OPTIMIZATION METHODS
FOR REGULARIZED CONVEX FORMULATIONS
IN MACHINE LEARNING

by

Sang Kyun Lee

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2011

To my mother.

ACKNOWLEDGMENTS

First I wish to thank my advisor, Stephen Wright, for his constant support for the six years of my graduate study in Madison. Without his encouragement and dedication, my research would not have been possible. He was also a great mentor for academic life and a best friend, teaching me many delightful virtues through his own actions and attitude. I also thank the other committee members: Michael Ferris, Benjamin Recht, Grace Wahba, and Jerry Zhu. A great portion of my research has been motivated from what I have learned from them in lectures and discussions.

It was a great fortune for me to join the optimization group in Madison. We had a list of superb courses offered, where I had practiced the craft of optimization, and also great minds who were always open for discussion. Among the renowned professors in our group, I like to thank Michael Ferris, Jeffrey Linderoth, Andrew Miller (now in Université Bordeaux 1 in France), Stephen Robinson, and again my advisor Stephen Wright. Our group recently had moved to a new building called the Wisconsin Institute of Discovery, and it was my pleasure to enjoy the new facility for some time, meeting with new people from various disciplines.

The support, encouragement, and prayer from my family, friends, and church members were the indispensable part of everything. I like to show my special gratitude to all of them, especially to Pastor Kwang-Woo Lee in South Korea and to the chorus team members in Korean Presbyterian Church of Madison.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

NOMENCLATURE

$\mathbb{R}$	Real numbers.
$\mathbb{R}_+$	Nonnegative real numbers.
$\overline{\mathbb{R}}$	Extended real numbers, $\mathbb{R} \cup \{+\infty\}$.
$\mathbb{Z}$	Integers.
$\mathbb{Z}_+$	Nonnegative integers.
$\mathbb{N}$	Natural numbers.
$\|\cdot\|$	The standard Euclidean norm $\|\cdot\|_2$.
$\mathbb{E}[X]$	The expectation of the random variable X.
$\mathbf{1}$	A column vector of ones, i.e. $(1, 1, \ldots, 1)^T$, with a proper length.
$\mathbf{0}$	A column vector of zeros, i.e. $(0, 0, \ldots, 0)^T$, with a proper length.
$\|M\|$	The cardinality of a finite set M.
$i.i.d.$	Independently and identically distributed.
$\lceil x \rceil$	The smallest integer larger than or equal to $x \in \mathbb{R}$.
$\lfloor x \rfloor$	The largest integer smaller than or equal to $x \in \mathbb{R}$.
$\mathrm{sign}(x)$	The signum function that returns $+1$ if $x > 0$, -1 if $x < 0$, and 0 otherwise.

dist (w, C) The distance between $w \in \mathcal{E}$ and a convex set $C \subset \mathcal{E}$ for a vector space $\mathcal{E}$, which is defined by dist $(w, C) := \inf_{c \in C} \|w - c\|$.

dom Ψ The effective domain of a function $\Psi : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$.

int S The interior of a set S.

ri S The relative interior of a set S.

ABSTRACT

We develop efficient numerical optimization algorithms for regularized convex formulations that appear in a variety of areas such as machine learning, statistics, and signal processing. Their objective functions consist of a loss term and a regularization term, where the latter controls the complexity of prediction models or induces a certain structure to the solution encoding our prior knowledge. The formulations become difficult to solve when we consider a large amount of data, or when we employ nonsmooth functions in the objective.

In this research we study algorithms in two different learning environments, large-scale/online and moderate-scale/batch learning. In online learning, we use only approximated or partial information about the objective at a time, but in batch learning we typically have full access to the objective. We focus on subgradient algorithms that are closely related to stochastic approximation methods for the former, and decomposition and cutting-plane techniques for the latter.

The low computational requirement of stochastic approximation methods makes them very appealing for large-scale and online learning, despite their slow asymptotic convergence. We study theoretical properties of a stochastic subgradient algorithm for regularized problems, revealing that the solution structure, called the optimal manifold, can be identified in finite iterations with high probability. This allows us developing a new algorithmic strategy that starts with a stochastic subgradient method but switches to another type of optimization on the near-optimal manifold. We also present a subgradient algorithm customized for the nonlinear support vector machines (SVMs), where kernels are approximated with low-dimensional surrogate mappings.

For moderate-sized learning tasks, batch approaches often find solutions much faster than online approaches. We discuss algorithms based on decomposition and cutting-plane techniques, exploiting the structure of SVMs for efficiency. Specifically, we address a decomposition approach

for semiparametric SVM regression problems that have extended dual formulations of the standard SVMs, and an improved cutting-plane algorithm for SVM classification that makes use of multiple cuts in every iteration.

Chapter 1

Introduction

This dissertation focuses on developing efficient numerical optimization algorithms for solving *regularized convex problems*, which can be described as follows:

$$\min_{w \in \mathcal{E}} \quad f(w) + \Psi(w), \tag{1.1}$$

where f and Ψ are convex functions and $\mathcal{E}$ is a finite-dimensional real vector space. The problems of this form appear in various areas such as machine learning, statistics, and signal processing. Typically, the first term f represents the deviation between the observed and the predicted values, where the predictions are made by certain models. The second term Ψ often controls the complexity of the models to avoid overfitting, or induces a certain structure (for instance, sparsity) to the solution reflecting one's prior knowledge. The minimization problem (1.1) becomes difficult when the function f involves abundant data, or when nonsmooth functions are employed in f or Ψ. We focus on these cases.

This manuscript is organized as follows. In Chapter 2 and Chapter 3, we study *online* optimization methods based on *stochastic approximation*. Each step of these methods evaluates an approximate subgradient of the objective at the current iterate, based on a small subset (perhaps a single item) of the data, thereby exhibiting low computational complexity per iteration and slow convergence to the optimal solution. In Chapter 4 and Chapter 5, we develop *batch* optimization techniques that evaluate information based on entire data at each iteration. For efficiency, these methods exploit decomposition or cutting-plane approaches to construct a sequence of small subproblems that will lead us to a minimizer.

In Chapter 2, we present theoretical properties of a subgradient descent algorithm, illustrating that we can design improved algorithms based on an understanding of the theoretical properties. In later chapters, we describe our optimization techniques specifically for the support vector machines (SVMs), one of the most popular methodologies for machine learning. We choose the SVMs since they are expressed with simple but fundamental mathematical formulations that appear in the heart of many other optimization problems. Although there have been many developments in the past twelve years, solving SVM problems efficiently for large amount data still remains challenging.

In this chapter, we briefly review the concepts and optimization techniques relevant to our discussion.

1.1 Stochastic Approximation Methods

In *stochastic approximation* (SA), we consider the following minimization problem:

$$\min_{w \in \mathcal{W}} \quad f(w) := \mathbb{E}_{\xi}[F(w;\xi)] = \int_{\Xi} F(w;\xi) dP(\xi), \tag{1.2}$$

where ξ is a random vector whose probability distribution P is supported on the set Ξ, $F(\cdot;\xi)$ is a convex function for each $\xi \in \Xi$, the expectation is well-defined and finite-valued for all $w \in \mathcal{W}$, and $\mathcal{W}$ is a nonempty closed and bounded convex set in $\mathcal{E}$. This problem can be regarded as a regularized convex problem (1.1), with f defined as above and $\Psi(w) := \delta_{\mathcal{W}}(w)$, where $\delta_{\mathcal{W}}(w)$ is the indicator function which is zero on $\mathcal{W}$ and $+\infty$ elsewhere.

The classical approaches to solve such problems first appeared in 1950's (Kushner and Yin, 2003), in forms of a recursive procedure for finding the root of a real-valued function (Robbins and Monro, 1951), or for finding the minimizer of a function using noisy estimates of the derivatives (Kiefer and Wolfowitz, 1952). SA methods have been widely adopted in stochastic optimization (Ermoliev, 1983; Pflug, 1996; Ruszczyński and Syski, 1986), and in signal processing because of its low memory footprint. Especially, SA methods have been popular recently in machine learning, because they scale very well with large data sets and provide solutions of low accuracy but of good generalization performance in practice (Bottou, 2004; Zhang, 2004; Shalev-Shwartz et al.,

2007). The elements of SA methods can be found in closely aligned literature such as incremental subgradient methods (Nedic and Bertsekas, 2001) and convex online learning (Zinkevich, 2003).

The most popular approach to solve SA problems is the *stochastic gradient descent* (SGD) method, which generates its iterates by the simple rule:

$$w_{t+1} = \Pi_{\mathcal{W}}\left(w_t - \eta_t g_t\right), \quad t = 1, 2, \ldots \tag{1.3}$$

where $g_t \in \partial F(w_t; \xi_t)$, and $\Pi_{\mathcal{W}}(\cdot)$ is the Euclidean projection onto the set $\mathcal{W}$. The analysis of convergence goes back to some old literature (Chung, 1954; Sacks, 1958), where f is assumed to be strongly convex and twice continuously differentiable, and $w^* \in \text{int}\, \mathcal{W}$. In this case the SGD algorithm exhibits $O(1/t)$ convergence (in terms of the objective value) with the stepsizes (also referred to as *learning rates*) $\eta_t = \theta/t$ for some positive constant $\theta > 0$. For reference, we present a convergence result from a recent literature in Theorem 1.1. This algorithm is known to be very sensitive to the choice of θ, and often performs poorly in practice (Spall, 2003, Section 4.5.3).

Theorem 1.1 (Nemirovski et al. (2009)) Suppose that f is twice continuously differentiable and strongly convex with the modulus $c > 0$, ∇f is Lipschitz continuous with the constant $L > 0$, and that the stepsizes are chosen by $\eta_t = \theta/t$ for some constant $\theta > 1/(2c)$. Also assume that there exists $M > 0$ such that $\mathbb{E}[\|g\|^2] \leq M^2$ for all $w \in \mathcal{W}$, where $g \in \partial F(w; \xi)$. Then the iterates w_t generated by the SGD algorithm satisfy

$$\mathbb{E}[f(w_t) - f(w^*)] \leq \frac{LQ(\theta)}{2t}, \quad t = 1, 2, \ldots$$

where $Q(\theta) := \max\{\theta^2 M^2 (2c\theta - 1)^{-1}, ||w_1 - w^*||^2\}$.

A more robust scheme was developed later (Nemirovski and Yudin, 1978; Polyak, 1990; Polyak and Juditsky, 1992), where longer stepsizes $\eta_t = O(1/\sqrt{t})$ and averaging of the obtained iterates were suggested. This method works for general convex functions (without the assumption of smoothness and strong convexity of the objective), exhibiting $O(1/\sqrt{t})$ convergence rate as shown in Theorem 1.2. These rates are known to be optimal for subgradient schemes under the assumption of *black-box* models (Nemirovski and Yudin, 1983).

Theorem 1.2 (Nemirovski et al. (2009)) Suppose that f is a convex function and there exist $M > 0$ such that $\mathbb{E}[\|g\|^2] \leq M^2$ for all $w \in \mathcal{W}$, where $g \in \partial F(w;\xi)$, and $D > 0$ such that $D = \max_{w\in\mathcal{W}} \|w - w_1\|$. If we choose $\eta_t = \frac{\theta D}{M\sqrt{t}}$ for some constant $\theta > 0$, and run the SGD algorithm for $t = 1, \ldots, N$, then

$$\mathbb{E}[f(\bar{w}_K^N) - f(w^*)] \leq C(r)\max\{\theta, \theta^{-1}\}\frac{DM}{\sqrt{N}}, \quad N = 1, 2, \ldots$$

where $\bar{w}_K^N = \frac{\sum_{j=K}^N \eta_j w_j}{\sum_{j=K}^N \eta_j}$, and $C(r)$ is a factor depending solely on the fraction $r \in (0, 1)$ such that $K = \lceil rN \rceil$.

The regularized convex problem (1.1) can be regarded as a stochastic approximation (1.2) when we consider the function $\Psi(w) := \delta_{\mathcal{W}}(w) + \psi(w)$, where $\delta_{\mathcal{W}}(w)$ is the indicator function which is zero on $\mathcal{W}$ and $+\infty$ elsewhere, and ψ is a convex function. Then we can apply the SGD algorithm with slight modification,

$$w_{t+1} = \Pi_{\mathcal{W}}\left(w_t - \alpha_t(g_t + h_t)\right), \quad t = 1, 2, \ldots \tag{1.4}$$

where $g_t \in \partial F(w_t;\xi_t)$ and $h_t \in \partial\psi(w_t)$. However, the SGD method does not exploit the problem structure explicitly which is induced by the regularizer ψ; as we can see in (1.4), the information associated with ψ only comes in terms of its subgradient in a way that we can combine $\psi(\cdot)$ into $F(\cdot;\xi_t)$ and treat them as a single function. Although in theory the sequence of iterates converges in expectation to an optimal solution as t goes to infinity, we expect high variations in the solution obtained with finite iterations. Also, for general convex cases the convergence is described with averaged iterates (referred to as *primal averages*). However, primal averages obtained with finite iterations are typically very dense, hardly revealing the desired structure encoded in the regularization term.

In Chapter 2, we focus on a subgradient-based method called the *regularized dual averaging* (RDA) algorithm (Xiao, 2010). Even though the RDA algorithm has similar convergence rate to the SGD algorithm, the finite-iteration behavior of RDA is much better in practice than SGD for regularization problems. In Chapter 2, we analyze the theoretical properties of the RDA algorithm, revealing that RDA actually identifies the structure (optimal manifold) induced by the regularizer

in finite iterations with high probability. We also present an algorithmic strategy that starts with RDA and switches to a different optimization method once a near-optimal manifold is identified.

1.2 Support Vector Machines

The support vector machines (SVMs) are among the most widely adopted techniques in machine learning. The first SVM was introduced by Vapnik et al. (Vapnik and Lerner, 1963; Vapnik and Chervonenkis, 1964), as a device for binary classification using hyperplanes. The idea was extended by Boser et al. (1992) to nonlinear SVMs, applying the *kernel trick* to the dual reformulation of SVMs, that is, substituting inner products of inputs with *kernels*. Unlike its predecessors, the SVM has a solid foundation on the *statistical learning theory* developed by Vapnik and Chervonenkis (Vapnik and Chervonenkis, 1991; Vapnik, 1998, 1999). Also, many practical algorithms have been developed for training SVMs, making them appealing for many applications.

In this section we overview the formulations of the SVMs for classification and regression tasks, and review a few representative algorithms for training SVMs.

1.2.1 Formulations

We review the primal and the dual formulations of SVMs for two primary learning tasks, classification and regression.

1.2.1.1 Formulations for Classification

In classification, we assume that a binary label $y_i \in \{-1, +1\}$ is associated with a training example $x_i \in \mathcal{H}$, representing the membership of x_i to one of the two different categories, for $i = 1, 2, \ldots, m$. We assume that the *feature space* $\mathcal{H}$ has well-defined inner products, and the examples x_i's are sampled from $\mathcal{H}$ independently according to an (unknown) identical distribution.

The fundamental idea of the SVM is to find the hyperplane of the form $\langle w, x\rangle + b = 0$ that bisects the shortest connection between the convex hulls of two input vector subsets correspond to the two categories (Scholkopf and Smola, 2001, Chapter 7). After proper scaling of w and b, we can consider the two hyperplanes $\langle w, x\rangle + b = 1$ and $\langle w, x\rangle + b = -1$ as 'support' hyperplanes, that

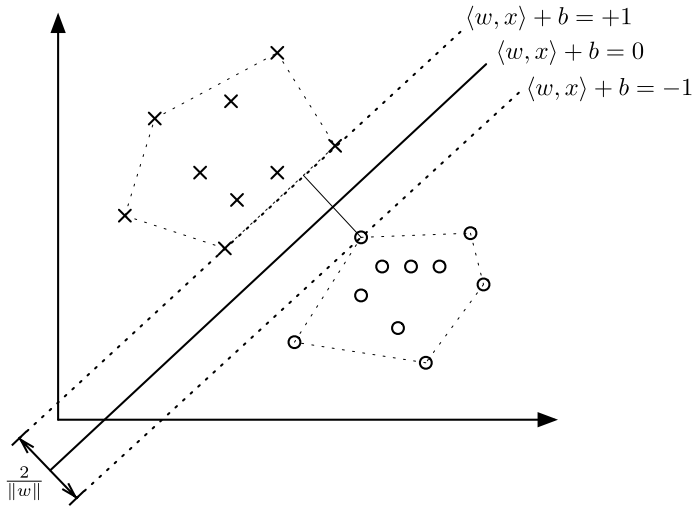


Figure 1.1 The support vector machine for binary classification (separable case) (Scholkopf and Smola, 2001). The crosses represent the data points with labels '$+1$', while circles represent those with '-1'. The SVM finds the hyperplane $\langle w, x\rangle + b = 0$ that bisects the shortest connection between the two convex hulls corresponds to the two categories.

separate the convex hull of each category to the other, with the margin of $1/||w||_2$ to the classifying hyperplane $\langle w, x\rangle + b = 0$. This is illustrated in Figure 1.1.

We try to find the classifier that maximizes the margin, since the classifiers with larger margin typically give better generalization over unseen test data points. (We assume that test examples are generated from the same probability distribution where we obtained the training data.) Theoretical justifications for finding maximal margin classifiers can be found in Scholkopf and Smola (2001, Theorem 7.3) and in statistical learning theory literature (Vapnik, 1999; Vapnik and Kotz, 2006).

This idea can be formulated into the convex regularization problem (1.1), defining

$$f(w, b) := \frac{c}{m}\sum_{i=1}^{m} \ell_h(w, b; x_i, y_i), \quad \text{and}$$
$$\Psi(w, b) := \frac{1}{2}\|w\|^2,$$

where $c > 0$ is a given parameter, and ℓ_h is the *hinge loss* function defined by

$$\ell_{\mathrm{h}}(w, b; x_i, y_i) := \max\{1 - y_i(\langle w, x_i\rangle + b), 0\}. \tag{1.5}$$

As the objective is nonsmooth due to the definition of ℓ_h, we consider the following reformulation called the *soft-margin* SVMs,

$$\begin{aligned} \min_{w,b,s} \quad & \frac{c}{m}\sum_{i=1}^{m} s_i + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(\langle w, x_i\rangle + b) \geq 1 - s_i \quad i = 1, 2, \ldots, m, \\ & s_i \geq 0 \qquad\qquad\qquad\quad i = 1, 2, \ldots, m. \end{aligned} \tag{1.6}$$

The parameter c is to be tuned by some external procedures, for example, using separate validation sets or via cross validation.

We derive the dual formulation of (1.6) using the first-order optimality conditions. First we construct the Lagrangian $\mathcal{L}$ of (1.6) introducing nonnegative dual variables $\alpha \in \mathbb{R}^m_+$ and $\beta \in \mathbb{R}^m_+$:

$$\mathcal{L}(w, b, s; \alpha, \beta) := \frac{c}{m}\sum_{i=1}^{m} s_i + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i\{y_i(\langle w, x_i\rangle + b) - 1 + s_i\} - \sum_{i=1}^{m} \beta_i s_i \ . \tag{1.7}$$

From the Karush-Kuhn-Tucker (KKT) first-order optimality conditions, we have

$$\begin{aligned} \nabla_w \mathcal{L}(w, b, s; \alpha, \beta) = \ & w - \sum_{i=1}^{m} \alpha_i y_i x_i = \mathbf{0} \\ \nabla_b \mathcal{L}(w, b, s; \alpha, \beta) = \ & -\sum_{i=1}^{m} \alpha_i y_i = 0 \\ \nabla_{s_i} \mathcal{L}(w, b, s; \alpha, \beta) = \ & \frac{c}{m} - \alpha_i - \beta_i = 0 \\ 0 \leq y_i(\langle w, x_i\rangle + b) + 1 - s_i \ \perp \alpha_i \geq 0, \quad & i = 1, 2, \ldots, m \\ 0 \leq s_i \ \perp \beta_i \geq 0, \quad & i = 1, 2, \ldots, m. \end{aligned} \tag{1.8}$$

Proper substitutions of the primal variables in (1.7) using (1.8) result in the dual formulation,

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^m} \quad & \frac{1}{2}\alpha^T D_y K D_y \alpha - \mathbf{1}^T \alpha \\ \text{s.t.} \quad & y^T \alpha = \mathbf{0} \\ & \mathbf{0} \leq \alpha \leq (c/m)\mathbf{1}, \end{aligned} \tag{1.9}$$

where $D_y := \mathrm{diag}(y)$ and $K \in \mathbb{R}^{m \times m}$ is defined by $K_{ij} := \langle x_i, x_j\rangle$.

1.2.1.2 Formulations for Regression

In regression, we have a real-valued response y_i for each training input x_i, which is assumed to be the evaluation of an unknown function with the input x_i. In support vector regression, we try to fit linear functions of the form $h(x) = \langle w, x\rangle + b$ to the training data. The SVM for regression can be formulated into the convex regularization problem (1.1) as follows,

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{i=1}^{m} \ell_\epsilon(h; x_i, y_i), \tag{1.10}$$

where ℓ_ϵ is the ϵ-*insensitive loss* function (Vapnik, 1999) defined by

$$\ell_\epsilon(h; x_i, y_i) := \max\{|y_i - h(x_i)| - \epsilon, 0\}$$

for a given constant $\epsilon > 0$. We can reformulate (1.10) as follows to have a differentiable objective,

$$\begin{aligned} \min_{w,b,s,s^*} \quad & \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{i=1}^{m}(s_i + s_i^*) \\ \text{s.t.} \quad & y_i - \langle w, x_i\rangle - b \le \epsilon + s_i \quad \text{for } i = 1, 2, \ldots, m \\ & \langle w, x_i\rangle + b - y_i \le \epsilon + s_i^* \quad \text{for } i = 1, 2, \ldots, m \\ & s_i \ge 0,\ s_i^* \ge 0 \qquad\qquad \text{for } i = 1, 2, \ldots, m\ . \end{aligned} \tag{1.11}$$

The corresponding dual formulation is,

$$\begin{aligned} \min_{\alpha,\alpha^* \in \mathbb{R}^m} \quad & \frac{1}{2}(\alpha - \alpha^*)^T K(\alpha - \alpha^*) + (\epsilon\mathbf{1} - y)^T\alpha + (\epsilon\mathbf{1} + y)^T\alpha^* \\ \text{s.t.} \quad & \mathbf{1}^T(\alpha - \alpha^*) = 0 \\ & 0 \le \alpha \le (c/m)\mathbf{1}, \quad 0 \le \alpha^* \le (c/m)\mathbf{1}\ , \end{aligned} \tag{1.12}$$

where $K \in \mathbb{R}^{m\times m}$ is defined by $K_{ij} = \langle x_i, x_j\rangle$. In Chapter 4, we discuss an extended formulation of (1.12) that has multiple equality constraints rather than a single equality constraint as in (1.12).

1.2.2 The Kernel Trick

The SVM formulations discussed above use linear decision functions in the space of input points. However, we often need to consider nonlinear functions for better prediction results. The

kernel trick is a convenient way to extend linear SVMs for nonlinear decision functions, using a feature mapping $\phi : \mathcal{X} \to \mathcal{H}$ from a input space $\mathcal{X}$ to a *Hilbert space* $\mathcal{H}$ (a feature space equipped with inner products).

When we replace all occurrences of x_i with $\phi(x_i)$ in the dual formulations (1.9) and (1.12), the images $\phi(x_i)$'s appear only in forms of inner products $\langle \phi(x_i), \phi(x_j) \rangle$. Therefore, instead of defining ϕ explicitly, we can substitute those inner products with a *kernel function* $\kappa(x_i, x_j)$ that satisfies the conditions of Mercer's theorem (Mercer, 1909). In other words, applying the kernel trick corresponds to simply redefining the positive semidefinite matrix K in (1.9) and (1.12) by $K_{ij} = \kappa(x_i, x_j)$. The popular choices of the kernels include

- Linear kernel: $\kappa(x_i, x_j) = \langle x_i, x_j \rangle$.
- Polynomial kernel: $\kappa(x_i, x_j) = (\langle x_i, x_j \rangle + 1)^d \,,\ d \in \mathbb{N}$.
- Gaussian kernel: $\kappa(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right),\ \gamma > 0$.

In particular, the kernel matrix defined by the Gaussian kernel has full rank, provided that all inputs x_i's are distinct (Scholkopf and Smola, 2001, Theorem 2.18). That is, the images $\phi(x_i)$, $i = 1, \ldots, m$ are linearly independent and thus span an m-dimensional subspace of $\mathcal{H}$. This implies the Gaussian kernel always defines a subspace which is big enough to embed all training data. For this reason the Gaussian kernel is widely adopted in many applications.

1.2.3 The Canonical Dual Form

We define the canonical form of the dual formulations (1.9) and (1.12) as follows,

$$\begin{aligned} \min_{z \in \mathbb{R}^{m'}} \quad & \frac{1}{2} z^T Q z + p^T z \\ \text{s.t.} \quad & q^T z = r \\ & \ell \leq z \leq u \ , \end{aligned} \tag{1.13}$$

where the vectors z, p, q, ℓ and u are has m' elements, r is a scalar, and Q is a positive semi-definite $m' \times m'$ matrix. We define these components for the classification and regression tasks in Table 1.1.

Table 1.1 Definitions of the components for the canonical SVM dual formulation. We define $D_y = \mathrm{diag}(y)$, and the kernel matrix K by $K_{ij} = \kappa(x_i, x_j)$.

	z	Q	p	q	r	ℓ	u
Classification	α	$D_y K D_y$	$-\mathbf{1}$	y	0	$\mathbf{0}$	$(c/m)\mathbf{1}$
Regression	$\begin{bmatrix}\alpha \\ \alpha^*\end{bmatrix}$	$\begin{bmatrix}K & -K \\ -K & K\end{bmatrix}$	$\begin{bmatrix}\epsilon\mathbf{1} - y \\ \epsilon\mathbf{1} + y\end{bmatrix}$	$\begin{bmatrix}\mathbf{1} \\ -\mathbf{1}\end{bmatrix}$	0	$\begin{bmatrix}\mathbf{0} \\ \mathbf{0}\end{bmatrix}$	$(c/m)\begin{bmatrix}\mathbf{1} \\ \mathbf{1}\end{bmatrix}$

1.3 Optimization Algorithms for SVMs

In this section we survey prior works for solving the SVM formulations. Since we focus on the classical SVM formulations discussed above, we exclude the approaches that make use of different formulations for faster optimization, for example, using $\|w\|_1$ instead of $\frac{1}{2}\|w\|_2^2$ (Mangasarian and Musicant, 2002), or using smooth loss functions (Mangasarian and Musicant, 2001; Ferris and Munson, 2004).

1.3.1 Solving the Dual Formulation

Due to the convenience of applying the kernel trick, the dual formulation (1.13) has been more popular than its primal counterpart. Many techniques have been developed to handle the dual Hessian Q efficiently, which is dense and ill-conditioned for typical choices of kernels.

Interior Point Methods: Fine and Scheinberg (2001) suggested a primal-dual interior-point method of predictor-corrector type to find the solution of (1.13). In theory this method takes $O(m' \ln(1/\epsilon))$ iterations to converge where ϵ is the relative accuracy, but in practice it converges mostly within 50 iterations regardless of the problem size. The most costly operation in each iteration is to solve the linear system

$$(Q + D)u = w \ ,$$

for u, twice per predictor step, where Q is the Hessian of the objective of (1.13) and D is a diagonal matrix. As $Q + D$ is a dense $m' \times m'$ symmetric positive definite matrix, it takes $O((m')^3)$ in

general to obtain the solution $u = (Q + D)^{-1}w$. However, if we have a rank-k approximation V to Q such that $Q = VV^T$ and $V \in \mathbb{R}^{m' \times k}$ with $k \ll m'$, then the vector u can be found in $O(k^2 m')$ operations. For many SVM problems Q is not necessarily of low-rank, so VV^T can only approximate Q with certain error. An incomplete Cholesky factorization is used in Fine and Scheinberg (2001) to acquire such V, which has the computation complexity of $O(k^2 m')$.

Decomposition Algorithms: Decomposition algorithms solve a sequence of subproblems, updating a small subvector of z of (1.13) leaving the remaining components untouched in each iteration. The sequential minimal optimization (SMO) algorithm (Platt, 1999), LIBSVM (Chang and Lin, 2009), and an online variant LASVM (Bordes et al., 2005) create subproblems with two components at a time, which can be solved analytically. On the other hand, SVM-Light (Joachims, 1999), SVMTorch (Collobert and Bengio, 2001), and GPDT (Serafini et al., 2004) choose larger subvectors, solving the subproblems with quadratic programming solvers.

Efficient implementations are often equipped with *kernel caching*, storing only recently used kernel matrix entries to save memory. Some solvers such as SVM-Light use *shrinking* heuristics, which freeze the values of the components of z that have stayed at bounds for a period of time.

In Chapter 4, we develop a decomposition framework for semiparametric SVM regression. The dual formulation of the semiparametric SVM has multiple equality constraints, rather than a single equality constraint as in the canonical SVM dual form (1.13).

1.3.2 Solving the Primal Formulation

There have been recent developments regarding the primal SVM formulations (1.1), (1.6) and (1.11). Since we cannot apply the kernel trick, methods in this category are only effective when the linear kernel suffices for our purpose, or when the chosen kernel function can be approximated well with low-rank surrogate mappings.

Stochastic Subgradient Methods: These methods generate iterates by solving simple subproblems similarly to SGD (1.3). In each iteration, they use the information (x_t, y_t) of a single

(random) data point and create an estimated subgradient $g_t \in \partial\ell_{\mathrm{h}}(w_t, b_t; x_t, y_t)$ for classification, or $g_t \in \partial\ell_{\epsilon}(h; x_t, y_t)$ for regression. The PEGASOS algorithm (Shalev-Shwartz et al., 2007) solves a slightly modified SVM formulation (1.1) with $\Psi(w, b) = \frac{1}{2}(\|w\|^2 + b^2)$ rather than $\Psi(w, b) = \frac{1}{2}\|w\|^2$, in order to have a strongly convex objective function and to choose the steplength $\eta_t \leq O(1/t)$. PEGASOS also requires linear kernels, as the explicit feature mapping ϕ is not available in general. In Chapter 3, we suggest an extended framework that operates with general convex SVMs, working with nonlinear kernels obtained via Nyström sampling or random projections.

Cutting-plane Methods: For classification, cutting-plane methods make use of the following formulation which is equivalent to (1.6):

$$\begin{aligned} \min_{w,b,\eta} \quad & \frac{c}{m}\eta + \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & \eta \geq \sum_{i=1}^{m} u_i\{1 - y_i(\langle w, x_i\rangle + b)\}, \quad \forall u \in \{0,1\}^m. \end{aligned}$$

(Regression problems can be reformulated similarly.) In Chapter 5, we derive this formulation using the idea of the Benders' reformulation. This formulation was also discovered independently by Joachims (2006) from a different perspective. Joachims' algorithm, called SVM-Perf, has been improved later by Franc and Sonnenburg (2007, 2008), exploiting efficient line-search steps. For nonlinear SVMs, Joachims et al. suggested a new version of SVM-Perf (Joachims et al., 2009) (which we refer to as CPNY) and CPSP (Joachims and Yu, 2009). These methods use inexact kernel information approximated with small dimensions.

Cutting-plane algorithms solve a sequence of relaxed subproblems, adding a single violated constraint (called a cut) to the subproblem in each iteration. These methods fit well for medium to large problems, since they usually require solving only a few subproblems to find solutions. However, the size of subproblems increases as we add cuts, so it may not be appropriate for very large problems. In Chapter 5, we discuss improvements for the existing cutting-plane algorithms, suggesting a generalized method that can generate multiple cuts per iteration to obtain more strengthened subproblems, reducing the number of iterations until convergence.

Chapter 2

Manifold Identification Approaches for Regularized Stochastic Online Learning

Iterative methods that take steps in approximate subgradient directions have proved to be useful for stochastic learning problems over large or streaming data sets. When the objective consists of a loss function plus a nonsmooth regularization term, whose purpose is to induce structure in the solution, the solution often lies on a low-dimensional manifold along which the regularizer is smooth.

In this chapter, we show that a regularized dual averaging algorithm can identify this manifold with high probability. This observation motivates an algorithmic strategy in which, once a near-optimal manifold is identified, we switch to an algorithm that searches only in this manifold, which typically has much lower intrinsic dimension than the full space, thus converging quickly to a near-optimal point with the desired structure. Computational results are presented to illustrate these claims.

2.1 Introduction

The online learning algorithms inspired by stochastic approximation have been proved to be effective dealing with large machine learning problems. Regarding the trade-offs with other errors in statistical learning theory, solutions with high optimization error may be accepted, but typically they do not suffice to identify the structures (such as sparsity) induced by the regularization term in the objective, which encodes our prior knowledge for the structure.

We focus on the regularized dual averaging (RDA) method developed by Xiao (2010), which is an extension to Nesterov (2009) and known to exploit the structure more effectively in online settings.

2.1.1 Regularized Stochastic Online Learning

In *regularized stochastic learning*, we consider the following problem:

$$\min_{w \in \mathbb{R}^n} \phi(w) := f(w) + \Psi(w), \tag{2.1}$$

where

$$f(w) := \mathbb{E}_\xi F(w;\xi) = \int_\Xi F(w;\xi) dP(\xi), \tag{2.2}$$

ξ is a random vector whose probability distribution P is supported on the set $\Xi \subset \mathbb{R}^d$. We assume that $\Psi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a proper convex function with $\operatorname{dom}\Psi$ closed. We also assume that there is an open neighborhood $\mathcal{O}$ of $\operatorname{dom}\Psi$ that is contained in the domain of $F(\cdot,\xi)$, for all $\xi \in \Xi$. We suppose that $F(w;\xi)$ is a smooth convex function for $w \in \mathcal{O}$ and every $\xi \in \Xi$, and the expectation in (2.2) is well defined and finite-valued for all $w \in \mathcal{O}$. We use w^* to denote a minimizer of (2.1).

The purpose of the regularization function Ψ is to promote certain desirable types of structure in the solution of (2.1). A common desirable property is *sparsity* (where w has few nonzero elements), which under certain conditions is promoted by setting $\Psi(\cdot) = \lambda\|\cdot\|_1$ for some parameter $\lambda > 0$. We focus on objectives that consist of a smooth loss function in conjunction with a nonsmooth regularizer. A classic problem of this form is ℓ_1-regularized logistic regression.

In *regularized stochastic online learning*, we encounter a previously unknown cost function $F(\cdot;\xi_t) : \mathbb{R}^n \to \mathbb{R}$ for $\xi_t \in \Xi$ in each time $t \geq 1$, where $\{\xi_t\}_{t\geq 1}$ forms an i.i.d. sequence of random samples generated from the distribution P. At each time t, we make a decision w_t using the information gathered up to the time t, and attempt to generate a sequence $\{w_t\}_{t\geq 1}$ such that

$$\lim_{t\to\infty} \mathbb{E}\left[F(w_t;\xi) + \Psi(w_t)\right] = f(w^*) + \Psi(w^*).$$

We denote the history of random variables up to time t by

$$\xi_{[t]} := \{\xi_1, \xi_2, \ldots, \xi_t\}.$$

In the algorithmic framework we use in this chapter, the iterate w_t depends on $\xi_1, \xi_2, \ldots, \xi_{t-1}$ but not on $\xi_t, \xi_{t+1}, \cdots$; we can emphasize this fact by writing $w_t = w_t(\xi_{[t-1]})$.

We will base our analyses on the *regret* $R_t(w)$ of the algorithms, defined in (2.3), which is a standard performance measure for online learning algorithms. The regret represents the difference between accumulated objective function values up to time t with respect to a single decision $w \in \operatorname{dom} \Psi$ and an instantiation of the random sequence of decisions $w_1, w_2, \ldots, w_t$.

$$R_t(w) := \sum_{j=1}^{t} [F(w_j; \xi_j) + \Psi(w_j)] - \sum_{j=1}^{t} [F(w; \xi_j) + \Psi(w)]. \tag{2.3}$$

2.1.2 Related Works

Xiao (2010) recently developed the *regularized dual averaging* (RDA) method, in which the smooth term is approximated by an averaged gradient, while the regularization term appears explicitly in each subproblem. The RDA method is known to exploit the structure of (2.1) more effectively than other methods for this setting. In each iteration of the RDA algorithm, we compute the next iterate by

$$w_{t+1} = \arg\min_{w \in \mathbb{R}^n} \left\{ \langle \bar{g}_t, w \rangle + \Psi(w) + \frac{\beta_t}{t} h(w) \right\}, \quad t = 1, 2, \ldots \tag{2.4}$$

where $\bar{g}_t = \frac{1}{t} \sum_{j=1}^{t} g_j$ and $g_j \in \partial F(w_j; \xi_j)$. The function $h(w)$ is a strongly convex function such as $\|w - w_1\|^2$, and we will introduce the formal definition of h in Section 2.3.

The main difference of the RDA method to other subgradient-based algorithms is that RDA makes use of the averaged gradient information collected at each iteration, referred as the *dual averages*, which tends to converge to the optimal gradient as the iterates converge to a solution. Taken in conjunction with the nondegeneracy condition $0 \in \text{ri } [\nabla f(w^*) + \partial \Psi(w^*)]$, this property provides the key to identification.

