Efficient Classification Based on Sparse Regression

by

Pardis Noorzad

A thesis submitted in partial fulfilment

of the requirements for the degree of

Master of Science

in

COMPUTER ENGINEERING

with an emphasis in Artificial Intelligence

at the

Department of Computer Engineering and Information Technology

AMIRKABIR UNIVERSITY OF TECHNOLOGY

Committee:

_____

Mohammad Rahmati, Supervisor

_____

Nasrollah Moghaddam Charkari

_____

Mohammad Mehdi Ebadzadeh

July 2012

Abstract

The ℓ_1-regularized square loss minimization problem has recently gained much attention. This optimization principle has two main applications in the machine learning literature. Specifically, the lasso or basis pursuit de-noising (although the two are not entirely equivalent in practice) is the optimization principle used for solving the linear inverse problem: $\mathbf{y} = \mathbf{X}\mathbf{a}$, under convex sparsity constraints. When the lasso is used for regression and classification, $\mathbf{y}$ is a vector of outputs. When it is used for sparse coding and feature learning, or in the context of sparse representation classification, $\mathbf{y}$ is the feature vector or signal itself.

The use of lasso for regression is already well-established. In this thesis, we argue that the use of lasso for classification also has its advantages. One might think that the square loss is not appropriate for the classification task, however, theoretical results show that all convex loss functions are Fisher consistent. Additionally, square loss minimization, like logistic loss minimization, and unlike hinge loss minimization, gives estimates of the posterior probability. The value of the posterior probability at a point tells us about the confidence of the classifier in its prediction. Another benefit of the lasso for classification is that ℓ_1-regularization leads to a sparse classifier, that once trained, can be evaluated quickly. Additionally, one has direct control over the sparsity of the solution through the regularization parameter. The only problem with the lasso is the stability of its solutions (Wang et al., 2011).

The second part of the thesis, is on the use of the lasso for signal and feature representation. The lasso or basis pursuit de-noising is also an integral part of the sparse representation classification method. We extend this method to the regression setting. Through experimental results we argue that one can easily achieve the same or even better results using simpler methods like k-nearest neighbor classification which is also better motivated theoretically. We conclude that ℓ_1-regularized square loss minimization is not worth it.

Keywords: square loss minimization, ℓ_1-regularization, binary classification, sparse representation

Acknowledgements

I thank my supervisor, Prof. Mohammad Rahmati, for accepting me as his student at the Image Processing and Pattern Recognition lab, and for making sure we had a comfortable place in which to work. I also thank him for his time and patience in putting up with my myriad ideas for the thesis, for providing me with directions, and for making sure I stayed on the right track during my studies at AUT.

I am truly indebted to Prof. Bob L. Sturm of Aalborg University Copenhagen. This document would have lacked in content and quality had I not had the wonderful privilege of working with him. He introduced me to sparse approximation algorithms that have become the motivation for this thesis. I thank him for being the inspiring mentor who has helped me gain the confidence and skills to pursue a future in research.

Special thanks go to my brother, Parham, for his technical advice on the Hilbert space of functions, inner products and norms, and for all the many hours of conversations. My greatest gratitude goes to my parents, for being extraordinary teachers, for supporting me, comforting me, and caring for me through every moment of my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "The resulting classification function depends only on so-called supporting patterns. These are those training examples that are closest to the decision boundary and are usually a small subset of the training data."
>
> —Boser et al. (1992)

This thesis considers two main applications of ℓ_1-regularized square loss minimization. The first application, covered in Chapter 2, is of employing the mentioned optimization problem for the linear binary classification task. The second application, covered in Chapter 3, is that of using ℓ_1-regularized square loss minimization for signal and feature representation. In Chapter 4, an equivalence between ℓ_1-regularized square loss minimization and ϵ-support vector regression is illustrated under a number of restricting conditions. In the last chapter, conclusions and some areas for future work are presented.

1.1 Linear Classification and the SVM

The support vector machine (SVM) classifier (Boser et al., 1992; Vapnik, 1995) has two well-known properties: excellent generalization performance—attributed to an ℓ_2 penalty on the weights; and a sparse coefficient vector—attributed to the the hinge loss as its data fitting term. The latter suggests that upon computing the output for a given test sample, the SVM classifier uses only a subset of the training samples, known as the *support vectors*, thus leading to faster evaluation. In practice, however, the number of support vectors is comparable to the number of training samples: the coefficient vector is not sparse. Thus, the resulting classifier is expensive and grows more so with increasing training samples (see Figure 1.1). The SVM optimization problem has the form

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} && \begin{cases} y_i(\mathbf{w}.\mathbf{x}_i - b) & \geq 1-\xi_i \\ \xi_i & \geq 0 \end{cases} \qquad \text{for } i = 1, \cdots, n, \end{aligned} \tag{1.1}$$

This can alternatively be written as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \xi_i \geq \max\Big(0, 1 - y_i(\mathbf{w}.\mathbf{x}_i - b)\Big) \quad \text{for } i = 1, \cdots, n. \end{aligned} \tag{1.2}$$

Introduce the notation

$$\xi_i \geq \Big[1 - y_i(\mathbf{w}.\mathbf{x}_i - b)\Big]_+, \text{ where } [x]_+ = \max(0, x) \tag{1.3}$$

This leads to the following formulation for the SVM optimization problem

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\Big[1 - y_i(\mathbf{w}.\mathbf{x}_i - b)\Big]_+ \tag{1.4}$$

which consists of two terms, namely, the hinge loss and ℓ_2-regularization of the weights.

Sparsity of the resulting classifier is important for real-time applications like speech recognition. It is also crucial for applications with many observations to evaluate, for e.g., in systems for reading cheques and zip codes. A sparse classifier also takes up less memory.

In this thesis, we focus on linear classifiers. There are two main reasons for this decision. Nonlinear classifiers, like nonlinear SVM take longer to train and evaluate. This is especially a setback in the multiclass case where many classifiers are to be trained for one task, in a one-versus-one or one-versus-all setting. Additionally, for high-dimensional problems, nonlinear classification does not provide a significant advantage. High-dimensional data spaces are usually sparse—except in some applications—and are therefore more likely to be linearly separable.

Towards building a faster classifier, we enforce sparsity by regularization with a sparsity-inducing norm. This also allows explicit control over the sparsity of the coefficient vector through the regularization parameter—unlike in the SVM optimization where such control is not straightforward (see Figure 1.2). Specifically, we use ℓ_1-regularized square loss minimization algorithms, especially SPGL1 (van den Berg and Friedlander, 2008), to solve the classification problem. This is what sets our work apart from the recent study by Yuan et al. (2010, 2011). Their analyses focuses only on algorithms optimizing the logistic or the squared hinge loss—also called the L_2 loss—and defined as,

$$\max(0, 1 - y\mathbf{a}^\mathsf{T}\mathbf{x})^2. \tag{1.5}$$

The logistic loss is twice differentiable as is the case for the square loss. However, the squared hinge loss is not.

Related Work in Square Loss Minimization for Classification

In his PhD thesis, Rifkin (2002) claims that the hinge loss is not the secret to SVM's success. Rifkin proposes Regularized Least Squares Classification (RLSC)

$$\min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda\|\mathbf{w}\|^2 \tag{1.6}$$

and the nonlinear case

$$\min_{\mathbf{c}} \|\mathbf{y} - \mathbf{K}\mathbf{c}\|^2 + \lambda \mathbf{c}^\mathsf{T}\mathbf{K}\mathbf{c} \tag{1.7}$$

where $\mathbf{K}$ is the matrix obtained by evaluating the kernel K on pairs of the training samples. Through some experimental studies, Rifkin shows that RLSC is as good as SVM on various data sets. However, RLSC does not lead to a sparse classifier. Additionally, nonlinear RLSC takes longer than SVM to train.

Least-squares optimization can be used for binary classification (Zhang and Oles, 2001; Rifkin et al., 2003; Hastie and Zhu, 2006). It can also be used in the context of generalized linear models—for e.g. logistic regression with iteratively reweighted least squares optimization (Tibshirani, 1996; Lee et al., 2006) and for m-estimation (Fox, 2002; Andersen, 2008). We obtain a faster classifier that still retains some generalization capabilities because of the ℓ_1 regularizer. However, the latter deserves further analysis (Girosi, 1998; Poggio et al., 2009; Xu et al., 2012).

Related Work in ℓ_1-regularization for Sparse Classifiers

Yuan et al. (2010) compare several sparse linear classifiers of the form

$$\min_{\mathbf{w}} \|\mathbf{w}\|_1 + C \sum_{i=1}^{n} \xi(\mathbf{w}; \mathbf{x}_i, y_i) \tag{1.8}$$

with the logistic, hinge, and square hinge loss defined as

- $\xi_{\log}(\mathbf{w}; \mathbf{x}_i, y_i) = \log\left(1 + \exp(-y\mathbf{w}^\mathsf{T}\mathbf{x})\right)$
- $\xi_{L_1}(\mathbf{w}; \mathbf{x}_i, y_i) = \max\left(1 - y\mathbf{w}^\mathsf{T}\mathbf{x}, 0\right)$
- $\xi_{L_2}(\mathbf{w}; \mathbf{x}_i, y_i) = \max\left(1 - y\mathbf{w}^\mathsf{T}\mathbf{x}, 0\right)^2$.

However, they don't consider optimizing the square loss for classification. As we show in the next chapter, the square loss has several good computational and statistical properties.

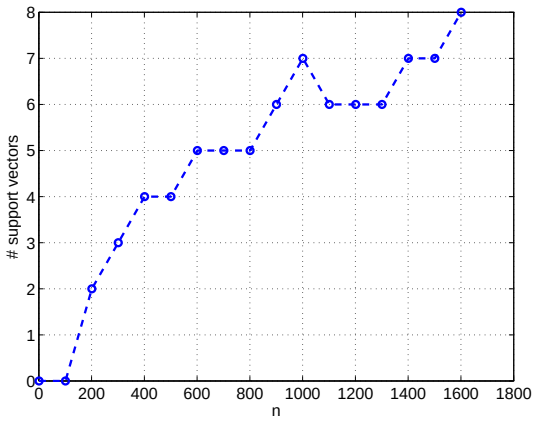
As we show through experiments and some simple proofs, there doesn't seem to be a significant advantage in preferring one convex loss function over the others for minimization in the classification setting. Additionally, there are some benefits of optimizing the square loss over the hinge loss and logistic loss (Zhang, 2004; Hastie and Zhu, 2006). When there are inexpensive square loss minimization algorithms, it seems reasonable and efficient to use them for our classification tasks.

Table 1.1 contains information about the datasets we use for our experiments in this chapter and the next. They are from the LIBSVM website for binary class data sets[1]. The sets are chosen to be representative of a variety of tasks that arise in practice.

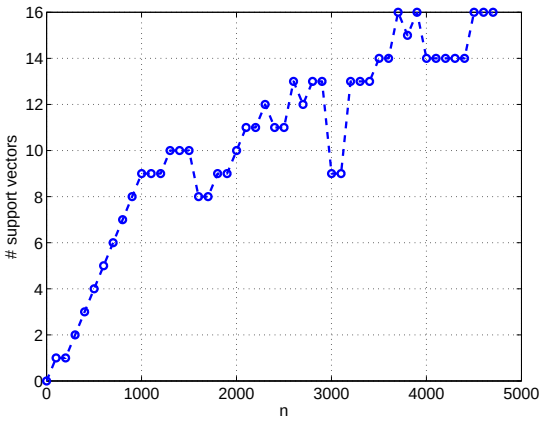
[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/binary.html

Table 1.1: Data set information: n denotes the number of observations and p denotes the number of attributes of each observation.

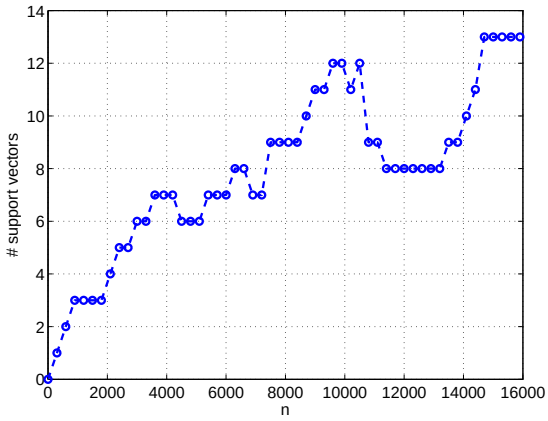
Data set	# observations (n)	# attributes (p)
adult1	1605	123
adult4	4781	123
adult7	16,100	123
adult9	32,561	123
australian	690	14
colon-cancer	62	2000
covertype	581,012	54
diabetes	768	8
heart	270	13
ionosphere	351	34
leu	38	7,129
liver-disorders	345	6
mushrooms	8124	112



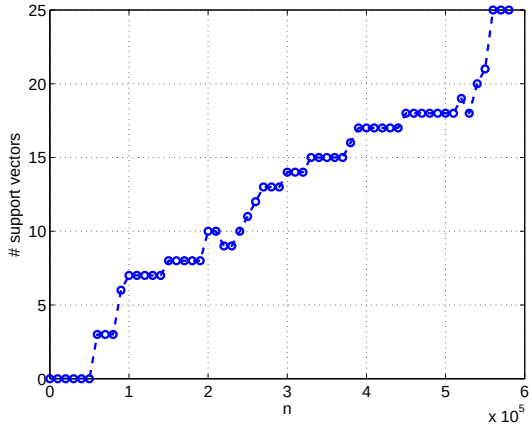
(a) adult1 dataset



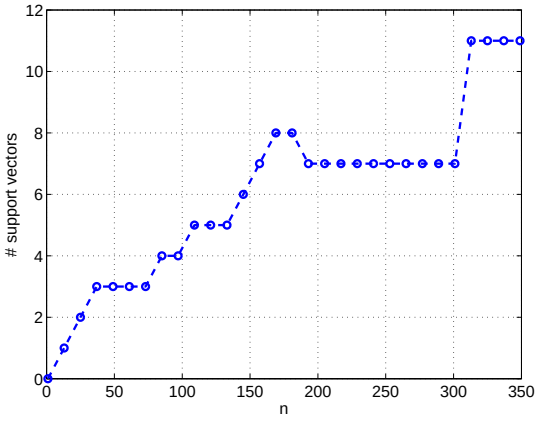
(b) adult4 dataset



(c) adult7 dataset

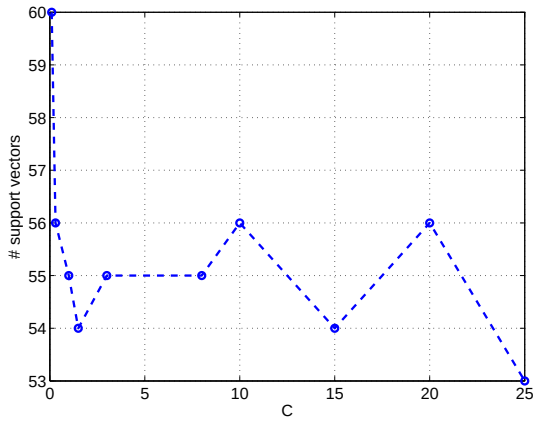


(d) covertype dataset

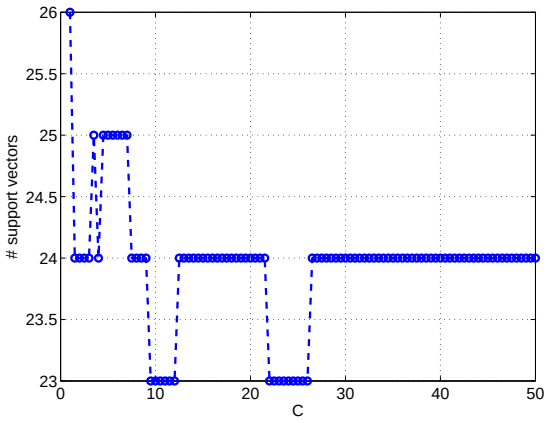


(e) Ionosphere dataset

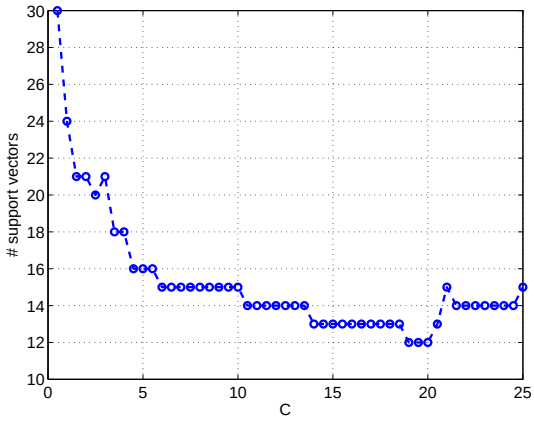
Figure 1.1: Graphs show that the number of support vectors increases as the number of training samples grows.



