May 2020

A Projection-Free Algorithm for Solving Support Vector Machine Models

Seyed Hamid Nazari
Clemson University, nazari.hamid.85@gmail.com

A projection-free algorithm for solving support vector machine models

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mathematics

---

by
Hamid Nazari
May 2020

Accepted by:
Dr. Yuyuan Ouyang, Committee Chair
Dr. Cristopher McMahan
Dr. Boshi Yang

Abstract

In this thesis our goal is to solve the dual problem of the *support vector machine* (SVM) problem, which is an example of convex smooth optimization problem over a polytope. To this goal, we apply the *conditional gradient* (CG) method by providing explicit solution to the linear programming (LP) subproblem. We also describe the *conditional gradient sliding* (CGS) method that can be considered as an improvement of CG in terms of number of gradient evaluations. Even though CGS performs better than CG in terms of optimal complexity bounds, it is not a practical method because it requires the knowledge of the Lipschitz constant and also the number of iterations. As an improvement of CGS, we designed a new method, *conditional gradient sliding with line search* (CGS-ls) that resolves the issues in CGS method. CGS-ls requires $\mathcal{O}(1/\sqrt{\epsilon})$ gradient evaluations and $\mathcal{O}(1/\epsilon)$ linear optimization calls that achieves the optimal complexity bounds in CGS method. We also compare the performance of our method with CG and CGS methods as numerical results by experimenting them in dual problem of SVM for binary classification of two subsets of the MNIST hand-written digits dataset.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis our problem of interest is

$$\min_{x \in \mathcal{X}} f(x) \tag{1.1}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a convex compact set and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. We assume that the gradient function $\nabla f(\cdot)$ is Lipschitz continuous (with respect to the norm $\|\cdot\|$) with Lipschitz constant $L > 0$, namely

$$\|\nabla f(x) - \nabla f(y)\|_* \leq L \|x - y\|, \ \forall x, y \in \mathbb{R}^n. \tag{1.2}$$

Here $\|\cdot\|$ is any norm and $\|\cdot\|_*$ is its dual norm, defined by

$$\|y\|_* := \sup_{\|x\| \leq 1} \langle x, y \rangle$$

for $y \in \mathbb{R}^n$. Also, we define the diameter of the set $\mathcal{X}$ as

$$D_{\mathcal{X}} \equiv D_{\mathcal{X}, \|\cdot\|} := \max_{x, y \in \mathcal{X}} \|x - y\|. \tag{1.3}$$

Throughout this chapter, we describe the properties and examples of problem (1.1), and also list several algorithms that can solve (1.1).

1.1 Properties of convex smooth functions

In this section we describe a few properties of convex smooth functions. These properties are necessary for analyzing the algorithms that we are going to introduce in the sequel. We start with the definition of a convex function.

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function if for any $x, y \in \mathbb{R}^n$ and any $\lambda \in [0, 1]$, we have*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)y. \tag{1.4}$$

We say that f is concave if $(-f)$ is convex. An important property of convex functions can be shown by an induction through our definition in (1.4):

$$f\left(\sum_{i=1}^{k} \lambda_i x_i\right) \leq \sum_{i=1}^{k} \lambda_i f(x_i), \tag{1.5}$$

for all $\lambda_1, \ldots, \lambda_k \in [0, 1]$ such that $\sum_{i=1}^{k} \lambda_i = 1$ and all $x_i \in \mathbb{R}^n,\ i = 1, \cdots, k$. In such case, $\sum_{i=1}^{k} \lambda_i x_i$ is called a convex combination of $x_1, \ldots, x_k$. We are now ready to introduce two important properties of convex smooth functions.

Proposition 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a smooth function that satisfies* (1.2)*, then*

$$|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \frac{L}{2} \|y - x\|^2,\ \forall x, y \in \mathbb{R}^n. \tag{1.6}$$

Proof. Let us fix any $x, y \in \mathbb{R}^n$ and define

$$h(\tau) := f((1 - \tau)x + \tau y),\ \forall \tau \in [0, 1].$$

Then we have $h(0) = f(x)$, $h(1) = f(y)$, and

$$h'(\tau) = \langle \nabla f((1 - \tau)x + \tau y), y - x \rangle.$$

By the fundamental theorem of calculus, we have

$$h(1) = h(0) + \int_0^1 h'(\tau) d\tau,$$

i.e.,

$$\begin{aligned} f(y) =& f(x) + \int_0^1 \langle \nabla f((1-\tau)x + \tau y), y - x \rangle \, d\tau \\ =& f(x) + \langle \nabla f(x), y - x \rangle + \int_0^1 \langle \nabla f((1-\tau)x + \tau y) - f(x), y - x \rangle \, d\tau. \end{aligned}$$

Therefore

$$\begin{aligned} |f(y) - f(x) - \langle \nabla f(x), y - x \rangle| &= \left| \int_0^1 \langle \nabla f((1-\tau)x + \tau y) - f(x), y - x \rangle \, d\tau \right| \\ &\leq \int_0^1 |\langle \nabla f((1-\tau)x + \tau y) - f(x), y - x \rangle| \, d\tau \\ &\leq \int_0^1 \|\nabla f((1-\tau)x + \tau y) - f(x)\|_* \|y - x\| \, d\tau \\ &\leq \int_0^1 L\tau \|y - x\|^2 \, d\tau = \frac{L}{2} \|y - x\|^2 . \end{aligned}$$

Here in the second inequality we use the Cauchy-Schwartz inequality.