When we consider ℓ_1-regularization, we can consider an alternative problem to (2.1) using *sample average approximation* (SAA) as follows:

$$\min_{w \in \mathbb{R}^n} \quad \tilde{\phi}_{\mathcal{N}}(w) := \tilde{f}_{\mathcal{N}}(w) + \Psi(w) \tag{2.5}$$

where $\tilde{f}_{\mathcal{N}}(w) := \frac{1}{|\mathcal{N}|}\sum_{j\in\mathcal{N}} F(w;\xi_j)$ with random samples ξ_j in $j \in \mathcal{N}$. When the sample set $\mathcal{N}$ is relatively small, this problem can be solved very efficiently with *batch* optimization, where the small magnitude components of the high-accuracy solution can be rounded toward zero to produce sparse solutions. On the contrary, simple rounding with the low-accuracy solutions from the SGD method (1.4) with ℓ_1-regularization would be very unreliable. The *forward-backward splitting* (FOBOS) (Duchi and Singer, 2009) and the *truncated gradient* (TG) (Langford et al., 2009) approaches provide principled ways to address this problems. For general regularizer Ψ, the iterations of FOBOS can be stated as below, following Xiao (2010):

$$w_{t+1} = \arg\min_{w\in\mathbb{R}^n} \left\{ \langle g_t, w\rangle + \Psi(w) + \frac{1}{2\alpha_t}\|w - w_t\|^2 \right\}. \tag{2.6}$$

For $\Psi(w) := \lambda\|w\|_1$, TG truncates the iterates obtained by the standard SGD method in every K steps. That is,

$$[w_{t+1}]_i = \begin{cases} \text{trnc}\left([w_t]_i - \alpha_t[g_t]_i,\ \lambda_t^{\text{TG}},\ \theta\right) & \text{if } \text{mod}(t,K) = 0, \\ [w_t]_i - \alpha_t[g_t]_i & \text{otherwise,} \end{cases} \tag{2.7}$$

where $\lambda_t^{\text{TG}} := \alpha_t\lambda K$, $\text{mod}(t,K)$ is the remainder on division of t by K, and

$$\text{trnc}(\omega,\ \lambda_t^{\text{TG}},\ \theta) = \begin{cases} 0 & \text{if } |\omega| \le \lambda_t^{\text{TG}} \\ \omega - \lambda_t^{\text{TG}}\text{sgn}(\omega) & \text{if } \lambda_t^{\text{TG}} < |\omega| \le \theta, \\ \omega & \text{otherwise.} \end{cases}$$

TG becomes identical to FOBOS (2.6) for ℓ_1-regularization when $K = 1$ and $\theta = +\infty$.

As we discuss later, the RDA algorithm has $O(\sqrt{t})$ regret bounds with $\beta_t = O(\sqrt{t})$ for general convex cases, and $O(\ln t)$ bounds with $\beta_t = O(\ln t)$ for strongly convex cases. These regret bounds are comparable to these of the SGD method. For general convex cases, when we use $\alpha_t = O(1/\sqrt{t})$, the SGD method achieves $O(1/\sqrt{t})$ regret bound (Zinkevich, 2003; Nemirovski et al., 2009), which cannot be improved in general. For the strongly convex case the SGD method has $O(\ln t)$ bound (Hazan et al., 2006; Bartlett et al., 2008) with $\alpha_t = O(1/t)$.

A characteristic of problems with nonsmooth regularizers is that the solution often lies on a manifold of low dimension. In ℓ_1-regularized problems, for instance, the number of nonzero

components at the solution is often a small fraction of the dimension of the full space. Where a reliable method for identifying an optimal (or near-optimal) manifold is available, we have the possibility of invoking an algorithm that searches just in the low-dimensional space defined by this manifold — possibly a very different algorithm to one that would be used on the full space. One example of this type of approach is seen in LPS (Shi et al., 2008; Wright, 2010), a batch optimization method for ℓ_1-regularized logistic regression, which takes inexact Newton steps on the space of apparently nonzero variables, to supplement the partial gradient steps that are used in the full space. In logistic regression, and probably in other cases, it can be much less expensive to compute first- and second-order information on a restricted subspace than on the full space.

Identification of optimal manifolds has been studied in the context of convex constrained optimization (Burke and Moré, 1994; Wright, 1993) and nonsmooth nonconvex optimization (Hare and Lewis, 2004). In the latter setting, the optimal manifold is defined to be a smooth surface passing through the optimum, parameterizable by relatively few variables, such that the restriction of the objective to the manifold is smooth. When a certain nondegeneracy condition is satisfied, this manifold may be identified *without knowing the solution*, usually as a by-product of an algorithm for solving the problem.

In this chapter, we investigate the ability of the RDA algorithm to identify the optimal manifold. We also investigate this behavior computationally for the case of ℓ_1-regularized logistic regression, and suggest a technique for switching to a different method once a near-optimal manifold is identified, thus avoiding the sublinear asymptotic convergence rate that characterizes stochastic gradient methods.

2.1.3 Terminology

We call a function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is *strongly convex* if there exists a constant $\sigma > 0$ (known as the *modulus* of strong convexity) such that $\forall w, w' \in \operatorname{dom} \varphi$ and $\forall \alpha \in [0, 1]$,

$$\varphi(\alpha w + (1-\alpha)w') \leq \alpha\varphi(w) + (1-\alpha)\varphi(w') - \frac{\sigma}{2}\alpha(1-\alpha)\|w - w'\|^2.$$

Strong convexity implies that for any $w \in \text{dom}\,\varphi$ and $w' \in \text{ri dom}\,\varphi$, we have

$$\varphi(w) \geq \varphi(w') + \langle s, w - w' \rangle + \frac{\sigma}{2}\|w - w'\|^2, \quad \forall s \in \partial\varphi(w').$$

2.2 Assumptions and Basic Results

We summarize here the fundamental assumptions that we make on the problem and its solution, together with some basic observations and results that will be used in the analysis of later sections.

2.2.1 Unbiasedness

As in Nemirovski et al. (2009), we assume the following *unbiasedness* property:

$$\nabla f(w) = \nabla_w \mathbb{E}_\xi F(w;\xi) = \mathbb{E}_\xi \nabla_w F(w;\xi) \tag{2.8}$$

for any w independent of ξ. (As the differentiation of F is taken only for its first argument, we omit the subscript in the following discussion.) Given that $w_t = w_t(\xi_{[t-1]})$, this implies

$$\mathbb{E}[\nabla F(w_t;\xi_t)] = \mathbb{E}\left[\mathbb{E}[\nabla F(w_t;\xi_t)|\xi_{[t-1]}]\right] = \mathbb{E}\left[\nabla f(w_t)\right].$$

2.2.2 Uniform Lipschitz Continuity

First we assume that each $F(w;\xi)$ is a smooth convex function of $w \in \mathcal{O}$ for every $\xi \in \Xi$, and is uniformly Lipschitz continuous in w over all ξ. That is, there exists a constant $L > 0$ such that

$$\|\nabla F(w;\xi) - \nabla F(w';\xi)\| \leq L\|w - w'\|, \quad \forall w, w' \in \mathcal{O}, \quad \forall \xi \in \Xi. \tag{2.9}$$

This assumption immediately leads to a Lipschitz property on ∇f. Before showing the property, we present a result that is used often in the analysis of later sections.

Lemma 2.1 For a vector-valued function $h : \Xi \to \mathbb{R}^n$ which is integrable with respect to P, we have

$$\|\mathbb{E}h\|_2 \leq \mathbb{E}\|h\|_2$$

Proof Using the dual of the Euclidean norm, we deduce that

$$\begin{aligned}\|\mathbb{E}h\|_2 &= \sup_{v\in\mathbb{R}^n,\|v\|_2=1} \langle v, \mathbb{E}h\rangle \\ &= \sup_{v\in\mathbb{R}^n,\|v\|_2=1} \int_\Xi \langle v, h(\xi)\rangle dP(\xi) \\ &\leq \sup_{v\in\mathbb{R}^n,\|v\|_2=1} \int_\Xi \left[\sup_{w\in\mathbb{R}^n,\|w\|_2=1} \langle w, h(\xi)\rangle\right] dP(\xi) \\ &= \int_\Xi \|h(\xi)\|_2 dP(\xi) = \mathbb{E}\|h\|_2.\end{aligned}$$

■

Lemma 2.2 If $\nabla F(w;\xi)$ satisfies the uniform Lipschitz continuity assumption (2.9) on $\mathcal{O}$, then $\nabla f(w)$ is also uniformly Lipschitz continuous on $\mathcal{O}$ with the same constant L.

Proof From unbiasedness, we have for $w, w' \in \mathcal{O}$ independent of ξ that

$$\begin{aligned}\nabla f(w) = \nabla\mathbb{E}[F(w;\xi)] = \mathbb{E}[\nabla F(w;\xi)] \quad &\text{from (2.8)} \\ = \mathbb{E}[\nabla F(w';\xi) + u_\xi] \quad &\text{for } u_\xi := \nabla F(w;\xi) - \nabla F(w';\xi) \\ = \nabla f(w') + \mathbb{E}[u_\xi] \quad &\text{from (2.8) again.}\end{aligned}$$

Since $\|u_\xi\| \leq L\|w - w'\|$, we have

$$\|\nabla f(w) - \nabla f(w')\| = \|\mathbb{E}u_\xi\| \leq \mathbb{E}\|u_\xi\| \leq L\|w - w'\|,$$

where the first inequality is due to Lemma 2.1. ■

2.2.3 Optimality and nondegeneracy

We specify several optimality conditions that are assumed to hold throughout the chapter. The optimality of w^* for the problem (2.1) can be characterized as follows:

$$0 \in \nabla f(w^*) + \partial\Psi(w^*). \tag{2.10}$$

We assume that w^* is a *nondegenerate* solution, which satisfies the following stronger condition:

$$0 \in \text{ri } [\nabla f(w^*) + \partial \Psi(w^*)]\,. \tag{2.11}$$

2.2.4 Manifolds and Partial Smoothness

In this section we discuss properties of differential manifolds and partial smoothness by repeating some definitions from Hare and Lewis (2004).

Definition 2.3 (Manifold) A set $\mathcal{M} \subset \mathbb{R}^n$ is a *manifold* about $\bar{z} \in \mathcal{M}$ if it can be described locally by a collection of $\mathcal{C}^p$ functions ($p \geq 2$) with linearly independent gradients. That is, there exists a map $H : \mathbb{R}^n \to \mathbb{R}^k$ that is $\mathcal{C}^p$ around $\bar{z}$ with $\nabla H(\bar{z})^T \in \mathbb{R}^{k \times n}$, surjective, such that points z near $\bar{z}$ lie in $\mathcal{M}$ if and only if $H(z) = 0$.

The *normal space* to $\mathcal{M}$ at $\bar{z}$, denoted by $N_{\mathcal{M}}(\bar{z})$, is the range space of $\nabla H(\bar{z})$, while the *tangent space* to $\mathcal{M}$ at $\bar{z}$ is the null space of $\nabla H(\bar{z})^T$. We assume without loss of generality that $\nabla H(\bar{z})$ has orthonormal columns.

We define *partial smoothness* as follows (Lewis, 2003, Section 2).

Definition 2.4 (Partial Smoothness) A function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ is ($\mathcal{C}^2$-) *partly smooth* at a point $\bar{z} \in \mathbb{R}^n$ relative to a set $\mathcal{M} \subset \mathbb{R}^n$ containing $\bar{z}$ if $\mathcal{M}$ is a manifold about $\bar{z}$ and the following properties hold:

(i) (Smoothness) The function φ restricted to $\mathcal{M}$, denoted by $\varphi|_{\mathcal{M}}$, is $\mathcal{C}^2$ near $\bar{z}$;

(ii) (Regularity) φ is subdifferentially regular at all points $z \in \mathcal{M}$ near $\bar{z}$, with $\partial\varphi(z) \neq \emptyset$.

(iii) (Sharpness) The affine span of $\partial\varphi(\bar{z})$ is a translate of $N_{\mathcal{M}}(\bar{z})$;

(iv) (Sub-continuity) The set-valued mapping $\partial\varphi : \mathcal{M} \rightrightarrows \mathbb{R}^n$ is continuous at $\bar{z}$.

We refer to $\mathcal{M}$ as the *active manifold*, and if it is associated with the minimizer w^*, we call it as the *optimal manifold*.

We assume that Ψ is partly smooth at w^* relative to the optimal manifold, which implies partly smoothness of ψ since f is smooth. (The smoothness of f follows from the smoothness of $F(\cdot;\xi)$ for each $\xi \in \Xi$.)

2.2.5 Strong Minimizer

We assume that w^* is a *strong local minimizer* of ϕ relative to the optimal manifold $\mathcal{M}$ with modulus $c_{\mathcal{M}} > 0$, that is, there exists $c_{\mathcal{M}} > 0$ and $r_{\mathcal{M}} > 0$ such that $\{w \in \mathbb{R}^n \mid \|w - w^*\| \leq r_{\mathcal{M}}\} \subset \mathcal{O}$ and

$$\phi|_{\mathcal{M}}(w) \geq \phi|_{\mathcal{M}}(w^*) + c_{\mathcal{M}}\|w - w^*\|^2, \quad \text{for all } w \in \mathcal{O} \text{ with } \|w - w^*\| \leq r_{\mathcal{M}}. \tag{2.12}$$

Under the given conditions, this implies that w^* is a strong local minimizer of $\phi(w)$.

Theorem 2.5 (Strong Minimizer for General Convex Case) Suppose that ϕ is partly smooth at w^* relative to the optimal manifold $\mathcal{M}$, that w^* is a strong local minimizer of $\phi|_{\mathcal{M}}$ with the modulus $c_{\mathcal{M}} > 0$ and radius $r_{\mathcal{M}} > 0$ as defined in (2.12), and that the nondegeneracy condition (2.11) holds at w^*. Then there exist $0 < c \leq c_{\mathcal{M}}$ and $0 < \bar{r} \leq r_{\mathcal{M}}$ such that

$$\phi(w) - \phi(w^*) \geq c\|w - w^*\|^2, \quad \text{for all } w \in \mathcal{O} \text{ with } \|w - w^*\| \leq \bar{r} \tag{2.13}$$

Proof See Appendix A.1. ■

This condition is similar to the quadratic growth condition proposed by Anitescu (2000) in the context of nonlinear programming. It was shown by Anitescu that this fundamental condition is weaker than many other second-order conditions that are widely use in nonlinear programming. We have the following immediate consequences.

Corollary 2.6 Suppose that w^* is a strong local minimizer of (2.1) that satisfies (2.13). For all $w \in \mathcal{O}$ with $\|w - w^*\| > \bar{r}$, we have

$$\phi(w) - \phi(w^*) > c\bar{r}\|w - w^*\|.$$

Proof Given $w \in \mathcal{O}$ with $\|w - w^*\| > \bar{r}$, we have from the convexity of ϕ that

$$\phi\left(w^* + \bar{r}\frac{w - w^*}{\|w - w^*\|}\right) \leq \phi(w^*) + \frac{\bar{r}}{\|w - w^*\|}(\phi(w) - \phi(w^*)).$$

From the strong local minimizer property of w^*, we also have

$$\phi\left(w^* + \bar{r}\frac{w-w^*}{\|w-w^*\|}\right) - \phi(w^*) \geq c\left\|\left(w^* + \bar{r}\frac{w-w^*}{\|w-w^*\|}\right) - w^*\right\|^2 = c\bar{r}^2.$$

Collecting the above two inequalities leads to the claim. ■

Corollary 2.7 (Strong Minimizer for Strongly Convex Case) Suppose that w^* is a strong local minimizer of (2.1) satisfying (2.13). If ϕ is strongly convex on $\operatorname{dom}\Psi$ with the modulus $\sigma > 0$, then w^* is a (globally) strong minimizer of (2.1) with the modulus $\min(c, \sigma/2)$, that is,

$$\phi(w) \geq \phi(w^*) + \min(c, \sigma/2)\|w - w^*\|^2, \quad \text{for all } w \in \mathcal{O}. \tag{2.14}$$

Proof Given $w \in \mathcal{O}$, if $\|w - w^*\| \leq \bar{r}$, then the claim follows from (2.13). If $\|w - w^*\| > \bar{r}$, then we have from the strong convexity of ϕ that

$$\begin{aligned}\phi\left(w^* + \bar{r}\frac{w-w^*}{\|w-w^*\|}\right) \quad \leq \quad & \phi(w^*) + \frac{\bar{r}}{\|w-w^*\|}(\phi(w) - \phi(w^*)) \\ & -\frac{\sigma}{2}\frac{\bar{r}}{\|w-w^*\|}\left(1 - \frac{\bar{r}}{\|w-w^*\|}\right)\|w-w^*\|^2.\end{aligned}$$

From the strong local minimizer property of w^*, we also have

$$\phi\left(w^* + \bar{r}\frac{w-w^*}{\|w-w^*\|}\right) - \phi(w^*) \geq c\bar{r}^2.$$

Combining the above two inequalities results in

$$\begin{aligned}\phi(w) - \phi(w^*) \quad &\geq \quad \left[\sigma/2 + \frac{\bar{r}}{\|w-w^*\|}(c - \sigma/2)\right]\|w-w^*\|^2 \\ &\geq \quad \min(c, \sigma/2)\|w-w^*\|^2.\end{aligned}$$

■

In the remainder of the chapter, we assume that all of the conditions discussed in this section are satisfied unless otherwise stated.

2.3 Regularized Dual Averaging Algorithm

We start this section by describing regret bounds for the regularized dual averaging (RDA) algorithm of Xiao (2010) following Nesterov (2009), focusing on its stochastic variant. We also describe the consequences for the analysis of the condition that the minimum is strong locally (2.13) or globally (2.14). We then analyze the properties of the averaged gradient; this analysis forms the basis of the manifold identification result in Section 2.4.

2.3.1 Regret Bounds

To derive the regret bounds from the results of Xiao (2010), we first recall our assumptions on the functions F, f, and Ψ from Section 2.1, and note that they are stronger than the corresponding conditions in Xiao (2010) (which require only subdifferentiability of $F(w;\xi_t)$ on $\operatorname{dom}\Psi$). We assume without loss of generality that $\min_w \Psi(w) = 0$.

We introduce a *prox-function* $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ which is proper, strongly convex on $\operatorname{dom}\Psi$, and subdifferentiable on $\operatorname{ri}\operatorname{dom}\Psi$. Without loss of generality, $h(w)$ is assumed to have modulus of convexity 1, with $\min_w h(w) = 0$. In addition, we require h to satisfy

$$\arg\min_w \; h(w) \in \arg\min_w \; \Psi(w).$$

We define the *prox-center* w_1 of $\operatorname{dom}\Psi$ with respect to h, (which will be the starting point of the RDA method) as follows:

$$w_1 := \arg\min_{w \in \operatorname{dom}\Psi} \; h(w).$$

Note that $h(w_1) = 0$ and $\Psi(w_1) = 0$ by the assumptions. (The terms "prox-function" and "prox-center" are borrowed from Nesterov (2009).) The most obvious prox-function is $h(\cdot) = \|\cdot - w_1\|^2$, where $w_1 \in \arg\min_x \Psi(w)$.

We now make some further assumptions, and define two constants that reappear throughout the analysis. First, choosing any $D > 0$, we consider a level set of the prox-function h defined as follows:

$$\mathcal{F}_D := \{w \in \operatorname{dom}\Psi \mid h(w) \leq D^2\}. \tag{2.15}$$

Algorithm 1 The RDA Algorithm.

1: Input:

- a prox-function $h(w)$ that is strongly convex on $\operatorname{dom}\Psi$ and also satisfies

$$\arg\min_w \ h(w) \in \arg\min_w \ \Psi(w).$$

- a positive and nondecreasing sequence $\{\beta_t\}, t \geq 1$.

2: Initialize: set $w_1 = \arg\min_w \ h(w)$ and $\bar{g}_0 = 0$.

3: **for** $t = 1, 2, \ldots$ **do**

4: Sample ξ_t from Ξ and compute a gradient $g_t = \nabla F(w_t; \xi_t)$.

5: Update the average gradient:

$$\bar{g}_t = \frac{t-1}{t}\bar{g}_{t-1} + \frac{1}{t}g_t.$$

6: Compute the next iterate:

$$w_{t+1} = \arg\min_{w \in \mathbb{R}^n} \left\{ \langle \bar{g}_t, w \rangle + \Psi(w) + \frac{\beta_t}{t} h(w) \right\}. \tag{2.19}$$

7: **end for**

Second, we assume that there exists a uniform bound G for which

$$\|\nabla F(w; \xi)\| \leq G, \ \ \forall w \in \mathcal{O}, \ \forall \xi \in \Xi. \tag{2.16}$$

At iteration t, the stochastic RDA algorithm samples a vector $\xi_t \in \Xi$, according to the distribution P, and evaluates an approximate gradient as follows:

$$g_t := \nabla_w F(w_t; \xi_t). \tag{2.17}$$

We assume that the random variables ξ_t are i.i.d. We form an averaged approximation to the gradient of f as follows:

$$\bar{g}_t := \frac{1}{t}\sum_{j=1}^{t} g_j = \frac{1}{t}\sum_{j=1}^{t} \nabla_w F(w_j; \xi_j) \tag{2.18}$$

which is called the *dual average*.

The RDA algorithm is specified precisely in Algorithm 1. As the objective function in the subproblem (2.19) is strongly convex for $\beta_t > 0$, w_{t+1} is uniquely defined. Note that w_{t+1} depends on the history of random variables $\xi_{[t]}$. In particular, we have that w_{t+1} is independent of later samples $\xi_{t+1}, \xi_{t+2}, \ldots$.

We are most interested in the case of $\beta_t = \gamma\sqrt{t}$ with some constant $\gamma > 0$ for general convex cases, and $\beta_t \leq O(\ln t)$ for strongly convex cases. The first key result is as follows.

Theorem 2.8 Suppose that the sequences $\{w_t\}$ and $\{g_t\}$ generated by the RDA algorithm using $\beta_t = \gamma\sqrt{t}$, and assume that (2.16) holds. We have for any $w \in \mathcal{F}_D$ and any $t \geq 1$ that

$$R_t(w) \leq \left(\gamma D^2 + \frac{G^2}{\gamma}\right)\sqrt{t}. \tag{2.20}$$

Moreover, when $\Psi(w)$ is strongly convex with the modulus $\sigma > 0$, then using $\beta_t = \sigma(1 + \ln t)$ results in a bound for $w \in \mathcal{F}_D$ and any $t \geq 1$,

$$R_t(w) \leq \left(\sigma D^2 + \frac{G^2}{2\sigma}\right)(1 + \ln t). \tag{2.21}$$

Proof See Xiao (2010, Corollary 2) for the general convex case, and Xiao (2010, Theorem 1 and Section 3.2) for the strongly convex cases. (For the latter we note that there are other choices of β_t, but all leads to similar bounds. The specific β_t is chosen to have a nondecreasing sequence as we do in the general convex case.) ■

The next result obtains bounds on the expected errors in the iterates generated by Algorithm RDA. For the purpose of this and future results, we define the indicator function $I_{(A)}$ for the event A to be such that $I_{(A)} = 1$ when the event A is true and $I_{(A)} = 0$ otherwise. For a random event A, $I_{(A)}$ becomes a random variable.

Theorem 2.9 (Expected Error Bounds of Iterates) Suppose that $w^* \in \mathcal{F}_D$ is a strong local minimizer of (2.1) according to the definition (2.13). Then for the iterates $w_1, w_2, \ldots, w_t$ generated by

the stochastic RDA algorithm with $\beta_t = \gamma\sqrt{t}$, we have

$$\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| \le \bar{r})}\|w_j - w^*\|^2\right] \le \frac{1}{c}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/2}, \tag{2.22a}$$

$$\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})}\|w_j - w^*\|\right] \le \frac{1}{c\bar{r}}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/2}. \tag{2.22b}$$

Moreover, when $\Psi(w)$ is strongly convex with the modulus $\sigma > 0$, then with $\beta_t = \sigma(1 + \ln t)$ we have

$$\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[\|w_j - w^*\|^2\right] \le \frac{1}{\min(\sigma/2, c)}\left(\sigma D^2 + \frac{G^2}{2\sigma}\right)\frac{1 + \ln t}{t}. \tag{2.23}$$

Proof See Appendix A.2.1. ■

2.3.2 Stochastic Behavior of the Dual Average

We now study the properties of the dual average, $\bar{g}_t$. We first state a blanket assumption that is useful in the remainder of the chapter.

Assumption 2.10 All of the conditions in Section 2.2, and the gradient bound (2.16) are satisfied. The iterates $w_1, w_2, \ldots$ are generated by Algorithm 1, with

- $\beta_t = \gamma\sqrt{t}$ when ϕ is a general convex function, or
- $\beta_t = \sigma(1 + \ln t)$ when ϕ is strongly convex due to its regularization component Ψ which is strongly convex with the modulus $\sigma > 0$.

Theorem 2.11 Suppose that Assumption 2.10 is satisfied. Defining

$$\Sigma^t := \mathbb{E}[(\bar{g}_t - \mathbb{E}\bar{g}_t)(\bar{g}_t - \mathbb{E}\bar{g}_t)^T],$$

we have for the general convex case:

(i) $\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| \le L\mu t^{-1/4}$

(ii) $\text{tr}\, \Sigma^t \leq 4G(G + 4L\mu)t^{-1/4}$

and for strongly convex case:

(i') $\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| \leq L\mu' \left(\frac{1+\ln t}{t}\right)^{1/2}$

(ii') $\text{tr}\, \Sigma^t \leq 4G(G + 4L\mu') \left(\frac{1+\ln t}{t}\right)^{1/2}$

where the constants μ and μ' are defined as follows,

$$\mu = \frac{1}{\sqrt{c}} \left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2} \left[1 + \frac{1}{\bar{r}\sqrt{c}} \left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2}\right], \tag{2.24a}$$

$$\mu' = \frac{1}{\sqrt{\min(c, \sigma/2)}} \left(\sigma D^2 + \frac{G^2}{2\sigma}\right)^{1/2}. \tag{2.24b}$$

Proof See Appendix A.2.2 ■

Using Theorem 2.11, we show the important property of the dual average that the probability $\bar{g}_t$ lies outside any given ball around $\nabla f(w^*)$ goes to zero as t increases.

Theorem 2.12 Suppose that Assumption 2.10 is satisfied. For the general convex cases, for every $\epsilon > 0$ and $t \geq 1$ we have

$$\mathbb{P}(\|\bar{g}_t - \nabla f(w^*)\| > \epsilon) < \epsilon^{-2}\nu t^{-1/4}.$$

And for the strongly convex cases we have

$$\mathbb{P}(\|\bar{g}_t - \nabla f(w^*)\| > \epsilon) < \epsilon^{-2}\nu' \left(\frac{1 + \ln t}{t}\right)^{1/2}.$$

The constants ν and ν' are defined as follows,

$$\nu := \left[L\mu + 2\sqrt{G}\,(G + 4L\mu)^{1/2}\right]^2, \tag{2.25a}$$

$$\nu' := \left[L\mu' + 2\sqrt{G}\,(G + 4L\mu')^{1/2}\right]^2. \tag{2.25b}$$

Proof For the general convex case, from Markov inequality, we obtain for every $\epsilon > 0$ that

$$\mathbb{P}(\|\bar{g}_t - \nabla f(w^*)\| > \epsilon) \;=\; \mathbb{P}(\|\bar{g}_t - \nabla f(w^*)\|^2 > \epsilon^2) \;<\; \epsilon^{-2}\mathbb{E}[\|\bar{g}_t - \nabla f(w^*)\|^2].$$

Since $\bar{g}_t - \nabla f(w^*) = \{\mathbb{E}\bar{g}_t - \nabla f(w^*)\} + \{\bar{g}_t - \mathbb{E}\bar{g}_t\}$, we have

$$\begin{aligned}
&\mathbb{E}[\|\bar{g}_t - \nabla f(w^*)\|^2] \\
&\leq \mathbb{E}\left[\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\|^2 + 2\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\|\|\bar{g}_t - \mathbb{E}\bar{g}_t\| + \|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2\right] \\
&\leq \|\mathbb{E}\bar{g}_t - \nabla f(w^*)\|^2 + 2\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\|\mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\| + \mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2 \\
&\leq \|\mathbb{E}\bar{g}_t - \nabla f(w^*)\|^2 + 2\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\|\left(\mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2\right)^{1/2} \\
&\quad + \mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2 \\
&= \left[\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| + \left(\mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2\right)^{1/2}\right]^2,
\end{aligned}$$

where the third inequality is due to Jensen's inequality. Each term in the last line can be bounded using Theorem 2.11 as follows,

$$\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| \leq L\mu t^{-1/4},$$

and,

$$\begin{aligned}
\mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2 = \mathbb{E}\left(\sum_{i=1}^{n}[\bar{g}_t - \mathbb{E}\bar{g}_t]_i^2\right) &= \sum_{i=1}^{n}\mathbb{E}\left([\bar{g}_t - \mathbb{E}\bar{g}_t]_i^2\right) = \operatorname{tr}\Sigma^t \\
&\leq 4G\left[G + 4L\mu\right]t^{-1/4}.
\end{aligned}$$

Collecting all results, we conclude that

$$\begin{aligned}
&\mathbb{P}(\|\bar{g}_t - \nabla f(w^*)\| > \epsilon) \\
&< \epsilon^{-2}\left[L\mu t^{-1/4} + \left\{4G(G + 4L\mu)t^{-1/4}\right\}^{1/2}\right]^2 \\
&\leq \epsilon^{-2}\left[L\mu + 2\sqrt{G}\left(G + 4L\mu\right)^{1/2}\right]^2 t^{-1/4}
\end{aligned}$$

as claimed.

For the strongly convex case, we use different bounds from Theorem 2.11 instead,

$$\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| \leq L\mu'\left(\frac{1 + \ln t}{t}\right)^{1/2}$$

and,

$$\mathbb{E}\|\bar{g}_t - \mathbb{E}\bar{g}_t\|^2 \leq 4G\left(G + 4L\mu'\right)\left(\frac{1+\ln t}{t}\right)^{1/2}.$$

Then similar arguments as above lead to the claim. ■

2.4 Manifold Identification

In this section we present that the RDA algorithm identifies the optimal manifold in finite iterations. Our analysis is based upon the important properties of the dual average discussed in the previous section, and the results for manifold identification.

2.4.1 Fundamental Results

We state a result from Hare and Lewis (2004) in a modified form that is more convenient for our analysis below.

Theorem 2.13 Suppose that ϕ is partly smooth at the minimizer w^* relative to the manifold $\mathcal{M}$ and that the nondegeneracy condition (2.11) holds. Then there exists a threshold $\bar{\epsilon} > 0$ such that for all $w \in \mathcal{O}$ with $\|w - w^*\| < \bar{\epsilon}$ and dist $(0, \partial\phi(w)) < \bar{\epsilon}$, we have $w \in \mathcal{M}$.

Proof Suppose for contradiction that no such $\bar{\epsilon}$ exists. Let $\{\epsilon_j\}_{j\geq 1}$ be any sequence of positive numbers such that $\epsilon_j \downarrow 0$. Then for each $j \geq 1$ we have w_j such that $\|w_j - w^*\| < \epsilon_j$, dist $(0, \partial\phi(w_j)) < \epsilon_j$ but $w_j \notin \mathcal{M}$. Considering the sequence $\{w_j\}_{j\geq 1}$, we have that $w_j \to w^*$, and dist $(0, \partial\phi(w_j)) \to 0$. With convexity, these imply $\phi(w_j) \to \phi(w^*)$, since for all $a_j \in \partial\phi(w_j)$ we have $\phi(w_j) - \phi(w^*) \leq a_j^T(w_j - w^*) \leq \|a_j\|\|w_j - w^*\|$. Convexity implies prox-regularity, so by applying Theorem 5.3 of Hare and Lewis (2004), we have that $w_j \in \mathcal{M}$ for all j sufficiently large. This contradicts our choice of w_j, so we conclude that $\bar{\epsilon} > 0$ with the claimed properties exists. ■

The next result is essentially from Wright (2010), which motivates the acceleration based on the Newton-type methods after identifying the optimal manifold. It relates the strong minimizer property for $\phi|_{\mathcal{M}}$ at w^* to the second-order sufficient conditions for an explicit representation of this function along the manifold.

Theorem 2.14 Suppose that ϕ is partly smooth at $w^* \in \mathbb{R}^n$ relative to the optimal manifold $\mathcal{M} \subset \mathbb{R}^n$. Suppose that $\mathcal{M}$ is characterized by C^2 mappings $F : \mathbb{R}^n \to \mathbb{R}^k$ and $G : \mathbb{R}^{n-k} \to \mathbb{R}^n$ and a point $y^* \in \mathbb{R}^{n-k}$, such that $F(w) = 0$ for all $w \in \mathcal{M}$ near w^*, $\nabla F(w^*)$ is orthonormal, $G(y) \in \mathcal{M}$ for all y near y^*, and $G(y) = w^* + Y(y - y^*) + O(\|y - y^*\|^2)$ for some matrix Y such that $[\nabla F(w^*)\ Y]$ is orthogonal. Then $\phi|_{\mathcal{M}}$ has a strong minimizer at w^* with modulus $c_{\mathcal{M}} > 0$ if and only if the function defined by

$$\psi(y) := \phi(G(y)) \tag{2.26}$$

is C^2 with $\nabla\psi(y^*) = 0$ and $\nabla^2\psi(y^*)$ positive definite, with minimum eigenvalue at least $2c_{\mathcal{M}}$.

2.4.2 Convergent Sequences

We start with two result that estimates the likelihood of w_j lying within a given radius of w^*, each for general convex and strongly convex objectives.

Lemma 2.15 (Convergent Sequences for General Convex Case) Suppose that the conditions in Assumption 2.10 hold for general convex objectives. Define the subsequence $\mathcal{S}$ by

$$\begin{aligned}\mathcal{S} := \Big\{ j \in \{1, 2, \dots\} \mid \mathbb{E}\left[I_{(\|w_j - w^*\| \le \bar{r})}\|w_j - w^*\|^2\right] \le j^{-1/4}, \text{ and} \\ \mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})}\|w_j - w^*\|\right] \le \bar{r}^{-1} j^{-1/4} \Big\}.\end{aligned} \tag{2.27}$$

For any $\epsilon > 0$, we then have

$$\mathbb{P}\left(\|w_j - w^*\| > \epsilon\right) < \frac{1}{\epsilon}\left(\frac{1}{\epsilon} + \frac{1}{\bar{r}}\right) j^{-1/4}, \quad \forall j \in \mathcal{S}. \tag{2.28}$$

Defining

$$S_t := \mathcal{S} \cap \{1, 2, \dots, t\}, \tag{2.29}$$

we have

$$\frac{1}{t}|S_t| > 1 - \frac{2}{c}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/4}, \tag{2.30}$$

that is, the density of S_t in $\{1, 2, \ldots, t\}$ is $1 - O(t^{-1/4})$.

Proof To measure the cardinality of the complement of $\mathcal{S}_t$, that is, $\mathcal{S}_t^c := \{1, 2, \ldots, t\} \setminus \mathcal{S}_t$, we first define indicator variables χ_-^j and χ_+^j for $j \geq 1$ as follows,

$$\chi_-^j := \begin{cases} 1 & \text{if } \mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})} \|w_j - w^*\|^2\right] > j^{-1/4}, \\ 0 & \text{otherwise.} \end{cases}$$

$$\chi_+^j := \begin{cases} 1 & \text{if } \mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})} \|w_j - w^*\|\right] > (1/\bar{r}) j^{-1/4}, \\ 0 & \text{otherwise.} \end{cases}$$

As the set $\mathcal{S}_t^c$ contains all indices $j \in \{1, 2, \ldots, t\}$ that satisfy $\chi_-^j = 1$ or $\chi_+^j = 1$, the cardinality of $\mathcal{S}_t^c$ is bounded above by $\sum_{j=1}^t (\chi_-^j + \chi_+^j)$. For $\sum_{j=1}^t \chi_-^j$, we note that

$$\begin{aligned} \sum_{j=1}^t \mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})} \|w_j - w^*\|^2\right] &\geq \sum_{j=1}^t \chi_-^j \, \mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})} \|w_j - w^*\|^2\right] \\ &> \sum_{j=1}^t \chi_-^j j^{-1/4} \quad \text{(due to the definition of } \chi_-^j\text{)} \\ &\geq t^{-1/4} \sum_{j=1}^t \chi_-^j. \end{aligned}$$

Using (2.22a), we deduce that

$$\frac{1}{t} \sum_{j=1}^t \chi_-^j \leq \frac{1}{c}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/4}.$$

Similar arguments for $\sum_{j=1}^t \chi_+^j$ with $\mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})} \|w_j - w^*\|\right]$, $j = 1, 2, \ldots, t$ and (2.22b) lead to

$$\frac{1}{t} \sum_{j=1}^t \chi_+^j \leq \frac{1}{c}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/4}.$$

Therefore, the fraction of the cardinality of $\mathcal{S}_t$ to $\{1, 2, \ldots, t\}$ is

$$\begin{aligned}\frac{1}{t}|\mathcal{S}_t| &= 1 - \frac{1}{t}|\mathcal{S}_t^c| \\ &\geq 1 - \frac{1}{t}\sum_{j=1}^{t}(\chi_-^j + \chi_+^j) \\ &> 1 - \frac{2}{c}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/4},\end{aligned}$$

thus proving (2.30).

To show (2.28), we first observe that for any $\epsilon > 0$,

$$\begin{aligned}\mathbb{P}(\|w_j - w^*\| > \epsilon) = \mathbb{P}(\|w_j - w^*\| > \epsilon,\ \|w_j - w^*\| \leq \bar{r}) \\ + \mathbb{P}(\|w_j - w^*\| > \epsilon,\ \|w_j - w^*\| > \bar{r})\end{aligned} \tag{2.31}$$

Focusing on the first term, we have for all $j \in \mathcal{S}$

$$\begin{aligned}\mathbb{P}(\|w_j - w^*\| > \epsilon,\ \|w_j - w^*\| \leq \bar{r}) &= \mathbb{P}(I_{(\|w_j - w^*\| \leq \bar{r})}\|w_j - w^*\| > \epsilon) \\ &< \epsilon^{-2}\mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})}\|w_j - w^*\|^2\right] \\ &\leq \epsilon^{-2} j^{-1/4}\end{aligned} \tag{2.32}$$

where the first inequality is due to Markov inequality and the second inequality is from the definition of $\mathcal{S}$ in (2.27). Similarly for the second term in (2.31), we have for all $j \in \mathcal{S}$

$$\begin{aligned}\mathbb{P}(\|w_j - w^*\| > \epsilon,\ \|w_j - w^*\| > \bar{r}) &= \mathbb{P}(I_{(\|w_j - w^*\| > \bar{r})}\|w_j - w^*\| > \epsilon) \\ &< \epsilon^{-1}\mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})}\|w_j - w^*\|\right] \\ &\leq \epsilon^{-1}\bar{r}^{-1} j^{-1/4}\end{aligned} \tag{2.33}$$

Applying (2.32) and (2.33) to (2.31) leads to the claim,

$$\mathbb{P}(\|w_j - w^*\| > \epsilon) < \epsilon^{-1}(\epsilon^{-1} + \bar{r}^{-1}) j^{-1/4}, \quad \forall j \in \mathcal{S}.$$

■

The next is a corresponding result for the strongly convex case.