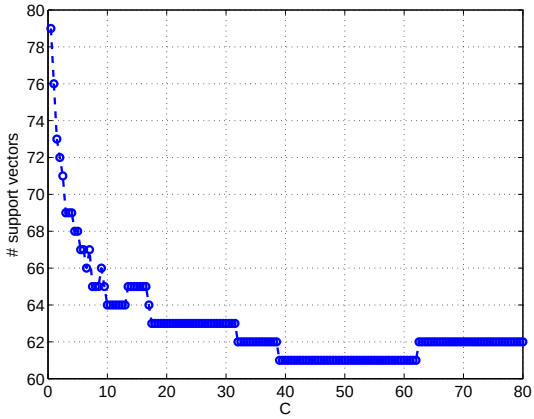
(a) australian dataset



(b) heart dataset



(c) ionosphere dataset



(d) liver-disorders dataset

Figure 1.2: Graphs show that the number of support vectors does not have a meaningful relation to the hyperparameter C in the SVM optimization.

1.2 Linear Inverse Problems, Regression, and Regularization

Given $\mathcal{T}_n$, our goal is to estimate the real regression coefficients $a_0, a_1, \ldots, a_p$ of the linear model

$$Y = \sum_{j=1}^{p} a_j X_j + a_0 \tag{1.9}$$

such that an empirical estimate of the expected value of a specified loss criterion is minimized. Note that the random variable Y is the *response* variable, and the random vector X contains the *predictor* or *explanatory variables* $X_1, \ldots, X_p$. In the absence of noise and using the vector-matrix notation, the regression problem boils down to finding a solution to the following system of equations

$$\mathbf{y} = \mathbf{X}\mathbf{a} \tag{1.10}$$

where $\mathbf{y} = [y_1, \ldots, y_n]^\mathsf{T}$ is the vector of observed responses, $\mathbf{X}$ is an $n \times p$ design matrix defined as $\mathbf{X} = [\mathbf{x}_1 | \mathbf{x}_2 | \ldots | \mathbf{x}_n]$, where $\mathbf{x}_i = [x_{i1}, \ldots, x_{ip}]^\mathsf{T}$, and $\mathbf{a} = [a_1, \ldots, a_p]^\mathsf{T}$ is the target vector of regression coefficients. In the rest of our discussion, we assume that the bias term a_0, is implicit in our representation, that is

$$\mathbf{x}_i^\mathsf{T} = [\mathbf{x}_i^\mathsf{T}, 1] \quad \text{and} \quad \mathbf{a}^\mathsf{T} = [\mathbf{a}^\mathsf{T}, a_0]. \tag{1.11}$$

One would wish to solve (1.10) directly, but $\mathbf{X}$ is almost never invertible in our problems. If $n = p$ or $n > p$, the system of equations characterized by (1.10) is *overdetermined*. In this case a solution does not exist. If $n < p$ the system is *underdetermined* and there exists infinitely many solutions. Instead of solving (1.10), one tries to minimize a certain measure of the residual between the sides of the equation. A common loss criterion for regression applications is the least squares loss function. The least squares error is the sum of squared residuals, where the residuals are defined as

$$r = y - \sum_{j=1}^{p} a_j x_j - a_0. \tag{1.12}$$

Putting it all together, the least squares error becomes

$$\|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2. \tag{1.13}$$

In other words, we allow for some noise in the model, and aim to minimize a certain function of that noise. One way to estimate the regression coefficients $\mathbf{a}$, is to minimize (1.13). The minimization of (1.13) is an unconstrained convex optimization problem with a differentiable objective. It has a unique global minimum. The objective in (1.13) is minimized when its gradient is zero with respect to $\mathbf{a}$. This yields the following system of equations, known as the *normal equations*

$$(\mathbf{X}^\mathsf{T}\mathbf{X})\hat{\mathbf{a}} = \mathbf{X}^\mathsf{T}\mathbf{y}. \tag{1.14}$$

The normal equations can be solved efficiently if $\mathbf{X}^\mathsf{T}\mathbf{X}$ has full rank—in this case, $\mathbf{X}$ must have full column rank. However, this condition may not hold in most applications and lead to numerical instability. This can also cause the least squares estimate to be

highly inaccurate. A solution to this problem is *Tikhonov regularization*. In Tikhonov regularization, the square of the ℓ_2 norm of the coefficients are added to the objective, thereby giving preference to solutions with smaller ℓ_2 norms. The idea is to incorporate a priori assumptions on the solution. Tikhonov regularization leads to the following optimization problem

$$\|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_2^2 \tag{1.15}$$

where λ is the regularization paramter that controls the tradeoff between the minimization of the least squares term and the minimization of the ℓ_2 penalty term. The objective in (1.15) remains convex and differentiable with a unique global minimum given as the solution of the following system of equations

$$(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})\hat{\mathbf{a}} = \mathbf{X}^\mathsf{T}\mathbf{y}. \tag{1.16}$$

Note that $\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I}$ is nonsingular even when $\mathbf{X}^\mathsf{T}\mathbf{X}$ is singular. The minimization of (1.15) is also known as *ridge regression*.

The ℓ_2 penalty is part of the *power family* of penalties, indexed by $\gamma \geq 0$

$$\sum_{j=1}^{p} |a_j|^\gamma. \tag{1.17}$$

This is the ℓ_γ norm of the parameter raised to the power of γ (Friedman, 2008). For $\gamma = 2$ we get Tikhonov regularization or ridge regression. It is known that when $\gamma = 2$, regularization results in a coefficient vector that is dense, i.e., almost all values are nonzero. Regularization with the ℓ_2 norm only has the *shrinkage* property—shrinking the coefficient absolute values. At the other end of the spectrum, $\gamma = 0$ produces a sparse coefficient vector. Hence, it is said to have the *variable selection* property. We have already seen this in play in Section 1.3. Least squares regression with $\gamma = 0$ is known as *all-subsets regression*. Penalizing with the ℓ_0 norm forces many values of the coefficient vector to zero but does not shrink any nonzero values. In between, there is $\gamma = 1$. Optimization with the ℓ_1 penalty holds the best of both worlds: shrinkage and selection. Least squares optimization with ℓ_1-regularization has been made popular by Chen et al. (1995) as Basis Pursuit Denoising (BPDN), and by Tibshirani (1996) as the Least Absolute Selection and Shrinkage Operator (LASSO).

Let us look more closely at the ℓ_1-regularized least squares problem

$$\min_{\mathbf{a}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1. \tag{1.18}$$

The objective in (1.18) is convex but no longer differentiable due to the ℓ_1 norm. Thus there no longer exists a closed form solution as was the case for (1.13) and (1.15). In Chapter 2 we go over several methods for solving (1.18).

We are also interested in the *elastic-net* penalty. Additionally, go over all three "equivalent" forms of Tikhonov regularization.

1.2.1 Logistic Regression

In logistic regression, the regression function has a nonlinear relation with the linear combination of the explanatory variables. This relation is modeled by the *probit function*.

In the classification setting, the response is a binary variable, i.e., $y_i \in \{-1, 1\}$. The response data are chosen to be realizations of a Bernoulli random variable Y with success probability $\eta = \mathbb{P}\{Y = 1\}$. The success probability is assumed to depend on the predictors, i.e., $\eta = \eta(\mathbf{x})$. For the Bernoulli distribution, it is known that $\mathbb{E}\{Y\} = \eta$. If the predictors are considered to be realizations of a random variable X, then $\eta(\mathbf{x})$ is the conditional expectation of Y

$$\mathbb{E}\{Y|X\} = \eta(\mathbf{x}). \tag{1.19}$$

In linear regression, the conditional expectation of Y given the value $\mathbf{x}$ of X is an affine function of $\mathbf{x}$

$$\mathbb{E}\{Y|X\} = \mathbf{a}^\mathsf{T}\mathbf{x}. \tag{1.20}$$

In classification however, a monotone *link function* g transforms the expectation to the linear combination of the predictors

$$g\big(\mathbb{E}\{Y|X\}\big) = \mathbf{a}^\mathsf{T}\mathbf{x}. \tag{1.21}$$

Models of this form are known as *generalized linear models*. In the case of *logistic regression*, the link function is chosen to be to logit function

$$g(a) = \ln \frac{a}{1-a}. \tag{1.22}$$

The inverse of the logit function is the logistic function, denoted as $\sigma(z)$

$$g^{-1}(z) = \sigma(z) = \frac{1}{1+\exp(-z)} \tag{1.23}$$

therefore the conditional expectation becomes,

$$\mathbb{E}\{Y|X\} = g^{-1}\big(\mathbf{a}^\mathsf{T}\mathbf{x}\big) = \sigma\big(\mathbf{a}^\mathsf{T}\mathbf{x}\big). \tag{1.24}$$

In order to estimate the unknown parameter $\mathbf{a}$ of the model, we begin by forming the likelihood function. We assume the observations are generated independently. The likelihood function is algebraically the same as the joint probability density function of the observations, thus

$$\begin{aligned} \mathcal{L}(\mathbf{a}) = p(\mathbf{y}|\mathbf{X};\mathbf{a}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i;\mathbf{a}) \\ &= \prod_{i=1}^{n} \sigma\big(\mathbf{a}^\mathsf{T}\mathbf{x}_i\big)^{y_i}\big(1-\sigma(\mathbf{a}^\mathsf{T}\mathbf{x}_i)\big)^{1-y_i}. \end{aligned} \tag{1.25}$$

For computational reasons we consider instead the logarithm of the likelihood function

$$\ell(\mathbf{a}) = \log \mathcal{L}(\mathbf{a}) = \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i;\mathbf{a}). \tag{1.26}$$

Maximizing the likelihood is equivalent to minimizing the negative log-likelihood. The optimization problem we aim to solve becomes

$$\min_{\mathbf{a}} \sum_{i=1}^{n} -\log p(y_i|\mathbf{x}_i;\mathbf{a}) = \min_{\mathbf{a}} \sum_{i=1}^{n} -\log \sigma\big(y_i\mathbf{a}^\mathsf{T}\mathbf{x}_i\big) \tag{1.27}$$

$$= \min_{\mathbf{a}} \sum_{i=1}^{n} \log\big(1+\exp\big(-y_i\mathbf{a}^\mathsf{T}\mathbf{x}_i\big)\big). \tag{1.28}$$

where (1.27) follows because $y_i \in \{-1, 1\}$. The function

$$\log\left(1 + \exp\left(-y_i \mathbf{a}^\mathsf{T} \mathbf{x}_i\right)\right) \tag{1.29}$$

in (1.28) is often referred to as the *logistic loss*. As can be seen, optimizing the logistic loss happens to produce a maximum likelihood estimate. Therefor, logistic regression is considered to be maximum-likelihood estimation if the posterior probability $\eta(x)$ can be expressed as $1/(1 + \exp(-f(x)))$ for some $f(x) \in \mathcal{F}$.

Let's assume instead, that we are interested in the maximum a posteriori (MAP) estimate of the parameters $\mathbf{a}$. We consider a Laplacian prior on the parameters. The multivariate Laplacian probability distribution has the form

$$p(\mathbf{a}) = \left(\frac{\lambda}{2}\right)^n \exp(-\lambda \|\mathbf{a}\|_1). \tag{1.30}$$

The optimization problem we need to solve is

$$\max_{\mathbf{a}} \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i; \mathbf{a}) p(\mathbf{a}) = \max_{\mathbf{a}} \sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i; \mathbf{a}) + \log p(\mathbf{a}) \tag{1.31}$$

$$= \min_{\mathbf{a}} \sum_{i=1}^{n} -\log p(y_i|\mathbf{x}_i; \mathbf{a}) + \lambda \|\mathbf{a}\|_1 \tag{1.32}$$

$$= \min_{\mathbf{a}} \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \mathbf{a}^\mathsf{T} \mathbf{x}_i\right)\right) + \lambda \|\mathbf{a}\|_1. \tag{1.33}$$

The optimization in (1.33) is referred to as the ℓ_1-regularized logistic regression problem.

1.3 Sparse Approximation

Methods for sparse approximation and sparse coding have recently been getting increasing attention (Bruckstein et al., 2009). Hence, there arises the need to exploit what these new or renewed results have to offer. Below we briefly go over the problem of sparse approximation.

Suppose we have a full rank matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, with $n < m$, and we wish to solve the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Clearly, the system of equations is underdetermined and so has infinitely many solutions. We constrain the problem by requiring that $\mathbf{x}$ be sparse, i.e. have few nonzero entries. More precisely, we define the ℓ_0 pseudo-norm as below

$$\|\mathbf{x}\|_0 = \#\{i : x_i \neq 0\}.$$

Based on this definition, $\mathbf{x}$ is *sparse* if $\|\mathbf{x}\|_0 \ll m$. The optimization problem we want to solve is as follows,

$$\text{minimize} \quad \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{1.34}$$

However, because of the combinatorial aspect of the ℓ_0 norm, the above optimization problem is NP-hard. Over the years, there have been many attempts at solving the sparse approximation problem of (1.34) approximately, see for e.g. (Mohimani et al., 2009). There have also been attempts at relaxing the constraints and objective of (1.34) to make it tractable. Many of these methods solve a relaxed version of this optimization problem exactly, see for e.g. (Chen et al., 1995; Tibshirani, 1996). We will repeatedly come back to this topic throughout the thesis.

1.3.1 Sparse Representation of Signals and Features

In recent years, sparse approximation has proven to be a successful *unsupervised feature learning* method (Coates and Ng, 2011). One can use sparse approximation (Section 1.3) to obtain a (higher-dimensional) sparse representation of a given feature vector. This is known in the literature as *sparse coding*. Yang et al. (2009) show that sparse coding improves the classification performance of linear SVM when used to learn features in an image classification task.

If the dictionary used in sparse coding is made up of training samples with known labels, the resulting sparse representation can be used for classification (Wright et al., 2009). We show that this setup can also be used for regression.

Chapter 2

On ℓ_1-regularized Square Loss Minimization for Classification

> "Lasso (ℓ_1-)penalties are useful for fitting a wide variety of models. Newly developed computational algorithms allow application of these models to large data sets, exploiting sparsity for both statistical and computation gains."
>
> —Tibshirani (2011)

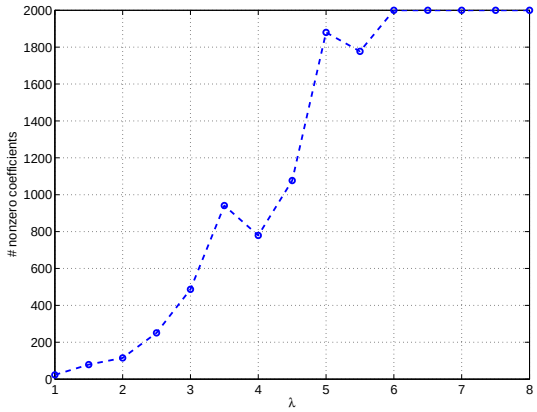
As was explained in Chapter 1, we are interested in solving the ℓ_1-regularized least squares optimization problem of (1.18). In this chapter we go over efficient methods for performing this optimization problem and more importantly, answer if it can be employed successfully in the classification setting. There are a wide variety of algorithms available to solve the ℓ_1-regularized least squares optimization problem (Mairal, 2010). In this thesis, we employ one of the most successful of these algorithms that is able to solve the ℓ_1-regularized square loss minimization problem optimally. This algorithm is called Spectral Projected Gradient Method for ℓ_1-minimization (SPGL1) (van den Berg and Friedlander, 2008) and solves the following optimization problem (it calls the lasso)

$$\min_{\mathbf{a}} \ \|\mathbf{y} - \mathbf{X}\mathbf{a}\|_2^2 \ \text{ s.t. } \ \|\mathbf{a}\|_1 \leq \lambda. \tag{2.1}$$

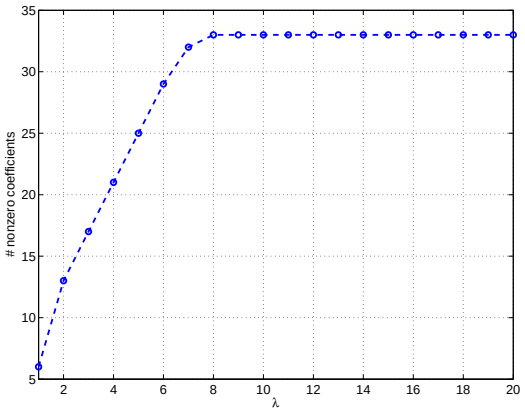
There are a variety of reasons for using the lasso optimization principle to train our classifier. As we have noted before, one of the reasons is explicit control over the sparsity of the solution. In Figure 2.1, we show that one can control the sparsity of the solution using the regularization parameter λ in the lasso formulation above. Another important reason is that the square loss has many computational and statistical benefits. The next section explains why all convex loss functions are suitable for the task of classification. It also explains an important statistical advantage of the square loss over the hinge loss.