□

Theorem 1. *A smooth function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle, \quad \forall x, y \in \mathbb{R}^n. \tag{1.7}$$

Proof. Let us fix any $x, y \in \mathbb{R}^n$, and denote $x_\lambda := \lambda x + (1-\lambda)y$. Suppose that (1.7) holds. Then for any $\lambda \in [0, 1]$ we have

$$f(x) \geq f(x_\lambda) + \langle \nabla f(x_\lambda), x - x_\lambda \rangle \text{ and } f(y) \geq f(x_\lambda) + \langle \nabla f(x_\lambda), y - x_\lambda \rangle.$$

Noting that $x - x_\lambda = (1-\lambda)(x - y)$ and $y - x_\lambda = \lambda(y - x)$, using the above relations, we have

$$\lambda f(x) + (1-\lambda) f(y) \geq \lambda \left[f(x_\lambda) + (1-\lambda) \langle \nabla f(x_\lambda), x - y \rangle \right] + (1-\lambda) \left[f(x_\lambda) + \lambda \langle \nabla f(x_\lambda), y - x \rangle \right]$$

$$= f(x_\lambda),\ \forall \lambda \in [0,1].$$

Therefore (1.4) holds and f is convex. Let us consider the other direction and suppose that (1.4) holds. Then for any $\lambda \in [0,1)$ we have

$$f(x_\lambda) \leq \lambda f(x) + (1-\lambda) f(y),$$

or

$$f(y) \geq \frac{f(x_\lambda) - \lambda f(x)}{1-\lambda} = f(x) + \frac{f(x_\lambda) - f(x)}{1-\lambda}. \tag{1.8}$$

Letting $\lambda \to 1$, we have

$$\lim_{\lambda \to 1} \frac{f(x_\lambda) - f(x)}{1-\lambda} = -\lim_{\lambda \to 1} \frac{f(y+\lambda(x-y)) - f(x)}{\lambda - 1} = -\frac{d}{d\lambda}\Big|_{\lambda=1} f(y+\lambda(x-y)) = -\langle f(x), x-y \rangle. \tag{1.9}$$

Combining (1.9) and (1.8) we obtain (1.7) and conclude the theorem.

□

From Proposition 1 and Theorem 1, we can prove the following corollary immediately.

Corollary 1. *Let f be a convex smooth function. We have,*

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y-x \rangle \leq \frac{L}{2}\|y-x\|^2,\ \forall x, y \in \mathbb{R}^n. \tag{1.10}$$

1.2 Support Vector Machine

In this section, we describe an example of (1.1). Suppose that we have a set of points in $\mathbb{R}^n$ and we would like to classify these points into k sets. This problem is called *classification* in *machine learning*. One classification model is the *support vector machine* (SVM). There are two types of SVMs, namely *hard-margin SVM* and *soft-margin SVM*. In this section, we describe the soft-margin SVM.

Suppose that we have $k = 2$ sets of data points. Let $X \in \mathbb{R}^{n \times m}$ be the matrix corresponding to the n points and b_i denotes the binary label of i^{th} point. Such classification problem is called

binary classification, namely, we are distinguishing two classes of data points. The optimization problem of the soft-margin SVM model [4] can be expressed as

$$\begin{aligned} \min_{w\in\mathbb{R}^m,\xi\in\mathbb{R}^n,w_0\in\mathbb{R}} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & b_i(\langle w, X_i\rangle - w_0) \geq 1-\xi_i, \quad i=1,\ldots,n \\ & \xi_i \geq 0. \end{aligned} \tag{1.11}$$

Here $\xi := (\xi_1, \cdots, \xi_n)^T$ and X_i^T denotes the i^{th} row of X. We will formulate the dual problem of (1.11). The Lagrangian function corresponding to (1.11) is

$$\begin{aligned} \mathcal{L}(w, w_0, \xi, x, \lambda) &= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n} x_i\left[b_i(w^T X_i + w_0) - 1 + \xi_i\right] - \sum_{i=1}^{n}\lambda_i\xi_i \\ &= \frac{1}{2}w^T w + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n} x_i b_i w^T X_i - \sum_{i=1}^{n} x_i b_i w_0 + \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i\xi_i - \sum_{i=1}^{n}\lambda_i\xi_i \\ &= \frac{1}{2}w^T w - w^T\sum_{i=1}^{n} x_i b_i X_i - \sum_{i=1}^{n} x_i b_i w_0 + (C - x_i - \lambda_i)\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n} x_i \end{aligned}$$

where $x = (x_1, \cdots, x_n)^T$ and $\lambda = (\lambda_1, \cdots, \lambda_n)^T$. So the Lagrangian dual will be

$$z_D = \sup_{\lambda\geq 0, x\geq 0}\ \inf_{w\in\mathbb{R}^n,\xi\in\mathbb{R}^m,w_0\in\mathbb{R}} \mathcal{L}(w, w_0, \lambda)$$

or

$$z_D = \sup_{\lambda\geq 0, x\geq 0}\left(\sum_{i=1}^{n} x_i + \inf_{w\in\mathbb{R}^m,\xi\in\mathbb{R}^n,w_0\in\mathbb{R}}\left(\frac{1}{2}w^T w - w^T\sum_{i=1}^{n} x_i b_i X_i - \sum_{i=1}^{n} x_i b_i w_0 + (C - x_i - \lambda_i)\sum_{i=1}^{n}\xi_i\right)\right)$$