Lemma 2.16 (Convergent Sequences for Strongly Convex Cases) Suppose that the conditions in Assumption 2.10 hold for strongly convex objectives. Define the subsequence $\mathcal{S}'$ by

$$\mathcal{S}' := \left\{ j \in \{1, 2, \dots\} \mid \mathbb{E}\left[\|w_j - w^*\|^2\right] \leq \left(\frac{1+\ln j}{j}\right)^{1/2} \right\}. \tag{2.34}$$

For any $\epsilon > 0$, we then have

$$\mathbb{P}\left(\|w_j - w^*\| > \epsilon\right) < \epsilon^{-2}\left(\frac{1+\ln j}{j}\right)^{1/2}, \quad \forall j \in \mathcal{S}'. \tag{2.35}$$

Defining

$$S'_t := \mathcal{S}' \cap \{1, 2, \dots, t\}, \tag{2.36}$$

we have

$$\frac{1}{t}|S'_t| > 1 - \frac{1}{\min(c, \sigma/2)}\left(\sigma D^2 + \frac{G^2}{2\sigma}\right)\left(\frac{1+\ln t}{t}\right)^{1/2}, \tag{2.37}$$

that is, the density of S'_t in $\{1, 2, \dots, t\}$ is $1 - O\left(\left(\frac{1+\ln t}{t}\right)^{1/2}\right)$.

Proof We first define indicator variables χ^j for $j \geq 1$ as follows,

$$\chi^j := \begin{cases} 1 & \text{if } \mathbb{E}\left[\|w_j - w^*\|^2\right] > \left(\frac{1+\ln j}{j}\right)^{1/2}, \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned} \sum_{j=1}^{t} \mathbb{E}\left[\|w_j - w^*\|^2\right] &\geq \sum_{j=1}^{t} \chi^j \mathbb{E}\left[\|w_j - w^*\|^2\right] \\ &> \sum_{j=1}^{t} \chi^j \left(\frac{1+\ln j}{j}\right)^{1/2} \quad \text{(due to the definition of } \chi^j\text{)} \\ &\geq \left(\frac{1+\ln t}{t}\right)^{1/2} \sum_{j=1}^{t} \chi^j. \end{aligned}$$

With (2.23), this leads to

$$\frac{1}{t}\sum_{j=1}^{t} \chi^j \leq \frac{1}{\min(c, \sigma/2)}\left(\sigma D^2 + \frac{G^2}{2\sigma}\right)\left(\frac{1+\ln t}{t}\right)^{1/2}.$$

Therefore,

$$\begin{aligned}\frac{1}{t}|\mathcal{S}'_t| &= 1 - \frac{1}{t}|\mathcal{S}'^c_t| \\ &\geq 1 - \frac{1}{t}\sum_{j=1}^{t}\chi^j \\ &> 1 - \frac{1}{\min(c,\sigma/2)}\left(\sigma D^2 + \frac{G^2}{2\sigma}\right)\left(\frac{1+\ln t}{t}\right)^{1/2},\end{aligned}$$

thus proving (2.37). The claim (2.35) follows from applying Markov inequality for any $\epsilon > 0$ and using the definition of $\mathcal{S}'$. ■

2.4.3 Identification

The next theorem is our main result, showing that the stochastic RDA algorithm identifies the optimal manifold with increasing probability as iterations proceed. This result requires a condition (2.38) on h that is trivially satisfied by the usual prox function $h(x) = (1/2)\|x - w_1\|^2$.

Theorem 2.17 (Identification for General Convex Cases) Suppose that Assumption 2.10 holds, $w^* \in \mathcal{F}_D$, $\mathcal{M}$ is the optimal manifold including w^*, and the RDA algorithm uses $h(w)$ such that

$$\sup_{b_j \in \partial h(w_j)} \|b_j\| \leq \eta\|w_j - w_1\|, \quad j = 1, 2, \ldots \tag{2.38}$$

for some $\eta > 0$. Given $\mathcal{S}$ defined in (2.27), we have

$$\mathbb{P}(w_j \in \mathcal{M}) \geq 1 - (\zeta_1 + \zeta_2)j^{-1/4}$$

for all $j \in \mathcal{S}$ sufficiently large, where

$$\zeta_1 := \frac{3\max(1,L)}{\bar{\epsilon}}\left(\frac{3\max(1,L)}{\bar{\epsilon}} + \frac{1}{\bar{r}}\right), \text{ and } \zeta_2 := 1.2\left(\frac{3}{\bar{\epsilon}}\right)^2\nu.$$

Proof Let $\bar{\epsilon} > 0$ be the threshold defined in Theorem 2.13. We focus on the iterate w_j and the random events associated with it. First we denote the following event as E_1:

$$E_1 : \quad \|w_j - w^*\| \leq \frac{\bar{\epsilon}}{3\max(L,1)}, \tag{2.39}$$

where $\bar{\epsilon}$ is defined in Theorem 2.13 and L is the Lipschitz constant of (2.9) and Lemma 2.2. Note that E_1 depends on the random history $\xi_{[j-1]}$. If E_1 is true, it trivially implies the condition $\|w_j - w^*\| \leq \bar{\epsilon}$ of Theorem 2.13. From Lemma 2.15, we have that

$$\mathbb{P}(\|w_j - w^*\| \leq \bar{\epsilon}) \geq \mathbb{P}(E_1) \geq 1 - \zeta_1 j^{-1/4}. \quad \text{for all } j \in \mathcal{S}, \tag{2.40}$$

We now examine the other condition in Theorem 2.13, namely

$$\text{dist}\left(0, \nabla f(w_j) + \partial\Psi(w_j)\right) \leq \bar{\epsilon}.$$

By adding and subtracting terms, we obtain

$$\begin{aligned}\nabla f(x_j) + a_j = (\nabla f(w_j) - \nabla f(w^*)) + (\nabla f(w^*) - \bar{g}_{j-1}) - \frac{\beta_{j-1}}{j-1} b_j \\ + \left(\bar{g}_{j-1} + a_j + \frac{\beta_{j-1}}{j-1} b_j\right).\end{aligned} \tag{2.41}$$

for any $a_j \in \partial\Psi(w_j)$ and $b_j \in \partial h(w_j)$. We choose the specific a_j and b_j that satisfy the optimality of the subproblem (2.19), that is,

$$0 = \bar{g}_{j-1} + a_j + \frac{\beta_{j-1}}{j-1} b_j. \tag{2.42}$$

This choice eliminates the last term in (2.41). For the other three terms, we have the following observations.

(i) For those w_j satisfying E_1, the Lipschitz property of ∇f (Lemma 2.2) implies that

$$\|\nabla f(w_j) - \nabla f(w^*)\| \leq L\|w_j - w^*\| \leq \frac{L}{3\max(L,1)}\bar{\epsilon} \leq \frac{\bar{\epsilon}}{3}.$$

Hence, E_1 implies the following event:

$$E_2: \quad \|\nabla f(w_j) - \nabla f(w^*)\| \leq \bar{\epsilon}/3$$

(ii) From Theorem 2.12, we have by setting $\epsilon = \bar{\epsilon}/3$ and $t = j - 1$ that

$$\mathbb{P}(\|\nabla f(w^*) - \bar{g}_{j-1}\| > \bar{\epsilon}/3) < \zeta_2 j^{-1/4}, \quad \forall j \geq 2.$$

Hence, denoting by E_3 the event

$$E_3: \quad \|\nabla f(w^*) - \bar{g}_{j-1}\| \leq \bar{\epsilon}/3,$$

we have that

$$\mathbb{P}(\neg E_3(j)) < \zeta_2 j^{-1/4}, \quad \text{for all } j \geq 2. \tag{2.43}$$

(iii) Since $\beta_{j-1} = \gamma(j-1)^{1/2}$, we have for w_j satisfying E_1 that

$$\begin{aligned}
\frac{\beta_{j-1}}{j-1}\|b_j\| &= \gamma(j-1)^{-1/2}\|b_j\| \\
&\leq \gamma\eta(j-1)^{-1/2}\|w_j - w_1\| \qquad \text{(from (2.38))} \\
&\leq \gamma\eta(j-1)^{-1/2}(\|w_j - w^*\| + \|w_1 - w^*\|) \\
&\leq \gamma\eta(j-1)^{-1/2}\left(\frac{\bar{\epsilon}}{3\max(L,1)} + D\right).
\end{aligned}$$

Therefore, E_1 implies the event

$$E_4: \quad \frac{\beta_{j-1}}{j-1}\|b_j\| \leq \frac{\bar{\epsilon}}{3},$$

whenever $j \geq j_0$, where we define j_0 by

$$j_0 := 1 + \left\lceil \frac{9\gamma^2\eta^2}{\bar{\epsilon}^2}\left(\frac{\bar{\epsilon}}{3\max(L,1)} + D\right)^2 \right\rceil.$$

Therefore for $j \in \mathcal{S}$ with $j \geq j_0$, by definition of the events E_1, E_2, E_3, and E_4 above, the probability that Theorem 2.13 will hold is

$$\begin{aligned}
&\mathbb{P}\Big(\|w_j - w^*\| \leq \bar{\epsilon} \ \wedge \ \text{dist}\,(0, \partial\phi(w_j)) < \bar{\epsilon}\Big) \\
&\geq \ \mathbb{P}\Big(E_1 \wedge E_2 \wedge E_3 \wedge E_4\Big) = \mathbb{P}(E_1 \wedge E_3) \\
&\geq \ 1 - \mathbb{P}(\neg E_1) - \mathbb{P}(\neg E_3) \geq 1 - (\zeta_1 + \zeta_2) j^{-1/4}
\end{aligned}$$

where the last inequality is due to (2.40) and (2.43). Our claim follows. ■

Theorem 2.18 (Identification for Strongly Convex Cases) Suppose that Assumption 2.10 holds, ϕ is strongly convex with the modulus $\sigma > 0$, $w^* \in \mathcal{F}_D$, $\mathcal{M}$ is the optimal manifold including w^*, and the RDA algorithm uses $h(w)$ satisfying (2.38). Given $\mathcal{S}'$ defined in (2.34), we have

$$\mathbb{P}(w_j \in \mathcal{M}) \geq 1 - (\zeta_1' + \zeta_2') \left(\frac{1 + \ln j}{j} \right)^{1/2}.$$

for all $j \in \mathcal{S}'$ sufficiently large, where

$$\zeta_1' := \left(\frac{3 \max(1, L)}{\bar{\epsilon}} \right)^2, \text{ and } \zeta_2' := 1.25 \left(\frac{3}{\bar{\epsilon}} \right)^2 \nu'.$$

Proof This proof is almost identical to that of Theorem 2.17; here we briefly mention the required changes for the strongly convex case. Consider $\bar{\epsilon} > 0$ and the event E_1 defined in the proof of Theorem 2.17. From Lemma 2.16 we have

$$\mathbb{P}(\|w_j - w^*\| \leq \bar{\epsilon}) \geq \mathbb{P}(E_1) \geq 1 - \zeta_1'[(1 + \ln j)/j]^{1/2}, \quad \text{for all } j \in \mathcal{S}'.$$

Instead of (ii) and (iii) in the proof of Theorem 2.17, we use the following:

(ii') From Theorem 2.12, we have by setting $\epsilon = \bar{\epsilon}/3$ and $t = j - 1$ that

$$\mathbb{P}(\|\nabla f(w^*) - \bar{g}_{j-1}\| > \bar{\epsilon}/3) < \zeta_2'[(1 + \ln j)/j]^{1/2}, \quad \forall j \geq 2.$$

Hence, denoting by E_3 the event $\|\nabla f(w^*) - \bar{g}_{j-1}\| \leq \bar{\epsilon}/3$, we have that

$$\mathbb{P}(\neg E_3) < \zeta_2'[(1 + \ln j)/j]^{1/2}, \quad \text{for all } j \geq 2.$$

(iii') With $\beta_{j-1} = \sigma(1 + \ln(j - 1))$, we can choose j_0' sufficiently large so that the event E_4 in the proof of Theorem 2.17 holds for all $j \geq j_0'$.

Using the modified probability bounds for E_1 and E_3, we have

$$\begin{aligned} P\Big(\|w_j - w^*\| \leq \bar{\epsilon} \ \wedge \ \text{dist}\,(0, \partial\phi(w_j)) < \bar{\epsilon} \Big) &\geq P\Big(E_1 \wedge E_3 \Big) \\ &\geq 1 - (\zeta_1' + \zeta_2')[(1 + \ln j)/j]^{1/2}, \end{aligned}$$

for all $j \in \mathcal{S}'$ with $j \geq j_0'$. Our claim follows. ■

Lemma 2.15 (respectively, Lemma 2.16) tells us that the sequence $\mathcal{S}$ (resp., $\mathcal{S}'$) is "dense" in $\{1, 2, \dots\}$, while Theorem 2.17 (resp., Theorem 2.18) states that for all sufficiently large $j \in \mathcal{S}$ (resp., $j \in \mathcal{S}'$), w_j lies on the optimal manifold with probability approaching one as j increases.

2.5 Dual Averaging with Manifold Identification

We present a simple strategy motivated by our analysis above, in which the RDA method gives way to a local optimization phase after a near-optimal manifold is identified.

2.5.1 RDA$^+$ Algorithm

Algorithm 2 summarizes our algorithm called RDA$^+$. This algorithm starts with RDA steps until it identifies a near-optimal manifold, then switching to the LPS algorithm (Wright, 2010) to search a reduced space until an optimality criterion is satisfied.

For choosing a manifold as a candidate, we use a simple heuristic inspired by Theorem 2.17 that if the past τ consecutive iterates have been on the same manifold $\mathcal{M}$, we take $\mathcal{M}$ to be approximately optimal. Before commencing the local phase, however, we "safeguard" by expanding $\mathcal{M}$ to incorporate additional dimensions that may yet contain the minimizer. Our simple approach will work provided that $\mathcal{M}$ is a *superset* of the optimal manifold, since LPS is able to move to more restricted submanifolds of $\mathcal{M}$.

2.5.2 Specification for ℓ_1-regularization

We describe the details of Algorithm 2 for ℓ_1-regularization, where $\Psi(w) = \lambda\|w\|_1$ for some $\lambda > 0$. (Thus $w_1 = 0$.) This is a simple and widely used regularizer that encourages sparsity in the solution w^*. The manifold embracing $w^* \in \mathbb{R}^n$ corresponds to the set of points in $\mathbb{R}^n$ that have the same sign and nonzero patterns as w^*. Also, we use the simple quadratic prox-function $h(w) = \|w - w_1\|^2$.

Computation of w_{j+1}: For these choices, we have a closed-form solution for the subproblem (2.19):

$$[w_{j+1}]_i = \frac{\sqrt{j}}{2\gamma}\text{soft}(-[\bar{g}_j]_i,\, \lambda), \quad i = 1, 2, \ldots, n,$$

where $\text{soft}(u, a) := \text{sgn}(u)\max\{|u| - a, 0\}$ is the well-known soft-threshold function.

Acceleration: To generate the approximate solution in the local optimization phase of Algorithm 2, we use an empirical estimate $\tilde{\phi}_{\mathcal{N}}$ in (2.5) as a surrogate objective function and then solve

$$\min_{w \in \mathcal{M}} \quad \tilde{\phi}_{\mathcal{N}}|_{\mathcal{M}}(w),$$

where $\mathcal{N}$ is drawn from available samples. LPS calculates first- and second-order information for $\tilde{\phi}_{\mathcal{N}}$ on the subset of components defined by $\mathcal{M}$. Since the intrinsic dimension of $\mathcal{M}$ is usually much smaller than the dimension n of the full space, these restricted gradients and Hessians are much cheaper to compute than their full-space counterparts.

Checking Optimality: From the optimality condition for (2.5), we define the optimality measure $\delta(w_j)$:

$$\delta(w_j) := \frac{1}{\sqrt{n}} \inf_{a_j \in \partial\Psi(w_j)} \|\nabla \tilde{f}_{\mathcal{N}}(w_j) + a_j\|. \tag{2.45}$$

Since $\delta(w^*) \approx 0$ for sufficiently large sample set $\mathcal{N}$ because of the law of large numbers, we can stop the algorithm when $\delta(w_j)$ drops below a certain threshold.

Safeguarding: For a more robust implementation, we augment $\mathcal{M}$ before starting the local phase, adding components i for which $[w_{j+1}]_i = 0$ but $[\bar{g}_j]_i$ is close to one of the endpoints of its allowable range; that is,

$$[w_{j+1}]_i = 0 \text{ and } |[\bar{g}_j]_i| > \rho\lambda \tag{2.46}$$

for some fixed $\rho \in (0, 1]$. This is motivated from Theorem 2.12, which indicates that $\bar{g}_j$ approaches $\nabla f(w^*)$ in probability as j increases.

Algorithm 2 RDA$^+$ Algorithm.

Input:

- a prox-function $h(w)$ that is strongly convex on $\operatorname{dom}\Psi$ and also satisfies

$$\arg\min_w h(w) \in \arg\min_w \Psi(w),$$
$$\sup_{b_j \in \partial h(w_j)} \|b_j\| \le \eta \|w - w_1\|, \quad \forall w \in \operatorname{dom}\Psi.$$

- a nonnegative and nondecreasing sequence $\{\beta_t\}, t \ge 1$.
- a positive integer τ.

Initialize: set $w_1 \in \arg\min_w \Psi(w)$ and $\bar{g}_0 = 0$.

Dual Averaging:

for $j = 1, 2, \ldots$ **do**

  Choose a random vector $\xi_j \in \Xi$.

  Compute a gradient $g_j \leftarrow \nabla F(w_j; \xi_j)$.

  Update the average gradient:

$$\bar{g}_j = \frac{j-1}{j}\bar{g}_{j-1} + \frac{1}{j} g_j.$$

  Compute the next iterate by solving the subproblem (2.19), which is

$$w_{j+1} = \arg\min_{w \in \mathbb{R}^n} \left\{ \langle \bar{g}_j, w \rangle + \Psi(w) + \frac{\beta_t}{t} h(w) \right\}. \tag{2.44}$$

  if there is $\mathcal{M}$ such that $w_{j+2-i} \in \mathcal{M}$ for $i = 1, 2, \ldots, \tau$ **then**

    Local Phase:

    Expand $\mathcal{M}$ and use LPS to search for solution on manifold $\mathcal{M}$, starting at w_{j+1};

  end if

end for

2.6 Computational Experiment

We consider binary classification tasks via logistic regression with ℓ_1-regularization. We construct the empirical estimate $\tilde{\phi}_{\mathcal{N}}$ from the full training set of size m, that is, $\mathcal{N} = \{1, 2, \ldots, m\}$. For the training example selected by ξ_t at time $t \geq 1$, we use its feature vector $x_t \in \mathbb{R}^{n-1}$ and label $y_t \in \{-1, 1\}$ to define the corresponding loss function for $\tilde{w} \in \mathbb{R}^{n-1}$, $b \in \mathbb{R}$ and $w = (\tilde{w}, b)$:

$$F(w; \xi_t) = \log\left(1 + \exp\left(-y_t(\tilde{w}^T x_t + b)\right)\right).$$

We choose $\Psi(w) = \lambda \|w\|_1$ as the regularizer for some $\lambda > 0$, and set $w_1 = 0$.

2.6.1 Manifold Identification

To investigate the identification behavior of the RDA algorithm in practical circumstances, we use five data sets from the UCI Machine Learning Repository with various sizes and dimensions shown in Table 2.1. We first apply the original LPS to acquire the solution $w^*_{\mathcal{N}}$ of (2.5), with the tight optimality threshold of 10^{-6}. We then tabulate how many iterations of RDA are required before it generates a point in the optimal manifold $\mathcal{M}$ containing $w^*_{\mathcal{N}}$.

We also check when the iterate of the RDA algorithm lies on a modest superset of the optimal manifold. Here we aim for a "$2\times$" superset composed of the points in $\mathbb{R}^n$ having the same sign pattern for the active components in $\mathcal{M}$, and up to twice as many nonzeros as the points in $\mathcal{M}$. For each data set we use three values of λ equally spaced in the log-scale range of $[0.3, 0.9]\lambda_{\max}$, where the $\lambda_{\max}$ is computed accordingly to Koh et al. (2007) beyond which the solution contains no nonzero components except for the intercept term.

Table 2.1 shows the number of iterations taken by the RDA algorithm, for 100 different random permutations of each data, until the algorithm identifies the optimal manifold and its $2\times$ superset. As the empirical distributions of the iterations are skewed, we show the median (rather than the mean) and the standard deviation. The table also shows the values of the optimality measure δ defined in (2.45) for the iterate at the moment we identify the optimal manifold. These results tell us that identifying the optimal manifold could require a huge number of iterations, whereas identifying the superset is often much easier. We note too that even when optimal identification is

Table 2.1 Manifold identification of the RDA algorithm for 100 different permutations. The median (and the standard deviation in parentheses) number of iterations until the identification of the optimal manifold $\mathcal{M}$ containing $w^*_{\mathcal{N}}$ and its $2\times$ superset are presented. δ represents the optimality measure at the moment of identifying $\mathcal{M}$.

Data set	λ	No. iterations				Optimality δ	NNZs
		$2\times$ Superset		Optimal $\mathcal{M}$			$w^*_{\mathcal{N}}$
`Glass` ($m = 214, n = 10$)	0.29	14	(25)	20	(27)	0.068	1
	0.17	13	(10)	116	(428)	0.063	2
	0.10	13	(11)	28392	(6907)	0.016	3
`Iono` ($m = 351, n = 35$)	0.22	38	(84)	122	(95)	0.015	2
	0.13	44	(28)	30812	(15575)	0.008	3
	0.07	86	(41)	404	(150)	0.019	5
`Arrhythmia` ($m = 452, n = 280$)	0.15	192	(110)	304	(141)	0.001	2
	0.09	272	(88)	2036	(1076)	0.002	8
	0.05	447	(195)	27750	(4590)	0.001	13
`Spambase` ($m = 4601, n = 58$)	0.17	137	(219)	357	(325)	0.006	1
	0.10	722	(2495)	4340	(3097)	0.004	8
	0.06	812	(1247)	4680	(2209)	0.004	17
`Pageblock` ($m = 5473, n = 11$)	0.11	26	(326)	58	(395)	0.063	1
	0.07	182	(941)	524	(1233)	0.038	3
	0.04	103	(913)	461	(1232)	0.040	4

achieved, the iterate is still far from being optimal, suggesting the need for alternatives to achieve tighter optimality.

2.6.2 Performance on the MNIST Data Set

We now focus on the effects of the local phase in the RDA$^+$ algorithm in Algorithm 2. For this we use the MNIST data set which consists of gray-scale images of digits represented by 28×28

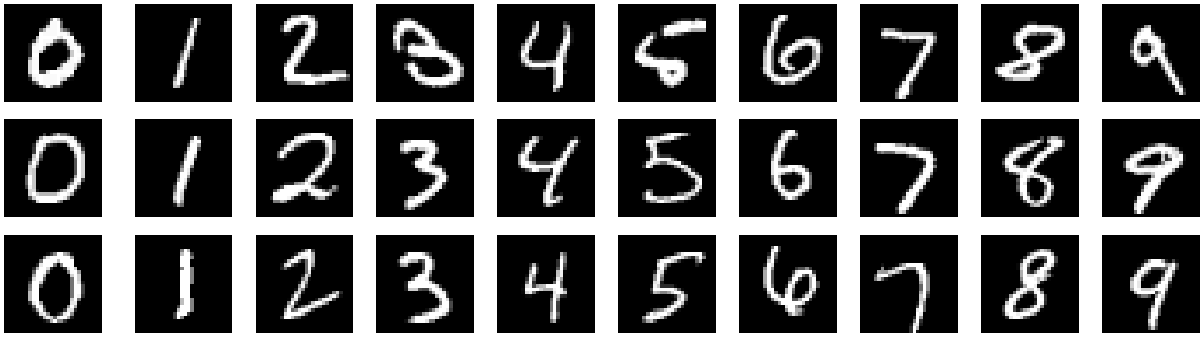


Figure 2.1 Examples of the input digits of MNIST data set.

pixels, choosing the digits 6 and 7 to have 12183 training and 1986 test examples. We show some digits from the data set in Figure 2.1. Although the MNIST data set has relatively small dimension, we choose this set because we can compare our results to those reported in Xiao (2010) for the original RDA algorithm.

We compare RDA$^+$ to several other algorithms: SGD, TG, RDA, and LPS. The *stochastic gradient descent* (SGD) method (for instance, Nemirovski et al., 2009) for ℓ_1-regularization consists of the iterations

$$[w_{t+1}]_i = [w_t]_i - \alpha_t\big([g_t]_i + \lambda \text{sgn}\left([w_t]_i\right)\big),\ i = 1, 2, \ldots, n.$$

The iteration of the *truncated gradient* (TG) algorithm (Langford et al., 2009) is described in (2.7), where use $\theta = \infty$ and $K = 10$ for enhanced regularization effect (Xiao, 2010). For the stepsize α_t in SGD and TG, we adopt a variable stepsize scheme (Zinkevich, 2003; Nemirovski et al., 2009), choosing

$$\alpha_t = \frac{D}{G}\sqrt{\frac{2}{t}}.$$

This gives SGD a bound $2\sqrt{2}GD\sqrt{t}$ for the regret $R_t(w^*)$ when $\lambda \ll G$ (see Appendix A.3 for details). This bound is comparable to the simplified regret bound of RDA, $R_t(w^*) \leq 2GD\sqrt{t}$, which is obtained with the best $\gamma^* = G/D$ that minimizes the expression $(\gamma D^2 + G^2/\gamma)$ in (2.20) for the general convex cases. (Then we can set the stepsizes of SGD and TG as $\alpha_t = (\gamma^*)^{-1}\sqrt{2/t}$.) In this setting we can run RDA$^+$ and RDA estimating only γ^* without knowing D and G; we use $\gamma^* = 5000$ for both, which is determined by cross validation with RDA and with a single scan

through the data set. For LPS and the local phase of RDA$^+$, we set the Newton threshold to 200 so that the Newton approach will be applied only when the number of active elements fall below this number, and use no sampling in the gradient and Hessian computation. We set $w_1 = 0$ for all algorithms.

For SGD, TG, and RDA, we keep track of the primal average $\bar{w}_T := \frac{1}{T}\sum_{t=1}^{T} w_t$ additionally, where T for each algorithm denotes the iteration number where the algorithm is stopped. We include these into comparison because the convergence of the stochastic subgradient algorithms are often described in term of the primal averages but we do not expect they will identify the optimal manifold as fast as the raw iterates in general. Note that RDA$^+$ and LPS does not operate with primal averages.

We first run the RDA$^+$ algorithm with random permutations of the training samples, stopping when $\tau = 100$ consecutive iterates have the same sparsity pattern, after seeing all samples at least once. (All repeated runs required at most 19327 iterations to stop, which was within two permutations.) In the safeguarding test (2.46), we use $\rho = 0.85$. Then we run the local phase of the RDA$^+$ algorithm until we obtain a solution with the optimality measure value in (2.45) less than 10^{-4}. We record the total runtime of the RDA$^+$ algorithm, and then run other algorithms SGD, TG, and RDA, up to the runtime of RDA$^+$ (they may stop earlier if they achieve the desired optimality).

Progress in Time: We compare the convergence speed of the algorithms, in terms of the optimality measure and the number of nonzero components. For this experiment, we run LPS up to the runtime of RDA$^+$ and include it into comparison. Figure 2.2 presents the plots for the iterates without averaging, for three different values of $\lambda \in \{10, 1.0, 0.1\}$. The optimality plots (on the left) show that RDA$^+$ achieves the target optimality much faster than other algorithms, including LPS. The RDA algorithm behaves better than SGD and TG, but it still hardly achieves the target value. Also, it is hard to notice the decreasing tendency of optimality values for SGD, TG and RDA, although it might be more distinct in a larger timeframe.

The plots on the right of Figure 2.2 show the number of nonzeros in the iterates. RDA tends to produce much sparser iterates with less fluctuation than SGD and TG, but it fails to reduce the

number of nonzeros to the smallest number identified by RDA$^+$ in the given time, apparently for $\lambda = 1.0$ and $\lambda = 0.1$.

We mark the events of switching to the local phase for RDA$^+$ with black dot-dashed lines. In the local phase, RDA$^+$ behaves very similarly to LPS, sharing the typical behavior of nonmonotonic decrease in optimality. However, the local phase often converges faster than LPS, because it can operate on the reduced space chosen by the initial phase of RDA$^+$. The number of nonzeros often increases in the event of switching, since the safeguarding can add more elements. This behavior can be diminished by using more conservative (larger) ρ values.

Figure 2.3 shows similar plots but for the averaged iterates (primal averages). We duplicate the plots of RDA$^+$ and LPS from Figure 2.2 for easy comparison. Regarding the number of nonzeros components, it is easy to tell the primal averages behave much worse than the iterates without averaging in finite time.

Quality of Solutions: Next we compare the quality of the solutions in terms of optimality, the number of nonzeros, and test error rate, in Figure 2.4. We present the results for the iterates without averaging on the left, and those for the primal averages on the right, averaged up to each observation point from the beginning. (The plots of RDA$^+$ and LPS on the left are duplicated to the right for easy comparison.) We run the algorithms with the same setting used in the previous experiments, except for LPS; now we run LPS without any time limit to use it as the baseline of comparison. (The runtime of LPS was about four times longer than that of RDA$^+$ on average.) The experiments are repeated for 100 different random seeds, for each of the seven λ values in the range of $[0.01, 10]$. ($\lambda_{\max}$ was 45.8.)

In Figure 2.4, only the solutions from RDA$^+$ achieve the desired optimality and the smallest number of nonzeros, with almost identical quality to the solutions from LPS. The solutions (both with and without averaging) from SGD, TG, and RDA are suboptimal, leaving much scope for zeroing out many more components of the iterates. RDA achieves a similar number of nonzeros to RDA$^+$ for large λ values, but more nonzeros on smaller values of λ. In terms of the test error

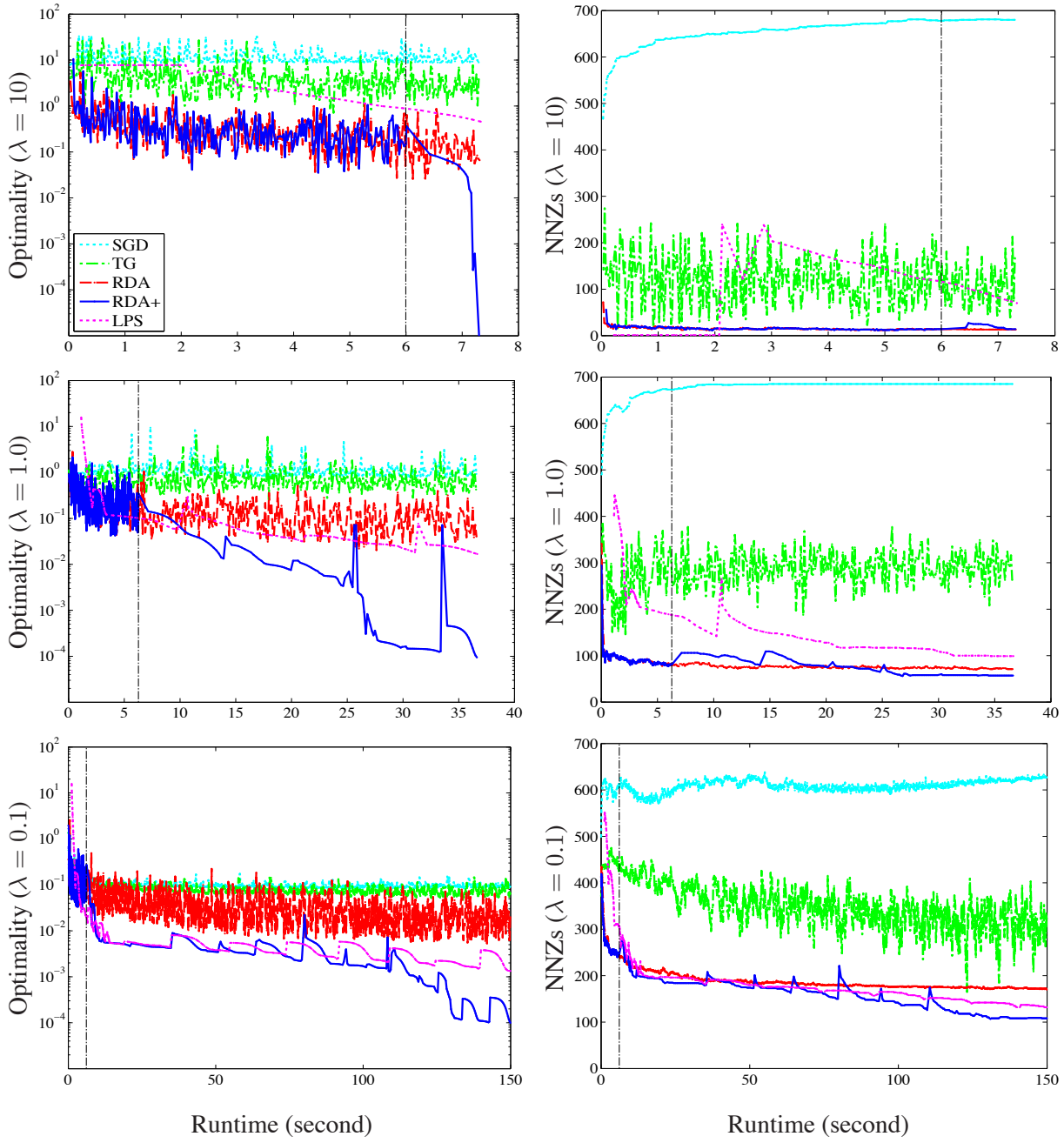


Figure 2.2 Convergence of iterates without averaging, for the task classifying the digits 6 and 7 in the MNIST data set. Convergence is measured in terms of the optimality measure (left) and the number of nonzero components in the iterates (right). SGD, TG, RDA and LPS are run up to the time taken for RDA$^+$ to achieve 10^{-4} optimality value. The black dot-dashed lines indicate the event of phase switching in RDA$^+$. Only the vertical axes on the left are in logarithmic scale.

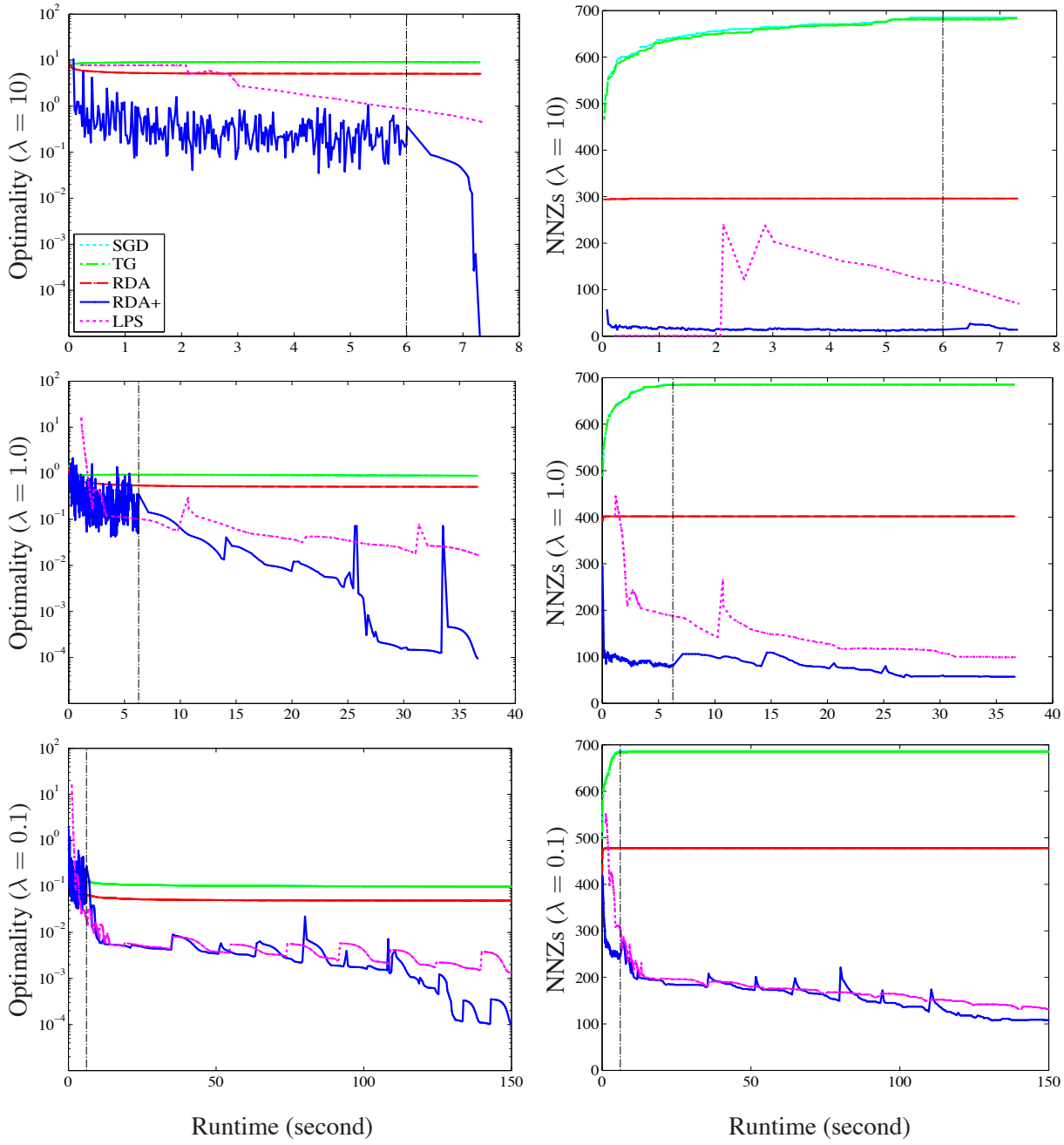


Figure 2.3 Convergence of averaged iterates, for the task classifying the digits 6 and 7 in the MNIST data set. Convergence is measured in terms of the optimality measure (left) and the number of nonzero components in the iterates (right). SGD, TG, RDA and LPS are run up to the time taken for RDA$^+$ to achieve 10^{-4} optimality value. The plots of RDA$^+$ and LPS are duplicated from Figure 2.2 for comparison. Only the vertical axes on the left are in logarithmic scale.

rate, RDA$^+$ produces slightly better solutions than SGD, TG, and RDA overall. Although the improvement is marginal, we note that high accuracy is difficult to achieve solely with the stochastic online learning algorithms in limited time. The averaged iterates of SGD and TG show smaller test error for $\lambda \geq 1$ than others, but they need a large number of nonzero components, despite the strong regularization imposed.