2.1 Classification and Convex Loss Minimization

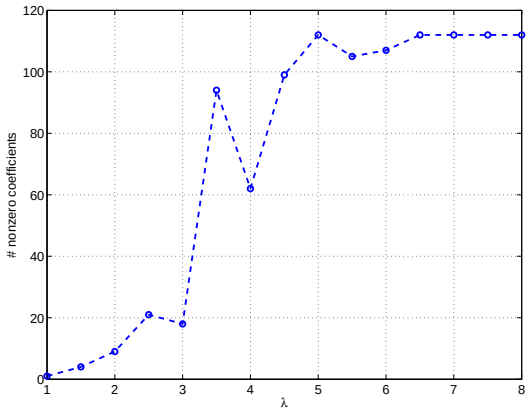
Pattern recognition, also known as *classification*, is the process of assigning a discrete label to an unknown observation (Devroye et al., 1996). In pattern recognition, the task is to find a function $g : \mathbb{R}^p \to \{1, \ldots, M\}$ which takes an *observation* represented by $\mathbf{x} \in \mathbb{R}^p$ and assigns it to $y \in \{1, \ldots, M\}$: one of M available *classes*. The function g is called a *classifier*.



(a) colon-cancer dataset



(b) ionosphere dataset



(c) mushrooms dataset

Figure 2.1: In these figures, we see that the number of nonzero elements of the solution increases as we increase the regularization parameter λ, thus providing us with means to control the sparsity of the solution.

For our present analysis, let X and Y be random variables taking their values from $\mathbb{R}^p$ and $\{1, \ldots, M\}$, respectively. Define the *probability of error* for a classifier g as

$$L(g) = \mathbb{P}\{g(X) \neq Y\}. \tag{2.2}$$

Consequently, the optimal classifier g^* is

$$g^* = \underset{g:\mathbb{R}^p \to \{1,\ldots,M\}}{\operatorname{argmin}} \mathbb{P}\{g(X) \neq Y\} \tag{2.3}$$

and is called the *Bayes classifier*. The corresponding probability of error—the minimum probability of error—is called the *Bayes error* and denoted by $L^* = L(g^*)$.

To compute g^*, one needs to know the distribution of (X, Y) which is unknown. We can however, given enough data, find a classifier g with low $L(g)$. For our classification

task, we have access to the data set $\mathcal{T}_n = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ of n observations, assumed to be sampled i.i.d. from the distribution of (X, Y).

In the rest of this thesis, we consider the case of binary classification, that is, $M = 2$. In this case, Y takes its values from the binary set $\{-1, +1\}$. Binary classification can readily be extended to the multiclass classification ($M > 2$) in a one-versus-all (OVA) setting (Rifkin and Klautau, 2004).

Before we further delve into the topic of classification, we introduce an importance concept in this setting: the posterior probability.

2.1.1 The Posterior Probability and Plug-in Classifiers

In *regression analysis*, the goal is to estimate Y given X by $r(X)$, where $r : \mathbb{R}^p \to \mathbb{R}$ is a function. One can show that the function minimizing the mean squared error in this setting is the posterior probability η

$$\eta(\mathbf{x}) = \mathbb{P}\{Y = 1 | X = \mathbf{x}\} = \mathbb{E}\{Y | X = \mathbf{x}\}. \tag{2.4}$$

That is, for all $r : \mathbb{R}^p \to \mathbb{R}$,

$$\mathbb{E}\{(\eta(X) - Y)^2\} \leq \mathbb{E}\{(r(X) - Y)^2\} \tag{2.5}$$

Proof. For each $\mathbf{x} \in \mathbb{R}^p$ we have

$$\begin{aligned}
&\mathbb{E}\{(r(X) - Y)^2 | X = \mathbf{x}\} \\
&\quad = \mathbb{E}\{(r(X) - \eta(X) + \eta(X) - Y)^2 | X = \mathbf{x}\} \\
&\quad = (r(\mathbf{x}) - \eta(\mathbf{x}))^2 + 2(r(\mathbf{x}) - \eta(\mathbf{x}))\mathbb{E}\{\eta(X) - Y | X = \mathbf{x}\} + \mathbb{E}\{(\eta(X) - Y)^2 | X = \mathbf{x}\} \\
&\quad = (r(\mathbf{x}) - \eta(\mathbf{x}))^2 + \mathbb{E}\{(\eta(X) - Y)^2 | X = \mathbf{x}\}.
\end{aligned} \tag{2.6}$$

Note that equality holds in (2.5), if and only if for all $\mathbf{x} \in \mathbb{R}^p$

$$r(\mathbf{x}) = \eta(\mathbf{x}). \qquad \square$$

The significance of the posterior probability is that, given η, one can construct a classifier with minimum probability of error. Define the classifier $g^* : \mathbb{R}^p \to \{-1, 1\}$ using the regression function η as

$$g^*(\mathbf{x}) = \begin{cases} -1 & \text{if } \eta(\mathbf{x}) \leq \frac{1}{2} \\ 1 & \text{otherwise.} \end{cases} \tag{2.7}$$

The claim is that g^* is the Bayes classifier, that is, it minimizes the error probability. To prove this, one needs to show that for any classifier $g : \mathbb{R}^p \to \{-1, 1\}$

$$\mathbb{P}\{g^*(X) \neq Y\} \leq \mathbb{P}\{g(X) \neq Y\}. \tag{2.8}$$

Proof. For a given $X = \mathbf{x}$, the conditional probability of error of the classifier g is

$$\begin{aligned}
&\mathbb{P}\{g(X) \neq Y | X = \mathbf{x}\} \\
&\quad = 1 - \mathbb{P}\{g(X) = Y | X = \mathbf{x}\} \\
&\quad = 1 - \Big(\mathbb{P}\{g(X) = 1, Y = 1 | X = \mathbf{x}\} + \mathbb{P}\{g(X) = -1, Y = -1 | X = \mathbf{x}\}\Big) \\
&\quad = 1 - \Big(\chi_{\{g(\mathbf{x})=1\}}\mathbb{P}\{Y = 1 | X = \mathbf{x}\} + \chi_{\{g(\mathbf{x})=-1\}}\mathbb{P}\{Y = -1 | X = \mathbf{x}\}\Big) \\
&\quad = 1 - \Big(\chi_{\{g(\mathbf{x})=1\}}\eta(\mathbf{x}) + \chi_{\{g(\mathbf{x})=-1\}}(1 - \eta(\mathbf{x}))\Big).
\end{aligned} \tag{2.9}$$

where $\chi_{\mathcal{A}}$ is the indicator function of the set $\mathcal{A}$. For every $\mathbf{x} \in \mathbb{R}^p$, one can write

$$\begin{aligned}
&\mathbb{P}\{g(X) \neq Y | X = \mathbf{x}\} - \mathbb{P}\{g^*(X) \neq Y | X = \mathbf{x}\} \\
&\quad = \eta(\mathbf{x})\Big(\chi_{\{g^*(\mathbf{x})=1\}} - \chi_{\{g(\mathbf{x})=1\}}\Big) + (1 - \eta(\mathbf{x}))\Big(\chi_{\{g^*(\mathbf{x})=-1\}} - \chi_{\{g(\mathbf{x})=-1\}}\Big) \\
&\quad = (2\eta(\mathbf{x}) - 1)\Big(\chi_{\{g^*(\mathbf{x})=1\}} - \chi_{\{g(\mathbf{x})=1\}}\Big) \geq 0
\end{aligned} \tag{2.10}$$

where the last equality holds because

$$\begin{aligned}
\chi_{\{g^*(\mathbf{x})=-1\}} &= 1 - \chi_{\{g^*(\mathbf{x})=1\}} \\
\chi_{\{g(\mathbf{x})=-1\}} &= 1 - \chi_{\{g(\mathbf{x})=1\}}.
\end{aligned} \tag{2.11}$$

One can now reach (2.8) by integrating over $\mathbf{x}$. □

The function $\eta(\mathbf{x})$ is unknown. To approximate the Bayes classifier, we employ the nonnegative function $\tilde{\eta}(\mathbf{x})$ as an approximation of $\eta(\mathbf{x})$ and plug it in the form of the Bayes classifier in Eq. (2.18)

$$g(\mathbf{x}) = \begin{cases} -1 & \text{if } \tilde{\eta}(\mathbf{x}) \leq \dfrac{1}{2} \\ 1 & \text{otherwise.} \end{cases} \tag{2.12}$$

The classifier g is called the *plug-in classifier*. Plug-in classifiers perform well. Formally, if $\tilde{\eta}(\mathbf{x})$ is close to $\eta(\mathbf{x})$—as measured by the expected L_1 norm—then the error probability of the plug-in classifier is close to the Bayes error, that is

$$\mathbb{P}\{g(X) \neq Y\} - L^* \leq 2\mathbb{E}\{|\eta(X) - \tilde{\eta}(X)|\}. \tag{2.13}$$

Proof. Note that the difference between the conditional error probabilities of g and g^* is zero when $g(\mathbf{x}) = g^*(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^p$. When $g(\mathbf{x}) \neq g^*(\mathbf{x})$, based on (2.10), one can write the difference as

$$\begin{aligned}
&\mathbb{P}\{g(X) \neq Y | X = x\} - \mathbb{P}\{g^*(X) \neq Y | X = x\} \\
&\quad = (2\eta(x) - 1)\Big(\chi_{\{g^*(x)=1\}} - \chi_{\{g(x)=1\}}\Big) \\
&\quad = \big|2\eta(x) - 1\big|\chi_{\{g(x)\neq g^*(x)\}} \\
&\quad = 2\big|\eta(x) - 1/2\big|\chi_{\{g(x)\neq g^*(x)\}} \\
&\quad \leq 2\big|\eta(x) - \tilde{\eta}(x)\big|.
\end{aligned} \tag{2.14}$$

The last inequality holds because $g(x) \neq g^*(x)$ implies

$$\big|\eta(x) - \tilde{\eta}(x)\big| \geq \big|\eta(x) - 1/2\big|. \tag{2.15}$$

To see why this is true, consider each case in turn

$$\begin{aligned}
&\text{if } g^*(x) = -1 \Rightarrow \eta(x) \leq 1/2 \text{ and } g(x) = 1 \text{ so } \tilde{\eta}(x) > 1/2 \\
&\qquad\qquad \big|\eta(x) - \tilde{\eta}(x)\big| = \tilde{\eta}(x) - \eta(x) \leq 1/2 - \eta(x) = \big|1/2 - \eta(x)\big|
\end{aligned}$$

if $g^*(x) = 1 \Rightarrow \eta(x) > 1/2$ and $g(x) = -1$ so $\tilde{\eta}(x) \leq 1/2$

$$\big|\eta(x) - \tilde{\eta}(x)\big| = \eta(x) - \tilde{\eta}(x) \leq \eta(x) - 1/2 = \big|1/2 - \eta(x)\big|.$$

Getting back to the proof, we can reach the claim by integrating both sides of the inequality in (2.14) over $\mathbf{x}$

$$\mathbb{P}\{g(X) \neq Y\} - L^* \leq 2\mathbb{E}\big\{\big|\eta(X) - \tilde{\eta}(X)\big|\big\}. \qquad \square$$

This result shows that a good estimate of η can produce a good plug-in classifier. What is illuminating to notice is that $\tilde{\eta}(x)$ can be far from $\eta(x)$ and one would obtain the same classifier as long as they are both on the same side of $1/2$.

We are now ready for a thorough treatment on ways to obtain a good classifier in practice.

2.1.2 Empirical Risk Minimization

Minimizing the probability of error (the probability of misclassification)

$$L(g) = \mathbb{E}\{\chi_{\{g(X)\neq Y\}}\} = \mathbb{P}\{g(X) \neq Y\} \tag{2.16}$$

is only possible with the knowledge of the joint distribution of X and Y. An estimate of the probability of error of a classifier g given $\mathcal{T}_n$ is the average error count

$$L_n(g) = \frac{1}{n}\sum_{i=1}^{n} \chi_{\{g(\mathbf{x}_i)\neq y_i\}}. \tag{2.17}$$

The estimate $L_n(g)$ is called the *empirical error* of g. For a rather thorough survey of recent advances consult (Boucheron et al., 2005).

We can now approach the classification task by considering a class $\mathcal{C}$ of classifiers $g : \mathbb{R}^p \to \{-1, 1\}$. Given $\mathcal{T}_n$, choose the classifier in $\mathcal{C}$ that results in the least empirical error $L_n(g)$. However, the problem of minimizing the empirical error is computationally intractable. To deal with this issue, one needs to modify the functional to be minimized. We consider classifiers of the form

$$g_f(\mathbf{x}) = \begin{cases} -1 & \text{if } f(\mathbf{x}) < 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.18}$$

where $f : \mathbb{R}^p \to \mathbb{R}$ is a real-valued function in $\mathcal{F}$. The probability of error of g_f is thus written as

$$\begin{aligned} L(g_f) = L(f) &= \mathbb{P}\{\operatorname{sgn}(f(X)) \neq Y\} \\ &= \mathbb{P}\{Yf(X) \leq 0\} \\ &= \mathbb{E}\{\chi_{\{Yf(X)\leq 0\}}\}. \end{aligned} \tag{2.19}$$

The quantity $yf(\mathbf{x})$ is called the *margin*, and is a recurring theme in the rest of this chapter. Given $\mathcal{T}_n$, once can estimate $L(f)$ by $L_n(f)$

$$L_n(f) = \frac{1}{n}\sum_{i=1}^{n} \chi_{\{y_i f(\mathbf{x}_i)\leq 0\}} \tag{2.20}$$

where $\chi_{\{yf(\mathbf{x})\leq 0\}}$ is the *0-1 loss function*. As explained before, minimizing the empirical error is computationally intractable (Arora et al., 1993). Instead, we seek to minimize

Table 2.1: Well-known convex loss functions and their corresponding minimizing function.

Loss function name	Form of $\phi(v)$	Form of $f^*_\phi(\eta)$
Square loss	$(1-v)^2$	$2\eta - 1$
Hinge loss	$\max(0, 1-v)$	$\text{sign}(2\eta - 1)$
Squared hinge loss	$\max(0, 1-v)^2$	$2\eta - 1$
Logistic loss	$\ln(1+\exp(-v))$	$\ln \frac{\eta}{1-\eta}$

a smooth convex upper bound of the 0-1 loss: $\chi_{\{yf(\mathbf{x})\leq 0\}}$. The smooth convex function ϕ of the margin $v = yf(x)$, called the *cost function*. Examples include the exponential loss function, $\phi(v) = \exp(-v)$, employed in AdaBoost (Freund and Schapire, 1997) and the hinge loss function, $\phi(v) = \max(0, 1-v)$, used in SVM (Boser et al., 1992). The cost functional becomes

$$A(f) = \mathbb{E}\{\phi(Yf(X))\} \tag{2.21}$$

with its corresponding empirical form being

$$A_n(f) = \frac{1}{n}\sum_{i=1}^{n} \phi(y_i f(\mathbf{x}_i)). \tag{2.22}$$

Note that ϕ is an upper bound on the 0-1 loss and hence, $L(f) \leq A(f)$ and $L_n(f) \leq A_n(f)$.

If $\mathcal{F}$ consists of functions that are linear in their parameters—thus making them convex, then minimizing the empirical cost $A_n(f)$ is a convex optimization problem. Thus, there exist efficient algorithms for obtaining the minimum $A_n(f)$ over $f \in \mathcal{F}$.

The loss functions we are interested in are listed in Table 2.1 and are shown in Figure 2.2. Real-valued loss functions act as surrogates to the 0-1 loss. This leads to a regression problem. The result for classification is then obtained by thresholding the output of the resulting function from regression. In what follows, we go over the fact that optimizing a convex surrogate leads to the Bayes classifier.

Minimizing a convex upper bound of the 0-1 loss, not only leads to tractable classification, but also a successful one. The empirical success of SVM and boosting reinforces this point. More importantly, we can show that the minimizer f^* of $A_n(f)$ is such that the induced classifier g^*_f is the Bayes classifier.