Calling the function inside the inf function as $h(w, \xi, w_0)$ where $w,\ \xi \in \mathbb{R}^n$ and $w_0 \in \mathbb{R}$, we can observe that the function h is convex with respect to w and is affine with respect to ξ and w_0. So, by taking gradient of h with respect to w, ξ, and w_0 we can find the optimal $w^*,\ \xi^*$ and w_0^* for

$\inf_{w,w_0} h(w, w_0)$. So for w we have:

$$\begin{aligned}
\nabla_w h(w, \xi, w_0) &= w - \sum_{i=1}^{n} x_i b_i X_i = 0 \quad \Rightarrow \quad w^* = \sum_{i=1}^{n} x_i b_i X_i && (1.12)\\
\nabla_\xi h(w, \xi, w_0) &= C - x_i - \lambda_i = 0, \quad i = 1, \cdots, n && (1.13)\\
\nabla_{w_0} h(w, \xi, w_0) &= \sum_{i=1}^{n} x_i b_i = 0. && (1.14)
\end{aligned}$$

Having found w^* we can find w_0^*. Since we require the intercept w_0^* to satisfy

$$w_0^* \leq -1 - \max_{i:b_i=-1} w^{*T} X_i \tag{1.15}$$

$$w_0^* \geq \ 1 - \min_{i:b_i=1} w^{*T} X_i, \tag{1.16}$$

we take w_0^* to be the average the two values (1.15) and (1.16). Hence,

$$w_0^* = -\frac{\max_{i:b_i=-1} w^{*T} X_i + \min_{i:b_i=1} w^{*T} X_i}{2}. \tag{1.17}$$

Also, for any w and w_0 we have $\xi_i = 1 - b_i(\langle w, X_i \rangle - w_0),\ i = 1, \cdots, n$. Therefore,

$$\xi_i^* = 1 - b_i(\langle w^*, X_i \rangle - w_0^*), \quad i = 1, \cdots, n. \tag{1.18}$$

Substituting the optimal solution w^*, w_0^* and ξ^* to $\inf_{w,w_0} h(w, \xi, w_0)$ we obtain:

$$\begin{aligned}
\inf_{w \in \mathbb{R}^m, \xi \in \mathbb{R}^n, w_0 \in \mathbb{R}} h(w, \xi, w_0) &= \frac{1}{2} \left(\sum_{j=1}^{m} x_j b_j X_j \right) \left(\sum_{i=1}^{n} x_i b_i X_i \right) - \left(\sum_{j=1}^{m} x_j b_j X_j \right) \left(\sum_{i=1}^{n} x_i b_i X_i \right)\\
&= -\frac{1}{2} \left(\sum_{j=1}^{m} x_j b_j X_j \right) \left(\sum_{i=1}^{n} x_i b_i X_i \right)\\
&= -\frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{n} x_i x_j b_i b_j X_i^T X_j\\
&= -\frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{n} x_i x_j b_i b_j \langle X_i, X_j \rangle.
\end{aligned}$$

Note that we must have $x_i, \lambda_i \geq 0$. Also, from (1.13) we have $\lambda_i = C - x_i$. Therefore, we must have $x_i \leq C$. Summarizing the above derivation, the Lagrangian dual of (1.11) can be written as

$$\begin{aligned} \sup_{x\in\mathbb{R}^n} \quad & \sum_{i=1}^{n} x_i - \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n} x_i x_j b_i b_j \langle X_i, X_j\rangle \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i b_i = 0 \\ & 0 \le x_i \le C \quad i = 1,\dots,n. \end{aligned} \tag{1.19}$$

Note that the above is a special case of (1.1) with convex quadratic objective function

$$f(x) := \sum_{i=1}^{n} x_i - \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n} x_i x_j b_i b_j \langle X_i, X_j\rangle$$

and polytope feasible set

$$\mathcal{X} := \left\{ x \ : \ \sum_{i=1}^{n} x_i b_i = 0,\ 0 \le x_i \le C,\ i = 1,\dots,n \right\}.$$

Now we are ready to introduce and analyze some types of algorithms to solve (1.1). In general, we will discuss two types of algorithms, *projection-based* algorithms and *projection-free* algorithms. At each part, we will discuss the advantages and also the drawbacks of each type of algorithm.

1.3 Projection-based Algorithms

Projection-based algorithms are of the type that need projection as their subproblems. The projections that appear in these algorithms might be different depending on the problem structure. In this section we will describe and analyze the *projected gradient* and *Nesterov's accelerated gradient descent* methods [10]. After analyzing these algorithms we will have a section to provide several examples of projection to different sets that might appear in these algorithms as a subproblem.