In Figure 2.5 we show the typical solutions from the algorithm runs, for the different λ values. The first three rows present the solutions acquired without averaging, and the last three rows present the ones obtained with primal averaging. The solutions from RDA$^+$ reveals almost identical sparsity pattern to those from the baseline algorithm LPS, achieving smallest nonzero patterns. For large λ values, RDA produces similar sparse solutions to RDA$^+$, but much denser solutions for smaller values. The solutions from primal averaging are typically denser than those without averaging.

2.6.3 Scaling Performance

In this section we benchmark RDA$^+$ and LPS in terms of the speed to achieve the solutions with 10^{-4} thresholds of convergence, varying problem sizes. Recall that the LPS algorithm is a batch optimization solving (2.5), operating with the full gradient in each iteration, possibly with the Hessian information, in the basic setting. However, it also features options for large-scale problems, to work with partial gradients and with the Hessian estimates evaluated with subsets of the given training set. Also, the behavior of the local phase of RDA$^+$ is affected by these parameters. We are interested in the efficiency of the two algorithms with the changes of these parameters.

For the benchmark, we use the entire MNIST data set with 60k examples, creating binary classification tasks differentiating even versus odd digits. The training sets of five different sizes in the range of $[10k, 50k]$ are created by random sampling from the entire set, maintaining the even versus odd ratio of the entire set. We measure the runtime of RDA$^+$ and LPS to achieve the solutions with 10^{-4} optimality, for 50 random training sets for each size. We use $\gamma^* = 5000$ as before, and change two parameters of LPS: the gradient fraction (*gf*) which controls the fraction of

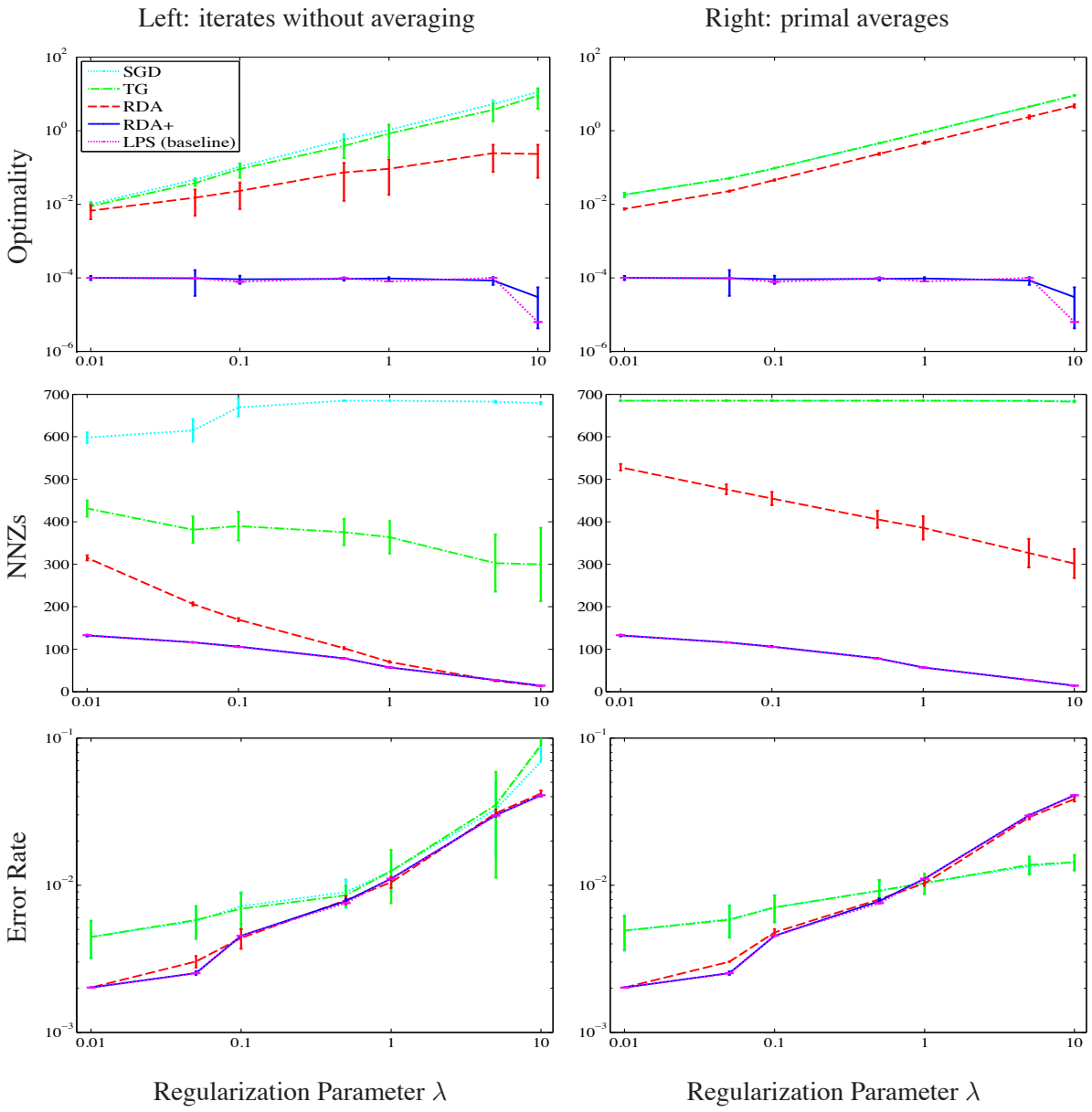


Figure 2.4 Quality of the solutions for classifying the digits 6 and 7 in the MNIST data set, in terms of the optimality, the number of nonzero components, and the test error rate (measured for 100 different random permutations). The plots on the left show the results for the iterates without averaging, and those on the right for the primal averages. SGD, TG, and RDA algorithms are run up to the time taken for RDA$^+$ to achieve 10^{-4} optimality value, whereas LPS is run without such limit. The plots for RDA$^+$ and LPS on the left are duplicated to the right for comparison. All axes are in logarithmic scale, except for the vertical axes in the second row.

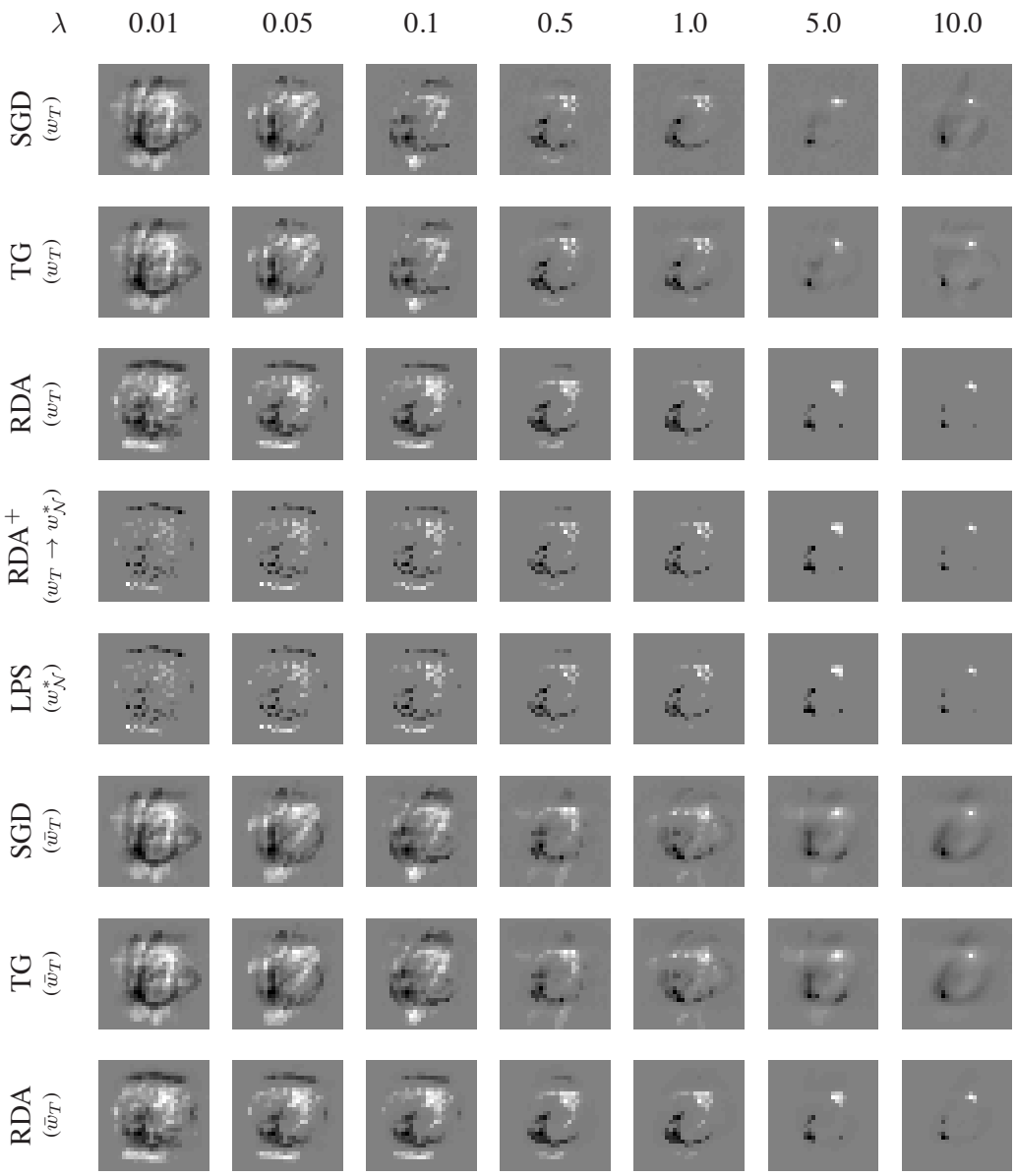


Figure 2.5 Sparsity patterns of the solutions, for classifying the digits 6 and 7 in the MNIST data set. The regularization parameter λ is varied in the range of $[0.01, 10]$. The spots represent the positive (bright) and negative (dark) values, where the gray background represents the zero value. The top three rows show the solutions acquired without averaging, and the bottom three rows show the ones from primal averaging. The two rows in the middle presents the solutions from RDA$^+$ and LPS. The algorithms SGD, TG, and RDA are run up to the time taken for RDA$^+$ to achieve a solution with 10^{-4} optimality value; the batch algorithm LPS is run without time limit.

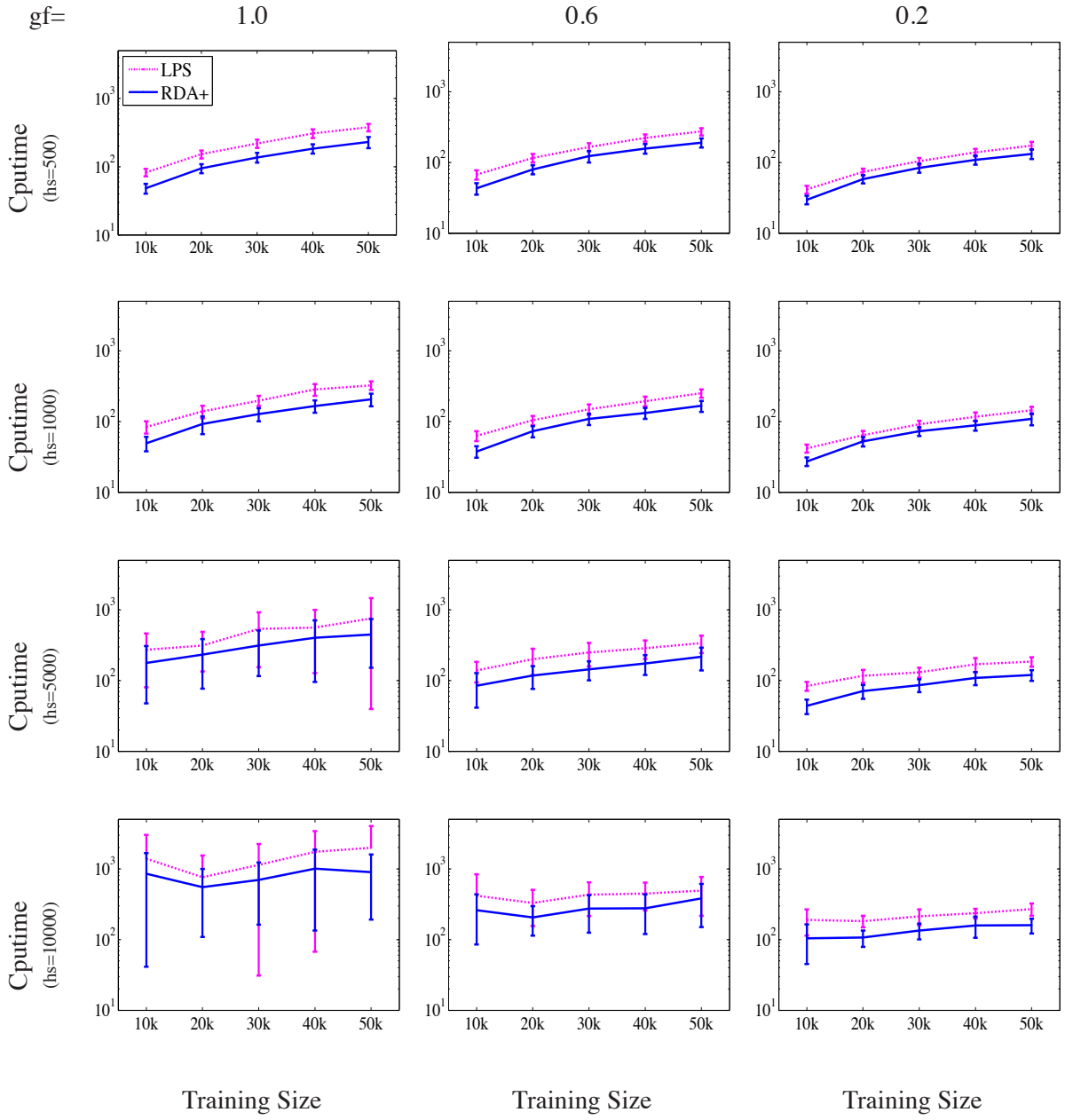


Figure 2.6 Scaling benchmark of RDA$^+$ and LPS. For even versus odd classification tasks, the training sets with different sizes in the range of $[10k, 50k]$ are created by random sampling from the entire MNIST data set, maintaining the original even versus odd ratio. The runtime of both algorithms are measured for 50 repetitions for each training set size, to achieve solutions with 10^{-4} optimality. Two large-scale parameters affecting LPS and the local phase of RDA$^+$ are varied: the gradient fraction (gf) and the Hessian sampling (hs). The vertical axes show runtime values in seconds, in logarithmic scale.

partial gradient to the full gradient; and the Hessian sampling (*hs*) which specifies the number of random training examples to be used for estimating the true Hessian.

We present the benchmark results in Figure 2.6, trying three gf values in $\{0.2, 0.6, 1.0\}$ and four hs values in the range of $[500, 10000]$. The best performance of LPS is acquired with small partial gradients (gf $= 0.2$) and rough Hessian estimates (hs $= 500$ or hs $= 1000$), but it is still about twice as slow as RDA$^+$ with the same settings. The parameter setting of Section 2.6.2 is similar to the configuration with gf $= 1.0$ and hs $= 10000$ (in the bottom left corner of Figure 2.6). This seems to be the slowest setting on average, and the experiment could have been performed about ten times faster by choosing different parameters, for instance gf $= 0.2$ and hs $= 500$. For the parameter values significantly smaller than those tried here, we experienced unstable convergence of LPS and the local phase of RDA$^+$.

2.7 Conclusion

We have shown that the RDA algorithm is effective for producing solutions with a smaller set of active elements than other subgradient methods, and also identifies the optimal manifold with probability approaching one as iterations proceed. This observation enables us to apply alternative optimization techniques with faster convergence rate on the near-optimal manifold, enabling more rapid convergence to near-optimal points than plain stochastic gradient approaches.

Chapter 3

Stochastic Subgradient Algorithms for Training Nonlinear Support Vector Machines

The support vector machine solvers using the subgradients of the primal form of SVMs are very successful in finding solutions for large-scale problems with linear kernels. However, the genuine SVM primal formulation does not satisfy the strong convexity property which is required for the existing algorithms. Moreover, the strong convexity breaks down when some tuning parameters approach zero in value.

In this chapter, we suggest an algorithmic framework equipped with a subgradient method which does not require strong convexity yet exhibits similar or even better performance to the existing methods in practice for the typical choices of the tuning parameters. Our framework utilizes the ideas of low dimensional approximation of kernels as well, which extends our method to nonlinear kernels both in batch and online settings.

3.1 Introduction

When data sets are extremely large, the computation required by some of the existing algorithms for training SVMs becomes excessive. We focus on subgradient methods that take simple steps, each typically based on a single training point, so can be implemented in a data-streaming context efficiently. While requiring a great many iterates to find accurate solutions, subgradient methods can calculate solutions that are "accurate enough" for the purposes at hand using much less computation than frameworks that more explicitly target an exact solution.

This chapter outlines an improved algorithm based on subgradient methods for solving primal SVM formulations. It extends current subgradient methods by allowing nonlinear kernels to be used, and not requiring strict convexity of the function to be minimized. This allows the classic SVM formulation with a non-penalized intercept term to be used, thus reclaiming the formulation on which many theoretic results have been built.

Our approach uses low-dimensional approximations to nonlinear kernels, obtained either by approximating the Gram matrix, or by constructing the subspace with random bases. The approximation yields a *linear* formulation with transformed feature vectors, which can be solved with the use of well known subgradient approaches. The approach has the added benefits that the approximate solution to the SVM yields an approximate classification function that can be evaluated cheaply, typically in time proportional to the dimension of approximation.

3.2 Nonlinear SVM in the Primal

In this section we develop a general theory for nonlinear SVM in the primal form focusing on classification, then show how it reformulates to a linear SVM problem by means of the low-dimensional approximation of a kernel. We discuss techniques for approximating the kernel and classifying data points efficiently.

3.2.1 Basic Derivation

Here we justify the primal SVM formulation with kernels, which was first introduced by Chapelle (2007). This analysis addresses a special case of the representer theorem (Kimeldorf and Wahba, 1970; Wahba, 1990), using the tools of convex analysis. Consider the training point and label pairs $\{(x_i, y_i)\}_{i=1}^m$, $x_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$ and feature mapping $\phi : \mathbb{R}^d \to \mathbb{R}^n$. Given a convex loss function $\ell : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$, the primal SVM problem can be formulated as follows:

$$\min_{w \in \mathbb{R}^n, b \in \mathbb{R}} \ \frac{\lambda}{2} w^T w + \frac{1}{m} \sum_{i=1}^{m} \ell(y_i(w^T \phi(x_i) + b)) \tag{3.1}$$

with $\lambda > 0$ and a convex loss function ℓ. The necessary and sufficient optimality conditions are

$$\lambda w + \frac{1}{m}\sum_{i=1}^{m} \chi_i y_i \phi(x_i) = 0, \tag{3.2a}$$

$$\frac{1}{m}\sum_{i=1}^{m} \chi_i y_i = 0, \tag{3.2b}$$

$$\text{for some} \quad \chi_i \in \partial\ell(y_i(w^T\phi(x_i) + b)), \quad i = 1, 2, \ldots, m. \tag{3.2c}$$

where $\partial\ell$ is the subdifferential of ℓ.

We now consider the following substitution:

$$w = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i) \tag{3.3}$$

(which mimics the form of (3.2a)) and, motivated by this expression, we formulate the following problem

$$\min_{\alpha\in\mathbb{R}^m, b\in\mathbb{R}} \ \frac{\lambda}{2}\alpha^T Q\alpha + \frac{1}{m}\sum_{i=1}^{m} \ell(Q_{i\cdot}\alpha + y_i b), \tag{3.4}$$

where $Q \in \mathbb{R}^{m\times m}$ is defined by

$$Q_{ij} := y_i y_j \phi(x_i)^T \phi(x_j), \ \ i, j = 1, 2, \ldots, m, \tag{3.5}$$

and $Q_{i\cdot}$ denotes the i-th row of Q. Optimality conditions for (3.4) are as follows:

$$\lambda Q\alpha + \frac{1}{m}\sum_{i=1}^{m} \beta_i Q_{i\cdot}^T = 0, \tag{3.6a}$$

$$\frac{1}{m}\sum_{i=1}^{m} \beta_i y_i = 0, \tag{3.6b}$$

$$\text{for some } \beta_i \in \partial\ell(Q_{i\cdot}\alpha + y_i b),\ i = 1, 2, \ldots, m. \tag{3.6c}$$

The following result shows that the solution of (3.4) can be used to derive a solution of (3.1). This observation is potentially interesting because (3.4) is formulated in terms of the kernel Q and does not require explicit knowledge of the feature mapping ϕ.

Proposition 3.1 Let $(\alpha, b) \in \mathbb{R}^m \times \mathbb{R}$ be a solution of (3.4). Then if we define w by (3.3), $(w, b) \in \mathbb{R}^n \times \mathbb{R}$ is a solution of (3.1).

Proof Since (α, b) solves (3.4), the conditions (3.6) hold, for some β_i, $i = 1, 2, \ldots, m$. To prove the claim, it suffices to show that (w, b) and χ satisfy (3.2), where w is defined by (3.3) and $\chi_i = \beta_i$ for all $i = 1, 2, \ldots, m$.

By substituting (3.5) into (3.6), we have

$$\lambda \sum_{i=1}^{m} y_i y_j \phi(x_j)^T \phi(x_i) \alpha_i + \frac{1}{m} \sum_{i=1}^{m} \beta_i y_i y_j \phi(x_j)^T \phi(x_i) = 0, \;\; j = 1, 2, \ldots, m,$$

$$\frac{1}{m} \sum_{i=1}^{m} \beta_i y_i = 0,$$

$$\beta_i \in \partial \ell \left(\sum_{j=1}^{m} y_i y_j \phi(x_j)^T \phi(x_i) \alpha_j + y_i b \right), \;\; i = 1, 2, \ldots, m.$$

From the first equality above, we have that

$$-\sum_{i=1}^{m} \left(\alpha_i + \frac{1}{\lambda m} \beta_i \right) y_i \phi(x_i) + \xi = 0,$$

for some $\xi \in \text{Null}\left(\left[y_j \phi(x_j)^T \right]_{j=1}^{m} \right)$. Since the two components in this sum are orthogonal, we have

$$0 = \left\| \sum_{i=1}^{m} \left(\alpha_i + \frac{1}{\lambda m} \beta_i \right) y_i \phi(x_i) \right\|^2 + \xi^T \xi,$$

which implies that $\xi = 0$. We can therefore rewrite the optimality conditions for (3.4) as follows:

$$\sum_{i=1}^{m} \left(\lambda \alpha_i + \frac{1}{m} \beta_i \right) y_i \phi(x_i) = 0, \tag{3.7a}$$

$$\frac{1}{m} \sum_{i=1}^{m} \beta_i y_i = 0, \tag{3.7b}$$

$$\beta_i \in \partial \ell \left(y_i \phi(x_i)^T \sum_{j=1}^{m} \alpha_j \phi(x_j) + y_i b \right), \;\; i = 1, 2, \ldots, m. \tag{3.7c}$$

By defining w as in (3.3) and setting $\chi_i = \beta_i$ for all i, we see that (3.7) is identical to (3.2), as claimed. ■

While Q is clearly symmetric positive semidefinite, the proof makes no assumption about non-singularity of this matrix, or uniqueness of the solution α of (3.4). However, (3.6a) suggests that

without loss of generality, we can constrain α to have the form $\alpha_i = -\beta_i/(\lambda m)$ where β_i is restricted to $\partial\ell$. In particular, if we use hinge loss function, that is,

$$\ell(\delta) := \max\{0, 1-\delta\}, \tag{3.8}$$

the subdifferential is

$$\partial\ell(\delta) = \begin{cases} \{-1\} & \text{if } \delta < 1, \\ [-1, 0] & \text{if } \delta = 1, \\ \{0\} & \text{if } \delta > 1. \end{cases}$$

Thus $\beta_i \in [-1, 0]$ for all $i = 1, 2, \ldots, m$.

3.2.2 Reformulation to a Linear SVM Problem

We consider the feature mapping $\phi^\circ : \mathbb{R}^d \to \mathcal{H}$ to a Hilbert space $\mathcal{H}$ induced by a kernel function $\kappa^\circ : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, where κ° satisfies the conditions of Mercer's theorem (Mercer, 1909) to guarantee the existence of ϕ° satisfying $\kappa^\circ(s, x) := \langle \phi^\circ(s), \phi^\circ(x) \rangle$. Suppose that we have a low-dimensional approximation $\phi : \mathbb{R}^d \to \mathbb{R}^n$ of ϕ° for which

$$\kappa^\circ(s, x) \approx \phi(s)^T \phi(x), \tag{3.9}$$

for all inputs s and x of interest. We construct a matrix $V \in \mathbb{R}^{m \times n}$ for training examples $x_1, x_2, \ldots, x_m$ by defining the i-th row as $V_{i\cdot} = y_i \phi(x_i)^T,\ i = 1, 2, \ldots, m$. Then V satisfies

$$Q := VV^T \approx Q^\circ := [y_i y_j \kappa^\circ(x_i, x_j)]_{i,j=1,2,\ldots,m}. \tag{3.10}$$

Note that Q is a rank-n, positive semidefinite approximation to Q°. By substituting this Q in (3.4), we obtain

$$\min_{\alpha \in \mathbb{R}^m, b \in \mathbb{R}} \ \frac{\lambda}{2} \alpha^T VV^T \alpha + \frac{1}{m} \sum_{i=1}^{m} \ell(v_i^T V^T \alpha + y_i b), \tag{3.11}$$

where $v_i := V_{i\cdot}^T$ is the transpose of the i-th row of V. By introducing the change of variables

$$\gamma = V^T \alpha, \tag{3.12}$$

we obtain the equivalent formulation

$$\min_{\gamma\in\mathbb{R}^n, b\in\mathbb{R}} \frac{\lambda}{2}\gamma^T\gamma + \frac{1}{m}\sum_{i=1}^{m}\ell(v_i^T\gamma + y_i b). \tag{3.13}$$

This problem is a *linear* SVM for the new input vectors $y_i v_i \in \mathbb{R}^n, i = 1, 2, \ldots, m$.

We can solve (3.13) by applying linear SVM techniques to find (γ, b). Any α that solves the overdetermined system (3.12) will yield a solution of (3.11). (Note that α satisfying (3.12) need have at most n nonzeros.) In Section 3.2.4, we provide an efficient way to classify data points without recovering α.

3.2.3 Approximating the Kernel

We discuss two techniques for finding V that satisfies (3.10). The first uses randomized linear algebra to calculate a low-rank approximation to the scaled Gram matrix Q°, as in (3.10). The second approach approximates the feature mapping $\phi^\circ(\cdot)$ explicitly by approximate feature mapping $\phi(\cdot)$ constructed using random projections.

3.2.3.1 Kernel Matrix Approximation

Our first approach is to use the Nyström method (Drineas and Mahoney, 2005), to find a good approximation of specified rank n to the $m \times m$ matrix Q° in (3.10). In this approach, we specify some integer s with $n \leq s < m$, and choose s elements at random from the index set $\{1, 2, \ldots, m\}$ to form a subset $\mathcal{S}$. We then find the best rank-n approximation to $(Q^\circ)_{\mathcal{S}\mathcal{S}}$ and denote it by $W_{s,n}$, with pseudo-inverse $W_{s,n}^+$. We then choose V so that

$$VV^T = (Q^\circ)_{\cdot\mathcal{S}} W_{s,n}^+ (Q^\circ)_{\cdot\mathcal{S}}^T, \tag{3.14}$$

where $(Q^\circ)_{\cdot\mathcal{S}}$ denotes the column submatrix of Q° defined by the indices in $\mathcal{S}$. The results in Drineas and Mahoney (2005) indicate that in expectation and with high probability, the rank-n approximation obtained by this process has an error that can be made as close as we wish to the *best* rank-n approximation by choosing s sufficiently large.

We calculate $W_{s,n}$ by forming the eigen-decomposition $(Q^\circ)_{\mathcal{S}\mathcal{S}} = PDP^T$, where P is $s \times s$ orthogonal and n is a diagonal matrix with decreasing nonnegative diagonal entries. Taking $\bar{n}$ to

be the smaller of n and the number of positive diagonals in n, we then have that

$$W_{s,n} = P_{\cdot,1..\bar{n}} D_{1..\bar{n},1..\bar{n}} P^T_{\cdot,1..\bar{n}},$$

(where $P_{\cdot,1..\bar{n}}$ denotes the first $\bar{n}$ columns of P, and so on). The pseudo-inverse is thus

$$W^+_{s,n} = P_{\cdot,1..\bar{n}} D^{-1}_{1..\bar{n},1..\bar{n}} P^T_{\cdot,1..\bar{n}}. \tag{3.15}$$

The matrix V satisfying (3.14) is therefore

$$V = (Q^\circ)_{\cdot\mathcal{S}} P_{\cdot,1..\bar{n}} D^{-1/2}_{1..\bar{n},1..\bar{n}}. \tag{3.16}$$

For practical implementation, rather than defining n a priori, we can choose a positive threshold ϵ_n with $0 < \epsilon_n \ll 1$, then choose n to be the largest integer in $1, 2, \ldots, s$ such that $D_{nn} \geq \epsilon_n$. (In this case, we have $\bar{n} = n$.)

Counting the time complexity of a single kernel evaluation $\kappa^\circ(s, x)$ for some $s, x \in \mathbb{R}^d$ as $O(k)$, the time complexity of the kernel approximation discussed above is $O(s^3 + sm(k+n))$. This consists of (i) a cost of $O(s^3)$ for the PDP^T factorization of $(Q^\circ)_{\mathcal{SS}}$, (ii) $O(smk)$ for computation of $(Q^\circ)_{\cdot,\mathcal{S}}$, and (iii) $O(smn)$ for the matrix multiplication of (3.16). Note that the cost of (ii) and (iii) dominates the cost of (i) since $n \leq s \ll m$.

3.2.3.2 Feature Mapping Approximation

The second approach, following Rahimi and Recht (2008), finds a mapping $\phi : \mathbb{R}^d \to \mathbb{R}^n$ that satisfies

$$\langle \phi^\circ(s), \phi^\circ(x) \rangle = \mathbb{E}\left[\langle \phi(s), \phi(x) \rangle \right],$$

where the expectation on the right hand side is over the random variables that determine ϕ. The mapping ϕ can be constructed explicitly by random projections. We write

$$\phi(x) = \sqrt{1/n} \left[\cos(\nu_1^T x + \beta_1), \cdots, \cos(\nu_n^T x + \beta_n) \right]^T \tag{3.17}$$

where $\nu_1, \ldots, \nu_n \in \mathbb{R}^d$ are i.i.d. samples from a distribution with density $p(\nu)$ and $\beta_1, \ldots, \beta_n \in \mathbb{R}$ are from the uniform distribution on $[0, 2\pi]$. The density function $p(\nu)$ is determined by the types

of the kernels we want to use. For the Gaussian kernel

$$\kappa^\circ(s,x) = \exp(-\|s-x\|^2/(2\sigma^2)), \tag{3.18}$$

it can be shown that

$$p(\nu) = \frac{1}{(2\pi)^{n/2}\sigma^2} \exp\left(-\frac{\|\nu\|^2}{2\sigma^2}\right),$$

from the Fourier transformation of κ°. We can define the matrix V satisfying (3.10) by setting

$$v_i := V_{i\cdot}^T = y_i\phi(x_i), \quad i = 1,2,\ldots,m, \tag{3.19}$$

thus setting up the formulation (3.13).

This method is suitable for online settings, since it takes only $O(dn)$ to prepare $\nu_1,\ldots,\nu_d$ (assuming that sampling each component of these vectors takes constant time) and $O(n)$ to process each data point. As we observe in Section 3.4, however, this approach tends to give lower classification accuracy than the first approach.

3.2.4 Efficient Classification

Given the solution (γ, b) of (3.4), we now describe how a new data point $x \in \mathbb{R}^d$ can be classified efficiently. The imposed low dimensionality of the approximate kernel in our approach can lead to significantly lower cost of classification, as low as a fraction of d/m of the cost of a full-space approach.

For the feature mapping approximation of Section 3.2.3.2, where ϕ is defined explicitly by (3.17), we use the classifier f suggested immediately by (3.1), that is, $f(x) = w^T\phi(x) + b$. By substituting from (3.3) and using the definition (3.19), we obtain

$$f(x) = \phi(x)^T \sum_{i=1}^m \alpha_i y_i \phi(x_i) + b = \phi(x)^T V^T \alpha + b = \phi(x)^T\gamma + b,$$

where we used (3.12) for the final equality. Note in particular that the classifier can be evaluated directly from γ; there is no need to recover α explicitly.

For the kernel approximation approach of Section 3.2.3.1, the classifier $w^T\phi(x) + b$ cannot be used directly, as we have no way to evaluate $\phi(x)$ for an arbitrary point x. We can however use the

approximation (3.9) to note that

$$\begin{aligned}\phi(x)^T w + b &= \sum_{i=1}^{m} \alpha_i y_i \phi(x)^T \phi(x_i) + b \\ &\approx \sum_{i=1}^{m} \alpha_i y_i \kappa^\circ(x_i, x) + b, \end{aligned} \tag{3.20}$$

so we can define the function (3.20) to be the classifier. To evaluate this function, we need only evaluate those kernels $\kappa^\circ(x_i, x)$ for which $\alpha_i \neq 0$. As noted in Section 3.2.2, we can satisfy (3.12) by using just n nonzero components of α, so (3.20) requires only n kernel evaluations.

If we set $\alpha_i = 0$ for all components $i \notin \mathcal{S}$, where $\mathcal{S}$ is the sample set from Section 3.2.3, we can compute α that approximately satisfies (3.12) without performing further matrix factorizations. Denoting the nonzero subvector of α by $\alpha_{\mathcal{S}}$, we have $V^T\alpha = V_{\mathcal{S}\cdot}^T \alpha_{\mathcal{S}} = \gamma$, so from (3.16) and the fact that $(Q^\circ)_{\mathcal{S}\mathcal{S}} = PDP^T$, we have

$$\gamma = D_{1..\bar{n},1..\bar{n}}^{-1/2} P_{\cdot,1..\bar{n}}^T (Q^\circ)_{\mathcal{S}\mathcal{S}} \alpha_{\mathcal{S}} = D_{1..\bar{n},1..\bar{n}}^{1/2} P_{\cdot,1..\bar{n}}^T \alpha_{\mathcal{S}}.$$

An approximate solution of this equation (which is exact when $\bar{n} = n = s$) is

$$\alpha_{\mathcal{S}} = P_{\cdot,1..\bar{n}} D_{1..\bar{n},1..\bar{n}}^{-1/2} \gamma.$$

3.3 Stochastic Approximation Algorithm

We describe here a stochastic approximation algorithm for solving the linear SVM reformulation (3.13). Consider the general convex optimization problem

$$\min_{z \in Z} f(z), \tag{3.21}$$

where f is a convex function and Z is a bounded closed convex set with the radius D_Z defined by

$$D_Z := \max_{z \in Z} ||z||_2. \tag{3.22}$$

The subdifferential of f at z is denoted by $\partial f(z)$, and we use $g(z)$ to denote a particular subgradient. By convexity of f, we have

$$f(z') - f(z) \geq g(z)^T (z' - z), \;\; \forall z, z' \in Z, \; \forall g(z) \in \partial f(z).$$

f is *strongly* convex when there exists $\mu > 0$ such that

$$(z' - z)^T [g(z') - g(z)] \geq \mu ||z' - z||^2, \tag{3.23}$$

for all $z, z' \in Z$, all $g(z) \in \partial f(z)$, and all $g(z') \in \partial f(z')$. Note that the objective in (3.13) is strongly convex in γ, but only weakly convex in b. PEGASOS (Shalev-Shwartz et al., 2007) requires f to be strongly convex in all variables and modifies the SVM formulation to have this property. The approach we describe below is suitable for the original SVM formulation.

3.3.1 The Algorithm

The algorithm assumes that at any $z \in Z$, we have available $G(z; \xi)$, a stochastic subgradient estimate depending on random variable ξ that satisfies $E[G(z; \xi)] = g(z)$ for some $g(z) \in \partial f(z)$. The norm deviation of the stochastic subgradients is measured by D_G defined as follows:

$$E[\|G(z; \xi)\|^2] \leq D_G^2 \quad \forall z \in Z. \tag{3.24}$$

At iteration j, the general algorithm takes the following step:

$$z^{j+1} = \Pi_Z(z^j - \eta_j G(z^j; \xi^j)), \quad j = 1, 2, \ldots, \tag{3.25}$$

where ξ^j is a random variable (i.i.d. with the random variables used at previous iterations), Π_Z is the projection onto Z, and $\eta_j > 0$ is a step length. For our function (3.13), we set $z^j = (\gamma^j, b^j)$, selecting ξ^j to be one of the indices $\{1, 2, \ldots, m\}$ with equal probability, and construct the subgradient estimate from the subgradient for the ξ^j-th term in the summation. Specifically, when the hinge loss ℓ from (3.8) is used, we have

$$G\left(\begin{bmatrix} \gamma^j \\ b^j \end{bmatrix}; \xi^j\right) = \begin{bmatrix} \lambda\gamma^j + d_j v_{\xi^j}^T \\ d_j y_{\xi^j} \end{bmatrix}, \tag{3.26}$$

where $d_j = -1$ if the kernelized training point ξ^j is currently misclassified and $d_j = 0$ otherwise. We define the set Z to be the Cartesian product of a ball in the γ component (due to strong duality, similarly to Shalev-Shwartz et al. (2007)) with an interval $[-B, B]$ for the b component:

$$Z = \left\{ \begin{bmatrix} \gamma \\ b \end{bmatrix} \in \mathbb{R}^n \times \mathbb{R} : \ \|\gamma\| \leq 1/\sqrt{\lambda}, \ |b| \leq B \right\}$$

for sufficiently large $B > 0$, resulting in $D_Z = \sqrt{1/\lambda + B^2}$.

