grant. He is currently a member of the Society of Industrial and Applied Mathematics (SIAM) and the American Mathematical Society (AMS). He also served as the vice-president of the UTEP SIAM Student Chapter. He has presented his research at international and national conference meetings including SIAM meetings, and the International and Conference for High Performance Computing.

The University of Texas at El Paso
Computational Science Program
500 West University Ave. Bell Hall
El Paso, Texas 79968-0514
rsanchezarias@miners.utep.edu
reinaldosanar@gmail.com