1.3.1 Projected gradient method

As we can see from the name of this algorithm, projected gradient method is a projection-based algorithm. Projected gradient is one of the most straight forward projection-based algorithms. The simplest interpretation of this algorithm is that at each step we go through the negative direction

of the objective; if we are out side of the feasible set then we project back the point to the feasible set and continue with the projected point. We will describe a more general form of the projected gradient method using *prox-function*.

The algorithm of projected gradient method for solving (1.1) is described bellow.

Algorithm 1 The gradient descent algorithm

Choose $x_0 \in \mathcal{X}$.

for $k = 1, \ldots, N$ **do**

$$x_k = \arg\min_{x \in \mathcal{X}} \langle \nabla f(x_{k-1}), x \rangle + \eta_k V(x_{k-1}, x) \tag{1.20}$$

end for

Output x_N.

Here $V(\cdot, \cdot)$ is a function, called the *prox function*, that satisfies the following two inequalities:

1. For any $x, y \in \mathbb{R}^n$,

$$V(x, y) \geq \frac{1}{2}\|x - y\|^2. \tag{1.21}$$

2. For any $g \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$, if y is the solution to the problem

$$\min_{x \in X} \langle g, x \rangle + \eta V(u, x),$$

then

$$\langle g, y - x \rangle \leq \eta[V(u, x) - V(u, y) - V(y, x)], \ \forall x \in \mathcal{X}. \tag{1.22}$$

The simplest choice of $V(\cdot, \cdot)$ is $V(x, y) := \|y - x\|_2^2/2$, when the norms $\|\cdot\|$ and $\|\cdot\|_*$ are both 2-norms. Note also that the gradient method is a special case of Algorithm 2 with $\gamma_k \equiv 1$, $\mathcal{X} = \mathbb{R}^n$, $V(x, y) := \|y - x\|_2^2/2$, and the norms $\|\cdot\|$ and $\|\cdot\|_*$ are both 2-norms. Note that if we set

$V(x_{k-1}, x) = \|x - x_{k-1}\|_2^2 / 2$, then (1.20) in Algorithm 1 becomes

$$\begin{aligned} x_k &= \arg\min_{x \in \mathbb{R}^n} \langle \nabla f(x_{k-1}), x \rangle + \frac{\eta_k}{2} \|x - x_{k-1}\|_2^2 \\ &= \arg\min_{x \in \mathbb{R}^n} \frac{\eta_k}{2} \left\| x - x_{k-1} + \frac{1}{\eta_k} \nabla f(x_{k-1}) \right\|^2 \\ &= x_{k-1} - \frac{1}{\eta_k} \nabla f(x_{k-1}). \end{aligned} \tag{1.23}$$

The geometric interpretation of above is clear; the new iteration x_k is computed by moving from x_{k-1} along the opposite direction of $\nabla f(x_{k-1})$ with stepsize $1/\eta_k$. The intuition is that the negative direction $-\nabla f(x)$ is the direction of the fastest local decrement of f at point x.

In the following theorem, we state the convergence result of the projected gradient algorithm in Algorithm 1, assuming constant stepsize.

Theorem 2. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. If the parameters η_k in Algorithm 1 satisfy*

$$\eta_k \equiv \eta \geq L, \tag{1.24}$$

then we have

$$f(\bar{x}_N) - f^* \leq \frac{\eta}{N} V(x_0, x^*),$$

where

$$\bar{x}_N := \frac{1}{N} \sum_{k=1}^{N} x_k.$$

Proof. Since f is a smooth convex function, from the Corollary 1 and Theorem 1 we have

$$\begin{aligned} f(x_k) \leq & f(x_{k-1}) + \langle \nabla f(x_{k-1}), x_k - x_{k-1} \rangle + \frac{L}{2} \|x_k - x_{k-1}\|_2^2 \\ = & f(x_{k-1}) + \langle \nabla f(x_{k-1}), x - x_{k-1} \rangle + \langle \nabla f(x_{k-1}), x_k - x \rangle + \frac{L}{2} \|x_k - x_{k-1}\|^2 && (1.25) \\ \leq & f(x) + \langle \nabla f(x_{k-1}), x_k - x \rangle + \frac{\eta}{2} \|x_k - x_{k-1}\|^2 && (1.26) \\ \leq & f(x) + \eta \left(V(x_{k-1}, x) - V(x_{k-1}, x_k) - V(x_k, x) \right) + \eta V(x_k, x_{k-1}) && (1.27) \end{aligned}$$

$$= f(x) + \eta \left(V(x_{k-1}, x) - V(x_k, x) \right),$$

where the equality (1.25) is from (1.24), inequality (1.26) is from convexity of f, and inequality (1.27) is from (1.22). Hence,

$$f(x_k) \leq f(x) + \eta \left(V(x_{k-1}, x) - V(x_k, x) \right).$$

Summing the above inequality up from $k = 1$ to N, we obtain

$$\begin{aligned} \sum_{k=1}^{N} f(x_k) &\leq N f(x) + \eta \left(V(x_{k-1}, x) - V(x_k, x) \right) \\ &\leq N f(x) + \eta V(x_0, x). \end{aligned}$$

Setting $x = x^*$ in above relation and using the convexity of f, we have

$$f(\bar{x}_N) \leq \frac{1}{N} \sum_{k=1}^{N} f(x_k) \leq f(x^*) + \frac{\eta}{N} V(x_0, x^*).$$

Therefore,

$$f(\bar{x}_N) - f^* \leq \frac{\eta}{N} V(x_0, x^*).$$

□

From the above theorem, we observe that in order to compute an approximate solution $\bar{x}_N$ such that $f(\bar{x}_N) - f^* \leq \varepsilon$, the number of iterations that are required is bounded by $\mathcal{O}\left({}^{LV(x_0,x^*)}/_{\epsilon}\right)$.