The solution of (3.21) is estimated not by the iterates z^j but rather by a weighted sum of the final few iterates. Specifically, if we define N to be the total number of iterates to be used and $K < N$ to be the point at which we start averaging, the final reported solution estimate would be

$$\bar{z}_K^N := \frac{\sum_{t=K}^N \eta_t z^t}{\sum_{t=K}^N \eta_t}. \tag{3.27}$$

These is no need to store all the iterates z^t, $t = K, K+1, \ldots, N$ in order to evaluate (3.27). Instead, a running average can be maintained over the last $N - K$ iterations, requiring the storage of only a single x.

The steplengths η_j require knowledge of the subgradient estimate variance D_G (3.24). We use a small sample of random variables $\xi^{(l)}$, $l = 1, 2, \ldots, M$, at the first iterate (γ^0, b^0), and estimate D_G^2 as

$$E\left\| G\left(\begin{bmatrix} \gamma^0 \\ b^0 \end{bmatrix}; \xi \right) \right\|^2 \approx \frac{1}{M^2} \sum_{l=1}^M d_l^2 (\|v_{\xi^{(l)}}\|^2 + y_{\xi^{(l)}}^2).$$

We summarize this framework in Algorithm 3 and refer it to as ASSET. The integer $K > 0$ specifies the iterate at which the algorithm starts averaging the iterates, which can be set to 1 to average all iterates, to a predetermined maximum iteration number to output the last iterate without averaging, or to a number in between.

3.3.2 Convergence

The analysis of robust stochastic approximation (Nemirovski et al., 2009; Nemirovski and Yudin, 1983) provides theoretical support for the algorithm above. Considering Algorithm 3 applied to the general formulation (3.21), and denoting the algorithm's output $\bar{z}_K^N := (\bar{\gamma}_K^N, \bar{b}_K^N)$, we have the following result.

Theorem 3.2 Given the output $\bar{z}_K^N$ and optimal function value $f(z^*)$, we have

$$E[f(\bar{z}_K^N) - f(z^*)] \leq C(\rho) \frac{D_Z D_G}{\sqrt{N}} \tag{3.28}$$

where $C(\rho)$ solely depends on the fraction $\rho \in (0, 1)$ for which $K = \lceil \rho N \rceil$.

Algorithm 3 ASSET Algorithm

Input: $T = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, Q°, λ, positive integers K and N with $0 < K < N$, and D_Z and D_G satisfying (3.22) and (3.24);

Set $(\gamma^0, b^0) \leftarrow (\mathbf{0}, 0)$, $j \leftarrow 1$;

Set $(\bar{\gamma}, \bar{b}) \leftarrow (\mathbf{0}, 0)$, $\bar{\eta} = 0$;

for $j = 1, 2, \ldots, N$ **do**

  $\eta_j \leftarrow \frac{D_Z}{D_G\sqrt{j}}$

  Choose $\xi^j \in \{1, \ldots, m\}$ independently at random.

  $$v_{\xi^j} = \begin{cases} V^T_{\xi^j \cdot} & \text{for } V \text{ as in (3.16), if we use the kernel matrix approximation.} \\ y_{\xi^j}\phi(x_{\xi^j}) & \text{for } \phi(\cdot) \text{ as in (3.17), if we use the feature mapping approximation.} \end{cases}$$

  $$d_j \leftarrow \begin{cases} -1 & \text{if } v_{\xi^j}\gamma^j + y_{\xi^j} b < 1 \\ 0 & \text{otherwise} \end{cases}$$

  $$\begin{bmatrix} \gamma^j \\ b^j \end{bmatrix} \leftarrow \Pi_Z \left(\begin{bmatrix} (1 - \eta_j\lambda)\gamma^{j-1} - \eta_j d_j v_{\xi^j} \\ b^{j-1} - \eta_j d_j y_{\xi^j} \end{bmatrix} \right)$$

  if $j \geq K$ **then**

    {update averaged iterate}

$$\begin{bmatrix} \bar{\gamma} \\ \bar{b} \end{bmatrix} \leftarrow \frac{\bar{\eta}}{\bar{\eta} + \eta_j} \begin{bmatrix} \bar{\gamma} \\ \bar{b} \end{bmatrix} + \frac{\eta_j}{\bar{\eta} + \eta_j} \begin{bmatrix} \gamma^j \\ b^j \end{bmatrix}.$$

$$\bar{\eta} \leftarrow \bar{\eta} + \eta_j.$$

  end if

end for

Define $\bar{\gamma}^N_K := \bar{\gamma}$ and $\bar{b}^N_K := \bar{b}$.

3.3.3 Strongly Convex Case

Suppose that we omit the intercept b from the linear formulation (3.13). Then its objective function $f(x)$ becomes strongly convex for all of its variables. In this special case we can apply

different steplength $\eta_j = 1/(\lambda j)$ to achieve faster convergence in theory. The algorithm remains the same as Algorithm 3 except that averaging is no longer needed and a faster convergence rate can be proved: essentially a rate of $1/j$ rather than $1/\sqrt{j}$ (see Nemirovski et al. (2009) for a general proof). And the set Z is simplified as follows

$$Z = \{\gamma \in \mathbb{R}^n : \|\gamma\| \leq 1/\sqrt{\lambda}\},$$

and the update steps are changed accordingly to omit the component b. The resulting algorithm, we refer it as ASSET*, is the same as PEGASOS (Shalev-Shwartz et al., 2007), except for our extension to nonlinear kernels.

Note that averaging like (3.27) may still be useful, as it can be shown to improve the convergence rate by some constant (Polyak and Juditsky, 1992).

3.4 Computational Results

We based our implemented our algorithms on the open-source PEGASOS code[1]. We refer our algorithms with kernel matrix approximation as ASSET and ASSET* (for the versions that do and do not allow an intercept term, resp.) and with feature mapping approximation as ASSET$_{on}$ and ASSET$^*_{on}$. In the interests of making direct comparisons with other codes, we do not include intercept terms in our experiments, since some of the other codes do not allow such terms to be used without penalization.

We run all experiments on load-free 64-bit Linux systems with 2.66 GHz processors and 8 GB memory. Kernel cache size is set to 1 GB when applicable. All experiments with randomness are repeated 50 times unless otherwise specified.

Table 3.1 summarizes the six binary classification tasks we use for the experiments[2]. The `ADULT` data set is randomly split into training/validation/test sets. In the `MNIST` data set, we obtain a binary problem by classifying the digits 0-4 versus 5-9. In the `CCAT` data set from the RCV1 collection (Lewis et al., 2004), we use the original test set as the training set, and divide the original

[1] Our code is available at `http://pages.cs.wisc.edu/~sklee/asset/`.
PEGASOS is from `http://mloss.org/software/view/35/`.

[2] `ADULT`, `MNIST`, `CCAT` and `COVTYPE` data sets are downloaded from the UCI Repository (Frank and Asuncion, 2010).

training set into validation and test sets. IJCNN is constructed by a random splitting of the IJCNN 2001 Challenge data set[3]. In COVTYPE, the binary problem is to classify type 1 against the other forest cover types. Finally, MNIST-E is an extended set of MNIST, generated with elastic deformation of the original digits[4]. Table 3.1 also indicates the values of the regularization parameter λ and Gaussian kernel parameter σ (3.18) selected by the SVM-Light solver (Joachims, 1999) to maximize the classification accuracy on each validation set. (For MNIST-E we use the same parameters as in MNIST.)

For the first five batch-mode tasks, we compare our algorithms against four publicly available codes. Two of these are the cutting-plane methods referred to by us as CPNY (Joachims et al., 2009) and CPSP (Joachims and Yu, 2009) that are implemented in the version 3.0 of of SVM-Perf. Both search for a solution as a linear combination of approximate basis functions, where the approximation is based on Nyström sampling (CPNY) or on constructing optimal bases (CPSP). The other two comparison codes are SVM-Light (Joachims, 1999), which solves the dual SVM formulation via a succession of small subproblems, and LASVM (Bordes et al., 2005), which makes a single pass over the data, selecting pairs of examples to optimize with the SMO algorithm. The original SVM-Perf (Joachims, 2006) and OCAS (Franc and Sonnenburg, 2008) are not included in the comparison because they cannot handle nonlinear kernels. For the final test — an online test with the large data set MNIST-E — we compare our online algorithms ASSET$_{on}$ and ASSET$^*_{on}$ to LASVM.

For our codes, the averaging parameter is set to $K = m - 100$ for all experiments, and the error values are computed using the efficient approximate classification schemes of Section 3.2.4.

3.4.1 Accuracy vs. approximation dimension

The first experiment investigates the effect of the dimension of the approximate kernel on classification accuracy on the test set. We set the dimension parameter s in Section 3.2.3 to values

[3] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[4] http://leon.bottou.org/papers/loosli-canu-bottou-2006/

Table 3.1 Data sets and Training Parameters.

Name	m (train)	valid/test	d	(density)	$\frac{1}{\lambda m}, \frac{1}{2\sigma^2}$
ADULT	32561	8140/8141	123	(11.2%)	1000, 0.001
MNIST	58100	5950/5950	784	(19.1%)	100, 0.01
CCAT	78127	11575/11574	47237	(1.6%)	10, 1.0
IJCNN	113352	14170/14169	22	(56.5%)	100, 1.0
COVTYPE	464809	58102/58101	54	(21.7%)	3.0, 1.0
MNIST-E	1000000	20000/20000	784	(25.6%)	100, 0.01

in the range $[2, 1024]$, with the eigenvalue threshold $\epsilon_n = 10^{-16}$. Note that s is an upper bound on the actual dimension n of approximation for ASSET$^{(*)}$, but is equal to n in the case of ASSET$^{(*)}_{on}$.

For the batch tasks, we ran our algorithms for 1000 epochs ($1000m$ iterations) so that they converged to a near-optimal value with small variation among different randomization.

The CPSP and CPNY have a parameter similar to s (as an upper bound of n); we compared by setting that parameter to the same values as for s. We obtained the baselines of batch tasks by running SVM-Light. SVM-Light do not have dimension parameters but can be expected to give the best achievable performance by the kernel-approximate algorithms as s approaches m.

Figure 3.1 shows the results. Since ASSET and ASSET* yield very similar results in all experiments, we do not plot ASSET*. (For the same reason we show only ASSET$_{on}$ for online settings.) We would expect the codes to perform well when the underlying kernel is well approximated by a low-dimensional surrogate. When σ in (3.18) is very large, as in Figure 3.2(a) of ADULT data set, all codes achieve good classification performance for small values of s. In other data sets, the chosen values of σ are smaller and the intrinsic dimension of the kernel is higher, so classification performance continues to improve as s increases. In particular, it is known that linear kernels work as well as nonlinear kernels on the CCAT. If linear kernels are optimal for CCAT, the optimal Gaussian kernel may choose a very small value of σ producing near-identity thus high-rank Gram matrix. ASSET$_{on}$ seems to suffer from approximating the kernel function rather than the kernel matrix;

the former is generally a more difficult problem. For a given dimension, the overall performance of ASSET$_{on}$ is worse than other methods, especially in the CCAT experiment.

CPSP generally requires lower dimension than the other methods to achieve the same classification performance. The power seems to come from the fact that CPSP spends extra time to construct good basis functions whereas the other methods depend on random sampling. However, all approximate-kernel methods including CPSP suffer considerably from the restriction in dimension for the COVTYPE task.

3.4.2 Speed of achieving similar test error

In performing timing comparisons, we ran all codes other than ours with their default stopping criteria. For ASSET and ASSET*, we checked the classification error on the test sets ten times per epoch, terminating when the error matched the performance of CPNY. (Since this code uses a similar Nyström approximation of the kernel, it is the one most directly comparable with ours in terms of classification accuracy.) The test error was measured using the iterate averaged over the 100 iterations immediately preceding each reporting point.

Results for the first five data sets are shown in Table 3.2 for the values $s = 512$ and $s = 1024$. (Note that LASVM and SVM-Light do not depend on s and so their results are the same in both tables.) The shortest time values to achieve similar test accuracy are marked as bold, showing that our methods are among the fastest in most cases. The best classification errors among the approximate codes are obtained by CPSP but the runtimes are considerably longer than for our methods. In fact, if we compare the performance of ASSET with $s = 1024$ and CPSP with $s = 512$, ASSET achieves similar test accuracy to CPSP (except for CCAT) but is faster by a factor between two and forty. CPNY requires an abnormally long run time on the ADULT data set; we surmise that the code may be affected by numerical difficulties associated with the highly ill conditioned kernel for this problem.

Interestingly, ASSET shows similar performance to ASSET* despite the less impressive theoretical error bound of the former. When the value of regularization parameter λ is near zero, the objective function loses strong convexity and thereby breaks the condition required for ASSET* to

work. We observe similar slowdown of PEGASOS and SGD when λ approaches zero for linear kernel SVMs.

3.4.3 Online performance on very large data sets

We take the final data set `MNIST-E` to be an online learning problem and compare the performance of ASSET$_{on}$ and ASSET$^*_{on}$ to the online SVM code LASVM. (Other algorithms such as CPSP, CPNY, and SVM-Light are less suitable for comparison because they operate in batch mode.) For a fair comparison, we fed the training samples to the algorithms in the same order.

Figure 3.2 shows the progress on a single run of our algorithms, with various approximation dimensions n in the range $[1024, 16384]$. Vertical bars in the graphs indicate the completion of training. ASSET$_{on}$ tends to converge faster and shows smaller test error values than ASSET$^*_{on}$, despite the theoretical slower convergence rate of the former. With $n = 16384$, ASSET$_{on}$ and ASSET$^*_{on}$ required 7.2 hours to finish with a solution of 2.7% and 3.5% test error, respectively. LASVM produced a better solution with only 0.2% test error, but it required 4.3 days of computation.

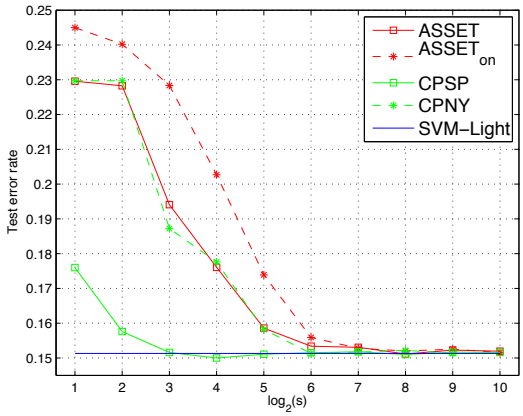
In Table 3.3, we represent the variability of solutions in the single runs of ASSET$_{on}$ and ASSET$^*_{on}$. We sampled ten test errors during the last 10000 iterations, evaluating the error values after each batch of 1000 iterations using the averaged solution over the 100 iterations immediately preceding each evaluation point. We also show the average time for the evaluations, and the total training time averaged over the single runs of ASSET$_{on}$ and ASSET$^*_{on}$ (these have negligible variations). The average error and the deviation of ASSET$_{on}$ up to $n = 8192$ tend to decrease as the dimension increases; one contributing factor would be that the approximate feature mapping approaches to the true function in exponentially increasing probability with the dimension growth (Rahimi and Recht, 2008). (This phenomenon was not as evident for ASSET$^*_{on}$.) We believe that the variability of stochastic subgradients eventually increases with dimension, leading to an increased variability in performance, as happens with $n = 16384$ for ASSET$_{on}$. Finally ASSET$_{on}$ seems to produce more accurate classifier than ASSET$^*_{on}$ with the same level of approximation.

The testing time depends on the controllable parameter n for our codes but it depends on the number of support vectors in the computed solution for LASVM (LASVM required 1504 seconds).

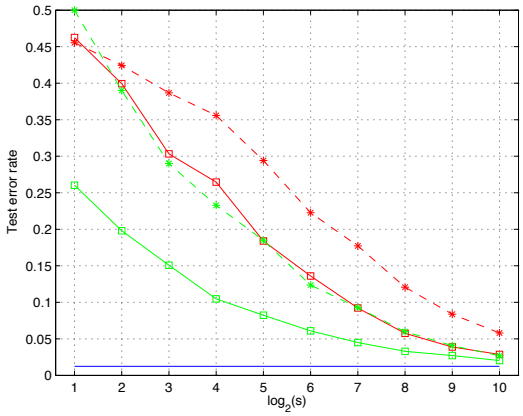
3.5 Conclusion

We have proposed a general framework for training support vector machines based on stochastic subgradients. Our algorithms can operate in batch and in online mode, and allows for the use of nonlinear kernels via kernel approximation and reformulation of the primal form. They do not require strong convexity. Our methods find solutions of reasonable quality for large problems, often in much shorter time than existing algorithms. Since the approaches require only (weak) convexity of the objective function, they can be extended easily to regression, ranking, and other learning problems.

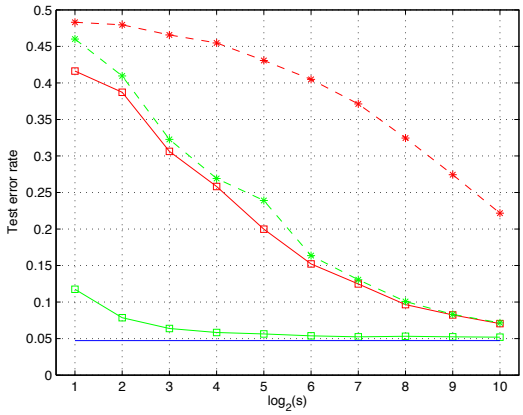
Figure 3.1 The effect of the approximation dimension n to the test error. The x-axis shows the values of the parameter s in logarithmic scale (base 2). For ASSET$_{on}$, $n = s$ and for the others $n \leq s$. The results from ASSET* and ASSET$^*_{on}$ are omitted since they are very similar to those from ASSET and ASSET$_{on}$, respectively.



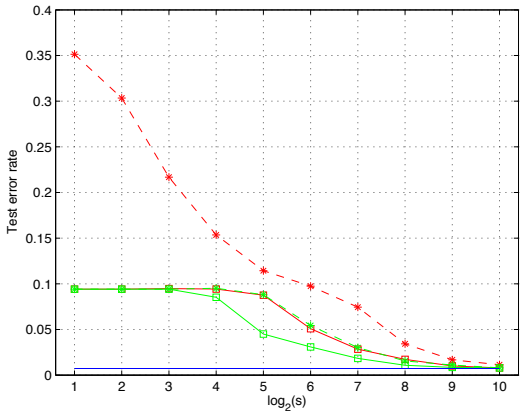
(a) ADULT



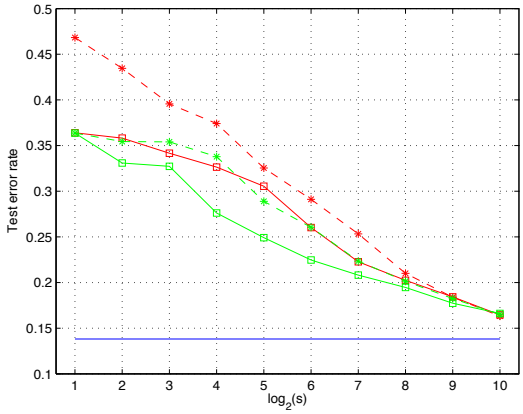
(b) MNIST



(c) CCAT



(d) IJCNN



(e) COVTYPE

Table 3.2 Training CPU time (s:seconds, h:hours) and test error in parentheses. Kernel approximation dimension is varied by setting $s = 512$ and $s = 1024$ for ASSET, ASSET*, CPSP and CPNY.

	Subgradient Methods		Cutting-plane		Decomposition	
$s = 512$	ASSET	ASSET*	CPSP	CPNY	LASVM	SVM-Light
ADULT	**22.5s** (15.06%)	23.9s (15.07±0.06%)	3020.0s (15.17%)	8.2h (15.13%)	1011.4s (18.02%)	856.8s (15.13%)
MNIST	**96.8s** (4.03%)	100.9s (4.03±0.04%)	549.6s (2.72%)	348.0s (4.07%)	587.5s (1.40%)	1322.6s (1.24%)
CCAT	95.0s (8.23%)	99.2s (8.26±0.06%)	799.9s (5.24%)	**62.0s** (8.31%)	2616.0s (4.71%)	3422.6s (4.72%)
IJCNN	**86.7s** (1.08%)	89.1s (1.08±0.02%)	726.8s (0.84%)	319.5s (1.1%)	288.1s (0.76%)	1331.3s (0.73%)
COVTYPE	697.2s (18.19%)	**585.7s** (18.18±0.07%)	1.8h (17.73%)	1841.5s (18.24%)	38.3h (13.46%)	52.7h (13.82%)
$s = 1024$	ASSET	ASSET*	CPSP	CPNY	LASVM	SVM-Light
ADULT	**77.6s** (15.10%)	83.2s (15.12±0.04%)	3398.5s (15.16%)	7.5h (15.17%)	1011.4s (18.02%)	856.8s (15.13%)
MNIST	**274.9s** (2.66%)	275.4s (2.67±0.02%)	1273.2s (2.03%)	515.4s (2.69%)	587.5s (1.40%)	1322.6s (1.24%)
CCAT	264.6s (7.09%)	278.4s (7.11±0.04%)	2949.9s (5.19%)	**122.9s** (7.15%)	2616.0s (4.71%)	3422.6s (4.72%)
IJCNN	307.1s (0.79%)	**297.0s** (0.79±0.01%)	1649.4s (0.78%)	598.0s (0.80%)	288.1s (0.76%)	1331.3s (0.73%)
COVTYPE	2259.4s (16.47%)	**2063.9s** (16.47±0.06%)	4.1h (16.61%)	3597.7s (16.52%)	38.3h (13.46%)	52.7h (13.82%)

Figure 3.2 Online progress of ASSET$_{on}$ and ASSET$^*_{on}$ to their completion (MNIST-E).

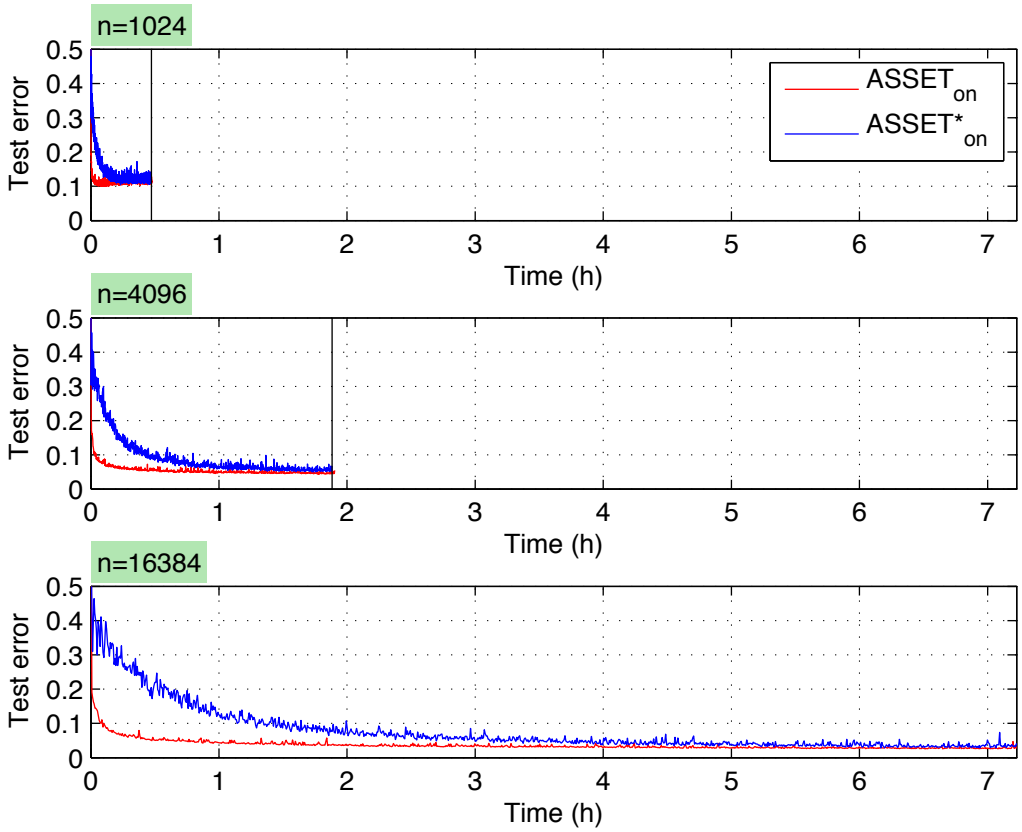


Table 3.3 Test error statistics (mean and standard deviation) for the last 10k iterations of online training (MNIST-E).

Dimension n	ASSET$_{on}$	ASSET$^*_{on}$	Avg. Time (sec)	
			Train	Test
1024	11.5±0.56%	11.8±0.57%	1705	34
2048	7.2±0.47%	8.1±0.45%	3418	69
4096	4.8±0.33%	5.8±0.54%	6824	139
8192	3.6±0.30%	4.3±1.06%	13375	270
16384	3.0±0.63%	3.5±0.29%	26053	546

Chapter 4

Decomposition Algorithms for Training Semiparametric SVMs

We describe an approach for solving large-scale semiparametric support vector machines for regression problems. Most of the approaches proposed to date for large-scale SVMs cannot be applied to semiparametric problems because of the multiple equality constraints that appear in the formulation, alongside bound constraints. Our approach uses a decomposition framework, with a primal-dual algorithm to find an approximate saddle point for the min-max formulation of each subproblem. We demonstrate that our approach scales well as the number of training examples grows, and compare with algorithms previously proposed for semiparametric SVMs.

4.1 Introduction

The power of SVM lies in the fact that it does not require the user to define the class of functions from which the observations might have been generated. In a sense, this is also a weakness, in that prior knowledge of the function class is often available for use. *Semiparametric* SVM formulations introduce parametric components into the model of the classifying / regression function, alongside the nonparametric contribution. The basis functions in the parametric part of the model can be chosen to embed prior knowledge and can be used for analyzing the effects of certain covariates, thus giving semiparametric SVM the potential advantages of both parametric and nonparametric methods.

Despite the benefits, semiparametric models have not drawn much attention from the machine learning community, possibly in part because the optimization problems arising from semiparametric SVMs are harder to solve than those generated by standard SVMs. This paper describes an efficient approach for finding solutions to large-scale semiparametric SVM problems. We focus on the formulation of semiparametric SVM regression first introduced in Smola et al. (1999), which gives rise to a dual problem which is a convex quadratic program (QP) with several equality constraints as well as bound constraints.

4.1.1 Motivation

To motivate our description of solvers for semiparametric SVMs, we discuss first the state of the art for solvers that tackle the standard SVM dual formulation, which is

$$\min_z \frac{1}{2} z^T Q z + p^T z \qquad \text{s.t.} \quad q^T z = 0, \qquad \mathbf{0} \leq z \leq c\mathbf{1} \ , \tag{4.1}$$

where z, p, q, and $\mathbf{1} := (1, 1, \ldots, 1)$ are column vectors of length n, and $c > 0$ is a given constant. Many effective algorithms for this problem solve a sequence of subproblems, each of which updates some subvector of z while leaving the remaining elements unchanged. These algorithms can be categorized into two distinct groups. In the first group, the subvector is very short, typically containing just two components. Since the subproblem can be solved analytically for such a small number of variables, no numerical solver is needed. The subproblems are inexpensive, but many iterations are usually needed to reach a solution with acceptable quality. Sequential Minimal Optimization (SMO) (Platt, 1999) and its variants such as LIBSVM (Chang and Lin, 2009) fall into this category. In the second group of solvers, the subvectors are longer, requiring the subproblems to be solved with a QP solver that exploits the structure of the application. Although we face the burden of designing an efficient, robust QP solver, methods in the second group often show faster convergence than those in the first group. Successful instances of methods in the second group include SVM-Light (Joachims, 1999) and GPDT (Serafini et al., 2004; Serafini and Zanni, 2005). The QP solvers used in the second group can be applied to the full problem, thus solving it in one "outer" iteration, though this approach is not usually effective for large data sets.

In general, the methods in both groups discussed above are specialized to handle the single equality constraint in (4.1) along with the bound constraints. The analytic subproblem solution in SMO can be acquired only when the subproblem has up to one (or two in case of the modified SMO (Keerthi and Gilbert, 2002)) equality constraint. The subproblem selection algorithm of SVM-Light strongly depends upon the existence of a single equality constraint; the same is true of GPDT, which uses a projection algorithm from Dai and Fletcher (2006). Semiparametric SVMs, however, require solution of the following generalization of (4.1):

$$\min_z \; F(z) := \frac{1}{2} z^T Q z + p^T z \qquad \text{s.t.} \quad Az = r, \qquad \mathbf{0} \leq z \leq c\mathbf{1}, \tag{4.2}$$

where $A \in \mathbb{R}^{K \times n}$ and $r \in \mathbb{R}^K$, where $K \geq 1$ is the number of parametric basis functions that we wish to include in the model. For semiparametric SVM regression, Smola et al. (1999) proposed to apply a primal-dual interior point method based on the code LOQO. The size of problems that can be handled is thus limited by the need to perform a full evaluation of the matrix Q and the need for repeated factorizations of matrices of about this size. (The approach could however be used as the inner loop of a decomposition method in the second group discussed above.) Kienzle and Schölkopf (2005) suggested a *Minimal Primal Dual* (MPD) algorithm. This algorithm use a variant of the method of multipliers to formulate a sequence of convex quadratic programs of dimension n with bound constraints only (no equalities), which are solved by a method that selects a single component for updating at each iteration. (In this sense, it is akin to the methods in the first group described above.) We give further details on MPD as we introduce our methods below. This approach does not scale well as the size n of the problem grows, but its performance can be improved by embedding it in a decomposition framework, as described below. We include both MPD and its decomposition variants in our computational tests of Sect. 4.5. In this chapter, we propose an approach that is related to MPD but that differs in several ways. First, it is a primal-dual approach; we alternate between steps in a subvector of z and steps in the Lagrange multipliers for the constraints $Az = r$. Second, subvectors of z with more than 1 element are allowed. Third, two-metric gradient projection techniques are used in taking steps in the z components. Throughout, we take account of the fact that n may be very large, that Q cannot practically be computed and stored in its entirety, and that operations involving even modest-sized submatrices of Q are expensive.

We compare our approach computationally with MPD as stand-alone solvers, and also in a decomposition framework.

The remainder of the chapter is structured as follows. In the next section, we define the semiparametric SVM regression problem and show that its dual has the form (4.2). Section 4.3 outlines the decomposition framework, while Section 4.4 describes the primal-dual method that we propose for solving the subproblems that arise from decomposition. Section 4.5 presents some computational results.

4.2 Semiparametric SVM Regression

We consider a regression problem for data $\{(x_i, y_i)\}_{i=1}^m$ where $x_i \in \mathbb{R}^N$ are feature vectors and $y_i \in \mathbb{R}$ are outcomes. We wish to find a function h that minimizes ϵ-insensitive loss function $\ell_\epsilon(h; x, y) := \max\{0, |y - h(x)| - \epsilon\}$, while maximizing the margin as in Boser et al. (1992). Following (Smola et al., 1999; Kienzle and Schölkopf, 2005), we formulate the semiparametric SVM regression problem as follows:

$$\min_{\boldsymbol{w},\boldsymbol{\beta},\boldsymbol{\xi},\boldsymbol{\xi}^*} \quad \frac{1}{2}w^T w + c\sum_{i=1}^{m}(\xi_i + \xi_i^*) \tag{4.3a}$$

$$\text{s.t.} \quad y_i - \langle w, \phi(x_i)\rangle - \sum_{j=1}^{K}\beta_j\psi_j(x_i) \le \epsilon + \xi_i \qquad \text{for } i = 1, \ldots, m \tag{4.3b}$$

$$\langle w, \phi(x_i)\rangle + \sum_{j=1}^{K}\beta_j\psi_j(x_i) - y_i \le \epsilon + \xi_i^* \qquad \text{for } i = 1, \ldots, m \tag{4.3c}$$

$$\xi \ge \mathbf{0},\ \xi^* \ge \mathbf{0}\ . \tag{4.3d}$$

where ϕ is a feature mapping function which defines a positive semidefinite kernel $\kappa(x_i, x_j) := \langle\phi(x_i), \phi(x_j)\rangle$, for all $i, j \in \{1, \ldots, m\}$, while $\{\psi_j\}_{j=1}^K$ are the basis functions for the parametric part of the model function. The model function is defined as an extended linear model of parametric and nonparametric parts, that is, $h(x) = \langle w, \phi(x)\rangle + \sum_{j=1}^K \beta_j\psi_j(x)$. We typically have $K \ll m$. If $K = 1$ and ψ_1 is a constant function, we recover the standard SVM regression problem.

The Wolfe-dual of (4.3) has the form (4.2), where

$$z = \begin{bmatrix} \alpha \\ \alpha^* \end{bmatrix} \in \mathbb{R}^{2m} \text{ for the dual vectors } \alpha \text{ and } \alpha^* \text{ of (4.3b) and (4.3c), resp.,}$$

$$p = [\epsilon - y_1, \ldots, \epsilon - y_m, \epsilon + y_1, \ldots, \epsilon + y_m]^T \in \mathbb{R}^{2m} \ ,$$

$$Q_{ij} = \begin{cases} y_i y_j \kappa(x_i, x_j) & \text{if } 1 \le i, j \le m, \text{or } m+1 \le i, j \le 2m \\ -y_i y_j \kappa(x_i, x_j) & \text{otherwise} \end{cases} \ ,$$

$$r = \mathbf{0} \ ,$$

and

$$A = \begin{bmatrix} \psi_1(x_1) & \cdots & \psi_1(x_m) & -\psi_1(x_1) & \cdots & -\psi_1(x_m) \\ \psi_2(x_1) & \cdots & \psi_2(x_m) & -\psi_2(x_1) & \cdots & -\psi_2(x_m) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \psi_K(x_1) & \cdots & \psi_K(x_m) & -\psi_K(x_1) & \cdots & -\psi_K(x_m) \end{bmatrix} \in \mathbb{R}^{K \times 2m} \ .$$

Introducing η as the Lagrange multipliers for the constraints $Az = r$ in (4.2), the Karush-Kuhn-Tucker (KKT) optimality conditions for (4.2), stated here for later reference, are as follows:

$$\left(Qz + p + A^T\eta\right)_i \ge 0 \qquad \text{if } z_i = 0 \tag{4.4a}$$

$$\left(Qz + p + A^T\eta\right)_i \le 0 \qquad \text{if } z_i = c \tag{4.4b}$$

$$\left(Qz + p + A^T\eta\right)_i = 0 \qquad \text{if } z_i \in (0, c) \tag{4.4c}$$

$$Az = r \tag{4.4d}$$

$$\mathbf{0} \le z \le c\mathbf{1} \ . \tag{4.4e}$$

If the kernel function κ is positive semidefinite, the Hessian matrix Q of (4.2) is also positive semidefinite, by definition. Therefore the objective function $F(\cdot)$ of (4.2) is convex, and as we only have linear constraints, the dual objective of (4.2) is a concave function in terms of the dual variable η. Therefore the primal-dual pair (z, η) satisfying the conditions in (4.4) is the saddle point of (4.2).

Moreover, η agrees with β in (4.3) Since η is the double dual variable of β (refer Scholkopf and Smola (2001) for details.) As our primal-dual solver discussed in Section 4.4 provides the optimal value of η, there is no need to compute β separately.

4.3 Decomposition Framework

In this section we outline the decomposition strategy, giving details of two key aspects.

4.3.1 Subproblem Definition

The convex quadratic program (4.2) becomes harder to solve as the number of variables $n := 2m$ grows (where m is the number of data points), as the Hessian Q in (4.2) is dense and poorly conditioned for typical choices of the kernel function κ. The decomposition framework can alleviate these difficulties by working with a subset $z_{\mathcal{B}}$, $\mathcal{B} \subset \{1, 2, \ldots, n\}$ of the variables at a time, fixing the other variables $z_{\mathcal{N}}, \mathcal{N} = \{1, 2, \ldots, n\} \setminus \mathcal{B}$ at their current values. We usually choose the number of elements $n_{\mathcal{B}}$ in $\mathcal{B}$ to be much smaller than n. By partitioning the data objects p, A, and Q in the obvious way, we obtain the following subproblem at outer iteration k:

$$\begin{aligned} \min_{z_{\mathcal{B}}} \quad & f(z_{\mathcal{B}}) := \frac{1}{2} z_{\mathcal{B}}^T Q_{\mathcal{B}\mathcal{B}} z_{\mathcal{B}} + (Q_{\mathcal{B}\mathcal{N}} z_{\mathcal{N}}^k + p_{\mathcal{B}})^T z_{\mathcal{B}} \\ \text{s.t.} \quad & A_{\mathcal{B}} z_{\mathcal{B}} = -A_{\mathcal{N}} z_{\mathcal{N}}^k + r, \qquad 0 \leq z_{\mathcal{B}} \leq c\mathbf{1}, \end{aligned} \tag{4.5}$$

where $z_{\mathcal{N}}^k$ contains the current values of the $\mathcal{N}$ components. This problem has the same form as (4.2); we discuss solution methods in Section 4.4.