To this aim, let us look at a closer look at the cost functional

$$\begin{aligned} A(f) &= \mathbb{E}\{\phi(Yf(X))\} \\ &= \mathbb{E}\Big\{\eta(X)\phi(f(X)) + (1-\eta(X))\phi(-f(X))\Big\} \end{aligned} \tag{2.23}$$

where $\eta(x)$ denotes the posterior probability $\mathbb{P}\{Y = 1|X = x\}$. Consider the following definition

$$A(f, \eta) = \eta\phi(f) + (1-\eta)\phi(-f) \tag{2.24}$$

where $f \in \mathbb{R}$ and $\eta \in [0, 1]$. Consider further that the function $f^*_\phi : [0, 1] \to \mathbb{R}$ is the minimizer of $A(f, \eta)$

$$f^*_\phi(\eta) = \operatorname*{argmin}_{f\in\mathbb{R}} A(f, \eta). \tag{2.25}$$

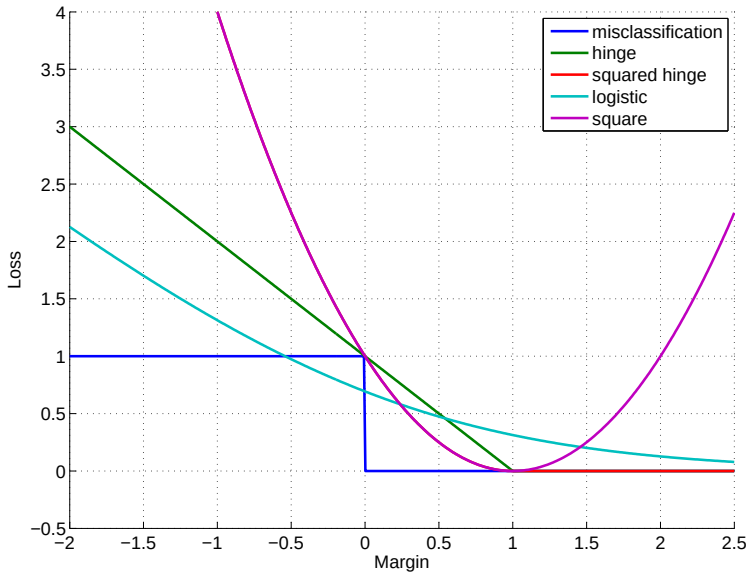


Figure 2.2: A comparison of convex loss functions. The misclassification loss is also shown.

By the definition of f_ϕ^* above, it is evident that $f_\phi^*(\eta(x))$ minimizes $A(f(x))$ in (2.21) among all functions $f(x)$. Given a convex loss function ϕ, the optimal minimizer f_ϕ^* is easy to compute. In the third column of Table 2.1 we list the optimal f_ϕ^* for the loss functions of interest. In these examples, it can readily be seen that $f_\phi^*(\eta) > 0$ only when $\eta > 1/2$. If we let $f(x) = f_\phi^*(\eta(x))$, thus minimizing (2.21), then the induced classifier

$$g^*(\mathbf{x}) = \begin{cases} -1 & \text{if } f^*(\mathbf{x}) < 0 \\ 1 & \text{otherwise} \end{cases}$$

has the same sign as the Bayes classifier. This allows us to conclude that the minimizer of the cost functional $A(f)$, an upper bound on the true classification error $L(f)$, is such that the induced classifier g_f is the Bayes classifier, thereby proving the *Fisher consistency* of convex cost functions (Zhang, 2004; Boucheron et al., 2005; Lin, 2002; Bartlett et al., 2003; Rosasco et al., 2004). Note that Fisher consistency in this context requires that the population minimizer f^* of the loss functional to have the same sign as the Bayes classifier, i.e., $\text{sign}(\eta - 1/2)$.

2.1.3 Estimation of the Posterior Probability

An important observation to be made based on Table 2.1 is that SVM estimates the binary classifier directly. That is, SVM estimates $\text{sign}(2\eta(x) - 1)$ directly, instead of the posterior probability $\eta(x)$. This means that SVM cannot inform us about the confidence of its predictions. Such information is especially useful for multiclass classification using a binary classifier in a one-versus-all setting. In a situation where the posterior probabilities for all classes are below $1/2$, SVM fails to make correct decisions.

On the other hand, the least squares classifier estimates the posterior probability $\eta(x)$ and can thus provide confidence information. For the least squares classifier, the

Table 2.2: A comparison of three other classifiers with the classification based on lasso on seven data sets. The hyperparameter is denoted by C or λ, depending on the algorithm. The number of nonzero entries in the solution vector is denoted by #nz. The percentage of correctly classified testing samples is denoted by Acc.

Data set	lasso			SVM			ℓ_1-reg L2SVM			ℓ_1-reg logreg		
	λ	Acc	#nz	C	Acc	#nz	C	Acc	#nz	C	Acc	#nz
australian	10	86	14	2	86	68	20	86	14	5	**87**	14
colon-cancer	1	77	16	1	**87**	11	10	75	112	10	83	91
diabetes	10	**80**	8	1	75	105	10	77	8	10	76	8
heart	6	**87**	13	1.5	84	30	10	80	13	5	83	13
ionosphere	1	77	16	1	**87**	11	10	75	28	2	82	31
liver-disorders	2	46	6	2	62	70	5	66	6	5	**67**	6
mushrooms	2	48	13	2	**100**	90	10	**100**	96	20	**100**	95

posterior probability estimate is $(f(x) + 1)/2$ truncated to $[0, 1]$. For more in-depth analysis of this subject see (Zhang, 2004).

2.2 Why does ℓ_1-regularization induce sparsity?

Figure 2.3, gives a good intuition for the reason that ℓ_1-regularization induces sparsity.

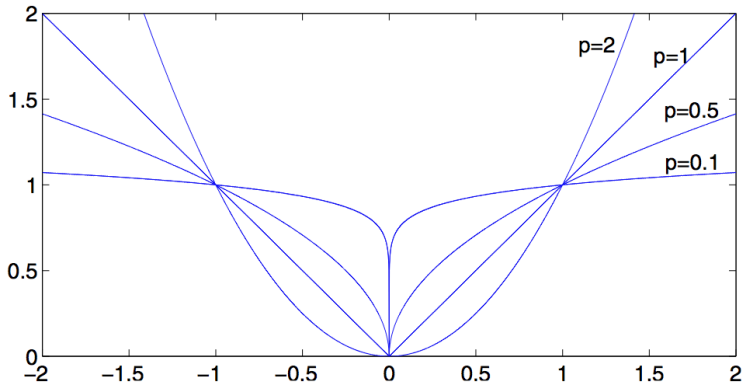


Figure 2.3: As p goes to 0, $|x|^p$ becomes the indicator function and $|\mathbf{x}|^p$ becomes a count of the nonzero entries in $\mathbf{x}$ (Bruckstein et al., 2009).

2.3 Empirical Evaluation of Lasso for Classification

In Table 2.2 we see the comparison of various methods for binary classification with the lasso. We see that lasso is able to obtain the sparsest solutions in higher-dimensional problems while maitaining reasonable accuracy. In Table 2.3, we see the results for ridge regression used as a classifier.

Table 2.3: Results for classification based on ridge regression. Note that the number of the number of nonzero entries of the solution equals the number of attributes of the observations.

Data set	ridge		
	λ	Acc	#nz
australian	30	86	14
colon-cancer	6	87	2000
diabetes	40	76	8
heart	9	86	13
ionosphere	10	74	34
liver-disorders	8	35	6
mushrooms	20	49	112

Chapter 3

On ℓ_1-regularized Square Loss Minimization for Reconstruction

> "Over the last several years, there has been an explosion of interest in alternatives to traditional signal representations. Instead of just representing signals as superpositions of sinusoids (the traditional Fourier representation) we now have available alternate dictionaries—collections of parameterized waveforms—of which the wavelets dictionary is only the best known. Wavelets, steerable wavelets, segmented wavelets, Gabor dictionaries, multiscale Gabor dictionaries, wavelet packets, cosine packets, chirplets, warplets, and a wide range of other dictionaries are now available."
>
> —Chen et al. (1995)

Sparse approximation schemes are known to be very useful for signal and feature representation (i.e., reconstruction). For example, there exists a successful multiclass classification algorithm based on sparse representations (Wright et al., 2009). In this chapter, we look at the reconstruction performance (in the unsupervised setting) of ℓ_1-regularized square loss minimization algorithms.

As we have seen in Chapter 2, ℓ_1-regularization is not as good as ℓ_2-regularization for generalization to unseen data. However, as experiments in sparse approximation show (Wright et al., 2009; Yang et al., 2009; Coates and Ng, 2011), ℓ_1-regularization quite suitable for feature representation. What is it that makes sparse approximation methods so successful at *feature learning and representation*? Is the success coming from the sparsity of the representation or the fact that it overfits the data? Some analysis has been done on the stability of ℓ_1-regularized algorithms (Poggio et al., 2009; Xu et al., 2012).

In the rest of this chapter we go over the application of ℓ_1-regularized square loss minimization for feature learning in an image classification task, especially, we go over the work of Coates and Ng (2011). We then review the classification algorithm by Wright et al. (2009), called sparse representation classification (SRC). Then we propose our new method for regression that is an extension of SRC to regression. We then argue that these approaches do not provide an improvement over much more efficient methods like k-means and kNN.

3.1 Sparse Coding and Dictionary Learning for Feature Learning

To explain the significance of ℓ_1-regularized square loss minimization for sparse coding, we go over a paper by Coates and Ng (2011). In the paper, "The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization", by "training" the authors mean learning the dictionary D, and by "encoding" they mean mapping the input x to feature f given the dictionary D. The authors argue that if we are able to break down each feature learning method into its comprising training and encoding routines, we can then combine the subroutines from different feature learning methods and get more efficient algorithms without sacrificing (and sometimes improving) performance on classification tasks. They report classification results using 5-fold cross-validation with linear SVM on CIFAR-10, NORB, and Caltech 101 (81.5%, 95%, and 72.6%, the first two being state-of-the-art results) (Coates and Ng, 2011). The authors experiment with six methods for populating the dictionary, and a few methods for encoding using the dictionary.

Here are the methods for learning/populating the dictionary (we are seeking dictionary $D \in \mathcal{R}^{n \times d}$ such that each atom (column) has unit ℓ_2-norm):

1. Sparse coding (SC) with coordinate descent:

$$\min_{D, s^{(i)}} \sum_i \|Ds^{(i)} - x^{(i)}\|_2^2 + \lambda \|s^{(i)}\|_1.$$

   One way to solve this optimization problem is to alternate minimization between the dictionary D and sparse codes $\{s^{(i)}\}$ (one is kept fixed while the objective function is minimized w.r.t. the other and so on). The authors obtain the parameter λ by minimizing its average cross-validation error over a grid of candid values.

2. Orthogonal matching pursuit (OMP-k):

$$\min_{D, s^{(i)}} \sum_i \|Ds^{(i)} - x^{(i)}\|_2^2 \tag{3.1}$$

$$\text{subject to } \|s^{(i)}\|_0 \leq k, \forall i, \tag{3.2}$$

   where k is an upper bound on the number of nonzero elements in $s^{(i)}$. To solve this optimization problem, one would alternate between minimizing D and $\{s^{(i)}\}$, just like above.

   - Coordinate descent and OMP are algorithms for obtaining sparse codes given a dictionary, the dictionary on the other hand, can be obtained using gradient descent.
   - Optimizing both 1. and 2., you get the sparse codes as a byproduct of learning the dictionary (the training and encoding phases are intertwined). However, this doesn't stop us from holding on only to the dictionary obtained in this step and computing the codes by other means.

3. Sparse restricted Boltzmann machine (RBM) and sparse auto-encoder.

4. Random downsampling of data matrix X containing normalized $x^{(i)}$

5. Random weights: One fills the dictionary with normalized columns sampled from the standard normal distribution.

And here are the methods for encoding:

1. SC: Same optimization problem as above, D fixed, possibly different λ, and setting the elements of feature f as

$$f_j = \max\{0, s_j\} \tag{3.3}$$
$$f_{j+d} = \max\{0, -s_j\}. \tag{3.4}$$

   Notice that instead of d dimensions, the feature f has $2d$ dimensions. The authors call this "polarity splitting".

2. OMP-k: Settings like 1.

3. Soft thresholding: For fixed threshold α, they assign f as follows,

$$f_j = \max\{0, D^{(j)T}x - \alpha\} \tag{3.5}$$
$$f_{j+d} = \max\{0, -D^{(j)T}x - \alpha\}. \tag{3.6}$$

4. The natural encoding: If the dictionary is learned with SC, then the already-learned codes are used. Same goes for OMP. For RBM and the autoencoder, one computes the activation at the hidden nodes using the logistic sigmoid function g:

$$f_j = g(W^{(j)}x + b) \tag{3.7}$$
$$f_{j+d} = g(-W^{(j)}x + b), \tag{3.8}$$

   where $W = D^{\mathsf{T}}$ and $W^{(j)}$ is the jth row of W. For 5. and 6., the authors use the dictionary as a linear map, i.e., $f = D^{\mathsf{T}}x$ (this is like random projection, except instead of decreasing dimensionality it increases it, assuming $d > n$).

The authors obtain the best result on CIFAR-10 by using OMP-1 for training, and soft thresholding for encoding (and showing that the wider the dictionary—e.g., d = 6000—the better). They achieve the best result for NORB using random patches as the dictionary, and SC for encoding. Same goes for Caltech 101, although this result trails behind the state-of-the-art by 3.1%.

Here is how the authors use the dataset in the unsupervised feature learning phase. In the case of CIFAR and NORB, they set $x^{(i)} \in \mathcal{R}^n$ to randomly-chosen, normalized, vectorized $(6 \times 6) \times 3$ patches. As for Caltech 101, the $x^{(i)}$ are the 128-dimensional SIFT descriptors extracted from each random 16×16 patch. Before sending it over to the dictionary training algorithm, they perform ZCA-whitening on the whole dataset $X = [x^{(1)}, \ldots, x^{(1600)}]$.

Given the feature mapping parameterized by D, here's an overview of the pipeline that the authors set up for performing classification (see Figure ??). First, they extract patches $\{x^{(i)}\}$ (with the size specified above) with a shift of one pixel for CIFAR-10 and NORB, and eight pixels for Caltech 101, covering the whole image. For CIFAR-10 and NORB, the $x^{(i)}$s are the raw pixel values for the patch, whereas for Caltech 101,

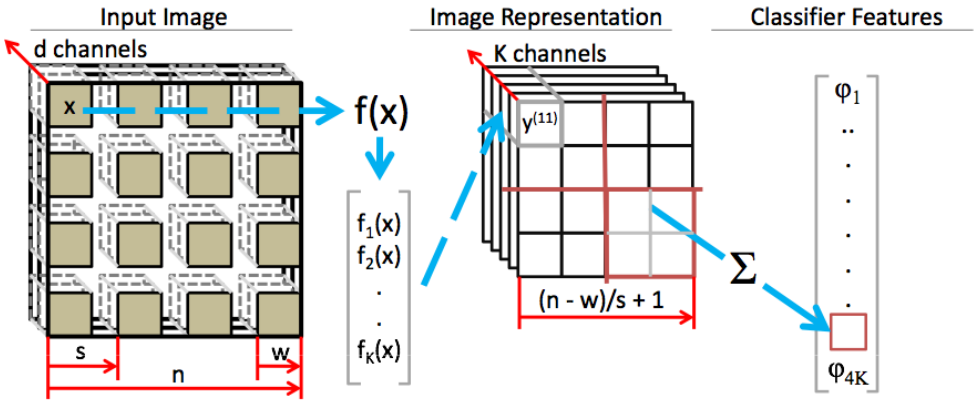


Figure 3.1: Image classification pipeline (Coates and Ng, 2011).

they are the values of the single SIFT descriptor extracted from the patch. For each pair of training/encoding method, the authors use the dictionary D to get feature $f^{(i)}$ for each $x^{(i)}$. So for example, for each 32×32 image in the CIFAR-10 dataset we get (given the settings specified) $27 \times 27 \times 1600 \times 2 = 2,332,800$ dimensions! To reduce the dimensionality of the feature space a pooling step is carried out (differently for each dataset):

- CIFAR-10: The authors average the feature values over the four quadrants of the image, yielding the final feature vector representing that image. (With pooling, we are down to $4 \times 1600 \times 2$ dimensions.)
- NORB: The authors perform two stages of downsampling on the original 108×108 images before extracting the patches. They do not mention their pooling strategy after the feature mapping is done.
- Caltech 101: Here, the authors perform "spatial pyramid" pooling. That is, they perform max-pooling on the features over 4×4, then 2×2, and 1×1 grids in a hierarchical manner. They concatenate the results to form the final feature vector representing the image.

Having thus obtained a single feature vector for each image in the test and training sets, the authors train a (actually many) linear SVM(s) to classify.

In my opinion, the results of this paper are interesting but not completely surprising. Consider PCA and random projection (although they do not result in wide dictionaries as the methods employed in this paper). We know that (in some tasks) random orthonormal weights are comparable to their "data-aware" and learned counterparts, i.e., the principal components. The results also partly explain why the k-means algorithm—employed in their preceding paper (Coates et al., 2011)—fared so well as the scheme for learning the atoms of the dictionary.

3.2 Sparse Representation Classification

To provide background for our proposed method for regression in the next section, we go over sparse representation classification (Wright et al., 2009). Sparse representation

classification (SRC) is a multiclass classification method. SRC requires each test sample to be reconstructed by only a few training samples. This is achieved by seeking sparse representation of each test sample with respect to the dictionary of training samples. In SRC, each sample is assigned a weight that accounts for the degree of its contribution in the reconstruction of a test sample. This information allows for a more informed decision on the class of a test sample.