1.3.2 Nesterov's accelerated gradient method

In the previous section we obtained an $\mathcal{O}({}^{1}/_{\epsilon})$ convergence result of the projected gradient descent method. In this section we introduce a method that has a better complexity. This algorithm is called Nesterov's *accelerated gradient descent* (AGD) method. Similar to projected gradient descent, AGD is a projection-based algorithm.

The algorithm of accelerated gradient descent method for solving (1.1) is described bellow.

Algorithm 2 Accelerated Gradient Descent Method

Choose $x_0 \in \mathcal{X}$ and set $y_0 = x_0$.
for $k = 1, \cdots, N$ **do**

$$z_k = (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1} \tag{1.28}$$
$$x_k = \arg\min_{x\in\mathcal{X}} \langle \nabla f(z_k), x\rangle + \eta_k V(x_{k-1}, x) \tag{1.29}$$
$$y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k \tag{1.30}$$

end for
Output y_N.

We present the convergence result of Algorithm 2. In Theorem 3 below we describe a general result.

Theorem 3. *Suppose that y_k and z_k in Algorithm 2 satisfy*

$$f(y_k) \le f(z_k) + \langle \nabla f(z_k), y_k - z_k\rangle + \frac{L_k}{2}\|y_k - z_k\|^2 \tag{1.31}$$

for some $L_k > 0$, and that the parameters in Algorithm 2 satisfy

$$\gamma_1 = 1,\ \gamma_k \in [0,1),\ \text{and } \eta_k \ge L_k\gamma_k, \quad \forall k \ge 1. \tag{1.32}$$

Letting Γ_k be a parameter that satisfies $\Gamma_1 > 0$ and

$$\Gamma_k = (1-\gamma_k)\Gamma_{k-1},\ \forall k > 1, \tag{1.33}$$

then we have

$$f(y_k) - f^* \le \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i\eta_i}{\Gamma_i}[V(x_{i-1}, x^*) - V(x_i, x^*)],$$

where x^* is a solution to (1.1).

Proof. Noting (1.28) and (1.30) we have $y_k - z_k = \gamma_k(x_k - x_{k-1})$, and also

$$\begin{aligned} y_k - z_k &= y_k - y_{k-1} + y_{k-1} - z_k \\ &\overset{(1.30)}{=} \gamma_k(x_k - y_{k-1}) + y_{k-1} - z_k \\ &= \gamma_k\left[(x_k - x) + (x - z_k) + (z_k - y_{k-1})\right] + y_{k-1} - z_k \\ &= (1-\gamma_k)(y_{k-1} - z_k) + \gamma_k\left((x - z_k) + (x_k - x)\right) \\ &= (1-\gamma_k)(y_{k-1} - z_k) + \gamma_k\left(x_k - z_k\right). \end{aligned} \tag{1.34}$$

Using above the inequality, (1.31) becomes

$$\begin{aligned} f(y_k) &\le f(z_k) + (1-\gamma_k)\langle \nabla f(z_k), y_{k-1} - z_k\rangle + \gamma_k\langle \nabla f(z_k), x_k - z_k\rangle + \frac{L_k\gamma_k^2}{2}\|x_k - x_{k-1}\|^2 \\ &= (1-\gamma_k)[f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle] + \gamma_k[f(z_k) + \langle \nabla f(z_k), x - z_k\rangle + \langle \nabla f(z_k), x_k - x\rangle] \\ &\quad + \frac{L_k\gamma_k^2}{2}\|x_k - x_{k-1}\|^2, \quad \forall x \in \mathcal{X}. \end{aligned}$$

Let us make three observations. First, by (1.10), we have

$$f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle \le f(y_{k-1}),$$

and

$$f(z_k) + \langle \nabla f(z_k), x - z_k\rangle \le f(x).$$

Second, by (1.22) (with $g = \nabla f(z_k)$, $y = x_k$, $u = x_{k-1}$, and $\eta = \eta_k$) we have

$$\langle \nabla f(z_k), x_k - x\rangle \le \eta_k[V(x_{k-1}, x) - V(x_{k-1}, x_k) - V(x_k, x)], \ \forall x \in \mathcal{X}.$$

Third, by (1.21), we have

$$\frac{L_k\gamma_k^2}{2}\|x_k - x_{k-1}\|^2 \le L_k\gamma_k^2 V(x_{k-1}, x_k).$$