Since our emphasis in this chapter is computational, we leave a convergence theory for this decomposition framework for future work. Suffice for the present to make a few remarks. If $\mathcal{B}$ is chosen so that the columns of $A_{\mathcal{B}}$ corresponding to components of $z_{\mathcal{B}}$ that are away from their bounds in (4.5) form a full-row-rank matrix, and if appropriate two-sided projections of $Q_{\mathcal{B}\mathcal{B}}$ are positive definite, then (4.5) has a primal-dual solution $(z_{\mathcal{B}}^*, \eta^*)$ that corresponds to a solution $(z^*, \eta^*) = (z_{\mathcal{B}}^*, z_{\mathcal{N}}^*, \eta^*)$ of (4.2), when $z_{\mathcal{N}}^k = z_{\mathcal{N}}^*$. Perturbation results can be used to derive a local convergence theory, and it may be possible to derive a global theory from appropriate generalizations of the results in Tseng and Yun (2010).

4.3.2 Working Set Selection

The selection of working set $\mathcal{B}$ at each outer iteration is inspired by the approach of Joachims (1999), later improved by Serafini and Zanni (2005). The size of the working set is fixed at some value $n_{\mathcal{B}}$, of which up to n_c are allowed to be "fresh" indices while the remainder are carried over from the current working set. Given the current primal-dual iterate (z^{k+1}, η^{k+1}), we find the indices corresponding to the nonzero components d_i obtained from the following problem:

$$\begin{aligned} \min_{d} \quad & \left(\nabla F(z^{k+1}) + (\eta^{k+1})^T A\right)^T d \\ \text{s.t.} \quad & \begin{aligned} 0 \le d_i \le 1 \qquad & \text{if } z_i^{k+1} = 0, \\ -1 \le d_i \le 0 \qquad & \text{if } z_i^{k+1} = c, \\ -1 \le d_i \le 1 \qquad & \text{if } z_i^{k+1} \in (0, c), \\ \#\{d_i | d_i \neq 0\} \le n_c. & \end{aligned} \end{aligned} \tag{4.6}$$

Note that the objective function of (4.6) is a linearization of the Lagrangian function of F at the current primal-dual pair (z^{k+1}, η^{k+1}). Our approach is motivated by the KKT conditions (4.4), and indeed can be solved by simply sorting the violations of these conditions. It contrasts with previous methods (Joachims, 1999; Serafini and Zanni, 2005; Tseng and Yun, 2010), in which the equality constraints are enforced explicitly in the working set selection subproblem. Our approach has no requirements on the size of n_c, yet it is still effective when η^{k+1} is close to the optimal value η^*.

Earlier analysis of decomposition algorithms based on working set selection schemes has been performed by Lin (2001), who shows linear convergence for the case of a single constraint, under positive definiteness assumptions on Q. Tseng and Yun (2010) proposed a decomposition framework for a formulation similar to (4.2) that includes multiple equality constraints. They present a convergence analysis which assumes that the subproblems at each step of decomposition are solved exactly, although they do not discuss techniques for solving the subproblem. Their working set selection algorithm requires relatively high complexity ($O(K^3 n^2)$) in general, compared with the $O(n \log n)$ complexity of our approach.

The (up to) n_c new components from (4.6) are augmented to a total of $n_{\mathcal{B}}$ entries by adding indices from the previous working set $\mathcal{B}$ according to a certain priority. We choose the indices of the

Algorithm 4 Decomposition Framework

1: **Initialization.** Choose an initial point z^1 of (4.2) (possibly infeasible), initial guess of the Lagrange multiplier η^1, positive integers $n_{\mathcal{B}} \geq K$ and $0 < n_c < n_{\mathcal{B}}$, and convergence tolerance `tolD`. Choose an initial working set $\mathcal{B}$.

2: **for** $k = 1, 2, \ldots$ **do**

3: **Subproblem.** Solve the subproblem (4.5) for the current working set $\mathcal{B}$, to obtain solution $z_{\mathcal{B}}^{k+1}$ together with Lagrange multiplier η^{k+1} of the equality constraints. Set $z^{k+1} = (z_{\mathcal{B}}^{k+1}, z_{\mathcal{N}}^{k})$.

4: **Gradient Update.** Evaluate the gradient of the Lagrangian of (4.2), by incrementally updating ∇F, as indicated here:

$$\nabla F(x^{k+1}) + (\eta^{k+1})^T A = \nabla F(z^k) + \begin{bmatrix} Q_{\mathcal{B}\mathcal{B}} \\ Q_{\mathcal{N}\mathcal{B}} \end{bmatrix} (z_{\mathcal{B}}^{k+1} - z_{\mathcal{B}}^{k}) + (\eta^{k+1})^T A \ .$$

5: **Convergence Check.** If the maximal violation of the KKT conditions (4.4) falls below `tolD`, terminate with the primal-dual solution (z^{k+1}, η^{k+1}).

6: **Working Set Update.** Find a new working set $\mathcal{B}$ as described in Section 4.3.2.

7: **end for**

off-bounds components ($0 < z_i^{k+1} < c$) first, and then those of lower and upper bounds. We reduce n_c as the change between two consecutive working sets decreases, as in Serafini and Zanni (2005). We observe that adaptive reduction of n_c provides better convergence of the Lagrange multiplier η^k, and helps avoid zigzagging between two working sets without making further progress. Adaptive reduction also helps not to degrade the benefit of optimizing many new components in a single decomposition step.

Our decomposition framework is summarized in Algorithm 4.

4.4 Subproblem Solver

Recalling that the decomposition framework requires both a primal solution $z_{\mathcal{B}}$ and Lagrange multipliers η to be obtained for the subproblem (4.5), we consider the following min-max formulation of (4.5):

$$\max_{\eta} \min_{z_{\mathcal{B}} \in \Omega} \quad \mathcal{L}(z_{\mathcal{B}}, \eta) \ , \tag{4.7}$$

where $\Omega = \{z \in \mathbb{R}^{n_{\mathcal{B}}} \mid \mathbf{0} \leq z \leq c\mathbf{1}\}$ and

$$\mathcal{L}(z_{\mathcal{B}}, \eta) := f(z_{\mathcal{B}}) + \eta^T (A_{\mathcal{B}} z_{\mathcal{B}} + A_{\mathcal{N}} z^k_{\mathcal{N}} - r) \ .$$

In this section we describe a primal-dual approach for solving (4.7), in which steps are taken in $z_{\mathcal{B}}$ and η in an alternating fashion. Scalings that include second-order information are applied to both primal and dual steps. We call the approach PDSG (for "Primal-Dual Scaled Gradient").

Our approach can be viewed as an extreme variant of the method of multipliers (Bertsekas, 1999), in which we do not attempt to minimize the augmented Lagrangian between updates of the Lagrange multiplier estimates, but rather take a single step along a partial, scaled, and projected gradient direction in the primal space. In describing the general form of each iteration, we use superscripts ℓ to denote iteration counts, bearing in mind that they refer to the *inner* iterations of the decomposition framework (and hence are distinct from the superscripts k of the previous section, which denote outer iterations).

$$z^{\ell+1}_{\mathcal{B}} \leftarrow z^{\ell}_{\mathcal{B}} + s(z^{\ell}_{\mathcal{B}}, \eta^{\ell}) \tag{4.8a}$$

$$\eta^{\ell+1} \leftarrow \eta^{\ell} + t(z^{\ell+1}_{\mathcal{B}}, \eta^{\ell}) \ , \tag{4.8b}$$

where $s(\cdot, \cdot)$ and $t(\cdot, \cdot)$ are steps, defined below. In computational testing, we found PDSG to be superior to methods more like traditional method-of-multiplier approaches, which would take multiple steps in $z_{\mathcal{B}}$ in between successive steps in η.

Primal Step. In the ℓ-th iteration of the subproblem solver, we choose a small sub-working set $\mathcal{W}^{\ell} \subset \mathcal{B}$ containing at most $n_{\mathcal{W}}$ elements (where $n_{\mathcal{W}}$ is a user-defined parameter), containing those indices in $\mathcal{B}$ that are among the $n_{\mathcal{W}}$ most-violated KKT conditions (4.4a)-(4.4c) for the subproblem

(4.5). We define the further subset $\bar{\mathcal{W}}^\ell$ by selecting those indices $i \in \mathcal{W}^\ell$ that are *not* at one of their bounds 0 and c. We then construct the block-diagonal $n_\mathcal{B} \times n_\mathcal{B}$ matrix H^ℓ, as follows:

$$H^\ell_{ij} = \begin{cases} Q_{ij} + \tau\delta_{ij} & \text{if } i \in \bar{\mathcal{W}}^\ell \text{ and } j \in \bar{\mathcal{W}}^\ell \\ Q_{ii} & \text{if } i = j \text{ and } i \in \mathcal{W}^\ell \setminus \bar{\mathcal{W}}^\ell \\ \infty & \text{if } i = j \text{ and } i \notin \mathcal{W}^\ell \\ 0 & \text{otherwise,} \end{cases} \tag{4.9}$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise, while τ is a small positive parameter (we use $\tau = 10^{-8}$) chosen to ensure that the "block" part of H^ℓ is numerically nonsingular. Since we apply the inverse of this matrix to the gradient in computing the step, the components of the matrix-vector product that correspond to the ∞ entries will evaluate to zero. Specifically, we obtain the search direction as follows:

$$d^\ell := z^\ell_\mathcal{B} - \Pi_\Omega \left(z^\ell_\mathcal{B} - \left(H^\ell\right)^{-1} \nabla_{z_\mathcal{B}} \mathcal{L}(z^\ell_\mathcal{B}, \eta^\ell) \right) \tag{4.10}$$

where $\Pi_\Omega(\cdot)$ is a projection operator to the set Ω, which is trivial to compute since this set is defined by simple bounds. This is essentially the two-metric gradient projection search direction (Gafni and Bertsekas, 1984) applied to the subvector defined by $\mathcal{W}^\ell$. Given this direction, the primal step s from (4.8a) is defined to be

$$s(z^\ell_\mathcal{B}, \eta^\ell) = \alpha_\ell d^\ell \ , \tag{4.11}$$

where $\alpha_\ell \in \mathbb{R}$ is the unconstrained minimizer of $\mathcal{L}(\cdot, \eta^\ell)$ along the line segment connecting $z^\ell_\mathcal{B}$ to $z^\ell_\mathcal{B} + d^\ell$.

Dual Update. The step in the dual variable η is a Newton-like step in the dual objective function for (4.5), which is

$$g(\eta) := \min_{z_\mathcal{B} \in \Omega} \mathcal{L}(z_\mathcal{B}, \eta).$$

This is a piecewise quadratic concave function. Since its second derivative does not exist, we cannot take a true Newton step. However, we use a slight modification of the procedure in Kienzle

Algorithm 5 Subproblem solver: PDSG

1: **Initialization.** Given a index set $\mathcal{B}$, choose initial points $z_{\mathcal{B}}^1$ and η^1. Choose $n_{\mathcal{W}}$ such that $1 \leq n_{\mathcal{W}} \leq n_{\mathcal{B}}$. Choose small positive convergence tolerance `tolS`.

2: **for** $\ell = 1, 2, \ldots$ **do**

3: **Sub-Working Set Selection.** Construct $\mathcal{W}^\ell$ (with at most $n_{\mathcal{W}}$ elements) and $\bar{\mathcal{W}}^\ell$ as described above.

4: **Primal-Dual Update.** Take the primal step according to (4.8a) and (4.11), then the dual step according to (4.8b) and (4.12).

5: **Convergence Check.** If the maximal KKT violation of the current primal-dual pair $(z_{\mathcal{B}}^{\ell+1}, \eta^{\ell+1})$ is less than `tolS`, exit.

6: **end for**

and Schölkopf (2005) to form a diagonal approximation G to this matrix. Their procedure progressively updates G by applying one step of Gauss-Jacobi-like procedure at each iteration of the MPD optimization scheme. Unlike MPD, our modification estimates G both internally and externally to the optimization loop. The external estimation ensures us to have an approximation with a certain quality before performing any dual updates. We refer the reader to Kienzle and Schölkopf (2005) for additional details. The dual step t in (4.8b) is thus simply

$$t(z_{\mathcal{B}}^{\ell+1}, \eta^{\ell}) = -G^{-1} \nabla_\eta \mathcal{L}(z_{\mathcal{B}}^{\ell+1}, \eta^{\ell}). \tag{4.12}$$

Our subproblem algorithm is summarized in Algorithm 5.

4.5 Experiments

We report on computational experiments that show the intrinsic benefits of the PDSG approach, as well as the benefits of the decomposition strategy, when applied to a simple semiparametric SVM regression problem. We compare PDSG with the MPD algorithm of Kienzle and Schölkopf (2005), which has slightly better performance and lower memory requirement than the interior-point-based approach used in Smola et al. (1999). We also show the advantage of semiparametric modeling on a real world problem.

Implementations. We implemented both the decomposition framework (Algorithm 4) and the PDSG subproblem solver (Algorithm 5) in C++. The code was developed by modifying the GPDT code of Serafini et al. (2004)[1], and retains many features of this code. Our code caches once-computed kernel entries for reuse, with the least-recently-used (LRU) replacement strategy. For efficiency, our subproblem solver exploits warm starting; the most recent values of the primal and dual variables are used as the starting points in the next invocation of the subproblem solver. We also implemented the MPD solver (Kienzle and Schölkopf, 2005) in C++, again basing the implementation on GPDT. Our codes can be invoked either with the decomposition framework, or in "stand-alone" mode, in which the solver is applied directly to the stated problem.

4.5.1 A Toy Problem

For the semiparametric regression test problem, we choose the modified Mexican hat function studied in (Smola et al., 1999; Kienzle and Schölkopf, 2005):

$$\omega(x) = \sin(x) + \text{sinc}\,(2\pi(x-5)) \quad .$$

To generate data, we sample the function ω at uniform random points $x_i \in \mathbb{R}$ in the interval $[0, 10]$, making m samples in total. The observations x_i's are corrupted with additive Gaussian noise ζ_i with mean 0 and standard deviation 0.2, that is, $y_i = \omega(x_i) + \zeta_i$. In the training process, we use Gaussian kernel $\kappa(x, y) = \exp(-\gamma||x - y||^2)$ with $\gamma = 0.25$, and set the insensitivity width ϵ of the loss function to $\epsilon = 0.05$, as in Smola et al. (1999). The optimal trade-off parameter value of $c = 0.5$ is found by 10-fold cross validation (CV) in Smola et al. (1999) using very small samples ($m = 50$). Since we are interested in the convergence behavior of algorithms with larger samples, we performed computational experiments with $c = 0.1$, $c = 1$, and $c = 10$. Our model is $h(x) = \langle w, \phi(x)\rangle + \sum_{j=1}^{K} \beta_j \psi_j(x)$, with two basis functions $\psi_1(x) = \sin(x)$ and $\psi_2(x) = \text{sinc}\,(2\pi(x-5))$ as in Kienzle and Schölkopf (2005).

The size of the sample data set m is varied from 500 to 100000. The subproblem size $n_{\mathcal{B}}$ and the maximum number of new components in each subproblem n_c are fixed to 500 and 100,

[1] GPDT is available at `http://mloss.org/software/view/54/`

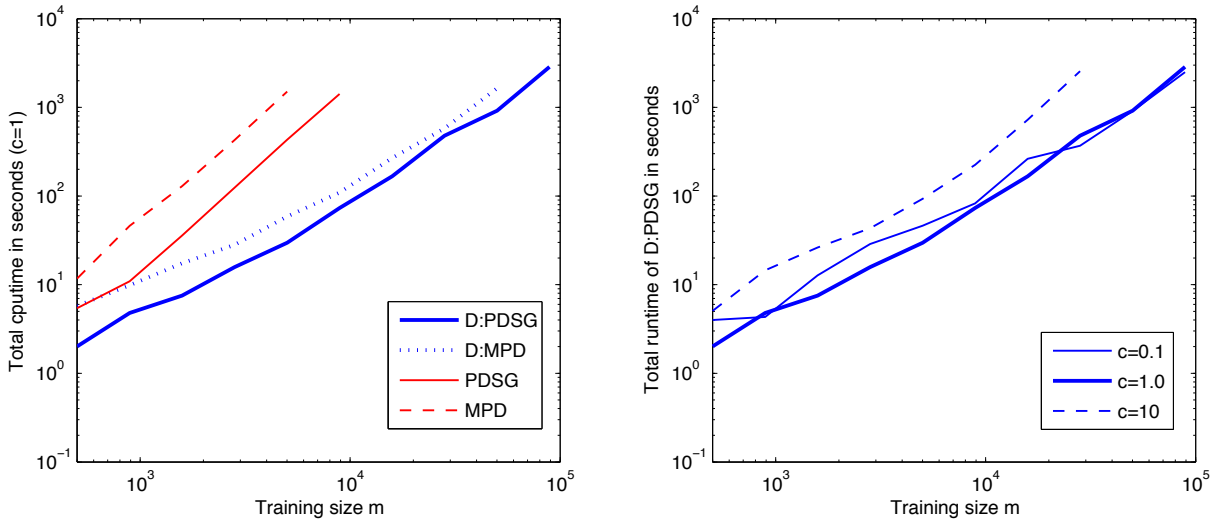


Figure 4.1 Left plot shows total runtimes using solvers PDSG and MPD in stand-alone mode and inside of the decomposition framework (D:PDSG and D:MPD) with $c = 1$. Right plot shows the total runtimes of D:PDSG (our proposed method) and MPD with different c values. For larger number of training examples m, updating of the full gradient in Step 3 of Algorithm 4 dominates the computation, blurring the distinction between PDSG and MPD as subproblem solvers (left plot). D:PDSG outperforms MPD for all c values tried (right plot). Stand-alone algorithms are run only for training-set size up to 10000 because of their high computational cost.

respectively, as these values gave good performance on the largest data set. Similarly, we fix the sub-working set size $n_{\mathcal{W}}$ to 2. (We tried various other values between 1 and 25, but 2 was slightly better than several alternatives.) In each setting, we use a kernel cache of 400MB in size.

Growth of the total runtime of the algorithms with increasing size of the data set is shown in Figure 4.1. When the decomposition framework is used, the stopping threshold values are set to $\texttt{tolD} = 0.001$ and $\texttt{tolS} = 0.0005$. In stand-alone settings, we set $\texttt{tolS} = 0.001$. We impose a slightly tighter threshold on subproblem solvers inside the decomposition framework to reduce the number of decomposition steps. Outer iterations in the decomposition framework become more costly as the number of variables increases, mainly because the full gradient update in Step 3 of Algorithm 4 becomes more expensive. The benefit of using decomposition framework becomes larger as the data set size grows. For instance, D:PDSG is about 100 times faster than MPD when $m = 10000$. In decomposition settings, using PDSG as the inner solver found the solution two to

three times faster than using MPD as the inner solver on average. Our proposed method D:PDSG shows quite stable scaling behavior for different values of c.

Convergence and Complexity. The different convergence behavior of PDSG and MPD is illustrated in Figure 4.2. Here both solvers are asked to solve a semiparametric regression problem discussed above with 1000 samples, in stand-alone mode. In the top and middle plots, the dual and primal infeasibility, respectively, are more rapidly reduced with PDSG than with MPD. (Note that since we project the iterates z^k to the bound constraints set, the KKT condition (4.4e) is always satisfied.) The bottom plot of Figure 4.2 shows the changes of the first Lagrange multiplier (the coefficient of the first basis function). In that, MPD is showing the typical behavior of the method of multipliers: sudden changes are made, but time gaps between such changes are rather large. In contrast, PDSG keeps making changes to the multiplier, resulting in a faster approach to the optimal value.

When the sub-working-set size $n_{\mathcal{W}}$ is smaller than the working-set size $n_{\mathcal{B}}$ of the subproblem (4.5), PDSG has computational complexity $O(Kn_{\mathcal{B}})$, the same as MPD, where K is the number of equality constraints in (4.2). Dual updates in Algorithm 5 requires $O(Kn_{\mathcal{B}})$ operations; all primal updates are done in $O(n_{\mathcal{B}})$. The effect of increasing K on the total time taken by D:PDSG is shown in Figure 4.3. We use the basis functions

$$\psi_j(x) = \begin{cases} \cos(j\pi x) & j = 0, 2, 4, \ldots \\ \sin(j\pi x) & j = 1, 3, 5, \ldots \end{cases}$$

and data sets of size $m = 1000$ randomly sampled from the Mexican hat function. Other settings are the same as the previous experiment. As expected, we observe linear scaling of total runtime with K.

4.5.2 Milan Respiratory Illness Data Set

We consider a data set[2] from the study on the effect of air pollution on respiratory illness in Milan, Italy, during 1980–89 (Vigotti et al., 1996). This data set consists of daily records of

[2]Available at `http://www.uow.edu.au/~mwand/webspr/data.html`

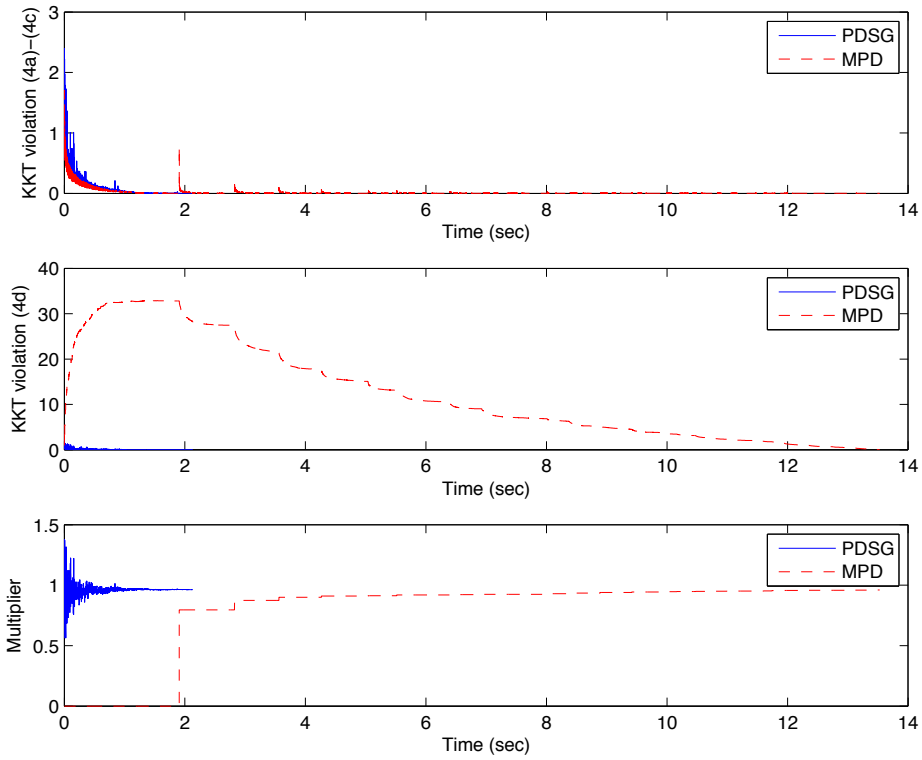


Figure 4.2 Convergence of PDSG and MPD in stand-alone mode (Mexican hat, data set size m=1000). PDSG requires about 2 seconds to reach convergence, whereas MPD takes about 14 seconds. (Top) maximum violation of the dual feasibility conditions (4.4a), (4.4b), (4.4c). (Middle) maximum violation of the primal equality constraints (4.4d). (Bottom) convergence of the first Lagrange multiplier to its optimal value of 1. The horizontal axis represents elapsed CPU time.

environmental conditions and the number of deaths due to respiratory diseases (total 3652 records, 9 features), where the details of the features are shown in Table 4.1.

All features are scaled linearly to the range $[0, 1]$. We construct a test set by holding out 20% of randomly chosen records from the data set, using the remaining records for training.

We hypothesize a simple semiparametric model to predict the number of respiratory deaths, inspired by Vigotti et al. (1996):

$$h_{\mathrm{sp}}(x) = \langle w, \phi(x) \rangle + \beta_1(x_{\mathrm{temp}}) + \beta_2(x_{\mathrm{SO_2}}) + \beta_3(x_{\mathrm{temp}})^2 + \beta_4(x_{\mathrm{SO_2}})^2 + \beta_5 \ ,$$

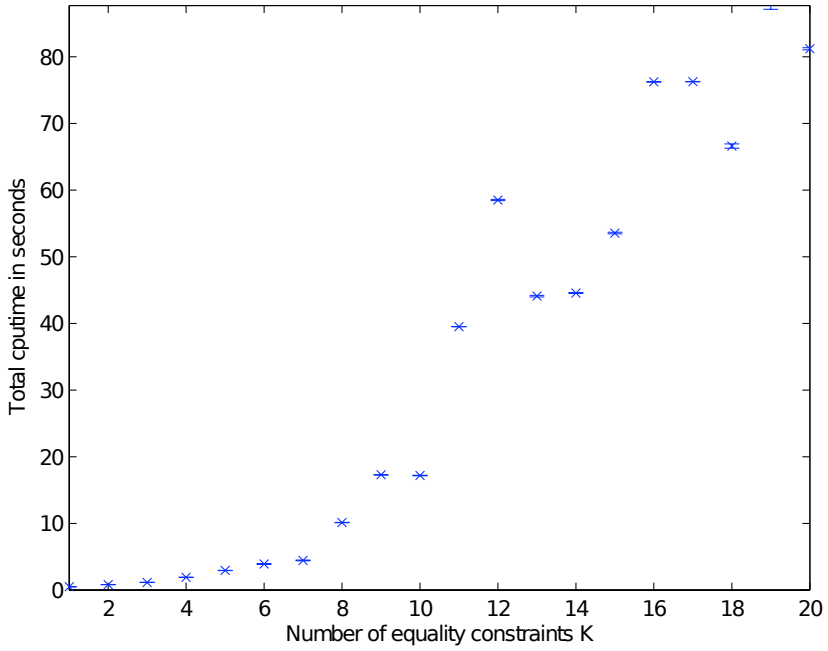


Figure 4.3 Total solution time for D:PDSG with increasing number of equality constraints K. Measurements are averaged over 10 repetitions with different random data sets ($m = 1000$) sampled from the Mexican hat function, and error bars (hardly visible) show the standard deviations. The time complexity of D:PDSG is $O(uKn_{\mathcal{B}})$ where u is the number of outer iterations. Solver time appears to increase linearly with K.

Table 4.1 The features of the Milan air pollution data set.

Feature name	Details
day.num	Number of days since 31st December, 1979.
day.of.week	1=Mon, 2=Tues, 3=Wed, 4=Thurs, 5=Fri, 6=Sat, 7=Sun.
holiday	Indicator of public holiday: 1=public holiday, 0=otherwise.
mean.temp	Mean daily temperature in degrees Celsius.
rel.humid	Relative humidity.
tot.mort	Total number of deaths.
resp.mort	Total number of respiratory deaths.
SO2	Measure of sulphur dioxide level in ambient air.
TSP	Total suspended particles in ambient air.

where the features x_{temp} and x_{SO_2} correspond to mean temperature and SO_2 level of the day, respectively. Our purpose is to study how those two elements affect the respiratory illness.

We fit our semiparametric model to the training data, and compare its prediction performance on the test set to that of a nonparametric model

$$h_{\text{np}}(x) = \langle w, \phi(x) \rangle + \beta_1 \ ,$$

and a parametric model

$$h_{\text{p}}(x) = \beta_1(x_{\text{temp}}) + \beta_2(x_{\text{SO}_2}) + \beta_3(x_{\text{temp}})^2 + \beta_4(x_{\text{SO}_2})^2 + \beta_5 \ .$$

With Gaussian kernel ($\gamma = 25.0$) and ϵ-insensitive loss function ($\epsilon = 0.01$), we perform 10-fold CV on the training set to determine the best balancing parameter c for each of semiparametric and nonparametric models independently.

The results are shown in Table 4.2. The semiparametric model attained smaller prediction error on the test set than the nonparametric model, indicating that the embedding of prior knowledge in h_{sp} while retaining the power of nonparametric approaches is beneficial. Moreover, the parametric

Table 4.2 Nonparametric and semiparametric regression on Milan data set. The loss penalty parameter c is determined by cross validation. Comparing the prediction performance on the test set by mean square error (MSE) values, the semiparametric model performed better than the nonparametric model by 2.8%. No significant difference of the number of support vectors (SVs) was found between the two methods.

Model	c	Fraction of SVs	Training Time (s)	Test Error (MSE)
Parametric	-	-	0.22	0.027911
Nonparametric (h_{np})	0.025	46.7%	1.17	0.019368
Semiparametric (h_{sp})	0.01	46.9%	5.35	0.018828

components in the trained semiparametric model

$$\begin{aligned} h_{\mathrm{sp}}(x) &= \langle w^*, \phi(x)\rangle - 0.30(x_{\mathrm{temp}}) + 0.26(x_{\mathrm{SO}_2}) + 0.22(x_{\mathrm{temp}})^2 - 0.07(x_{\mathrm{SO}_2})^2 + 0.22 \\ &= \langle w^*, \phi(x)\rangle + 0.22(x_{\mathrm{temp}} - 0.47)^2 + 0.26(x_{\mathrm{SO}_2}) - 0.07(x_{\mathrm{SO}_2})^2 + 0.12 \ . \end{aligned}$$

reveal that (i) deaths are lower in the middle of the temperature range, and (ii) there is an almost linear increase of death rate with SO_2 level. These results broadly agree with the outcomes of Vigotti et al. (1996), which were acquired from completely different statistical analysis techniques. It is difficult to perform model interpretation of this type with nonparametric approaches.

4.6 Conclusions

We have presented a new algorithm for semiparametric SVM regression problems, which extends a number of previous approaches in being able to handle multiple equality constraints. Our method combines a decomposition framework with a primal-dual scaled gradient solver for the subproblems. Computational tests indicate that the approach improves on previously proposed methods.

Future directions include reducing the cost of the full gradient update by using a randomized sampling procedure for the components of the gradient, as has been tried in a different context in Shi et al. (2008). While the concept is simple, it is not straightforward to implement this technique

in conjunction with caching of kernel entries, which is so important to efficient implementation of SVM solvers based on QP formulations.

Chapter 5

Cutting-Plane Methods for SVMs

The cutting-plane methods provide other ways to handle the primal support vector machine formulations for large amount of data. In these methods, we make use of a special formulation for which we can create relaxed subproblems. The algorithm proceeds adding violated constraints to subproblems, until no such constraint can be found. We call these violated constraints as *cuts*.

For the support vector machines, we derive a cutting-plane formulation using the idea of the Benders' reformulation (Benders, 1962). The same cutting-plane formulation is also suggested by Joachims (2006) independently, but starting from a different perspective. These works have been extended for monotonic convergence (Franc and Sonnenburg, 2007, 2008) and for nonlinear kernels (Joachims et al., 2009; Joachims and Yu, 2009). The extensions for nonlinear kernels use approximations to the kernel functions similar to our techniques for the subgradient methods in Chapter 3.

In this chapter we present our derivation of the cutting-plane formulation, suggesting some improvements to an existing method called the optimal cutting plane algorithm (Franc and Sonnenburg, 2007, 2008). We also present a generalized cutting-plane formulation where multiple violated cuts, instead of a single cut, can be admitted to a subproblem at a time, speeding up the procedure by reducing the number of iterations.

5.1 Introduction

To simplify our discussion, we focus on the primal SVM formulation (1.6) for classification introduced in Chapter 1, which is

$$\begin{aligned} \min_{w,b,s} \quad & \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{i=1}^{m} s_i \\ \text{s.t.} \quad & y_i(w^T x_i + b) \geq 1 - s_i, \quad i = 1, 2, \ldots, m, \\ & s_i \geq 0 \qquad\qquad\qquad\quad i = 1, 2, \ldots, m \end{aligned} \tag{5.1}$$

for the training inputs $x_i \in \mathbb{R}^n$ and their labels $y_i \in \{-1, +1\}$, $i = 1, 2, \ldots, m$, where $w \in \mathbb{R}^n$, $b \in \mathbb{R}$, and $s \in \mathbb{R}^m$. We derive a cutting-plane formulation equivalent to (5.1), inspired by the Benders' reformulation.

5.1.1 The Benders' Reformulation

We first discuss the reformulations with special structures, developed by Benders (1962) for mixed-integer programming problems. This technique is often used to handle complicating constraints, considering the dual of complicating variables (Nemhauser and Wolsey, 1988; Wolsey, 1998). In the following mixed-integer program,

$$\begin{aligned} z = \min_{x,y} \quad & c^T x + h^T y \\ \text{s.t.} \quad & Ax + Gy \geq b \\ & x \in X \in \mathbb{Z}^n_+, \; y \in \mathbb{R}^p_+ \end{aligned} \tag{5.2}$$

we can regard the integer variables x as complicating variables, because this program would be a simple linear program without x. In the first step of the reformulation, we suppose that x have been fixed. The resulting linear program is

$$z_{\text{LP}}(x) = \min_{y} \{h^T y \mid Gy \geq b - Ax, \; y \in \mathbb{R}^p_+\}.$$

Suppose that this linear program has a bounded optimal solution and $G \in \mathbb{R}^{m\times p}$. Then by the strong duality (Ferris et al., 2007), we have

$$z_{\text{LP}}(x) = \max_{u} \{u^T(b - Ax) \mid G^T u \leq h,\ u \in \mathbb{R}^m_+\}.$$

Let us denote by Q the feasible set of the above dual linear program, that is, $Q := \{u \in \mathbb{R}^m_+ \mid G^T u \leq h\}$. Then we consider the sets $\mathcal{U}$ and $\mathcal{V}$ such that

$$\mathcal{U} := \left\{u \in \mathbb{R}^m_+ \mid\ u \text{ is an extreme point of } Q\right\}$$
$$\mathcal{V} := \left\{v \in \mathbb{R}^m_+ \mid\ v \text{ is an extreme ray of } \{u \in \mathbb{R}^m_+ \mid G^T u \leq 0\}\right\}.$$

Note that if Q is nonempty, $\mathcal{V}$ is also the set of extreme rays of Q. Using these definitions and the strong duality, we can rewrite (5.2) as follows,

$$z = \min_{x} \left\{ c^T x + \max_{u\in\mathcal{U}}\ u^T(b - Ax) \mid v^T(b - Ax) \leq 0\ \ \forall v \in \mathcal{V},\ x \in X \right\}.$$

This leads to the Benders' representation of (5.2):

$$\begin{aligned} z = \min_{x,\eta}\ & \eta \\ \text{s.t. }\ & \eta \geq c^T x + u^T(b - Ax) && \forall u \in \mathcal{U} \\ & v^T(b - Ax) \leq 0 && \forall v \in \mathcal{V} \\ & x \in X, \eta \in \mathbb{R}. \end{aligned} \tag{5.3}$$

The equivalence of (5.2) and (5.3), including the cases when the linear program in (5.2) is infeasible or has an unbounded optimal value, can be found for example in Nemhauser and Wolsey (1988, Theorem 7.2).

Solving the Benders' reformulation (5.3), we often consider relaxed problems composed of a subset of extreme points and extreme rays, since the formulation typically has a large number of constraints.

5.1.2 A Reformulation of SVMs

Now we present a reformulation of the support vector machines in (5.1) inspired by Benders' idea. In the objective of (5.1), we treat s as the 'complicating' variable and consider the linear

program obtained by fixing w and b, that is,

$$\eta := \min_{s \geq 0} \left\{ \mathbf{1}^T s \mid s_i \geq 1 - y_i(w^T x_i + b),\ i = 1, 2, \ldots, m \right\}. \tag{5.4}$$

We denoted by η the optimal value of the linear program. Since this linear program is always feasible (the vector s defined by $s_i = \max\{1 - y_i(w^T x_i + b), 0\},\ i = 1, \ldots, m$, is a typical feasible point) and the objective is bounded below by zero, η should match with the dual optimal value by the strong duality. That is,

$$\eta = \max_{\mathbf{0} \leq u \leq \mathbf{1}} \sum_{i=1}^{m} u_i \{1 - y_i(w^T x_i + b)\}.$$

For this dual linear program, the set of extreme points is $\mathcal{U} = \{0, 1\}^m$ and there is no extreme ray since the feasible region is a bounded polyhedron. Therefore we can restate (5.1) as follows, similarly to the Benders' reformulation (5.3),

$$\begin{aligned} \min_{w,b,\eta} \quad & \frac{1}{2}\|w\|^2 + \frac{c}{m}\eta \\ \text{s.t.} \quad & \eta \geq \sum_{i=1}^{m} u_i \{1 - y_i(w^T x_i + b)\}, \quad \forall u \in \mathcal{U} = \{0, 1\}^m. \end{aligned} \tag{5.5}$$

This formulation has 2^m constraints. In the following section, we will consider an algorithm that constructs relaxed subproblems with a small number of constraints from (5.5).

The same formulation was derived independently by Joachims (2006) from a different perspective, acquiring (5.5) by aggregating the inequalities of (5.1) and showing the equivalence between (5.1) and (5.5) afterwards.

5.2 Cutting-Plane Algorithms for SVMs

We discuss the algorithms to obtain the solutions of the cutting-plane formulation (5.5).

5.2.1 A Naive Algorithm

We consider an algorithm with delayed constraints generation, which iteratively solves relaxed problems consisting of the objective and a subset of the constraints from (5.5).

We define the *cut-defining set* $\mathcal{D}$ as a subset of the set of extreme points $\mathcal{U} = \{0,1\}^m$ discussed in Section 5.1.2. Given a set $\mathcal{D}$, the algorithm solves the following relaxed problem characterized by $\mathcal{D}$ in each iteration,

$$\begin{aligned} \min_{w,b,\eta} \quad & \phi_{\mathcal{D}}(w,b,\eta) := \frac{1}{2}\|w\|^2 + \frac{c}{m}\eta \\ \text{s.t.} \quad & \eta \geq \sum_{i=1}^{m} d_i\{1 - y_i(w^T x_i + b)\}, \quad \forall d \in \mathcal{D} \\ & \eta \geq 0 \ . \end{aligned} \tag{5.6}$$

Note that we have added an extra constraint $\eta \geq 0$ for the case when $\mathcal{D} = \emptyset$. Suppose that the algorithm obtains w^t, b^t and η^t at t-th iteration by solving (5.6). If we can find a vector $u \in \mathcal{U}$ such that

$$\eta^t < \sum_{i=1}^{m} u_i\{1 - y_i\left((w^t)^T x_i + b^t\right)\}, \tag{5.7}$$

then we say u defines a cut of (5.5). The algorithm tries to identify the most violated cut by finding the vector u that maximizes the right hand side of (5.7). For our problem, such u can be computed analytically by

$$u_i = \begin{cases} 1 & \text{if } 1 - y_i\left((w^t)^T x_i + b^t\right) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

for $i = 1, 2, \ldots, m$. We stop the algorithm using a threshold $\epsilon > 0$ if the maximal constraint violation

$$\sum_{i=1}^{m} u_i \left\{1 - y_i\left((w^t)^T x_i + b^t\right)\right\} - \eta^t$$

falls below the specified threshold. Otherwise, we augment the set $\mathcal{D}$ with u and repeat the algorithm. We summarize this procedure in Algorithm 6.