Consider the training set $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, where $\mathbf{x}_i \in \mathbb{R}^p$ is a training feature vector and $y_i \in \{1, \ldots, C\}$, its corresponding class label. The feature vectors $\mathbf{x}_i, i = 1, \ldots, n$, make up the columns of the dictionary $\mathbf{D} \in \mathbb{R}^{p \times n}$. We assume that the dictionary is overcomplete ($p < n$), otherwise we would first perform dimensionality reduction on the features. Further, we assume that each column is normalized to unit norm. For each test point $\mathbf{x}$, we seek a sparse representation with respect to the dictionary of training points. Solving the following optimization problem results in a sparse representation $\mathbf{a}$ for the test point $\mathbf{x}$,

$$\min_{\mathbf{a} \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1. \tag{3.9}$$

Based on the values in the vector $\mathbf{a}$, a decision can be made about the class label of $\mathbf{x}$. Let $\mathcal{I}_c \in \{1, \ldots, n\}$ be the indices of the columns of $\mathbf{D}$ that correspond to points belonging to class c. Additionally, let the vector $\delta_c(\mathbf{a})$ be equal to $\mathbf{a}$ at the indices $\mathcal{I}_c$ and zero elsewhere. A test sample $\mathbf{x}$ is assigned to the same class as the samples whose linear combination best reconstructs $\mathbf{x}$ in the least squares sense, i.e.,

$$\hat{c} = \operatorname*{argmin}_{c \in \{1, \ldots, C\}} \|\mathbf{x} - \mathbf{D}\delta_c(\mathbf{a})\|_2. \tag{3.10}$$

In the regression setting, the outputs $y_i, i = 1, \ldots, n$ are real numbers. Let $\mathbf{y} = (y_1, \ldots, y_n)^\mathsf{T}$ be the vector containing the corresponding outputs for all of the training samples in $\mathbf{D}$. For a given test sample $\mathbf{x}$, after solving the optimization problem in (3.9), we predict its output $\hat{y}$, based on the following formula,

$$\hat{y} = \mathbf{y}^\mathsf{T}\mathbf{a}. \tag{3.11}$$

This is reminiscent of k-nearest neighbor regression, with the advantage of having the weight of the contribution of each sample provided in $\mathbf{a}$.

3.3 SPARROW: SPARse appROximation Weighted regression

Section 3.3 is based on published work in the following paper:

P. Noorzad and B. L. Sturm. Regression with Sparse Approximations of Data. In *Proceedings of the European Signal Processing Conference*, August 2012.

In this section we propose and study a new nonparametric method for local multivariate regression — sparse approximation weighted regression (SPARROW) — which employs the sparse approximation of a point in terms of the regressors. A similar nonparametric approach is k-nearest neighbor regression (k-NNR) (Härdle and Linton,

1994), which assumes that the k regressors nearest to a test point produce similar regressands. Both approaches can be considered variants of local polynomial kernel regression (LPKR) (Ruppert and Wand, 1994), which estimates the regression function at a point by fitting a polynomial at that point.

In addition to local methods like k-NNR and LPKR, considerable research has been aimed at global nonparametric methods, for example, additive models (AMs) (Buja et al., 1989), and sparse additive models (SpAM) (Ravikumar et al., 2009). In AMs, univariate methods are employed to estimate a smooth function of each regressor — in a model consisting of the sum of such univariate component functions — avoiding the need to deal directly with multidimensional inputs. In SpAM, the aim is to reduce the number of component functions of an additive model (Ravikumar et al., 2009). Projection pursuit regression (PPR) (Friedman and Stuetzle, 1981) is an extension to AMs that is able to model a more general class of functions.

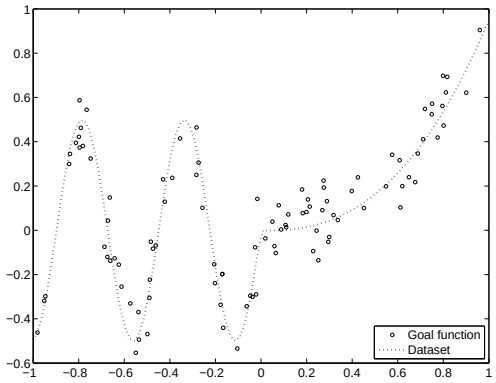
Although methods for global parametric and nonparametric regression might minimize the mean error over the entire dataset, it may not provide a good local fit. Figure 3.2, illustrates the ability of local methods in modeling data generated by an unknown distribution. Local methods like LPKR assume a local parametric model for the data (Cleveland and Devlin, 1988). In LPKR, one estimates the regression function at each point by fitting a Taylor polynomial about that point. This can produce models that are locally constant, locally linear, locally quadratic, etc., based on the order of the polynomial. Central to this procedure is the minimization of a weighted sum of squares error. Typically, the weights are defined by a decreasing function of the distance between two points. SPARROW defines these weights using the sparse approximation of the test point. Implicit in this is the assumption that a test point is better modeled by a sparse linear combination of the regressors than by its proximity to them.

The advantages of data modeling with sparsity constraints are well-documented (Chen et al., 1998; Elad, 2010; Mallat, 2009), e.g., in uncovering the physiological code of the mammalian primary visual cortex (Olshausen and Field, 1996), and in producing sparse codes of natural sounds (Lewicki, 2002), images (Yang et al., 2009), musical audio (Plumbley et al., 2010). Within the field of supervised learning, sparse representation classification (Wright et al., 2009) can outperform standard approaches in difficult settings, e.g., speech recognition in noise (Gemmeke et al., 2011), and face recognition with occlusions, misalignments, and illumination variation (Wright et al., 2009; Wagner et al., 2011). Sparsity has also been applied to variable selection, most notably in the LASSO (Tibshirani, 1996). In the next sections, we define SPARROW, and show how it is a variant of k-NNR and LPKR. Then we present several experimental results comparing SPARROW with these and other well-known approaches.

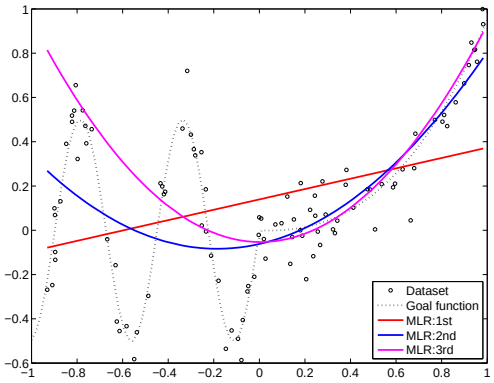
3.3.1 Sparse Approximation Weighted Regression

Consider a dataset (or dictionary) of N observations, $\mathcal{D} := \{(\mathbf{x}_i, y_i)\}_{i\in\Omega}$, where the input $\mathbf{x}_i = [x_{1i}, \ldots, x_{Mi}]^\mathsf{T} \in \mathbb{R}^M$ is associated with the output $y_i \in \mathbb{R}$. Let $\Omega := \{1, 2, \ldots, N\}$ index the dictionary. In nonparametric regression, one assumes $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $f(\mathbf{x})$ is an unknown but smooth function and ϵ_i is some error independent of $\mathbf{x}_i$. Given $\mathcal{D}$ and a point $\mathbf{z}$, SPARROW estimates the regression function $f(\mathbf{z})$ by a linear combination of the outputs

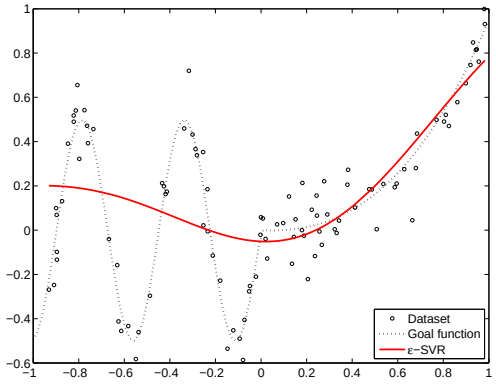
$$\hat{f}(\mathbf{z}) := \sum_{i\in\Omega} l_i(\mathbf{z}, \mathcal{D}) y_i \tag{3.12}$$



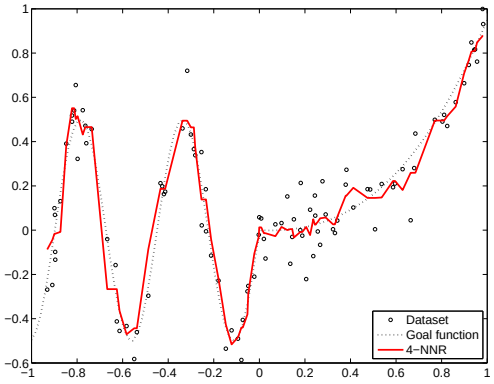
(a) Our generated data set.



(b) Multiple linear regression with first-, second-, and third-order terms.



(c) ϵ-support vector regression with an RBF kernel.



(d) 4-nearest neighbor regression.

Figure 3.2: These figures illustrate the ability of local regression methods to model data with an unknown distribution. The function generating the data is: $y_i = f(x_i) + \epsilon_i$, where $f(x) = (x^3 + x^2)\,\mathrm{I}(x) + \sin(x)\,\mathrm{I}(-x)$.

where $l_i(\mathbf{z}, \mathcal{D})$ is the ith *effective weight*, which SPARROW defines as a function of the sparse approximation of $\mathbf{z}$ in $\mathcal{D}$.

Instead of fitting a single model to the entire dataset, as in global parametric and nonparametric regression, SPARROW fits parametric models about each test point $\mathbf{z}$ by using, e.g., a zeroth, first-, or second-order Taylor expansion. We now discuss how SPARROW defines the effective weights in (3.12) to estimate the regression function at a given point.

3.3.2 Definition of Effective Weights

To obtain the local quadratic estimate of the regression function at $\mathbf{z}$, we can approximate $f(\mathbf{x})$ about $\mathbf{z}$ by a Taylor polynomial of degree two

$$f(\mathbf{x}) \approx f(\mathbf{z}) + (\mathbf{x}-\mathbf{z})^{\mathsf{T}}\boldsymbol{\theta}_{\mathbf{z}} + \frac{1}{2}(\mathbf{x}-\mathbf{z})^{\mathsf{T}}\mathbf{H}_{\mathbf{z}}(\mathbf{x}-\mathbf{z}) \tag{3.13}$$

with $\boldsymbol{\theta}_{\mathbf{z}} := \nabla f(\mathbf{z})$ the gradient of $f(\mathbf{x})$, and $\mathbf{H}_{\mathbf{z}} := \nabla^2 f(\mathbf{z})$ its Hessian, both evaluated at $\mathbf{z}$. The problem now is to find $f(\mathbf{z}), \theta_{\mathbf{z}}$ and $\mathbf{H}_{\mathbf{z}}$, such that we minimize the locally weighted squared error about $\mathbf{z}$ for all measurements in $\mathcal{D}$, i.e.,

$$\min_{f(\mathbf{z}),\boldsymbol{\theta}_{\mathbf{z}},\mathbf{H}_{\mathbf{z}}} \sum_{i\in\Omega} \alpha_i(\mathbf{z})\Big[y_i - f(\mathbf{z}) - (\mathbf{x}_i-\mathbf{z})^{\mathsf{T}}\boldsymbol{\theta}_{\mathbf{z}} - \frac{1}{2}(\mathbf{x}_i-\mathbf{z})^{\mathsf{T}}\mathbf{H}_{\mathbf{z}}(\mathbf{x}_i-\mathbf{z})\Big]^2 \tag{3.14}$$

where $\alpha_i(\mathbf{z})$ is the ith *observation weight*, which can be defined in several ways, e.g., by a kernel function (Hastie and Loader, 1993; Härdle and Linton, 1994), or by sparse approximation as done by SPARROW.

Now define the parameter supervector (Ruppert and Wand, 1994)

$$\boldsymbol{\Theta}_{\mathbf{z}} := \big[f(\mathbf{z}), \boldsymbol{\theta}_{\mathbf{z}}, \operatorname{vech}(\mathbf{H}_{\mathbf{z}})\big]^{\mathsf{T}} \tag{3.15}$$

where $\operatorname{vech}(\mathbf{H})$ denotes the half-vectorization of the symmetric $M \times M$ matrix, i.e., the $M(M+1)/2$-vector formed by stacking the diagonal and lower triangular entries of $\mathbf{H}_{\mathbf{z}}$. Define the diagonal matrix $\mathbf{A}_{\mathbf{z}}$ where its ith diagonal element is the observation weight $\alpha_i(\mathbf{z})$. By defining the matrix

$$\mathbf{X}_{\mathbf{z}} := \begin{bmatrix} 1 & (\mathbf{x}_1-\mathbf{z})^{\mathsf{T}} & \operatorname{vech}^{\mathsf{T}}[(\mathbf{x}_1-\mathbf{z})(\mathbf{x}_1-\mathbf{z})^{\mathsf{T}}] \\ \vdots & \vdots & \vdots \\ 1 & (\mathbf{x}_N-\mathbf{z})^{\mathsf{T}} & \operatorname{vech}^{\mathsf{T}}[(\mathbf{x}_N-\mathbf{z})(\mathbf{x}_N-\mathbf{z})^{\mathsf{T}}] \end{bmatrix} \tag{3.16}$$

we can express the minimization in (3.14) as

$$\min_{\boldsymbol{\Theta}_{\mathbf{z}}} \Big\| \mathbf{A}_{\mathbf{z}}^{1/2}\big[\mathbf{y} - \mathbf{X}_{\mathbf{z}}\boldsymbol{\Theta}_{\mathbf{z}}\big] \Big\|_2^2 \tag{3.17}$$

where the regressands vector $\mathbf{y} := [y_1, y_2, \ldots, y_N]^{\mathsf{T}}$. The parameters defined by the least-squares solution is (Ruppert and Wand, 1994)

$$\widehat{\boldsymbol{\Theta}}_{\mathbf{z}} = \big(\mathbf{X}_{\mathbf{z}}^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}\mathbf{X}_{\mathbf{z}}\big)^{-1}\mathbf{X}_{\mathbf{z}}^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}\mathbf{y} \tag{3.18}$$

provided $\mathbf{X}_{\mathbf{z}}^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}\mathbf{X}_{\mathbf{z}}$ is invertible. Finally, the local quadratic estimate of the regression function at $\mathbf{z}$ is just the first element of $\boldsymbol{\Theta}_{\mathbf{z}}$, i.e.,

$$\hat{f}(\mathbf{z}) = \mathbf{e}_1^{\mathsf{T}}\big(\mathbf{X}_{\mathbf{z}}^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}\mathbf{X}_{\mathbf{z}}\big)^{-1}\mathbf{X}_{\mathbf{z}}^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}\mathbf{y} = \sum_{i\in\Omega}\beta_i y_i \tag{3.19}$$

where $\mathbf{e}_1$ has a one in its first row, and zeros in all others. Hence, we see the ith effective weight in (3.12) is

$$l_i(\mathbf{z},\mathcal{D}) = \mathbf{e}_i^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}^{\mathsf{T}}\mathbf{X}_{\mathbf{z}}\big(\mathbf{X}_{\mathbf{z}}^{\mathsf{T}}\mathbf{A}_{\mathbf{z}}\mathbf{X}_{\mathbf{z}}\big)^{-1}\mathbf{e}_1 \tag{3.20}$$

In summary, SPARROW estimates the regression function at a point $\mathbf{z}$ by computing (3.12) with effective weights given by (3.20). If we use only the first column of $\mathbf{X}_\mathbf{z}$ in (3.20), we produce a locally constant estimate of $f(\mathbf{z})$, i.e.,

$$\hat{f}(\mathbf{z}) = (\mathbf{1}^\mathsf{T}\mathbf{A}_\mathbf{z}\mathbf{1})^{-1}\mathbf{1}^\mathsf{T}\mathbf{A}_\mathbf{z}\mathbf{y} = \frac{\sum_{i\in\Omega}\alpha_i(\mathbf{z})y_i}{\sum_{k\in\Omega}\alpha_k(\mathbf{z})}. \tag{3.21}$$

Using $M+1$ columns of $\mathbf{X}_\mathbf{z}$ produces a locally linear estimate. And using all of $\mathbf{X}_\mathbf{z}$ results in a locally quadratic estimate. Using higher order polynomials as the local parametric model reduces the bias of the estimate (Hastie and Loader, 1993; Ruppert and Wand, 1994), but this comes at the price of increased variance and computation time because the number of local parameters to be estimated increases exponentially. Additionally, higher order polynomials do not offer significant improvement over the quadratic model unless one seeks to estimate the gradient and the Hessian, i.e., $\boldsymbol{\theta}_\mathbf{z}$ and $\mathbf{H}_\mathbf{z}$ in (3.14) (Ruppert, 1996).