Summarizing the three observations, we have

$$\begin{aligned} f(y_k) &\le (1-\gamma_k) f(y_{k-1}) + \gamma_k f(x) + \gamma_k \eta_k [V(x_{k-1}, x) - V(x_{k-1}, x_k) - V(x_k, x)] \\ &\quad + L_k \gamma_k^2 V(x_{k-1}, x_k) \\ &= (1-\gamma_k) f(y_{k-1}) + \gamma_k f(x) + \gamma_k \eta_k [V(x_{k-1}, x) - V(x_k, x)] \\ &\quad - \gamma_k (\eta_k - L_k \gamma_k) V(x_{k-1}, x_k) \\ &\le (1-\gamma_k) f(y_{k-1}) + \gamma_k f(x) + \gamma_k \eta_k [V(x_{k-1}, x) - V(x_k, x)]. \end{aligned}$$

Here the last inequality is from (1.32). Summing up the above two inequalities, we have

$$f(y_k) \le (1-\gamma_k) f(y_{k-1}) + \gamma_k f(x) + \gamma_k \eta_k [V(x_{k-1}, x) - V(x_k, x)].$$

In particular, letting $x = x^*$ where x^* is a solution to (1.1), we can reformulate the above to

$$f(y_k) - f^* \le (1-\gamma_k)(f(y_{k-1}) - f^*) + \gamma_k \eta_k [V(x_{k-1}, x^*) - V(x_k, x^*)].$$

Dividing both sides by Γ_k, and using (1.33) and (1.32), we have

$$\frac{1}{\Gamma_k}(f(y_k) - f^*) \le \frac{1}{\Gamma_{k-1}}(f(y_{k-1}) - f^*) + \frac{\gamma_k \eta_k}{\Gamma_k}[V(x_{k-1}, x^*) - V(x_k, x^*)], \quad \forall k > 1.$$

Also, when $k = 1$, noting that $\gamma_1 = 1$ by (1.32), we have

$$\frac{1}{\Gamma_1}(f(y_k) - f^*) \le \frac{\gamma_1 \eta_1}{\Gamma_1}[V(x_0, x^*) - V(x_1, x^*)].$$

Using induction on the two inequalities above, we conclude that

$$\frac{1}{\Gamma_k}(f(y_k) - f^*) \le \sum_{i=1}^{k} \frac{\gamma_i \eta_i}{\Gamma_i}[V(x_{i-1}, x^*) - V(x_i, x^*)].$$

□

In the corollary below, we describe an example of parameter setting of Algorithm 2.

Corollary 2. *If we set*

$$\gamma_k = \frac{2}{k+1}, \quad \eta_k = \frac{2L}{k} \tag{1.35}$$

in Algorithm 2, then

$$f(y_k) - f^* \leq \frac{4L}{k(k+1)} V(x_0, x^*). \tag{1.36}$$

Proof. Clearly (1.31) and (1.32) hold, and $\Gamma_k = 2/k(k+1)$ satisfies (1.33) with $\Gamma_1 = 1$. Therefore, by Theorem 3, we have

$$f(y_k) - f^* \leq \frac{2}{k(k+1)} \sum_{i=1}^{k} 2L[V(x_{i-1}, x^*) - V(x_i, x^*)] = \frac{4L}{k(k+1)}[V(x_0, x^*) - V(x_k, x^*)]. \tag{1.37}$$

From (1.21), we have $V(x_k, x^*) \geq 0$. Thus

$$f(y_k) - f^* \leq \frac{4L}{k(k+1)} V(x_0, x^*). \tag{1.38}$$

□

From Corollary 2 we can see that in order to compute an approximate solution such that $f(y_k) - f^* \leq \varepsilon$, we need

$$k \geq \sqrt{\frac{4LV(x_0, x^*)}{\varepsilon}}. \tag{1.39}$$

Therefore, the iteration complexity upper bound is $\mathcal{O}(\sqrt{1/\varepsilon})$. Note that the Nesterov's accelerated gradient method is optimal for solving smooth convex optimization with Lipschitz constant L [10].

1.4 Examples of projections

In this section we have some examples of projections of different sets. As we have seen in previous sections these projections arise from the subproblems of the projection-based algorithms.

1.4.1 Projection onto the standard simplex under the Euclidean prox-function

Assume that $\mathcal{X}$ in (1.1) is a standard simplex Δ_n. We will study the projection subproblem (1.29) in Algorithm 2 where the prox-function is defined as

$$V(x,y) = \frac{1}{2}\|x-p\|^2, \quad p \in \mathbb{R}^n.$$

Note that similar analysis can be performed for Algorithm 1.

The subproblem (1.29) in this case can be formulates as

$$\min_{x\in\mathbb{R}^n} \quad \langle \nabla f(z_k), x\rangle + \frac{\eta_k}{2}\|x-p\|^2 \tag{1.40}$$

$$\text{s.t.} \quad -x^{(i)} \le 0 \quad i = 1,\cdots,n \tag{1.41}$$

$$\sum_{i=1}^{n} x^{(i)} - 1 = 0 \tag{1.42}$$

and without loss of generality we can assume that $p \in \mathbb{R}^n$ satisfies $p^{(1)} \le p^{(2)} \le \cdots \le p^{(n)}$.