The correctness of Algorithm 6 is shown for example in Joachims (2006, Theorem 3). That is, the algorithm returns a triplet (w, b, η) that produces a better objective value than the optimal solution triplet (w^*, b^*, η^*) of (5.5), and $(w, b, \eta + \epsilon)$ is feasible for (5.5). It is also shown that Algorithm 6 terminates in a finite number of iterations. The upper bounds on the number of iterations until termination appeared as $O(\frac{1}{\epsilon^2})$ in early literature (Tsochantaridis et al., 2005; Joachims,

Algorithm 6 A Naive Cutting-Plane Algorithm

1: Initialize: $\mathcal{D} \leftarrow \emptyset$.
2: **for** $t = 1, 2, \ldots$ **do**
3: Solve the relaxed problem (5.6) characterized by $\mathcal{D}$, obtaining w^t, b^t and η^t.
4: Generate u by (5.8).
5: If $\sum_{i=1}^m u_i \left\{1 - y_i \left((w^t)^T x_i + b^t\right)\right\} - \eta^t \leq \epsilon$, stop the algorithm.
6: Otherwise, $\mathcal{D} \leftarrow \mathcal{D} \cup \{u\}$.
7: **end for**

2006), and tightened to $O(\frac{1}{\epsilon})$ later with improved analysis (Teo et al., 2007; Smola et al., 2008). These upper bounds are independent of the number of training examples m.

5.2.2 Solving the Relaxed Problems

In each iteration of Algorithm 6, we need to obtain the solutions of the relaxed problem (5.6) characterized by $\mathcal{D}$. For efficient implementation, we prefer to using the dual formulation of (5.6), since the dual formulation has close proximity to the standard SVM dual formulation (1.9) that has been studied very well.

To derive the dual formulation, we construct the Lagrangian $\mathcal{L}$ of (5.6) introducing dual variables $\alpha \in \mathbb{R}_+^{|\mathcal{D}|}$ and $\beta \in \mathbb{R}_+$. To simplify the notation we denote by $\alpha_d \in \mathbb{R}_+$ the element of the vector α associated with a cut-defining vector $d \in \mathcal{D}$.

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + \frac{c}{m}\eta - \sum_{d \in \mathcal{D}} \alpha_d \left[\eta - \sum_{i=1}^m d_i \left\{1 - y_i(w^T x_i + b)\right\}\right] - \beta\eta \ . \tag{5.9}$$

From the KKT conditions we obtain

$$\begin{aligned}
\nabla_w \mathcal{L} &= w - \sum_{d\in\mathcal{D}} \alpha_d \sum_{i=1}^{m} d_i y_i x_i = 0 \\
\nabla_b \mathcal{L} &= \sum_{d\in\mathcal{D}} \alpha_d \sum_{i=1}^{m} d_i y_i = 0 \\
\nabla_\eta \mathcal{L} &= \frac{c}{m} - \sum_{d\in\mathcal{D}} \alpha_d - \beta = 0 \\
\mathbf{0} \le \alpha_d \quad &\perp \quad \eta - \sum_{i=1}^{m} d_i\{1 - y_i(w^T x_i + b)\} \ge 0, \quad \forall d \in \mathcal{D} \\
0 \le \beta \quad &\perp \quad \eta \ge 0 \ .
\end{aligned}$$

Substituting the primal variables w, b, and η in (5.9) using the above equalities leads to the desired dual formulation,

$$\min_{\alpha\in\mathbb{R}^{|\mathcal{D}|}} \ \frac{1}{2}\sum_{d\in\mathcal{D}}\sum_{d'\in\mathcal{D}} \alpha_d\alpha_{d'} \sum_{i=1}^{m}\sum_{j=1}^{m} d_i d'_j y_i y_j \langle x_i, x_j\rangle - |\mathcal{D}| \sum_{d\in\mathcal{D}} \alpha_d \tag{5.10a}$$

$$\text{s.t.} \ \sum_{d\in\mathcal{D}} \alpha_d \sum_{i=1}^{m} d_i y_i = 0 \tag{5.10b}$$

$$\sum_{d\in\mathcal{D}} \alpha_d \le \frac{c}{m} \tag{5.10c}$$

$$\alpha_d \ge 0 \quad \forall d \in \mathcal{D} \ . \tag{5.10d}$$

This is a convex quadratic program similar to the standard SVM dual formulation (1.13), except for the extra constraint (5.10c). Since we augment the cut-defining set $\mathcal{D}$ by a single element in each iteration of Algorithm 6, the dimension of the dual problem (5.10) is increased by one, and thus *warm-starting* would be beneficial for solving the sequence of dual subproblems.

5.2.3 An Improved Cutting-Plane Algorithm

In Algorithm 6, the computation cost of each iteration increases as we add more cuts to the relaxed problem (5.6). Therefore, it would be a good idea to modify the algorithm so that the

iterates result in monotonic decrease of the SVM objective, that is,

$$\phi(w,b) := \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{i=1}^{m} \ell_h(w,b;x_i,y_i),$$

where ℓ_h is the hinge loss function (1.5) discussed in Chapter 1,

$$\ell_{\mathrm{h}}(w,b;x_i,y_i) := \max\{1 - y_i(\langle w, x_i\rangle + b), 0\}.$$

Franc and Sonnenburg (2007, 2008) introduced an efficient line-search step with the time complexity of $\big(m \log m\big)$ to generates such iterates. Their algorithm produces a sequence of best-so-far iterates $w_b^1, w_b^2, \ldots$, ensuring that $\phi(w_b^1), \phi(w_b^2), \ldots$ be a monotonic decreasing sequence. This modified cutting plane algorithm, called the optimized cutting-plane algorithm for SVMs (OCAS), is shown in Algorithm 7.

To solve the reduced problems (5.6), OCAS uses a quadratic program solver developed by Franc and Hlaváč (2006). This solver is based on an active-set method similar to other dual SVM solvers, and extended to handle the extra constraint (5.10c).

5.2.3.1 Safeguarding

In Algorithm 7, we want to avoid the situation that $(w_b^t, b_b^t) = (w_b^{t-1}, b_b^{t-1})$, because in that case there will be no improvement in objective values. Algorithm 7 uses a 'safeguarding' heuristic to avoid such situation, generating cuts using perturbed pairs $(w_c^1, b_c^1), (w_c^2, b_c^2), \ldots$, instead of the best-so-far pairs $(w_b^1, b_b^1), (w_b^2, b_b^2), \ldots$. The perturbed pairs are computed by (5.12), controlled by a parameter $\lambda \in (0, 1]$. In Franc and Sonnenburg (2008), this heuristic was applied in every iteration with $\lambda = 0.1$ for the experiments.

However, we claim that it is not necessary to perform this heuristic in every iteration. Suppose that we modify Algorithm 7 so that a user specifies $\lambda_0 \in (0, 1]$ instead of λ, and we use λ_t for safeguarding as follows,

$$\begin{bmatrix} w_c^t \\ b_c^t \end{bmatrix} = (1 - \lambda_t) \begin{bmatrix} w_b^t \\ b_b^t \end{bmatrix} + \lambda_t \begin{bmatrix} w^t \\ b^t \end{bmatrix}. \tag{5.14}$$

Then we can set $\lambda_t = \lambda_0$ if we want to perform the safeguard, or $\lambda_t = 0$ otherwise.

Algorithm 7 Optimized Cutting-plane Algorithm (OCAS) (Franc and Sonnenburg, 2007, 2008)

1: Input: Real numbers $\epsilon > 0$ and $\lambda \in (0, 1]$.

2: Initialize: $\mathcal{D} \leftarrow \emptyset$; $w_b^0 = \mathbf{0}$; $b_b^0 = 0$.

3: **for** $t = 1, 2, \ldots$ **do**

4: Solve the relaxed problem (5.6) characterized by $\mathcal{D}$, obtaining w^t, b^t, and η^t.

5: Compute the best so far solution (w_b^t, b_b^t) by

$$\begin{bmatrix} w_b^t \\ b_b^t \end{bmatrix} = (1 - k^*) \begin{bmatrix} w_b^{t-1} \\ b_b^{t-1} \end{bmatrix} + k^* \begin{bmatrix} w^t \\ b^t \end{bmatrix}$$

where k^* is acquired from a line-search procedure, that is,

$$k^* = \arg\min_{k \geq 0} \phi\left((1-k)w_b^{t-1} + kw^t, (1-k)b_b^{t-1} + kb^t\right) \quad . \tag{5.11}$$

6: Compute the cut generation point

$$\begin{bmatrix} w_c^t \\ b_c^t \end{bmatrix} = (1 - \lambda) \begin{bmatrix} w_b^t \\ b_b^t \end{bmatrix} + \lambda \begin{bmatrix} w^t \\ b^t \end{bmatrix}. \tag{5.12}$$

7: Generate u by

$$u_i = \begin{cases} 1 & \text{if } 1 - y_i\left((w_c^t)^T x_i + b_c^t\right) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad i = 1, 2, \ldots, m. \tag{5.13}$$

8: If $\phi(w_b^t, b_b^t) - \phi_{\mathcal{D}}(w^t, b^t, \eta^t) \leq \epsilon |\phi(w_b^t, b_b^t)|$, stop the algorithm.

9: Otherwise, $\mathcal{D} \leftarrow \mathcal{D} \cup \{u\}$.

10: **end for**

With this modified safeguarding mechanism, we now explain how to avoid the bad situation that $(w_c^t, b_c^t) = (w_c^{t-1}, b_c^{t-1})$, in order to guarantee monotonic decrease in the SVM objective values. In fact, the bad situation happens only when all of the following conditions are satisfied at t-th

iteration:

$$\begin{cases} w_{\mathrm{c}}^{t-1} = w_{\mathrm{b}}^{t-1} & \text{(i.e., } \lambda_{t-1} = 0\text{)}, \\ w_{\mathrm{b}}^{t} = w_{\mathrm{b}}^{t-1} & \text{(i.e., } k^* = 0\text{)}, \\ w_{\mathrm{c}}^{t} = w_{\mathrm{b}}^{t} & \text{(i.e., } \lambda_t = 0\text{)}. \end{cases}$$

This implies that we need to apply the safeguarding (that is, setting $\lambda_t > 0$) only when $\lambda_{t-1} = 0$ and the line search step has failed ($k^* = 0$) at t-th iteration.

5.2.3.2 Caching and Shrinking

For solving the relaxed dual problem (5.10), we need to compute the Hessian of the objective denoted by $Q \in \mathbb{R}^{|\mathcal{D}|\times|\mathcal{D}|}$. For convenience we use $Q_{dd'}$ to represent the entry of Q associated with vectors d and d' in $\mathcal{D}$. That is,

$$Q_{dd'} = \left\langle \sum_{i=1}^{m} d_i y_i x_i, \sum_{j=1}^{m} d'_j y_j x_j \right\rangle \quad \forall d, d' \in \mathcal{D}.$$

In each iteration of Algorithm 7, the cardinality of the cut-defining set $\mathcal{D}$ increases by one. Suppose that in the previous iteration we have stored the Hessian Q of the relaxed dual problem and created a new cut defined by a vector $u \in \mathcal{U}$. To solve the relaxed problem in the current iteration defined with $\mathcal{D} \cup \{u\}$, we have to augment the matrix Q by one column and one row. If we have stored the sum of vectors $\sum_{i=1}^{m} d_i y_i x_i$ for all $d \in \mathcal{D}$, the cost of augmentation becomes $O\left(n(m + |\mathcal{D}|)\right)$. For $x_i \in \mathbb{R}^n$, the cost consists of $O(mn)$ operations to compute $\sum_{i=1}^{m} u_i y_i x_i$ for u, and $O(n|\mathcal{D}|)$ for inner product computations.

However, the subproblem solver (Franc and Hlaváč, 2006) usually does not reference all entries of Q. Also, the storage requirement for Q grows quadratically with $|\mathcal{D}|$. Therefore we suggest using a cache with least-recently-used (LRU) replacement strategy to store once computed and recently referenced entries of Q for reuse, instead of storing the entire matrix Q. In our caching scheme, we store the columns of Q corresponding to active dual variables. We augment the cached columns whenever the size of $\mathcal{D}$ is increased.

We also suggest a *shrinking* strategy for efficient implementation, which is similar to the shrinking heuristic in SVM-Light (Joachims, 1999). With shrinking, we ignore some of the cut-defining

vectors in $\mathcal{D}$ that have been inactive for a certain period of time. As a result the size of subproblems can be decreased, possibly speeding up the entire algorithm. We periodically check the status of the ignored cuts and put them back into consideration if they become active again.

5.2.4 Multiple-cut Generation Approaches

In each iteration of Algorithm 7, we compute a vector $u \in \mathcal{U}$ using (5.13) that characterizes the most violated constraint. This step requires to evaluate the following expression for all training examples,

$$1 - y_i((w_c^t)^T x_i + b_c^t) \quad i = 1, \ldots, m \ , \tag{5.15}$$

for the current cut-generation points w_c^t and b_c^t. The time complexity of this step is $O(mn)$, which is rather costly to generate a single cut.

We propose an extended cutting-plane algorithm that generates multiple cuts per iteration rather than a single cut. This algorithm uses the same output from (5.15) to generate multiple cuts. By adding multiple cuts, we can obtain more restricted subproblems, possibly reducing the number of subproblems we need to create until we find a solution.

5.2.4.1 Formulations

To derive a multiple-cut formulation, we consider nonoverlapping and nonempty partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_P$ of the index set $\{1, 2, \ldots, m\}$. We define the cut-defining sets associated with the partitions by

$$\mathcal{D}_k := \{0, 1\}^{|\mathcal{P}_k|}, \ \ k = 1, 2, \ldots, P.$$

Recall that in Section 5.1.2 we reformulated SVMs as follows using the Benders' representation,

$$\begin{aligned} \min_{w,b,\eta} \quad & \frac{1}{2}\|w\|^2 + \frac{c}{m}\eta \\ \text{s.t.} \quad & \eta = \max_{\mathbf{0} \le u \le \mathbf{1}} \sum_{i=1}^{m} u_i\{1 - y_i(w^T x_i + b)\}. \end{aligned} \tag{5.16}$$

We focus on the fact that the maximization in the second line is separable over the partitions. Let us denote by $u_{\mathcal{P}_k}$ the subvector of $u \in \{0,1\}^m$ corresponding to the partition $\mathcal{P}_k$, and define

$$\eta_k := \max_{\mathbf{0} \le u_{\mathcal{P}_k} \le \mathbf{1}} \sum_{i \in \mathcal{P}_k} u_i \{1 - y_i(w^T x_i + b)\}, \quad k = 1, 2, \ldots, P,$$

so that $\eta = \sum_{k=1}^{P} \eta_k$. Then we can write an equivalent formulation of (5.16) using partitions,

$$\begin{aligned} \min_{w,b,\eta_1,\eta_2,\ldots,\eta_P} \quad & \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{k=1}^{P} \eta_k \\ \text{s.t.} \quad & \eta_k \ge \sum_{i \in \mathcal{P}_k} u_i \left\{1 - y_i(w^T x_i + b)\right\}, \quad \forall u \in \mathcal{U}, \quad k = 1, 2, \ldots, P. \end{aligned}$$

This leads to a relaxed formulation characterized by the cut-defining sets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_P$,

$$\begin{aligned} \min_{w,b,\eta_1,\ldots,\eta_P} \quad & \phi_{\mathcal{D}_1,\ldots,\mathcal{D}_P}(w, b, \eta_1, \ldots, \eta_P) := \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{k=1}^{P} \eta_k \\ \text{s.t.} \quad & \eta_k \ge \sum_{i=1}^{|\mathcal{P}_k|} d_i^k \left\{1 - y_i(w^T x_i + b)\right\}, \qquad \forall d^k \in \mathcal{D}_k, \ k = 1, 2, \ldots, P, \\ & \eta_k \ge 0, \qquad\qquad\qquad\qquad\qquad\qquad k = 1, 2, \ldots, P \ . \end{aligned} \tag{5.17}$$

Note that this formulation reverts to the single-cut formulation (5.5) when $P = 1$ and $\mathcal{P}_1 = \{1, 2, \ldots, m\}$. If we choose $P > 1$ and partitions of equal sizes, that is, $|\mathcal{P}_k| = O(m/P)$, then the number of constraints in (5.17) is bounded by $O(P2^{m/P})$. This is much smaller than the number of constraints $O(2^m)$ in the single-cut formulation.

To derive the dual formulation of (5.17), we consider the Lagrangian $\mathcal{L}$ introducing dual variables $\alpha_{d^k} \in \mathbb{R}_+$, $k = 1, 2, \ldots, P$, and $\beta \in \mathbb{R}_+^P$:

$$\begin{aligned} \mathcal{L} = & \frac{1}{2}\|w\|^2 + \frac{c}{m}\sum_{k=1}^{P} \eta_k - \sum_{k=1}^{P} \beta_k \eta_k \\ & - \sum_{k=1}^{P} \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} \left[\eta_k - \sum_{i=1}^{|\mathcal{P}_k|} d_i^k \left\{1 - y_i(w^T x_i + b)\right\} \right]. \end{aligned}$$

From the KKT conditions we have

$$\begin{aligned}
\nabla_w \mathcal{L} &= w - \sum_{k=1}^{P} \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} \sum_{i=1}^{|\mathcal{P}_k|} d_i^k y_i x_i = \mathbf{0} \\
\nabla_b \mathcal{L} &= -\sum_{k=1}^{P} \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} \sum_{i=1}^{|\mathcal{P}_k|} d_i^k y_i = 0 \\
\nabla_{\eta_k} \mathcal{L} &= \frac{c}{m} - \beta_k - \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} = 0 \\
0 \le \alpha_{d^k} \quad &\perp \quad \eta_k - \sum_{i=1}^{|\mathcal{P}_k|} d_i^k \left\{1 - y_i(w^T x_i + b)\right\} \ge 0, \quad \forall d^k \in \mathcal{D}_k,\ k = 1, 2, \ldots, P, \\
0 \le \beta_k \quad &\perp \quad \eta_k \ge 0 \ \ k = 1, 2, \ldots, P \ .
\end{aligned}$$

Substituting the primal variables in $\mathcal{L}$ using the above equalities leads to the dual formulation,

$$\begin{aligned}
\min_{\alpha} \ & \frac{1}{2} \sum_{k=1}^{P} \sum_{\ell=1}^{P} \sum_{d^k \in \mathcal{D}_k} \sum_{d^\ell \in \mathcal{D}_\ell} \alpha_{d^k} \alpha_{d^\ell} \sum_{i=1}^{|\mathcal{P}_k|} \sum_{j=1}^{|\mathcal{P}_\ell|} d_i^k d_j^\ell y_i y_j \langle x_i, x_j \rangle - \sum_{k=1}^{P} |\mathcal{P}_k| \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} \\
\text{s.t.} \ & \sum_{k=1}^{P} \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} \sum_{i=1}^{|\mathcal{P}_k|} d_i^k y_i = 0, \\
& \sum_{d^k \in \mathcal{D}_k} \alpha_{d^k} \le \frac{c}{m}, \qquad k = 1, 2, \ldots, P, \\
& \alpha_{d^k} \ge 0, \qquad\qquad k = 1, 2, \ldots, P \ .
\end{aligned} \tag{5.18}$$

The minimization is over the vector $\alpha \in \mathbb{R}^{\sum_{k=1}^{P} |\mathcal{D}_k|}$ that is defined by

$$\alpha := \left(\alpha_{d^{1,1}}, \alpha_{d^{1,2}}, \ldots, \alpha_{d^{1,|\mathcal{D}_1|}},\ \alpha_{d^{2,1}}, \ldots, \alpha_{d^{2,|\mathcal{D}_2|}},\ \ldots,\ \alpha_{d^{P,1}}, \ldots, \alpha_{d^{P,|\mathcal{D}_P|}}\right)^T$$

where we enumerate the elements of $\mathcal{D}_k$ by $d^{k,1}, d^{k,2}, \ldots, d^{k,|\mathcal{D}_k|}$. This formulation is very similar to the dual of the single-cut relaxed problem (5.10). In Algorithm 8, we present our cutting-plane algorithm that generates multiple cuts per iteration and performs the modified safeguarding mechanism discussed in Section 5.2.3.1.

Algorithm 8 Multiple Cutting-Plane Algorithm (MCPA)

1: Input:

- Real numbers $\epsilon > 0$ and $\lambda_0 \in (0, 1]$.
- The partitions $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_P$ of the training example indices $\{1, 2, \ldots, m\}$.

2: Initialize: $\mathcal{D}_k \leftarrow \emptyset,\ k = 1, 2, \ldots, P$; $w_b^0 = \mathbf{0}$; $b_b^0 = 0$.

3: **for** $t = 1, 2, \ldots$ **do**

4: Solve the relaxed problem (5.17) characterized by $\mathcal{D}$, obtaining $\eta_1^t, \ldots, \eta_P^t$, w^t, and b^t.

5: Compute the best so far solution (w_b^t, b_b^t) by

$$\begin{bmatrix} w_b^t \\ b_b^t \end{bmatrix} = (1 - k^*) \begin{bmatrix} w_b^{t-1} \\ b_b^{t-1} \end{bmatrix} + k^* \begin{bmatrix} w^t \\ b^t \end{bmatrix}, \text{ where}$$

$$k^* = \arg\min_{k \geq 0} \phi\left((1-k)w_b^{t-1} + kw^t, (1-k)b_b^{t-1} + kb^t\right) \ .$$

6: Determine the safeguard parameter $\lambda_t = \begin{cases} \lambda_0 & \text{if } \lambda_{t-1} = 0 \text{ and } k^* = 0, \\ 0 & \text{otherwise.} \end{cases}$

7: Compute the cut generation point

$$\begin{bmatrix} w_c^t \\ b_c^t \end{bmatrix} = (1 - \lambda_t) \begin{bmatrix} w_b^t \\ b_b^t \end{bmatrix} + \lambda_t \begin{bmatrix} w^t \\ b^t \end{bmatrix},$$

8: For $k = 1, 2, \ldots, P$, generate $u^k \in \{0, 1\}^{|\mathcal{P}_k|}$ by

$$u_i^k = \begin{cases} 1 & \text{if } 1 - y_i\left((w_c^t)^T x_i + b_c^t\right) > 0 \\ 0 & \text{otherwise} \end{cases}, \quad \forall i \in \mathcal{P}_k. \tag{5.19}$$

9: If $\phi(w_b^t, b_b^t) - \phi_{\mathcal{D}_1,\ldots,\mathcal{D}_P}(w^t, b^t, \eta_1^t, \ldots, \eta_P^t) \leq \epsilon|\phi(w_b^t, b_b^t)|$, stop the algorithm.

10: Otherwise, update $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{u^k\}$ for $k = 1, 2, \ldots, P$.

11: **end for**

5.3 Experiments

We benchmark the performance of our multiple-cut generation algorithm in Algorithm 8, comparing it to the single-cut algorithm OCAS in Algorithm 7. We call our implementation as MCPA, which is based on the open-source OCAS code[1]. We also implement the kernel cache and the shrinking heuristic in Section 5.2.3.2.

For training SVMs, we use three binary classification data sets summarized in Table 5.1 from the UCI machine learning repository (Frank and Asuncion, 2010). We classify: the digits 0-4 from 5-9 with `MNIST`; the texts with with 'ccat' category from others with `CCAT`; and the forest of type 1 from others with `COVTYPE` data set. For more details, we refer to Section 3.4. The table also shows the tuning parameters c/m determined by using separate validation sets.

For all experiment, we set the stopping threshold ϵ to 10^{-2} and the size of kernel cache to 512MB. For shrinking, we ignore the cuts that have been inactive for five consecutive iterations, testing them for reactivation in every 50 iterations.

Table 5.1 Benchmark Data Sets.

Name	m (train)	valid/test set size	dimension n	(density)	c/m
`MNIST`	58100	5950/5950	784	(19.1%)	1.0
`CCAT`	78127	11575/11575	47237	(1.6%)	0.01
`COVTYPE`	464809	57102/57101	54	(21.7%)	3.0

5.3.1 Comparison of the Safeguarding Mechanisms

We first test our modified safeguarding mechanism discussed in Section 5.2.3.1. For the benchmark data sets, we compared the total number of iterations and training time between MCPA with $P = 1$ and OCAS. (OCAS uses the original safeguarding heuristic.) The results in Table 5.2

[1] Available at `http://cmp.felk.cvut.cz/~xfrancv/ocas/html/`

Table 5.2 Comparison of safeguarding mechanisms. The total number of iterations and training time in seconds (in parentheses) of MCPA with $P = 1$ and OCAS are shown for three benchmark data sets.

Data set	No. iterations (cputime:sec)		
	OCAS	MCPA (no shrinking)	MCPA (with shrinking)
MNIST	150 (8.50)	105 (5.35)	109 (5.54)
CCAT	8 (0.43)	7 (0.37)	7 (0.38)
COVTYPE	83 (7.48)	64 (5.43)	66 (5.65)

indicate that the modified heuristic in MCPA is beneficial for reducing the total number of iterations and training time. With shrinking, MCPA spent little more iterations than MCPA without shrinking, but the numbers were still smaller than those of the OCAS algorithm.

5.3.2 The Effect of Multiple Cut Generation

To investigate the effect of generating multiple cuts, we varied the number of partitions P in the range of $[1, 100]$, measuring the number of iterations and total cputime until convergence. We note that when $P = 1$, MCPA becomes almost identical to OCAS, except for the modified safeguarding, kernel caching, and shrinking mechanisms. For the experiment we used three variants of MCPA: MCPA with both kernel caching and shrinking, MCPA with caching but no shrinking, and MCPA without any of the two features.

The results are reported in Figure 5.1. We can observe that the number of iterations decreases as we generate more cuts per iteration. However, the total cputime does not always decrease with the increment of P. The reason is that the size of relaxed problems grows faster as we add more cuts in each iteration. The best runtime of MCPA was acquired with small number of partitions overall. We expected that the runtime with larger P values would be reduced further by more aggressive shrinking. To check the claim, we tried shrinking with ignoring cuts if they were inactive for two iterations, testing them for revival in every tenth iteration. The result is shown in Figure 5.2. We

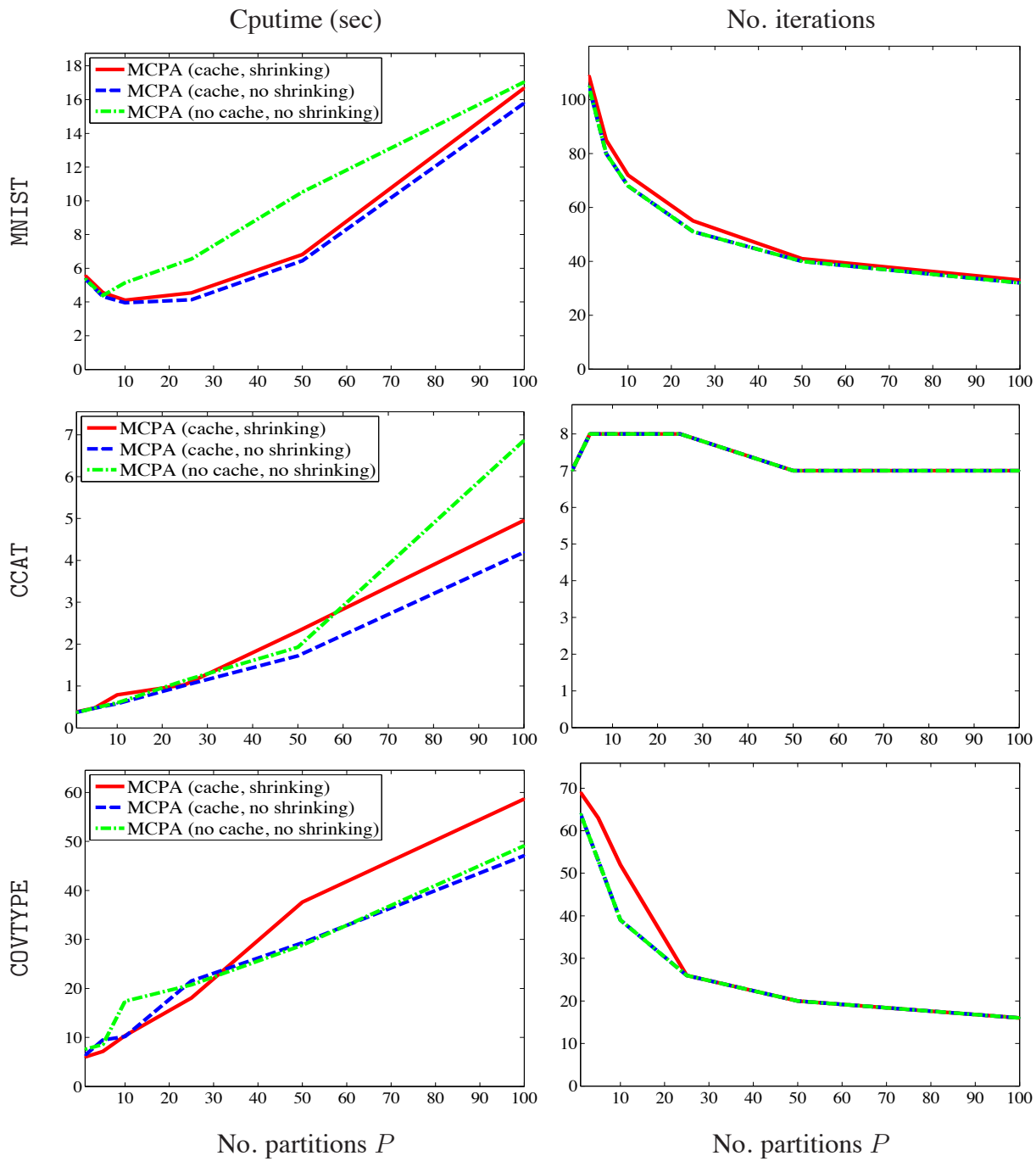


Figure 5.1 The total runtime (left) and the number of iterations (right) of MCPA, varying the number of partitions P, for the benchmark data sets `MNIST`, `CCAT`, and `COVTYPE`. Three variants of MCPA are presented in the plots: MCPA with both kernel caching and shrinking, MCPA with caching but no shrinking, and MCPA without any of the two features.

can observe that our algorithm performs better with multiple cuts for MNIST and COVTYPE. For CCAT it seemed to be hard to reduce the size of subproblems effectively with shrinking, since the algorithm terminated with very few iterations.

5.4 Conclusion

In this chapter we presented cutting-plane formulations for SVMs and a multiple-cut generation algorithm with improved features. When we change the number of cuts to be added in iterations, there happens a trade-off between the total number of iterations and the size of subproblems, where the latter can be reduced by shrinking mechanisms.

We expect that multiple-cut approaches will be more useful on platforms such as graphical processing units (GPUs), that have many but simple computation units with relatively small memory. The cut-generation task for each partition is simple to parallelize and depends on a subset of training data which would fit into the memory of GPUs. More research on subproblem solvers and shrinking mechanisms would be necessary, however.

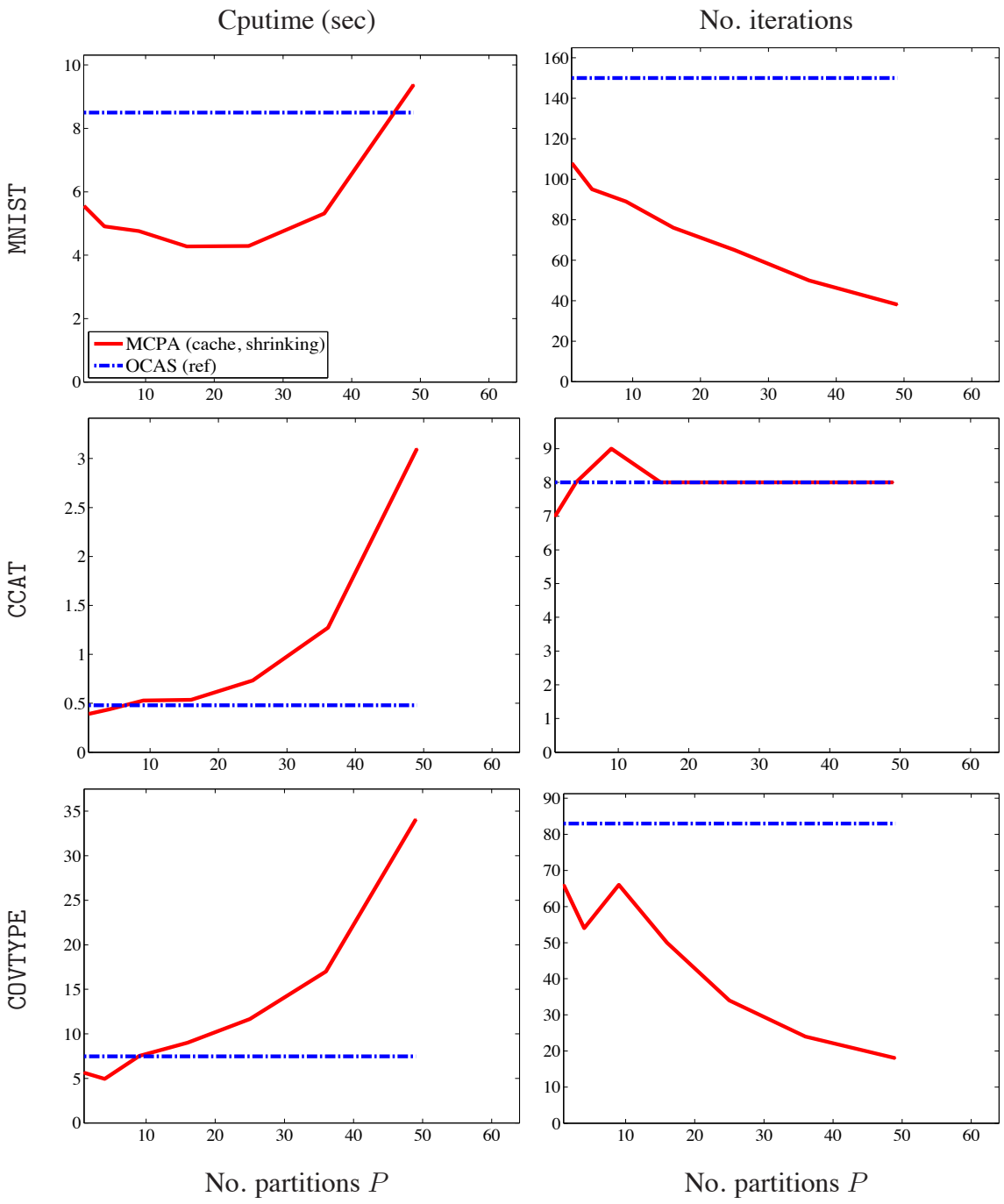


Figure 5.2 The total runtime (left) and the number of iterations (right) of MCPA with aggressive shrinking, varying the number of partitions P for the benchmark data sets `MNIST`, `CCAT`, and `COVTYPE`. We also show the results from OCAS for comparison, where OCAS always uses $P = 1$ (the values are duplicated for other P values).

APPENDIX
Details for Chapter 2

A.1 Strong Minimizer Property

In this section we show that Theorem 2.5 is true, based on the results from manifold analysis. Our proof is similar to the proof of Wright (2010, Theorem 2.5) but simpler. We first state an elementary result on manifold characterization, which is proved in Vaisman (1984, Sections 1.4-1.5) and Wright (2010, Appendix A.).

Lemma A.1 Let the manifold $\mathcal{M} \subset \mathbb{R}^n$ containing $\bar{z}$ be characterized by a $\mathcal{C}^p$ ($p \geq 2$) function $H : \mathbb{R}^n \to \mathbb{R}^k$. Then there is $\bar{y} \in \mathbb{R}^{n-k}$ and a $\mathcal{C}^p$ function G mapping some neighborhood of $\bar{y}$ to $\mathbb{R}^n$ such that $G(y) \in \mathcal{M}$ for all y near $\bar{y}$. Moreover, $G(y) - \bar{z} = Y(y - \bar{y}) + O(\|y - \bar{y}\|^2)$, where $Y \in \mathbb{R}^{n\times(n-k)}$ is an orthonormal matrix whose columns span the tangent space to $\mathcal{M}$ at $\bar{z}$.

The next result from Wright (2010) shows how perturbations from a point at which the objective function is partly smooth can be decomposed according to the manifold characterization above.

Lemma A.2 Let the manifold $\mathcal{M} \subset \mathbb{R}^n$ be characterized in a neighborhood of $\bar{z} \in \mathcal{M}$ by $\mathcal{C}^p$ mappings $H : \mathbb{R}^n \to \mathbb{R}^k$ and $G : \mathbb{R}^{n-k} \to \mathbb{R}^n$ and the point $\bar{y}$ described in Lemma A.1. Then for all z near $\bar{z}$, there are unique vectors $y \in \mathbb{R}^{n-k}$ and $v \in \mathbb{R}^k$ with $\|(y^T - \bar{y}^T, v^T)\| = O(\|z - \bar{z}\|)$ such that $z = G(y) + \nabla H(\bar{z})v$.

We also make use of a result from Wright (2010, Lemma A.1).