3.3.3 Definition of observation weights

Since the effective weights in (3.20) are a function of the observation weights in (3.14), i.e., $\{\alpha_i(\mathbf{z}) : i \in \Omega\}$, the remaining problem is to define the observation weights. If we define them in the locally constant model (3.21) by a kernel function, we produce the Nadaraya-Watson regression (NWR) estimate (Nadaraya, 1964). In this direction, we can define the weights by

$$\alpha_i(\mathbf{z}) := K\big(S(\mathbf{z},\mathbf{x}_i)/h\big) \tag{3.22}$$

where $K : \mathbb{R} \mapsto \mathbb{R}_+$ is a kernel function, $h > 0$ is the bandwidth, and $S(\mathbf{z},\mathbf{x}_i)$ is the distance

$$S(\mathbf{z},\mathbf{x}_i) := (\mathbf{z}-\mathbf{x}_i)^\mathsf{T}\mathbf{V}^{-1}(\mathbf{z}-\mathbf{x}_i) \tag{3.23}$$

where $\mathbf{V}$ is either a diagonal matrix of the unbiased estimates of the variances observed in the dimensions of the regressors in $\mathcal{D}$ (in which case (3.23) is the scaled Euclidean distance), or the unbiased estimate of the covariance of the regressors (in which case (3.23) is the Mahalanobis distance).

When we define the weights of the locally constant model

$$\alpha_i(\mathbf{z}) := \begin{cases} d(\mathbf{z},\mathbf{x}_i), & i \in N_k(\mathbf{z}) \subset \Omega \\ 0, & \text{else} \end{cases} \tag{3.24}$$

where $N_k(\mathbf{z})$ is the index set of the k nearest regressors of $\mathbf{z}$ in $\mathcal{D}$, then (3.21) produces k-NNR (Härdle and Linton, 1994). If $d(\mathbf{z},\mathbf{x}_i) := 1$, then the bandwidth of the constant kernel from $\mathbf{z}$ is at least as big as the largest distance between pairs of observations and $\mathbf{z}$, i.e., $h \geq \max_{i\in N_k(\mathbf{z})} S(\mathbf{z},\mathbf{x}_i)$. In weighted k-NNR (Wk-NNR), we define this weight as the reciprocal of the distance $d(\mathbf{z},\mathbf{x}_i) := 1/S(\mathbf{z},\mathbf{x}_i)$.

Contrary to NWR and k-NNR, SPARROW instead defines the observation weights from the sparse approximation of $\mathbf{z}$ in $\mathcal{D}$. First, consider the matrix form of the normalized regressors of the dictionary

$$\mathbf{D} := \left[\frac{\mathbf{x}_1}{\|\mathbf{x}_1\|_2}, \frac{\mathbf{x}_2}{\|\mathbf{x}_2\|_2}, \ldots, \frac{\mathbf{x}_N}{\|\mathbf{x}_N\|_2}\right]. \tag{3.25}$$

Table 3.1: Data set information. The last column indicates the tuned parameter k in the experiments involving k-NNR and Wk-NNR.

Data set	# observations (N)	# attributes (M)	k
abalone	4,177	8	9
bodyfat	252	14	4
housing	506	13	2
mpg	392	7	4

For an input $\mathbf{z}$, SPARROW finds a solution to $\mathbf{z} \approx \mathbf{D}\mathbf{s}$ such that $\mathbf{s} = [s_1, s_2, \ldots, s_N]^\mathsf{T}$ has many zero elements. There are a variety of ways to produce sparse approximations (see (Bruckstein et al., 2009; Tropp and Wright, 2010; Elad, 2010) for reviews). In this work, we use the principle of basis pursuit denoising (BPDN) (Chen et al., 1995), which poses the problem

$$\min_{\mathbf{s}\in\mathbb{R}^N} \|\mathbf{s}\|_1 \quad \text{subject to} \quad \frac{\|\mathbf{z} - \mathbf{D}\mathbf{s}\|_2^2}{\|\mathbf{z}\|_2^2} \leq \epsilon^2 \tag{3.26}$$

where $\epsilon^2 > 0$ limits the signal to approximation error ratio. Finally, SPARROW defines the ith observation weight using the sparse approximation weights

$$\alpha_i(\mathbf{z}) := \left[\frac{S(\mathbf{z}, \mathbf{x}_i)}{\min_{j\in\Omega} S(\mathbf{z}, \mathbf{x}_j)}\right]^{-1} \frac{s_i}{\|\mathbf{z}\|_2} \tag{3.27}$$

where s_i is the ith element of $\mathbf{s}$. The purpose of the first coefficient is to weight by 1 the regressand of the regressor closest to $\mathbf{z}$; and the purpose of dividing the sparse approximation weight by $\|\mathbf{z}\|_2$ is remove the influence of its length. Thus, as for Wk-NNR, SPARROW weights more heavily an observation closer to the query, but unlike Wk-NNR, only if it has a nonzero coefficient in its sparse approximation with $\mathcal{D}$. When we substitute the weights $\alpha_i(\mathbf{z})$ from (3.27) into (3.21) we obtain the constant SPARROW (C-SPARROW) estimate. And when we use these weights in (3.20), but use only the first $M + 1$ columns of $\mathbf{X}_\mathbf{z}$, (3.12) produces the linear SPARROW (L-SPARROW) estimate. Using all columns of $\mathbf{X}_\mathbf{z}$ produces the quadratic SPARROW (Q-SPARROW) estimate.

3.4 Empirical Evaluation of SPARROW

We now compare the performance of SPARROW against several other other well-established methods for local regression. In all cases, we use the standardized Euclidean distance in (3.23). We test NWR and its linear counterpart, local linear kernel regression (LLKR) (Härdle and Linton, 1994; Hastie and Loader, 1993), which solves (3.18) using the first $M + 1$ columns of $\mathbf{X}_\mathbf{z}$ in (3.16). For both NWR and LLKR we adopt the Gaussian kernel in (3.22)

$$K(x) := \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \tag{3.28}$$

We also test k-NNR, Wk-NNR (Härdle and Linton, 1994), for which we tune k by nested cross-validation. For a baseline, we test the global parametric approach of multiple linear

regression (MLR) (Hastie et al., 2009), which assumes a linear form of the regression function

$$f(\mathbf{x}) = [1, \mathbf{x}^\mathsf{T}]\mathbf{b} \tag{3.29}$$

and $\mathbf{b}$ is defined to minimize the mean squared error

$$\mathbf{b} = \underset{\mathbf{b}' \in \mathbb{R}^{M+1}}{\operatorname{argmin}} \|\mathbf{y} - [\mathbf{1}\ \mathbf{X}^\mathsf{T}]\mathbf{b}\|_2^2 \tag{3.30}$$

where the ith column of $\mathbf{X}$ is $\mathbf{x}_i$. To produce the sparse approximation for a test point in (3.26), we use the Spectral Projected Gradient Method for ℓ_1-minimization (SPGL1) (van den Berg and Friedlander, 2008), with at most 20 iterations, and $\epsilon := 10^{-6}$.

We use four different datasets commonly used in regression (see Table 3.1).[1] Except for bodyfat, we standardize each dataset such that its dimensions are zero-mean and have the same variance. Figure 3.3 shows the mean squared error (MSE) estimates of these algorithms from 10 independent trials of 10-fold cross-validation. We see that while MLR performs well for bodyfat and abalone, it performs poorly for mpg and housing. On the other hand, we see that LLKR does extremely well for all datasets. This gain in performance comes with an increase in computation as LLKR must compute (3.18). Except for abalone and housing, we see that C-SPARROW performs nearly the same as k-NNR and Wk-NNR. For housing, C-SPARROW appears to be almost as good as LLKR. This is surprising since, 1) C-SPARROW makes no assumption of the number of neighbors to be used for each test point, and 2) it is constructing a local constant estimation.

Table 3.2 shows the performance of L-SPARROW as compared to C-SPARROW. One might expect that L-SPARROW would perform better than C-SPARROW since it is a higher-order model. However, a problem with local polynomial regression for higher order polynomials (i.e., first- and second-order) is that when the input is locally rank deficient, the solutions to (3.18) become unstable. We resolve the problem by solving a regularized form of the weighted least squares optimization in (3.14). We use the ℓ_2-norm of the local parameters as the regularization term thus solving a ridge regression problem (Hoerl and Kennard, 1970), i.e., instead of solving (3.17), we solve

$$\min_{\boldsymbol{\Theta}_\mathbf{z}, \lambda} \left\| \mathbf{A}_\mathbf{z}^{1/2} \left[\mathbf{y} - \mathbf{X}_\mathbf{z} \boldsymbol{\Theta}_\mathbf{z} \right] \right\|_2^2 + \lambda \|\boldsymbol{\Theta}_\mathbf{z}\|_2^2 \tag{3.31}$$

where $\lambda \geq 0$ is the *ridge parameter*. For a given λ, the solution becomes (Hastie et al., 2009)

$$\widehat{\boldsymbol{\Theta}}(\mathbf{z}) = \left(\mathbf{X}_\mathbf{z}^\mathsf{T} \mathbf{A}_\mathbf{z} \mathbf{X}_\mathbf{z} + \lambda \mathbf{I} \right)^{-1} \mathbf{X}_\mathbf{z}^\mathsf{T} \mathbf{A}_\mathbf{z} \mathbf{y}. \tag{3.32}$$

We tune λ in the same way as we do k, described above. Nevertheless, while we see the performance of L-SPARROW improve with respect to using (3.18), it remains inferior to C-SPARROW.

3.4.1 Conclusion

In this work, we have proposed an adaptive variation of local polynomial regression methods: NWR, LLKR, k-NNR and Wk-NNR. NWR and LLKR use the entire dataset,

[1] mpg, abalone and housing are from `http://archive.ics.uci.edu/ml/`; bodyfat is from `http://lib.stat.cmu.edu/datasets/`.

Table 3.2: A comparison of the MSE estimates obtained on four data sets by 10 trials of 10-fold cross-validation of C-SPARROW and L-SPARROW with and without regularization. The last column denotes the ridge parameter used to obtain the L-SPARROW estimate.

Data set	C-SPAR.	L-SPAR. w/ R.	L-SPAR.	λ
abalone	5	16	988	10^{-3}
bodyfat	5×10^{-5}	35×10^{-5}	960×10^{-5}	10^{-6}
housing	10	45	4304	10^{-4}
mpg	7	8	6335	10^{-3}

Table 3.3: A comparison of the accuracy obtained by kNN and SRC on five multiclass classification data sets.

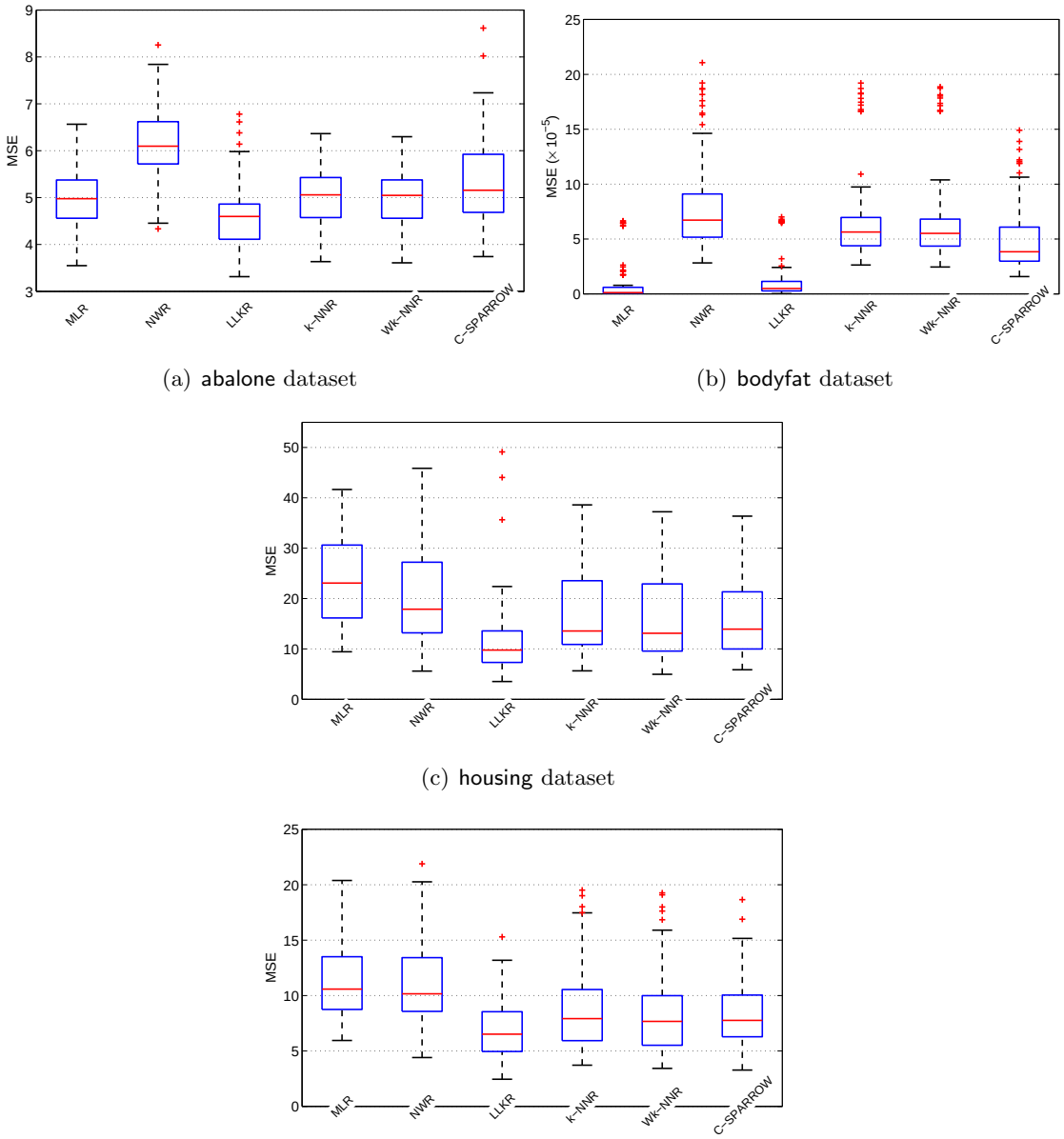
Data set	n	p	#classes	k	kNN	SRC
dna	2000	180	3	125	86	86
glass	214	9	6	2	70	65
iris	150	4	3	6	95	72
vowel	528	10	11	2	94	84
wine	178	13	3	7	97	99

and weight the regressand of each regressor by a kernel function. Alternatively, k-NNR and Wk-NNR use the regressands of the k regressors closest to a point, to locally estimate the regression function. With SPARROW, we propose using sparse approximation to adaptively select which regressors to use, and the weights of their regressands to estimate the regression function at a given point. Our experiments show that constant SPARROW can be a competitive regression algorithm. Our future work will analyze the situations where it makes sense to describe data as a linear combination (including negative weights) of labeled data. Furthermore, one can use other sparse approximation algorithms, such as greedy approaches, which are typically less computationally expensive than convex optimization approaches like BPDN.

3.5 Comparing kNN with SRC

In Table 3.3, we present a comparison of the k-nearest neighbor classification algorithm and sparse representation classification on multiclass classification data sets. The data sets were chosen from the LIBSVM website for multiclass data sets[2]. We see that sparse representation classification does not offer significant improvement over k-nearest neighbor classification on most data sets.

[2] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/multiclass.html



(a) abalone dataset

(b) bodyfat dataset

(c) housing dataset

(d) mpg dataset

Figure 3.3: Boxplots for 10-fold cross-validation estimate of mean squared error (100 independent runs) for four different datasets. Each box delimits 25 to 75 percentiles, and the red line marks the median. Extrema are marked by whiskers, and outliers by pluses.

Chapter 4

An Equivalence Between ϵ-SVR and BPDN

> "In ϵ-SV regression (Vapnik, 1995), our goal is to find a function $f(x)$ that has at most ϵ deviation from the actually obtained targets y_i for all the training data, and at the same time is as flat as possible. In other words, we do not care about errors as long as they are less than ϵ, but will not accept any deviation larger than this. This may be important if you want to be sure not to lose more than ϵ money when dealing with exchange rates, for instance."
>
> —Smola and Schölkopf (2004)

In the paper entitled, "An Equivalence Between Sparse Approximation and Support Vector Machines", Girosi derives the (kernel) support vector regression (SVR) quadratic programming problem in the framework of regularization theory. This is in contrast to Vapnik's original derivation of SVR using the structural risk minimization principle of statistical learning theory. Girosi shows that under minor provisions and assumptions, the technique of basis pursuit de-noising (BPDN) yields the same quadratic programming problem. Chen, Donoho, and Saunders proposed BPDN as a tractable technique for sparse approximation.