To solve the problem (1.4.1) since the Slater's condition holds we may consider the KKT points

a) Primal feasibility

$$-x^{(i)} \le 0 \quad i = 1,\cdots,n$$
$$\sum_{i=1}^{n} x^{(i)} - 1 = 0$$

b) Complementary slackness

$$-u^{(i)}x^{(i)} = 0, \ \forall i = 1,\cdots,n$$

c) Dual feasiblity

$$\nabla f(z_k)^{(i)} + x^{(i)} - p^{(i)} - u^{(i)} + v = 0, \ \forall i = 1,\cdots,n$$
$$u^{(i)} \ge 0, \ \forall i = 1,\cdots,n$$

For any $i = 1, \cdots, n$ if $u^{(i)} = 0$, then by dual feasibility we have

$$x^{(i)} = p^{(i)} - v - \nabla f(z_k)^{(i)}.$$

Since $x^{(i)} \geq 0$, we need $p^{(i)} \geq v + \nabla f(z_k)^{(i)}$. If $u^{(i)} > 0$, then from complementary slackness we have $x^{(i)} = 0$ and by dual feasibility

$$u^{(i)} = v - p^{(i)} + \nabla f(z_k)^{(i)}$$

and since $u^{(i)} > 0$, then $p^{(i)} < v + \nabla f(z_k)^{(i)}$. In summary,

$$x^{(i)} = \begin{cases} p^{(i)} - v - \nabla f(z_k)^{(i)} & \text{if } p^{(i)} \geq v + \nabla f(z_k)^{(i)} \\ 0 & \text{otherwise.} \end{cases}$$

Note that from primal feasibility and dual feasibility we have

$$1 - \sum_{i=1}^{n} (p^{(i)} + u^{(i)} - \nabla f(z_k)^{(i)}) + nv = 0,$$

which implies that

$$\begin{aligned} v &= \frac{1}{n} \left(\sum_{i=1}^{n} (p^{(i)} + u^{(i)} - \nabla f(z_k)^{(i)}) \right) - \frac{1}{n} \\ &= \frac{1}{n} \left(\sum_{i:p^{(i)} \geq v + \nabla f(z_k)^{(i)}} (p^{(i)} - \nabla f(z_k)^{(i)}) + \sum_{i:p^{(i)} < v + \nabla f(z_k)^{(i)}} v \right) - \frac{1}{n} \end{aligned}$$

or

$$\sum_{i:p^{(i)} \geq v + \nabla f(z_k)^{(i)}} v = \sum_{i:p^{(i)} \geq v + \nabla f(z_k)^{(i)}} (p^{(i)} - \nabla f(z_k)^{(i)}) - 1.$$

Therefore, we have the following cases:

i) $v \leq p^{(1)} - \nabla f(z_k)^{(1)}$. In this case

$$nv = \sum_{i=1}^{n} (p^{(i)} - \nabla f(z_k)^{(i)}) - 1 \qquad \text{or} \qquad v = \frac{1}{n} \sum_{i=1}^{n} (p^{(i)} - \nabla f(z_k)^{(i)}) - \frac{1}{n}$$

ii) $p^{(j)} - \nabla f(z_k)^{(j)} < v \leq p^{(j+1)} - \nabla f(z_k)^{(j+1)}$. In this case

$$(n+j)v = \sum_{i=j+1}^{n} (p^{(i)} - \nabla f(z_k)^{(i)}) - 1$$

iii) $v > p^{(n)} - \nabla f(z_k)^{(n)}$ which is infeasible.

Hence, we have n possible choices of v as

$$v = \begin{cases} \frac{1}{n}\sum_{i=1}^{n}(p^{(i)} - \nabla f(z_k)^{(i)}) - \frac{1}{n} & \text{if } np^{(1)} \geq \sum_{i=1}^{n}(p^{(i)} - \nabla f(z_k)^{(i)}) - 1 \\ \frac{1}{n+j}\sum_{i=j+1}^{n}(p^{(i)} - \nabla f(z_k)^{(i)}) - \frac{1}{n+j} & \text{if } (n+j)p^{(j)} \geq \sum_{i=1}^{n}(p^{(i)} - \nabla f(z_k)^{(i)}) - 1 \leq (n+j)p^{(j+1)} \end{cases}$$

where $j = 1, \cdots, n-1$. The optimal solution is of the form

$$(x^*)^{(i)} = \begin{cases} p^{(i)} - v - \nabla f(z_k)^{(i)} & \text{if } p^{(i)} \geq v + \nabla f(z_k)^{(i)} \\ 0 & \text{otherwise.} \end{cases}$$

1.4.2 Projection onto the standard simplex under entropy prox-function

Assume that $\mathcal{X}$ in (1.1) is a standard simplex Δ_n. We will study the projection subproblem (1.29) in Algorithm 2.