Lemma A.3 Consider a function $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$, a point $\bar{z} \in \mathbb{R}^n$, and a manifold $\mathcal{M}$ containing $\bar{z}$ such that φ is partly smooth at $\bar{z}$ with respect to $\mathcal{M}$. If the nondegeneracy condition $0 \in \text{ri } \partial\varphi(\bar{z})$

holds, then there exists $\epsilon > 0$ such that

$$\sup_{g \in \partial\varphi(\bar{z})} \langle g, d\rangle \geq \epsilon\|d\|, \quad \forall d \in N_{\mathcal{M}}(\bar{z}).$$

Now we assume that the following conditions of Theorem 2.5 hold:

(i) ϕ is partly smooth at w^* relative to the optimal manifold $\mathcal{M}$.

(ii) w^* is a strong local minimizer of $\phi|_{\mathcal{M}}$ with the modulus $c_{\mathcal{M}} > 0$ and radius $r_{\mathcal{M}} > 0$, and

(iii) the nondegeneracy condition (2.11) holds at w^*.

For the minimizer w^* of (2.1) and the optimal manifold $\mathcal{M}$ containing w^*, we consider the mappings H and G, the matrix Y, and the point $\bar{y} = y^* \in \mathbb{R}^{n-k}$ satisfying Lemma A.1 and Lemma A.2, associated with $\bar{z} = w^* \in \mathbb{R}^n$. From Lemma A.2, for all w satisfying $\|w - w^*\| \leq \bar{r} \leq r_{\mathcal{M}}$ with small enough $\bar{r} > 0$, we can find unique vectors $y \in \mathbb{R}^{n-k}$ and $v \in \mathbb{R}^k$ with $\|(y^T - \bar{y}^T, v^T)\| = O(\|w - w^*\|)$ such that $w = G(y) + \nabla H(w^*)v$. Therefore we have

$$\phi(w) - \phi(w^*) = [\phi(G(y) + \nabla H(w^*)v) - \phi(G(y))] - [\phi(G(y)) - \phi(w^*)]\,. \tag{A.1}$$

From the assumption (ii) above, the strong local minimizer property relative to $\mathcal{M}$, for the second bracketed term we have

$$\phi(G(y)) - \phi(w^*) = \phi|_{\mathcal{M}}(G(y)) - \phi|_{\mathcal{M}}(w^*) \geq c_{\mathcal{M}}\|G(y) - w^*\|^2 \tag{A.2}$$

for all y near y^*.

Now we consider the first bracketed term of (A.1). From Lemma A.3, we have $\epsilon > 0$ such that $\sup_{g \in \partial\phi(w^*)}\langle g, d\rangle \geq \epsilon\|d\|$ for all $d \in N_{\mathcal{M}}(w^*)$. From the sub-continuity property (iv) from Definition 2.4, we can choose a neighborhood of w^* sufficiently small that for all $g \in \partial\phi(w^*)$, there exists $\hat{g} \in \partial\phi(G(y))$ such that $\|\hat{g} - g\| \leq \epsilon/2$. These facts, with the convexity of ϕ, lead to for all y near y^* and v that

$$\begin{aligned}
\phi(G(y) + \nabla H(w^*)v) - \phi(G(y)) &\geq \sup_{\hat{g} \in \phi(G(y))} \langle \hat{g}, \nabla H(w^*)v\rangle \\
&\geq \sup_{g \in \partial\phi(w^*)} \langle g, \nabla H(w^*)v\rangle - (\epsilon/2)\|\nabla H(w^*)v\| \\
&\geq \epsilon\|\nabla H(w^*)v\| - (\epsilon/2)\|\nabla H(w^*)v\|.
\end{aligned}$$

Together with (A.1) and (A.2), we have

$$\phi(w) - \phi(w^*) \geq (\epsilon/2)\|\nabla H(w^*)v\| + c_{\mathcal{M}}\|G(y) - w^*\|^2.$$

By further reducing $\bar{r}$ if necessary, we can choose the neighborhood of w^* small enough to ensure that $\|\nabla H(w^*)v\| \leq 1$, and therefore

$$\begin{aligned}
\phi(w) - \phi(w^*) &\geq (\epsilon/2)\|\nabla H(w^*)v\|^2 + c_{\mathcal{M}}\|G(y) - w^*\|^2 \\
&\geq \min(\epsilon/2, c_{\mathcal{M}}) \left[\|\nabla H(w^*)v\|^2 + \|G(y) - w^*\|^2\right] \\
&\geq \frac{1}{2}\min(\epsilon/2, c_{\mathcal{M}}) \left[\|\nabla H(w^*)v\| + \|G(y) - w^*\|\right]^2 \\
&\geq \frac{1}{2}\min(\epsilon/2, c_{\mathcal{M}})\|w - w^*\|^2,
\end{aligned}$$

showing that w^* indeed is the strong local minimizer of ϕ, without the restriction to the manifold $\mathcal{M}$, with the modulus $c := \min(\epsilon/2, c_{\mathcal{M}})$ and the radius $\bar{r}$.

A.2 Properties of the RDA Algorithm

In this section we provides the analyses for the results discussed in Section 2.3, regarding the iterates generated by the RDA algorithm under our assumptions.

A.2.1 Expected Error Bounds of the Iterates

First we prove Theorem 2.9, which provides the bounds of the expected errors in the iterates generated by the RDA algorithm. For the general convex case, with $\beta_t = \gamma\sqrt{t}$, we consider the

expected regret up to time t with respect to w^*, and obtain

$$\begin{aligned}
\mathbb{E}R_t(w^*) &= \mathbb{E}\left[\sum_{j=1}^{t}(F(w_j;\xi_j)+\Psi(w_j)) - \sum_{j=1}^{t}(F(w^*;\xi_j)+\Psi(w^*))\right] \\
&= \sum_{j=1}^{t}\mathbb{E}\left[\mathbb{E}\left\{(F(w_j;\xi_j)+\Psi(w_j)-F(w^*;\xi_j)-\Psi(w^*)) \,|\, \xi_{[j-1]}\right\}\right] \\
&= \sum_{j=1}^{t}\mathbb{E}\left[f(w_j)+\Psi(w_j)-f(w^*)-\Psi(w^*)\right] \\
&= \sum_{j=1}^{t}\mathbb{E}\left[\phi(w_j)-\phi(w^*)\right]. \qquad \text{(A.3)}
\end{aligned}$$

Noting that $I_{(\|w_j-w^*\|\le\bar{r})} + I_{(\|w_j-w^*|>\bar{r})} = 1$, we can split the right-hand side into two sums and obtain

$$\begin{aligned}
\mathbb{E}R_t(w^*) = &\sum_{j=1}^{t}\mathbb{E}\left[I_{(\|w_j-w^*\|\le\bar{r})}\{\phi(w_j)-\phi(w^*)\}\right] \\
&+ \sum_{j=1}^{t}\mathbb{E}\left[I_{(\|w_j-w^*\|>\bar{r})}\{\phi(w_j)-\phi(w^*)\}\right]. \qquad \text{(A.4)}
\end{aligned}$$

Note that both terms on the right-hand side of (A.4) are nonnegative. For the first term, we have by using the regret bound (2.20) and the strong local minimizer property (2.13) that

$$\begin{aligned}
\left(\gamma D^2 + \frac{G^2}{\gamma}\right)t^{1/2} &\ge \mathbb{E}R_t(w^*) \\
&\ge \sum_{j=1}^{t}\mathbb{E}\left[I_{(\|w_j-w^*\|\le\bar{r})}\{\phi(w_j)-\phi(w^*)\}\right] \\
&\ge c\sum_{j=1}^{t}\mathbb{E}\left[I_{(\|w_j-w^*\|\le\bar{r})}\|w_j-w^*\|^2\right],
\end{aligned}$$

proving the first inequality (2.22a). For the second inequality, we have from (A.4), the regret bound (2.20), and Corollary 2.6 that

$$\begin{aligned}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{1/2} &\geq \mathbb{E} R_t(w^*) \\ &\geq \sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})}\{\phi(w_j) - \phi(w^*)\}\right] \\ &\geq c\bar{r} \sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})} \|w_j - w^*\|\right],\end{aligned}$$

thus proving (2.22b).

For the strongly convex objective with $\beta_t = \sigma(1 + \ln t)$, we apply the other regret bound (2.21) to (A.3), resulting in

$$\begin{aligned}\left(\sigma D^2 + \frac{G^2}{2\sigma}\right)(1 + \ln t) &\geq \mathbb{E} R_t(w^*) \\ &\geq \sum_{j=1}^{t} \mathbb{E}\{\phi(w_j) - \phi(w^*)\} \\ &\geq \min(c, \sigma/2) \sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\|^2,\end{aligned}$$

where for the last inequality we use the fact that w^* is a (global) strong minimizer with the modulus $\min(c, \sigma/2)$, as shown in Corollary 2.7. This proves (2.23).

For the general convex cases, the following result combines the two inequalities (2.22a) and (2.22b) into one which will be more handy in later discussion.

Corollary A.4 Under the assumptions of Theorem 2.9 with general convex objective and $\beta_t = \gamma\sqrt{t}$, we have

$$\frac{1}{t} \sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\| \leq \mu t^{-1/4} \tag{A.5}$$

for the constant μ defined in (2.24a):

$$\mu := \frac{1}{\sqrt{c}} \left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2} \left[1 + \frac{1}{\bar{r}\sqrt{c}} \left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2}\right].$$

Moreover, for strongly convex objective with $\beta_t = \sigma(1 + \ln t)$, we have

$$\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\| \leq \mu' \left(\frac{1+\ln t}{t}\right)^{1/2} \tag{A.6}$$

for the constant μ' defined in (2.24b):

$$\mu' := \frac{1}{\sqrt{\min(c, \sigma/2)}} \left(\sigma D^2 + \frac{G^2}{2\sigma}\right)^{1/2}.$$

Proof We start with the general convex case. From the Cauchy-Schwartz inequality $\|z\|_1 \leq \sqrt{m}\|z\|_2$ for a vector $z \in \mathbb{R}^m$ and Jensen's inequality, we have

$$\begin{aligned}
\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})}\|w_j - w^*\|\right] &\leq \frac{\sqrt{t}}{t}\left[\sum_{j=1}^{t}\left\{\mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})}\|w_j - w^*\|\right]\right\}^2\right]^{1/2} \\
&\leq \left[\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})}\|w_j - w^*\|^2\right]\right]^{1/2} \\
&\leq \frac{1}{\sqrt{c}}\left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2} t^{-1/4}
\end{aligned}$$

where the last inequality is from (2.22a). Together with (2.22b), this leads to

$$\begin{aligned}
&\frac{1}{t}\sum_{j=1}^{t}\mathbb{E}\|w_j - w^*\| \\
&\quad= \frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| \leq \bar{r})}\|w_j - w^*\|\right] + \frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[I_{(\|w_j - w^*\| > \bar{r})}\|w_j - w^*\|\right] \\
&\quad\leq \frac{1}{\sqrt{c}}\left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2} t^{-1/4} + \frac{1}{\bar{r}c}\left(\gamma D^2 + \frac{G^2}{\gamma}\right) t^{-1/2} \\
&\quad\leq \mu t^{-1/4},
\end{aligned}$$

for μ defined in (2.24a).

For the strongly convex case, again using Cauchy-Schwarz and Jensen's inequalities we have

$$\begin{aligned}
\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\| &\leq \left[\frac{1}{t}\sum_{j=1}^{t}\{\mathbb{E}\|w_j - w^*\|\}^2\right]^{1/2} \\
&\leq \left[\frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\|^2\right]^{1/2}.
\end{aligned}$$

Applying the bound in (2.23) to the last line leads to (A.6). ■

A.2.2 Properties of the Dual Average

Theorem 2.11 is one of our key results in Chapter 2 which reveals important properties of the dual average; in this section we show how the result is acquired, basing the expected error bounds in the iterates discussed above. For the general convex case, we have

(i) $\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| \leq L\mu t^{-1/4}$

(ii) $\text{tr}\, \Sigma^t \leq 4G\,(G + 4L\mu)\, t^{-1/4}$.

And for the strongly convex case, we have:

(i') $\|\mathbb{E}\bar{g}_t - \nabla f(w^*)\| \leq L\mu' \left(\frac{1+\ln t}{t}\right)^{1/2}$

(ii') $\text{tr}\, \Sigma^t \leq 4G(G + 4L\mu') \left(\frac{1+\ln t}{t}\right)^{1/2}$

where the constants μ and μ' are defined as follows,

$$\mu = \frac{1}{\sqrt{c}} \left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2} \left[1 + \frac{1}{\bar{r}\sqrt{c}} \left(\gamma D^2 + \frac{G^2}{\gamma}\right)^{1/2}\right],$$

$$\mu' = \frac{1}{\sqrt{\min(c, \sigma/2)}} \left(\sigma D^2 + \frac{G^2}{2\sigma}\right)^{1/2}.$$

We first examine (i), noting that $w_j = w_j(\xi_{[j-1]})$ is independent of the future random variables $\xi_j, \xi_{j+1}, \ldots, \xi_t$. Then

$$\begin{aligned}
\mathbb{E}[\bar{g}_t] &= \frac{1}{t}\sum_{j=1}^{t} \mathbb{E}[\nabla F(w_j; \xi_j)] \\
&= \frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\left[\mathbb{E}_j[\nabla F(w_j; \xi_j) \mid \xi_{[j-1]}]\right] \\
&= \frac{1}{t}\sum_{j=1}^{t} \mathbb{E}\nabla f(w_j) && \text{by (2.8)} \\
&= \nabla f(w^*) + \frac{1}{t}\sum_{j=1}^{t} \mathbb{E}v_j, && \text{where } v_j := \nabla f(w_j) - \nabla f(w^*).
\end{aligned}$$

From Lemma 2.2, we have $\|v_j\| \leq L\|w_j - w^*\|$ for $j = 1, 2, \ldots, t$, so that

$$\begin{aligned}
\|\mathbb{E}[\bar{g}_t] - \nabla f(w^*)\| &= \left\| \frac{1}{t} \sum_{j=1}^{t} \mathbb{E} v_j \right\| \\
&\leq \frac{1}{t} \sum_{j=1}^{t} \|\mathbb{E} v_j\| \\
&\leq \frac{1}{t} \sum_{j=1}^{t} \mathbb{E}\|v_j\| \qquad \text{by Lemma 2.1} \\
&\leq \frac{L}{t} \sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\| \\
&\leq L\mu t^{-1/4} \qquad \text{by (A.5),}
\end{aligned} \tag{A.7}$$

proving (i). For the corresponding strongly convex case (i'), instead of (A.7) we use

$$\begin{aligned}
\|\mathbb{E}[\bar{g}_t] - \nabla f(w^*)\| &= \left\| \frac{1}{t} \sum_{j=1}^{t} \mathbb{E} v_j \right\| \\
&\leq \frac{1}{t} \sum_{j=1}^{t} \mathbb{E}\|v_j\| \qquad \text{by Lemma 2.1} \\
&\leq \frac{L}{t} \sum_{j=1}^{t} \mathbb{E}\|w_j - w^*\| \\
&\leq L\mu' \left(\frac{1 + \ln t}{t} \right)^{1/2} \qquad \text{by (A.6).}
\end{aligned} \tag{A.8}$$

Next, for (ii), we observe that

$$\begin{aligned}
&\bar{g}_t - \mathbb{E}\bar{g}_t \\
&= \frac{1}{t}\sum_{j=1}^{t} (\nabla F(w_j;\xi_j) - \mathbb{E}\nabla F(w_j;\xi_j)) \\
&= \frac{1}{t}\sum_{j=1}^{t} \left(\nabla F(w_j;\xi_j) - \mathbb{E}\left[\mathbb{E}[\nabla F(w_j;\xi_j)\,|\,\xi_{[j-1]}]\right]\right) \\
&= \frac{1}{t}\sum_{j=1}^{t} (\nabla F(w_j;\xi_j) - \mathbb{E}\nabla f(w_j)) \\
&= \frac{1}{t}\sum_{j=1}^{t} \Big(\{\nabla F(w^*;\xi_j) - \nabla f(w^*)\} \\
&\quad + \{\nabla F(w_j;\xi_j) - \nabla F(w^*;\xi_j)\} - \mathbb{E}\{\nabla f(w_j) - \nabla f(w^*)\}\Big) \\
&= \frac{1}{t}\sum_{j=1}^{t} (\varphi_j + \vartheta_j)
\end{aligned}$$

where we have defined

$$\begin{aligned}
\varphi_j &:= \nabla F(w^*;\xi_j) - \nabla f(w^*), \\
\vartheta_j &:= \nabla F(w_j;\xi_j) - \nabla F(w^*;\xi_j) - \mathbb{E}\{\nabla f(w_j) - \nabla f(w^*)\}.
\end{aligned}$$

We now derive bounds on φ_j and ϑ_j. Note first that $\varphi_j = \varphi_j(\xi_j)$ and $w_j = w_j(\xi_{[j]})$. By unbiasedness, the assumption (2.16), the triangle inequality, and Lemma 2.1, we have

$$\mathbb{E}\varphi_j = \mathbb{E}[\nabla F(w^*;\xi_j) - \nabla f(w^*)] = 0, \tag{A.9}$$

$$\begin{aligned}
\|\varphi_j\| &\le \|\nabla F(w^*;\xi_j)\| + \|\nabla f(w^*)\| \\
&= \|\nabla F(w^*;\xi_j)\| + \|\mathbb{E}\nabla F(w^*;\xi_j)\| \\
&\le \|\nabla F(w^*;\xi_j)\| + \mathbb{E}\|\nabla F(w^*;\xi_j)\| \le 2G.
\end{aligned} \tag{A.10}$$

For ϑ_j, we have from (2.9) and Lemma 2.2, and the triangle inequality and Lemma 2.1, that

$$\begin{aligned}
\|\vartheta_j\| &\le \|\nabla F(w_j;\xi_j) - \nabla F(w^*;\xi_j)\| + \|\mathbb{E}(\nabla f(w_j) - \nabla f(w^*))\| \\
&\le \|\nabla F(w_j;\xi_j) - \nabla F(w^*;\xi_j)\| + \mathbb{E}\|\nabla f(w_j) - \nabla f(w^*)\| \\
&\le L\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right).
\end{aligned} \tag{A.11}$$

On the other hand, using (2.16) we have (with unbiasedness (2.8) and Lemma 2.1),

$$\begin{aligned}\|\vartheta_j\| &\le \|\nabla F(w_j;\xi_j)\| + \|\nabla F(w^*;\xi_j)\| + \|\mathbb{E}\nabla f(w_j)\| + \|\nabla f(w^*)\| \\ &\le \|\nabla F(w_j;\xi_j)\| + \|\nabla F(w^*;\xi_j)\| + \|\mathbb{E}\nabla F(w_j;\xi_j)\| + \|\mathbb{E}_j\nabla F(w^*;\xi_j)\| \\ &\le \|\nabla F(w_j;\xi_j)\| + \|\nabla F(w^*;\xi_j)\| + \mathbb{E}\|\nabla F(w_j;\xi_j)\| + \mathbb{E}\|\nabla F(w^*;\xi_j)\| \\ &\le 4G \end{aligned} \tag{A.12}$$

Focusing on the trace of Σ^t, we have

$$\begin{aligned} t^2(\operatorname{tr}\Sigma^t) &= t^2\sum_{\ell=1}^{n}\Sigma_{\ell\ell}^t \\ &= \mathbb{E}\sum_{j=1}^{t}\sum_{\ell=1}^{n}[\varphi_j]_\ell^2 + \sum_{\ell=1}^{n}\sum_{j\neq k}\mathbb{E}([\varphi_j]_\ell[\varphi_k]_\ell) \\ &\quad + \mathbb{E}\sum_{j,k}\left(\sum_{\ell=1}^{n}[\varphi_j]_\ell[\vartheta_k]_\ell + \sum_{\ell=1}^{n}[\vartheta_j]_\ell[\varphi_k]_\ell + \sum_{\ell=1}^{n}[\vartheta_j]_\ell[\vartheta_k]_\ell\right). \end{aligned} \tag{A.13}$$

Each term of (A.13) can be bounded as follows, using (A.9), (A.10), (A.11) and (A.12) as required:

(a) $\displaystyle\sum_{\ell=1}^{n}[\varphi_j]_\ell^2 = \|\varphi_j\|^2 \le 4G^2,$

(b) $\mathbb{E}([\varphi_j]_\ell[\varphi_k]_\ell) = [\mathbb{E}(\varphi_j\varphi_k)]_\ell = [\mathbb{E}\varphi_j]_\ell[\mathbb{E}\varphi_k]_\ell = 0, \quad \text{for } j \neq k,$

(c) $\displaystyle\sum_{\ell=1}^{n}[\varphi_j]_\ell[\vartheta_k]_\ell \le \|\varphi_j\|\|\vartheta_k\| \le 2GL\left(\|w_k - w^*\| + \mathbb{E}\|w_k - w^*\|\right),$

$\displaystyle\sum_{\ell=1}^{n}[\vartheta_j]_\ell[\varphi_k]_\ell \le \|\vartheta_j\|\|\varphi_k\| \le 2GL\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right),$

(d) $\displaystyle\sum_{\ell=1}^{n}[\vartheta_j]_\ell[\vartheta_k]_\ell \le \|\vartheta_j\|\|\vartheta_k\| \le 4GL\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right).$

Bound (a) uses (A.10). (b) uses (A.9) and independence of ξ_j and ξ_k, for $j \neq k$. (c) uses the Cauchy-Schwartz inequality, (A.10), and (A.11). (d) uses Cauchy-Schwartz, (A.11) and (A.12). Combining these results with (A.13) results in

$$t^2(\operatorname{tr}\Sigma^t) \le 4G^2t + 8GLt\,\mathbb{E}\sum_{j=1}^{t}\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right). \tag{A.14}$$

The second term can be bounded as follows:

$$\begin{aligned} & t\mathbb{E}\sum_{j=1}^{t}\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right) \\ &= t\sum_{j=1}^{t}\mathbb{E}\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right) \\ &= 2t^2\left\{\frac{1}{t}\sum_{j=1}^{t}\mathbb{E}\|w_j - w^*\|\right\} \\ &\leq 2t^2\mu t^{-1/4}, \end{aligned}$$

where the final inequality follows from (A.5). Applying this bound to (A.14), we obtain

$$\operatorname{tr}\Sigma^t \leq 4G^2t^{-1} + 16GL\mu t^{-1/4} \leq 4G[G + 4L\mu]t^{-1/4},$$

which implies (ii). For the corresponding strongly convex case (ii'), we use instead of (A.11),

$$\begin{aligned} \|\vartheta_j\| &\leq \|\nabla F(w_j;\xi_j) - \nabla F(w^*;\xi_j)\| + \|\mathbb{E}(\nabla f(w_j) - \nabla f(w^*))\| \\ &\leq \|\nabla F(w_j;\xi_j) - \nabla F(w^*;\xi_j)\| + \mathbb{E}\|\nabla f(w_j) - \nabla f(w^*)\| \\ &\leq L\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right). \end{aligned} \tag{A.15}$$

Then instead of (c) and (d) above, we have the following:

$$\begin{aligned} \text{(c')}\quad & \sum_{\ell=1}^{n}[\varphi_j]_\ell[\vartheta_k]_\ell \leq \|\varphi_j\|\|\vartheta_k\| \leq 2GL\left(\|w_k - w^*\| + \mathbb{E}\|w_k - w^*\|\right), \\ & \sum_{\ell=1}^{n}[\vartheta_j]_\ell[\varphi_k]_\ell \leq \|\vartheta_j\|\|\varphi_k\| \leq 2GL\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right), \\ \text{(d')}\quad & \sum_{\ell=1}^{n}[\vartheta_j]_\ell[\vartheta_k]_\ell \leq \|\vartheta_j\|\|\vartheta_k\| \leq 4GL\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right). \end{aligned}$$

Combining (a), (b), (c') and (d') with (A.13) results in

$$t^2(\operatorname{tr}\Sigma^t) \leq 4G^2t + 8GLt\,\mathbb{E}\sum_{j=1}^{t}\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right) \tag{A.16}$$

for which the second term can be bounded as follows:

$$t\mathbb{E}\sum_{j=1}^{t}\left(\|w_j - w^*\| + \mathbb{E}\|w_j - w^*\|\right)$$
$$\leq 2t^2\mu'\left(\frac{1+\ln t}{t}\right)^{1/2},$$

due to (A.6). Applying this bound to (A.16) we obtain

$$\operatorname{tr}\Sigma^t \leq 4G^2t^{-1} + 16GL\mu'\left(\frac{1+\ln t}{t}\right)^{1/2} \leq 4G(G+4L\mu')\left(\frac{1+\ln t}{t}\right)^{1/2},$$

which implies (ii').

A.3 The regret bound of SGD with variable stepsizes

In this section we discuss the regret bounds of the SGD algorithm with a variable stepsize scheme. As in Section 2.1.2, when we consider the regularizer $\Psi(w) := \delta_{\mathcal{W}}(w) + \psi(w)$ with the indicator function $\delta_{\mathcal{W}}$ for a compact convex set $\mathcal{W}$, the SGD algorithm consists of the iterations

$$w_{t+1} = \Pi_{\mathcal{W}}\left(w_t - \alpha_t(g_t + h_t)\right), \quad t = 1, 2, \ldots$$

where $g_t \in \partial F(w_t; \xi_t)$ and $h_t \in \partial\psi(w_t)$. For SGD we assume that $\|h_t\| \leq H$ for some constant H for all $t = 1, 2, \ldots$, in addition to the assumption of RDA that $\|g_t\| \leq G$.

In the variable stepsize scheme (Nemirovski et al., 2009) for general convex objectives, we choose the stepsizes for some $\theta > 0$,

$$\alpha_t = \frac{D}{G}\frac{\theta}{\sqrt{t}}.$$

With this, we present a regret bound of the SGD algorithm, extending Theorem 1 in Zinkevich (2003). We include the details for completeness.

Theorem A.5 Suppose that the iterates $w_1, w_2, \ldots$ are generated by the SGD algorithm with variable stepsizes $\alpha_t = (D/G)(\theta/\sqrt{t})$, where $\max_{w_t \in \mathcal{W}} \|w_t - w_1\| \leq D$, $\max_{g_t \in \partial F(w_t;\xi_t)} \leq G$ and $\max_{h_t \in \partial\psi(w_t)} \|h_t\| \leq H$. Then the regret of SGD algorithm up to the iteration t with respect to w^* can bounded as

$$R_t(w^*) \leq D\left(\frac{2G}{\theta} + \frac{(G+H)^2}{G}\theta\right)\sqrt{t}.$$

Proof Let $\phi_j(w) := F(w;\xi_j) + \Psi(w)$. From the convexity of ϕ_j, we have for all $\varphi_j \in \partial\phi_j(w_j)$

$$\phi_j(w_j) - \phi_j(w^*) \leq \langle \varphi_j, w_j - w^* \rangle, \quad j = 1, 2, \ldots, t.$$

And we can choose $\varphi_j = g_j + h_j$. Summing up both sizes for $j = 1, 2, \ldots, t$ leads to

$$R_t(w^*) = \sum_{j=1}^{t} \{\phi_j(w_j) - \phi_j(w^*)\} \leq \sum_{j=1}^{t} \langle \varphi_j, w_j - w^* \rangle. \tag{A.17}$$

Let $v_{j+1} = w_j - \alpha_j(g_j + h_j)$ and $w_{j+1} = \mathit{\Pi}_{\mathcal{W}}(v_{j+1})$. Then

$$\begin{aligned}
\|w_{t+1} - w^*\|^2 &\leq \|v_{j+1} - w^*\|^2 \qquad \text{(contraction due to projection)} \\
&= \|(w_j - w^*) - \alpha_j \varphi_j\|^2 \\
&= \|w_j - w^*\|^2 - 2\alpha_j \langle \varphi_j, w_j - w^* \rangle + \alpha_j^2 \|\varphi_j\|^2.
\end{aligned}$$

Rearranging the terms, we have

$$\langle \varphi_j, w_j - w^* \rangle \leq \frac{1}{2\alpha_j} \left(\|w_j - w^*\|^2 - \|w_{t+1} - w^*\|^2 \right) + \frac{\alpha_j}{2} \|\varphi_j\|^2.$$

With (A.17), $\max_{w,w' \in \mathcal{W}} \|w - w'\| \leq 2D$, and $\|\varphi_j\| \leq G + H$, this implies

$$\begin{aligned}
R_t(w^*) &\leq \sum_{j=1}^{t} \left[\frac{1}{2\alpha_j} \left(\|w_j - w^*\|^2 - \|w_{t+1} - w^*\|^2 \right) + \frac{\alpha_j}{2} (G + H)^2 \right] \\
&\leq \frac{1}{2\alpha_1} \|w_1 - w^*\|^2 + \frac{1}{2} \sum_{j=2}^{t} \left(\frac{1}{\alpha_j} - \frac{1}{\alpha_{j-1}} \right) \|w_j - w^*\|^2 + \frac{(G+H)^2}{2} \sum_{j=1}^{t} \alpha_j \\
&\leq (2D)^2 \left\{ \frac{1}{2\alpha_1} + \frac{1}{2} \sum_{j=2}^{t} \left(\frac{1}{\alpha_j} - \frac{1}{\alpha_{j-1}} \right) \right\} + \frac{(G+H)^2}{2} \sum_{j=1}^{t} \alpha_j \\
&\leq \frac{2D^2}{\alpha_t} + \frac{(G+H)^2}{2} \sum_{j=1}^{t} \alpha_j
\end{aligned}$$

For $\alpha_j = \frac{D}{G} \frac{\theta}{\sqrt{j}}$, we have

$$\begin{aligned}
R_t(w^*) &\leq 2DG \frac{\sqrt{t}}{\theta} + D \frac{(G+H)^2}{2G} \theta \sum_{j=1}^{t} \frac{1}{\sqrt{j}} \\
&\leq D \left(\frac{2G}{\theta} + \frac{(G+H)^2}{G} \theta \right) \sqrt{t}
\end{aligned}$$

where the last inequality uses the fact that $\sum_{j=1}^{t} \frac{1}{\sqrt{j}} \leq 1 + \int_1^t t^{-1/2} dt \leq 2\sqrt{t}$. ■

Note that this bound is with respect to w^*, not to the minimizer of the second term in the definition of the regret (2.3) as in Zinkevich (2003).

By choosing the optimal $\theta^* = \sqrt{2}G/(G + H)$ to minimize the expression involving θ, we can simplify the bound as

$$R_t(w^*) \leq 2\sqrt{2}D(G + H)\sqrt{t}.$$

For ℓ_1-regularization ($\phi(w) = \lambda\|w\|_1$), we have $H = \lambda$, which is often much smaller than G. If it is the case, the regret bound of SGD can be further simplified to $2\sqrt{2}DG\sqrt{t}$.

Bibliography

M. Anitescu. Degenerate nonlinear programming with a quadratic growth condition. *SIAM Journal on Optimization*, 10(4):1116–1135, 2000.

P. L. Bartlett, E. Hazan, and A. Rakhlin. Adaptive online gradient descent. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 65–72, Cambridge, MA, 2008. MIT Press.

J. F. Benders. Partitioning prodecures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.

A. Bordes, S. Ertekin, J. Weston, and L. Bottou. Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research*, 6:1579–1619, 2005.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual workshop on Computational learning theory (COLT)*, pages 144–152, NY, USA, 1992. ACM Press.

L. Bottou. Stochastic learning. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Artificial Intelligence*, pages 146–168. Springer Verlag, 2004.

J. V. Burke and J. J. Moré. Exposing constraints. *SIAM Journal on Optimization*, 4(3):573–595, 1994.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, April 2009.

O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19:1155–1178, 2007.

K. Chung. On a stochastic approximation method. *The Annals of Mathematical Statistics*, 25(3): 463–483, 1954.

R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

Y. H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming, Series A*, 106:403–421, 2006.

P. Drineas and M. W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

Y. Ermoliev. Stochastic quasigradient methods and their application to system optimization. *Stochastics*, 9:1–36, 1983.

M. C. Ferris and T. S. Munson. Semismooth support vector machines. *Math. Program.*, 101(1): 185–204, 2004.

M. C. Ferris, O. L. Mangasarian, and S. J. Wright. *Linear Programming with* MATLAB. SIAM, November 2007.

S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–164, 2001.

V. Franc and V. Hlaváč. A novel algorithm for learning support vector machines with structured output spaces. Technical Report CTU-CMP-2006-04, Center for Machine Perception, Czech Technical University, May 2006.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. Technical report, Fraunhofer Institute FIRST, December 2007.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 320–327, 2008.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

E. M. Gafni and D. P. Bertsekas. Two-metric projection methods for constrained optimization. *SIAM Journal on Control and Optimization*, 22:936–964, 1984.

W. L. Hare and A. S. Lewis. Identifying active constraints via partial smoothness and prox-regularity. *Journal of Convex Analysis*, 11(2):251–266, 2004.

E. Hazan, A. Kalai, S. Kale, and A. Agarwal. Logarithmic regret algorithms for online convex optimization. In *19th Annual Conference on Learning Theory (COLT)*, pages 499–513, 2006.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

T. Joachims. Training linear svms in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, NY, USA, 2006. ACM Press.

T. Joachims and C.-N. J. Yu. Sparse kernel svms via cutting-plane training. *Machine Learning*, 76 (2-3):179–193, 2009. Special Issue for European Conference on Machine Learning (ECML).

T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

S. S. Keerthi and E. G. Gilbert. Convergence of a generalized smo algorithm for svm classifier design. *Machine Learning*, 46(1-3):351–360, 2002.

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Statist.*, 23:462–466, 1952.

W. Kienzle and B. Schölkopf. Training support vector machines with multiple equality constraints. In *Machine Learning: ECML 2005*, volume 16, October 2005.

G. Kimeldorf and G. Wahba. A correspondence between bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics*, 41:495–502, 1970.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

H. J. Kushner and G. G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer, NY, USA, second edition, 2003.

J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

A. S. Lewis. Active sets, nonsmoothness, and sensitivity. *SIAM Journal on Optimization*, 13: 702–725, 2003.

D. D. Lewis, Y. Yang, T. G. Rose, G. Dietterich, F. Li, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

C. J. Lin. Linear convergence of a decomposition method for support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, 2001.

O. L. Mangasarian and D. R. Musicant. Active set support vector machine classification. In *Advances in Neural Information Processing Systems*, volume 13, pages 577–583. MIT Press, 2001.

O. L. Mangasarian and D. R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46(1-3):255–269, 2002.

J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society, London*, 209:415–446, 1909.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12:109–138, 2001.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, NY, USA, 1988.

A. Nemirovski and D. Yudin. On cezari's convergence of the steepest descent method for approximating saddle point of convex-concave functions. *Soviet Math. Dokl.*, 19, 1978.

A. Nemirovski and D. B. Yudin. *Problem complexity and method efficiency in optimization*. John Wiley, 1983.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120:221–259, 2009.

G. C. Pflug. *Optimization of Stochastic Models: The Interface Between Simulation and Optimization*. Kluwer, Boston, 1996.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 12, pages 185–208. MIT Press, Cambridge, MA, 1999.

B. T. Polyak. New stochastic approximation type procedures. *Automat. i Telemekh.*, 7:98–107, 1990.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, pages 1177–1184. MIT Press, 2008.

H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22:400–407, 1951.

A. Ruszczyński and W. Syski. A method of aggregate stochastic subgradients with on-line stepsize rules for convex stochastic programming problems. volume 28 of *Mathematical Programming Studies*, pages 113–131. Springer Berlin Heidelberg, 1986.

J. Sacks. Asymptotic distribution of stochastic approximation procedures. *The Annals of Mathematical Statistics*, 29(2):373–405, 1958.

B. Scholkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA, 2001.

T. Serafini and L. Zanni. On the working set selection in gradient projection-based decomposition techniques for support vector machines. *Optimization Methods and Software*, 20:583–596, 2005.

T. Serafini, G. Zanghirati, and L. Zanni. Gradient projection methods for large quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20 (2–3):353–378, 2004.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 807–814, 2007.

W. Shi, G. Wahba, S. J. Wright, K. Lee, R. Klein, and B. Klein. LASSO-Patternsearch algorithm with application to opthalmology data. *Statistics and its Interface*, 1:137–153, January 2008.

A. Smola, S. V. N. Vishwanathan, and Q. Le. Bundle methods for machine learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 1377–1384. MIT Press, Cambridge, MA, 2008.

A. J. Smola, T. T. Frieß, and B. Schölkopf. Semiparametric support vector and linear programming machines. In *Advances in Neural Information Processing Systems 11*, pages 585–591, Cambridge, MA, USA, 1999. MIT Press.

J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley, Hoboken, NJ, 2003.

C. H. Teo, A. J. Smola, S. V. N. Vishwanathan, and Q. V. Le. A scalable modular convex solver for regularized risk minimization. In *Knowledge Discovery and Data Mining*, pages 727–736, 2007.

P. Tseng and S. Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, 47:179–206, 2010.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

I. Vaisman. *A First Course in Differential Geometry*. Monographs and Textbooks in Pure and Applied Mathematics. Marcel Dekker, 1984.

V. Vapnik. *Statistical Learning Theory*. Wiley, NY, 1998.

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, NY, USA, second edition, 1999.

V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

V. Vapnik and A. Chervonenkis. The necessary and sufficient conditions for consistency in the empirical risk minimization method. *Pattern Recognition and Image Analysis*, 1(3):283–305, 1991.

V. Vapnik and S. Kotz. *Estimation of Dependences Based on Empirical Data*. Information Science and Statistics. Springer, NY, USA, second edition, 2006.

V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

M. A. Vigotti, G. Rossi, L. Bisanti, A. Zanobetti, and J. Schwartz. Short term effects of urban air pollution on respiratory health in milan, italy, 1980-89. *Journal of Epidemiology Community Health*, 50:s71–s75, 1996.

G. Wahba. *Splines Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, 1990.

L. A. Wolsey. *Integer Programming*. Series in Discrete Mathematics and Optimization. John Wiley and Sons, NY, USA, 1998.

S. J. Wright. Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31(4):1063–1079, 1993.

S. J. Wright. Accelerated block-coordinate relaxation for regularized optimization. Technical report, University of Wisconsin-Madison, August 29 2010.

L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, October 2010.

T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 928–936, 2003.