4.1 Reproducing Kernel Hilbert Space

Consider $\mathcal{H}$ as an inner product space on a set $\mathcal{X}$ over $\mathbb{R}$. If $\mathcal{H}$ is complete with respect to the metric induced by the inner product, then $\mathcal{H}$ is a Hilbert space. We denote the inner product of the vectors $f, g \in \mathcal{H}$ by $\langle f, g \rangle$.

Definition 4.1.1. *We say that $\mathcal{H}$ is a reproducing kernel Hilbert space if for every $x \in \mathcal{X}$, the linear functional $F_x : \mathcal{H} \to \mathbb{R}$, where $F_x f = f(x)$, is bounded*

Since F_x is a bounded linear functional, from the Riesz representation theorem it follows that for every $x \in \mathcal{X}$, there exists a unique $K_x \in \mathcal{H}$, with the following reproducing property

$$F_x[f] = \langle K_x, f \rangle = f(x) \tag{4.1}$$

where f is a function in $\mathcal{H}$. For $t \in \mathcal{X}$, K_t is also in $\mathcal{H}$ and by the reproducing property in Eq. (4.1) it follows that

$$K_t(x) = \langle K_x, K_t \rangle. \tag{4.2}$$

We define the function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for all $y \in \mathcal{X}$ as

$$K(x, y) = K_y(x) \tag{4.3}$$

and call it the reproducing kernel (RK) of $\mathcal{H}$. It follows from the uniqueness of K_y that K is determined entirely by $\mathcal{H}$. The following theorem establishes the relationship between an RK and its corresponding reproducing kernel Hilbert space (RKHS):

Theorem 4.1.2. *For every RKHS there exists a unique symmetric positive definite function (called the RK), and conversely, for every symmetric, positive definite function K on $\mathcal{X} \times \mathcal{X}$, there exists a unique RKHS of functions on $\mathcal{X}$ with K as its RK.*

The proof can be found in the first chapter of Wahba's book on splines. We only go over what the proof constructs. In particular, if $\mathcal{H}$ is an RKHS, then the RK is

$$K(x, y) = \langle K_x, K_y \rangle \tag{4.4}$$

where for each $x, y \in \mathcal{X}$, K_x and K_y are the unique representer functions. Conversely, given K, for all $x, y \in \mathcal{X}$ we define the representer function as

$$K_x(y) = K(x, y). \tag{4.5}$$

We construct the RKHS as the completion of the space of functions $\mathcal{H}_0$ spanned by $\{K_x : x \in \mathcal{X}\}$, with the inner product defined as

$$\langle f, g \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i \beta_j K(x_i, y_j) \tag{4.6}$$

where f and g are functions in $\mathcal{H}_0$,

$$f = \sum_{i=1}^{m} \alpha_i K_{x_i} \tag{4.7}$$

$$g = \sum_{j=1}^{n} \beta_j K_{y_j}. \tag{4.8}$$

From these definitions we derive

$$K(x, x_i) = \langle K_x, K_{x_i} \rangle, \tag{4.9}$$

and also

$$\begin{aligned} \langle K_x, f \rangle &= \Big\langle K_x, \sum_{i=1}^{m} \alpha_i K_{x_i} \Big\rangle \\ &= \sum_{i=1}^{m} \alpha_i \langle K_x, K_{x_i} \rangle \\ &= \sum_{i=1}^{m} \alpha_i K(x, x_i) \\ &= \sum_{i=1}^{m} \alpha_i K_{x_i}(x) \\ &= f(x). \end{aligned} \tag{4.10}$$

This last equation shows that the kernel is the representer of evaluation.

So far, we know that an RKHS $\mathcal{H}$ has a unique reproducing kernel K. Since K is a positive definite function, it has a spectral decomposition of the form

$$K(x,y) = \sum_{n=1}^{\infty} \lambda_n \phi_n(x)\phi_n(y) \tag{4.11}$$

where $\phi_1, \phi_2, \ldots$ is an orthonormal sequence of eigenfunctions in $L_2[\mathcal{X}]$, and $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$, the corresponding eigenvalues. The eigenfunctions of the RK span its RKHS $\mathcal{H}$. Hence, every function in $\mathcal{H}$ can be represented as

$$\hat{f}(x) = \sum_{n=1}^{\infty} c_n \phi_n(x) \tag{4.12}$$

and has the following norm

$$\|\hat{f}\|_{\mathcal{H}}^2 = \langle \hat{f}, \hat{f} \rangle_{\mathcal{H}} = \sum_{n=1}^{\infty} \frac{c_n^2}{\lambda_n}. \tag{4.13}$$

Functionals of the form in (4.13), known as smoothness functionals, associate smaller values to smoother functions.

4.2 Support Vector Machines for Regression

We have a data set, $D = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1, \ldots, l\}$, obtained by random sampling (in the absence of noise) of an unknown function f. Our goal is to recover f, or an estimate thereof, from D. This regression problem has many solutions because many functions pass through a given set of points. We constrain the problem by assuming that among all interpolating functions, the solution to our problem is the most smooth (where smoothness entails close points to have close values). Let $\Phi[\hat{f}]$ be a smoothness functional. In ϵ-SVR, our goal is to find a function $\hat{f}$ with at most ϵ deviation from the training data, i.e., $|\hat{f}(\mathbf{x}_i) - y_i| \leq \epsilon$, for $i = 1, \ldots, l$. The optimization problem is

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\Phi[\hat{f}] \\ &\text{subject to} && \begin{cases} y_i - \hat{f}(\mathbf{x}_i) \leq \epsilon \\ \hat{f}(\mathbf{x}_i) - y_i \leq \epsilon \end{cases} \qquad \text{for } i = 1, \cdots, l. \end{aligned} \tag{4.14}$$

We introduce slack variables ζ_i, ζ_i^* to deal with the possible infeasibility of the constrained optimization problem in (4.14). The optimization problem becomes

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\Phi[\hat{f}] + C\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \\ &\text{subject to} && \begin{cases} y_i - \hat{f}(\mathbf{x}_i) & \leq \epsilon + \zeta_i \\ \hat{f}(\mathbf{x}_i) - y_i & \leq \epsilon + \zeta_i^* \\ \zeta_i & \geq 0 \\ \zeta_i^* & \geq 0 \end{cases} \qquad \text{for } i = 1, \cdots, l \end{aligned} \tag{4.15}$$

where the free parameter $C > 0$, controls the tradeoff between the smoothness of $\hat{f}$ and the amount of deviation of $\hat{f}$ — beyond ϵ — from the training data. The problem in (4.15) is a convex programming problem and therefore has a unique minimum. By introducing Lagrange multipliers (dual variables) to add the constraints to the objective function, we form the Lagrangian of the problem in (4.15)

$$\begin{aligned}\mathcal{L}(\hat{f}, \boldsymbol{\zeta}, \boldsymbol{\zeta}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \mathbf{r}, \mathbf{r}^*) = \frac{1}{2}\Phi[\hat{f}] + C\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) + \sum_{i=1}^{l}\alpha_i^*(y_i - \hat{f}(\mathbf{x}_i) - \epsilon - \zeta_i^*) \\ + \sum_{i=1}^{l}\alpha_i(\hat{f}(\mathbf{x}_i) - y_i - \epsilon - \zeta_i) - \sum_{i=1}^{l}(r_i\zeta_i + r_i^*\zeta_i^*)\end{aligned} \tag{4.16}$$

such that $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^*$, $\mathbf{r}$, and $\mathbf{r}^*$ satisfy nonnegativity constraints. The Lagrangian in (4.16) has a saddle point at the optimal solution. Therefore, the optimization involves minimizing (4.16) with respect to the primal variables $\hat{f}$, $\boldsymbol{\zeta}$, and $\boldsymbol{\zeta}^*$, and maximizing with respect to the dual variables $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}^*$, $\mathbf{r}$, and $\mathbf{r}^*$.

Notice that up to now, we haven't considered any structure for $\hat{f}$ or the smoothing function. But as Cucker and Smale say

> *Learning processes do not take place in a vacuum.*

In order to find the function $\hat{f}$, we need to specify the hypothesis space we are considering for our search. Here, we make the assumption that the interpolating function $\hat{f}$ belongs to an RKHS $\mathcal{H}$. Therefore, it can be represented as in (4.12), with its norm having the form in (4.13). However, following Vapnik's derivation of ϵ-SVR, we consider an explicit bias for $\hat{f}$, that is

$$\hat{f}(\mathbf{x}) = \sum_{n=1}^{\infty} c_n\phi_n(\mathbf{x}) + b. \tag{4.17}$$

Substituting (4.17) for $\hat{f}$ and (4.13) for $\Phi[\hat{f}]$, the Lagrangian becomes

$$\begin{aligned}\mathcal{L}(b, \mathbf{c}, \boldsymbol{\zeta}, \boldsymbol{\zeta}^*; \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \mathbf{r}, \mathbf{r}^*) = \frac{1}{2}\sum_{n=1}^{\infty}\frac{c_n^2}{\lambda_n} + C\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \\ + \sum_{i=1}^{l}\alpha_i^*\Big(y_i - \sum_{n=1}^{\infty}c_n\phi_n(\mathbf{x}_i) - b - \epsilon - \zeta_i^*\Big) \\ + \sum_{i=1}^{l}\alpha_i\Big(\sum_{n=1}^{\infty}c_n\phi_n(\mathbf{x}_i) + b - y_i - \epsilon - \zeta_i\Big) \\ - \sum_{i=1}^{l}(r_i\zeta_i + r_i^*\zeta_i^*).\end{aligned} \tag{4.18}$$

To get the dual objective function, we need to minimize the Lagrangian with respect to the primal variables and eliminate them by substitution. We require that the partial

derivatives of $\mathcal{L}$ with respect to the primal variables $\hat{f}$ (b and c_n), ζ_i, and ζ_i^* are zero

$$\frac{\partial \mathcal{L}}{\partial c_n} = 0 \implies c_n = \lambda_n \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)\phi_n(\mathbf{x}_i) \tag{4.19}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \implies \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) = 0 \tag{4.20}$$

$$\frac{\partial \mathcal{L}}{\partial \zeta_i} = 0 \implies r_i = C - \alpha_i \tag{4.21}$$

$$\frac{\partial \mathcal{L}}{\partial \zeta_i^*} = 0 \implies r_i^* = C - \alpha_i^* \tag{4.22}$$

We substitute (4.19) in our model for the interpolating function

$$\begin{aligned}
\hat{f}(\mathbf{x}) &= \sum_{n=1}^{\infty} c_n \phi_n(\mathbf{x}) + b \\
&= \sum_{n=1}^{\infty} \Big(\lambda_n \sum_{i=1}^{l} (\alpha_i^* - \alpha_i)\phi_n(\mathbf{x}_i) \Big) \phi_n(\mathbf{x}) + b \\
&= \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) \sum_{n=1}^{\infty} \lambda_n \phi_n(\mathbf{x}_i)\phi_n(\mathbf{x}) + b \\
&= \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) K(\mathbf{x}, \mathbf{x}_i) + b
\end{aligned} \tag{4.23}$$

where the last equality follows from (4.11). Similarly, we substitute (4.19) in our equation for the norm

$$\begin{aligned}
\|\hat{f}\|^2 &= \sum_{n=1}^{\infty} \frac{c_n^2}{\lambda_n} \\
&= \sum_{n=1}^{\infty} \lambda_n \Big(\sum_{i=1}^{l} (\alpha_i^* - \alpha_i)\phi_n(\mathbf{x}_i) \Big) \Big(\sum_{j=1}^{l} (\alpha_j^* - \alpha_j)\phi_n(\mathbf{x}_j) \Big) \\
&= \sum_{n=1}^{\infty} \lambda_n \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\phi_n(\mathbf{x}_i)\phi_n(\mathbf{x}_j) \\
&= \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \sum_{n=1}^{\infty} \lambda_n \phi_n(\mathbf{x}_i)\phi_n(\mathbf{x}_j) \\
&= \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(\mathbf{x}_i, \mathbf{x}_j).
\end{aligned} \tag{4.24}$$

Substituting (4.21), (4.22), (4.23), and (4.24) in the Lagrangian we obtain

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) &= \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(\mathbf{x}_i, \mathbf{x}_j) + C\sum_{i=1}^{l}(\zeta_i + \zeta_i^*) \\ &+ \sum_{i=1}^{l}\alpha_i^*\Big(y_i - \sum_{j=1}^{l}(\alpha_i^* - \alpha_i)K(\mathbf{x}_i, \mathbf{x}_j) - b - \epsilon - \zeta_i^*\Big) \\ &+ \sum_{i=1}^{l}\alpha_i\Big(\sum_{j=1}^{l}(\alpha_i^* - \alpha_i)K(\mathbf{x}_i, \mathbf{x}_j) + b - y_i - \epsilon - \zeta_i\Big) \\ &- \sum_{i=1}^{l}(C - \alpha_i)\zeta_i - \sum_{i=1}^{l}(C - \alpha_i^*)\zeta_i^* \\ &= -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(\mathbf{x}_i, \mathbf{x}_j) - \epsilon\sum_{i=1}^{l}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{l}y_i(\alpha_i^* - \alpha_i).\end{aligned} \tag{4.25}$$

Given (4.21) and (4.22) and the fact that the dual variables satisfy nonnegativity constraints, we have the constraint, $0 \leq \alpha_i, \alpha_i^* \leq C$, for $i = 1, \cdots, l$. Thus, the dual problem has the form

$$\begin{aligned}&\text{minimize} && -\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) \\ &\text{subject to} && \begin{cases} 0 \leq \alpha_i, \alpha_i^* \leq C & \text{for } i = 1, \cdots, l \\ \sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0 & \\ \alpha_i\alpha_i^* = 0 & \text{for } i = 1, \cdots, l \end{cases}\end{aligned} \tag{4.26}$$

where instead of maximizing $\mathcal{L}$, we're minimizing $-\mathcal{L}$. The last constraint is automatically satisfied, otherwise, it would imply the existence of nonzero slacks in both directions. We include it for our comparisons later.

Generally, in a convex minimization problem, the primal objective function is always greater than or equal to the dual objective function at feasible primal and dual variable values. The difference is called the duality gap. For the quadratic programming problem in (4.26), it is shown that the duality gap is zero. Therefore, we find the optimal solution by solving the dual problem. This means that for the regularized minimization problem in (4.15), minimizing over the space of Hilbert functions amounts to minimizing over $\mathbb{R}^l$. This result is consistent with Kimeldorf and Wahba's representer theorem (Kimeldorf and Wahba, 1971).

4.3 ϵ-SVR and Sparsity

At a local minimum of a constrained optimization problem, the Karush-Kuhn-Tucker conditions hold. Among them is the condition of complementary slackness, stating that at a solution $\mathbf{x}_i$, the product between constraints and dual variables is zero. For the problem in (4.26), two of these conditions are

$$\begin{aligned}\alpha_i(f(\mathbf{x}_i) - y_i - \epsilon - \zeta_i) &= 0 \\ \alpha_i(y_i - f(\mathbf{x}_i) - \epsilon - \zeta_i) &= 0.\end{aligned} \tag{4.27}$$

According to (4.27), when $|f(\mathbf{x}_i) - y_i| < \epsilon$, α_i and α_i^* are forced to be zero. When $|f(\mathbf{x}_i) - y_i| \geq \epsilon$, α_i or α_i^* may be nonzero. Points with nonzero α_i or α_i^* are called support vectors. By increasing the free parameter ϵ, we decrease the number of support vectors, thereby increasing the sparsity of the solution in (4.23). The effect of the free parameter C on the sparsity of the solution can only be shown empirically. When there is no noise in the data, such as in our current problem setting, the optimal value for C is infinity. Therefore ϵ is the only free parameter of this formulation (not counting the kernel parameters).

4.4 Connection to Sparse Approximation

In sparse approximation, our goal is to approximate an unknown function f with a linear combination of a fixed set Φ of functions

$$\hat{f}(\mathbf{x}; \mathbf{a}) = \sum_{i=1}^{n} a_i \phi_i(\mathbf{x}) \tag{4.28}$$

where $\Phi = \{\phi_i(\mathbf{x}) : i = 1, \ldots, n\}$ is called the dictionary. The dictionary is usually overcomplete. This implies that $\mathbf{a}$ in (4.28) is not unique since some elements of Φ are linear combinations of other elements. We constrain the problem by requiring that $\mathbf{a}$ be the sparsest solution: the solution with the minimum number of nonzero elements. The following cost function represents a formulation of this problem

$$E(\mathbf{a}) = \frac{1}{2}\|f(\mathbf{x}) - \sum_{i=1}^{n} a_i \phi_i(\mathbf{x})\|_{L_2}^2 + \lambda \|\mathbf{a}\|_{L_0}^p \tag{4.29}$$

where $\|.\|_{L_0}$ counts the number of nonzero elements of a vector, and $\|.\|_{L_2}$ is the L_2 norm. However, because of the L_0 "norm", minimizing the cost function in (4.29) is NP-hard. To deal with the intractability of (4.29), Chen et al. (1995) proposed to minimize the following convex cost function

$$E(\mathbf{a}) = \frac{1}{2}\|f(\mathbf{x}) - \sum_{i=1}^{n} a_i \phi_i(\mathbf{x})\|_{L_2}^2 + \lambda \|\mathbf{a}\|_{L_1} \tag{4.30}$$

and called it basis pursuit de-noising. The authors used the L_1 norm as an approximation to the L_0 norm.

We assign the dictionary functions by the help of the reproducing kernel K of an RKHS $\mathcal{H}$, that is,

$$\phi_i(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}_i), \quad i = 1, \ldots, l \tag{4.31}$$

where $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, l\}$ is the dataset obtained by sampling f in the absence of noise. Further, we replace the L_2 criterion in (4.30) with the $\mathcal{H}$ norm, and set ϵ as the regularization parameter. We get the following cost function

$$E(\mathbf{a}) = \frac{1}{2}\|f(\mathbf{x}) - \hat{f}(\mathbf{x}, \mathbf{a})\|_{\mathcal{H}}^2 + \epsilon \|\mathbf{a}\|_{L_1} \tag{4.32}$$

where the approximating function is

$$\hat{f}(\mathbf{x}, \mathbf{a}) = \sum_{i=1}^{l} a_i K(\mathbf{x}, \mathbf{x}_i). \tag{4.33}$$

Assuming that the target function f has zero mean in $\mathcal{H}$, i.e., it's projection on the constant function is zero (we're not assuming that the constant function is in $\mathcal{H}$)

$$\langle f, 1\rangle_{\mathcal{H}} = 0 \tag{4.34}$$

we require that the approximating function to also have zero mean in $\mathcal{H}$

$$\langle \hat{f}, 1\rangle_{\mathcal{H}} = 0. \tag{4.35}$$

For this, we normalize K such that

$$\langle 1, K(\mathbf{x}, \mathbf{y})\rangle = 1. \tag{4.36}$$

By substituting (4.33) in (4.35) and using (4.36) we arrive at the required constraint

$$\begin{aligned}\langle \hat{f}, 1\rangle &= \Big\langle \sum_{i=1}^{l} a_i K(\mathbf{x}, \mathbf{x}_i), 1 \Big\rangle \\ &= \sum_{i=1}^{l} a_i \langle K(\mathbf{x}, \mathbf{x}_i), 1\rangle \\ &= \sum_{i=1}^{l} a_i = 0.\end{aligned} \tag{4.37}$$

We expand the $\mathcal{H}$ norm in equation (4.32) of the cost function, that is

$$\begin{aligned}E(\mathbf{a}) = \frac{1}{2}\|f\|_{\mathcal{H}}^2 - \sum_{i=1}^{l} a_i \langle f(\mathbf{x}), K(\mathbf{x}, \mathbf{x}_i)\rangle_{\mathcal{H}} \\ + \frac{1}{2}\sum_{i,j=1}^{l} a_i a_j \langle K(\mathbf{x}, \mathbf{x}_i), K(\mathbf{x}, \mathbf{x}_j)\rangle_{\mathcal{H}} + \epsilon\|\mathbf{a}\|_{L_1}.\end{aligned} \tag{4.38}$$

Recall the following two properties of a reproducing kernel

$$\langle K(\mathbf{x}, \mathbf{x}_i), K(\mathbf{x}, \mathbf{x}_j)\rangle_{\mathcal{H}} = K(\mathbf{x}_i, \mathbf{x}_j) \tag{4.39}$$

$$\langle f(\mathbf{x}), K(\mathbf{x}, \mathbf{x}_i)\rangle_{\mathcal{H}} = f(\mathbf{x}_i) \tag{4.40}$$

where in the last equation, $f(\mathbf{x}_i) = y_i$, because the data are noiseless. The cost function becomes

$$E(\mathbf{a}) = \frac{1}{2}\|f\|_{\mathcal{H}}^2 - \sum_{i=1}^{l} a_i y_i + \frac{1}{2}\sum_{i,j=1}^{l} a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) + \epsilon\|\mathbf{a}\|_{L_1}. \tag{4.41}$$

The vector $\mathbf{a}$ can be decomposed into its positive and negative parts. This implies that its L_1 norm can be written as

$$\|\mathbf{a}\|_{L_1} = \sum_{i=1}^{l} |a_i| = \sum_{i=1}^{l} (a_i^+ + a_i^-) \tag{4.42}$$

such that $a_i^+, a_i^- \geq 0$ and $a_i^+ a_i^- = 0$ for $i = 1, \ldots, l$. Using Eq. (4.42) and the fact that $\|f\|_{\mathcal{H}}^2$ is constant with respect to a_i^+ and a_i^-, we arrive at the following quadratic programming problem

$$\begin{aligned} \underset{a_i^+, a_i^-}{\text{minimize}} \quad & -\sum_{i=1}^{l}(a_i^+ - a_i^-)y_i + \frac{1}{2}\sum_{i=1}^{l}(a_i^+ - a_i^-)(a_j^+ - a_j^-)K(\mathbf{x}_i, \mathbf{x}_j) + \epsilon\sum_{i=1}^{l}(a_i^+ + a_i^-) \\ \text{subject to} \quad & \begin{cases} a_i^+, a_i^- \geq 0 & \text{for } i = 1, \cdots, l \\ \sum_{i=1}^{l}(a_i^+ - a_i^-) = 0 & \\ a_i^+ a_i^- = 0 & \text{for } i = 1, \cdots, l \end{cases} \end{aligned} \tag{4.43}$$

where the second constraint is the same as in Eq. (4.37). Renaming the coefficient a_i^+ to α_i^*, and a_i^- to α_i, makes it clear that Eq. (4.43) specifies the same quadratic programming problem as Eq. (4.26). We conclude that if

- the data are noiseless, i.e., $y_i = f(\mathbf{x}_i)$,
- the L_2 norm is replaced by the $\mathcal{H}$ norm in the data fitting term of BPDN,
- the function $\hat{f}$ has zero mean in the RKHS $\mathcal{H}$,
- the atoms of the dictionary used in BPDN are defined as in (4.31),
- and, the regularization parameter in SVR tends to zero, in other words, $C \to \infty$

then BPDN and ϵ-SVR are equivalent because they amount to the same quadratic programming problem. The last condition translates into the fact that ϵ-SVR will result in an interpolating function that is bound to overfit the data (because the effect of the regularization term is dampened). It would be an interesting to analyze what this says about the sparse approximation scheme that is this updated version of BPDN.

Chapter 5

Conclusion and Future Work

> "Penalized logistic regression is not the only model that performs similarly to the SVM; replacing the hinge loss with any sensible loss function will give a similar result, for example, the exponential loss function of boosting (Freund and Schapire, 1997) and the squared error loss (Zhang and Oles, 2001; Bühlmann and Yu, 2003). These loss functions are all Bayes consistent. The binomial deviance and the exponential loss are margin-maximizing loss functions, but the squared error loss is not. The distance weighted discrimination (Marron and Todd, 2002) is designed specifically for not maximizing the margin and works well with high-dimensional data, which in a way also justifies that margin maximization is not the key to the success of the SVM."
>
> —Hastie and Zhu (2006)

In this thesis, we have shown that ℓ_1-regularized square loss minimization for classification is a success, both computationally and statistically. We have also shown that ℓ_1-regularized square loss minimization for reconstruction is not worth it. Simpler methods like kNN classification and WkNNR are at least as good.

We propose four areas that hold promise for future investigation. One is the case where the design is filled with nonlinear mappings of the observed variables. How will this affect the optimization problem, and how will it affect the classification performance. Rifkin et al. (2003) answer this problem for ridge regression (or regularized least-squares classification). We are also interested to look more closely at the SVM dual problem. Can this problem be solved efficiently with added sparsity constraints on the dual variable $\boldsymbol{\alpha}$? Will this lead to a faster classifier that is comparable in performance to the SVM? An important disadvantage of the lasso optimization principle is that lasso solutions are not stable. We find it interesting to look at ways this problem can solved. Finally, we are interested to know if kNN or WkNN are just as good for feature learning as compared to sparse approximation methods in learning sparse representations in for e.g., image classification tasks. Below, we will look at each of these ideas in turn.

5.1 Nonlinear Regression

Consider the set $\{\phi_i, i = 1, \ldots, p\}$ of p predetermined functions, with at least one nonlinear member. One can apply these functions to the original explanatory variables

$\{\mathbf{x}_i, i = 1, \ldots, p\}$ and get a nonlinear regression function resulting from the linear combination of these mappings,

$$\sum_{i=1}^{p} a_i \phi_i(\mathbf{x}_i) + a_0. \tag{5.1}$$

One can fit this model using the same algorithms for fitting linear models, with the benefit of obtaining a solution that models a nonlinear relationship between the response and explanatory variables (Hesterberg et al., 2008). The same algorithms can be used because the model is still linear in the parameters (the regression coefficients). The model is only nonlinear in the explanatory variables.

One example of the application of this approach is Kernel Matching Pursuit (Vincent and Bengio, 2002). In this setting, $\phi_i(\mathbf{x}) = k(\mathbf{x}_i, \mathbf{x})$, where k is a bivariate function that does not necessarily have to satisfy Mercer's conditions. Another example is the kernelized LASSO formulation (Wang et al., 2007).

5.2 Regularizing $\boldsymbol{\alpha}$: the SVM Dual Variable

Recall that our motivation for ℓ_1-regularization was to get a classifier that was faster than SVM in the sense that there are fewer kernel evaluations at test time,

$$f(x) = \sum \alpha_i k(x_i, x). \tag{5.2}$$

Also recall that we said we don't have direct control over the sparsity of $\boldsymbol{\alpha}$, given the SVM optimization problem. What if we aim to make $\boldsymbol{\alpha}$ sparse directly? Let's take another look at the SVM dual optimization problem,

5.3 Instability and Non-uniqueness of Lasso Solutions

Lasso solutions are not unique when $\text{rank}(X) \neq p$ (Tibshirani, 2012). This is especially the case in $p > n$ settings. In this case, there exist multiple solutions to the lasso optimization problem. The solution of the lasso optimization problem should thus be expressed as

$$\mathbf{a} \in \underset{\mathbf{a}}{\text{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{a}\|^2 + \lambda\|\mathbf{a}\|_1. \tag{5.3}$$

This causes two major issues. The first is that one solution can have a positive ith coefficient while the other can have a negative ith coefficient. Additionally, two different solutions can have different supports altogether.

This is problematic when the task at hand is that of variable selection not prediction. However, what we are interested in is to study the effects of the inconsistencies of the lasso solutions on its prediction performance and on its stability.

5.4 kNN or Matching Pursuit for Feature Learning

If sparse approximation is good for sparse coding, we would like to know if matching pursuit or kNN can also be effective. The benefit of matching pursuit and kNN is that they are computationally less expensive than sparse approximation by convex relaxation methods like SPGL1.

References

Robert Andersen. Robust regression for the linear model. In *Modern Methods for Robust Regression*, pages 47–70. Sage Publications, 2008.

S. Arora, L. Babai, J. Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. In *Proceedings of the IEEE 34th Annual Foundations of Computer Science*, pages 724–733, Washington, DC, USA, 1993. IEEE Computer Society.

Peter L. Bartlett, Michael I. Jordan, and Jon D. McAuliffe. Convexity, classification, and risk bounds. Technical Report 638, Department of Statistics, U.C. Berkeley, 2003.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, 1992.

Stéphane Boucheron, Olivier Bousquet, and Gabor Lugosi. Theory of classification : A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.

Alfred M. Bruckstein, David L. Donoho, and Michael Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1): 34–81, 2009.

Peter Bühlmann and Bin Yu. Boosting with the L_2 loss: regression and classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

Andreas Buja, Trevor Hastie, and Robert Tibshirani. Linear smoothers and additive models. *The Annals of Statistics*, 17(2):435–555, 1989.

S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, Aug. 1998.

Scott S. Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. Technical Report 479, Department of Statistics, Stanford University, May 1995.

W. S. Cleveland and S. J. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.

Adam Coates and Andrew Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning (ICML)*, pages 921–928, 2011.

Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. *Journal of Machine Learning Research - Proceedings Track*, 15:215–223, 2011.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2002.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

M. Elad. *Sparse and redundant representations: From theory to applications in signal and image processing*. Springer, 2010.

John Fox. Robust regression: Appendix to an R and S-PLUS companion to applied regression, 2002.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1, part 2):119–139, 1997.

Jerome H. Friedman. Fast sparse regression and classification. In Paul Eilers, editor, *Proceedings of the 23rd International Workshop on Statistical Modelling*, pages 27–57. Statistical Modelling Society, 2008.

Jerome H. Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981.

Jort F. Gemmeke, Tuomas Virtanen, and Antti Hurmalainen. Exemplar-based sparse representations for noise robust automatic speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 19(7):2067–2080, 2011.

Frederico Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10:1455–1480, August 1998.

W. Härdle and O. Linton. Applied nonparametric methods. Technical Report 1069, Yale University, 1994.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2 edition, 2009.

T. J. Hastie and C. Loader. Local regression: Automatic kernel carpentry. *Statistical Science*, 8(2):120–129, 1993.

Trevor Hastie and Ji Zhu. Comment. *Statistical Science*, 21:352–357, 2006.

Tim C. Hesterberg, Nam H. Choi, Lukas Meier, and Chris Fraley. Least angle and ℓ_1 penalized regression: A review. *Statistics Surveys*, 2:61–93, 2008.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

George S. Kimeldorf and Grace Wahba. Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y. Ng. Efficient L1 regularized logistic regression. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 2006.

M. S. Lewicki. Efficient coding of natural sounds. *Nature Neuroscience*, 5(4):356–363, Mar. 2002.

Yi Lin. A note on margin-based loss functions in classification. Technical report, Department of Statistics, University of Wisconsin, Madison, 2002.

Julien Mairal. *Sparse Coding for Machine Learning, Image Processing and Computer Vision*. PhD thesis, Ecole Normale Superieure de Cachan, 2010.

S. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Academic Press, Elsevier, Amsterdam, 3rd edition, 2009.

J. Marron and M. Todd. Distance weighted discrimination. Technical report, School of Operations Research and Industrial Engineering, Cornell University, 2002.

Hosein Mohimani, Massoud Babaie-Zadeh, and Christian Jutten. A fast approach for overcomplete sparse decomposition based on smoothed ℓ_0 norm. *Transactions on Signal Processing*, 57:289–301, January 2009.

E. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9 (1):141–142, 1964.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Mark D. Plumbley, Thomas Blumensath, Laurent Daudet, Rémi Gribonval, and Mike E. Davies. Sparse Representations in Audio and Music: from Coding to Source Separation. *Proceedings of the IEEE*, 98(6):995–1005, 2010.

Tomaso Poggio, Lorenzo Rosasco, and Andre Wibisono. Sufficient conditions for uniform stability of regularization algorithms. Technical Report CBCL-284, Center for Biological and Computational Learning, MIT, December 2009.

Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse additive models. *Journal of The Royal Statistical Society (Series B)*, 71(5):1009–1030, 2009.

Ryan Rifkin. *Everything old is new again: a fresh look at historical approaches in machine learning*. PhD thesis, Sloan School of Management, Massachusetts Institute of Technology, 2002.

Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, December 2004.

Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification. In *Advances in Learning Theory: Methods, Model and Applications*, volume 190, pages 131–153. IOS Press, 2003.

Lorenzo Rosasco, Ernesto De Vito, Andrea Caponnetto, Michele Piana, and Alessandro Verri. Are loss functions all the same? *Neural Computation*, 16(5):1063–107, 2004.

D. Ruppert. Local polynomial regression and its applications in environmental statistics. Technical report, Cornell University, 1996.

D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *The Annals of Statistics*, 22:1346–1370, 1994.

Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, August 2004.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society (Series B)*, 73(3):273–282, 2011.

Ryan J. Tibshirani. The lasso problem and uniqueness. arXiv:1206.0313v1, 2012.

J. A. Tropp and S. J. Wright. Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–958, June 2010.

E. van den Berg and M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

Vladimir N. Vapnik. *The nature of statistical learning theory*. Statistics for Engineering and Information Science. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48: 165–187, September 2002.

A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Towards a practical face recognition system: Robust alignment and illumination via sparse representation. To appear in *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.

Grace Wahba. *Spline models for observational data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990.

Gang Wang, Dit-Yan Yeung, and Frederick H. Lochovsky. The kernel path in kernelized lasso. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

Sijian Wang, Bin Nan, Saharon Rosset, and Ji Zhu. Random Lasso. *Annals of Applied Statistics*, 5(1):468–485, 2011.

John Wright, Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:210–227, 2009.

Huan Xu, Constantine Caramanis, and Shie Mannor. Sparse algorithms are not stable: A no-free-lunch theorem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):187–193, 2012.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale ℓ_1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. Recent advances of large-scale linear classification. *Submitted*, 2011.

Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–134, March 2004.

Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2001.