If we define the prox-function in (1.29) as

$$V(x,y) = \omega(x) - \omega(y) - \langle \nabla\omega(y), x - y\rangle, \tag{1.43}$$

where $\omega(x) = \sum_{i=1}^{n} x^{(i)} \log x^{(i)}$, then we have

$$\begin{aligned} &\arg\min_{x\in\Delta_n} \langle \nabla f(z_k), x\rangle + \eta_k \left(\omega(x) - \omega(x_{k-1}) - \langle \nabla\omega(x_{k-1}), x - x_{k-1}\rangle\right) \\ &\quad = \arg\min_{x\in\Delta_n} \langle \nabla f(z_k), x\rangle \\ &\qquad + \eta_k \left(\sum_{i=1}^{n} x^{(i)} \log x^{(i)} - \sum_{i=1}^{n} (x_{k-1})^{(i)} \log x_{k-1}^{(i)} - \sum_{i=1}^{n} (\log x_{k-1}^{(i)} + 1)(x^{(i)} - x_{k-1}^{(i)})\right) \\ &\quad = \arg\min_{x\in\Delta_n} \langle \nabla f(z_k), x\rangle + \eta_k \left(\sum_{i=1}^{n} x^{(i)} \log \frac{x^{(i)}}{x_{k-1}^{(i)}} - \sum_{i=1}^{n} x^{(i)} + \sum_{i=1}^{n} x_{k-1}^{(i)}\right) \\ &\quad = \arg\min_{x\in\Delta_n} \langle \nabla f(z_k), x\rangle + \eta_k \sum_{i=1}^{n} x^{(i)} \log \frac{x^{(i)}}{x_{k-1}^{(i)}} \end{aligned}$$

$$= \arg\min_{x\in\Delta_n} \langle \nabla f(z_k), x\rangle + \eta_k \sum_{i=1}^n x^{(i)} \log x^{(i)} - \eta_k \sum_{i=1}^n x^{(i)} \log x_{k-1}^{(i)}$$

$$= \arg\min_{x\in\Delta_n} \left\langle \frac{1}{\eta_k}\nabla f(z_k) - \begin{bmatrix} \log x_{k-1}^{(1)} \\ \vdots \\ \log x_{k-1}^{(n)} \end{bmatrix}, x \right\rangle + \sum_{i=1}^n x^{(i)} \log x^{(i)}.$$

Now we rewrite the subproblem (1.29) as

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \langle \nabla f(z_k), x\rangle + \eta_k \sum_{i=1}^n x^{(i)} \log \frac{x^{(i)}}{x_{k-1}^{(i)}} \\ \text{s.t.} \quad & -x^{(i)} \le 0 \\ & \sum_{i=1}^n x^{(i)} - 1 = 0 \end{aligned}$$

Or equivalently

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \langle g, x\rangle + \sum_{i=1}^n x^{(i)} \log x^{(i)} \\ \text{s.t.} \quad & -x^{(i)} \le 0 \\ & \sum_{i=1}^n x^{(i)} - 1 = 0 \end{aligned} \tag{1.44}$$

where $g = \frac{1}{\eta_k}\nabla f(z_k) - \begin{bmatrix} \log x_{k-1}^{(1)} \\ \vdots \\ \log x_{k-1}^{(n)} \end{bmatrix}$. The optimization problem (1.44) is the subproblem of the accelerated gradient descent method with prox-function defined in (1.43). Proposition 2 gives the solution to the above.

Proposition 2. *Let $\mathcal{X}$ in subproblem of Algorithm 2 be the standard simplex defined as*

$$\Delta_n = \{x \in \mathbb{R}^n \ : \ \sum_{i=1}^n x^{(i)} = 1,\ x^{(i)} \ge 0,\ i = 1, \cdots, n\}.$$

and the prox-function $V(\cdot,\cdot)$ be of the form (1.43)*, where $\omega(x) := \sum_{i=1}^n x^{(i)} \log x^{(i)}$. Then the i-th*

element of the optimal solution x^* to the subproblem (1.44) *is*

$$(x^*)^{(i)} = \frac{e^{-g^{(i)}}}{\sum_{i=1}^{n} e^{-g^{(i)}}}.$$

Proof. To solve the problem (1.44) since the Slater's conditions hold we check the KKT points

a) Primal feasibility:

$$\sum_{i=1}^{n} x^{(i)} = 1$$
$$x^{(i)} \geq 0$$

b) Complementary slackness:

$$-u^{(i)} x^{(i)} = 0, \ \forall i = 1, \cdots, n$$

c) Dual feasibility:

$$g^{(i)} + \log x^{(i)} + 1 - u^{(i)} + v = 0, \ \forall i = 1, \cdots, n$$
$$u^{(i)} \geq 0, \ \forall i = 1, \cdots, n$$

Since for any i, $x^{(i)} > 0$, then we must have $u^{(i)} = 0, \ \forall i = 1, \cdots, n$. This implies that $\log x^{(i)} = -g^{(i)} - 1 - v$ and so

$$x^{(i)} = \frac{1}{e^{1+v} \times e^{g^{(i)}}}.$$

Since $\sum_{i=1}^{n} x^{(i)} = 1$ we conclude that

$$1 = \frac{1}{e^{1+v}} \sum_{i=1}^{n} e^{-g^{(i)}}$$

Taking log from both sides and writing the equality with respect to v we get

$$v = \log \left(\sum_{i=1}^{n} e^{-g^{(i)}} \right) - 1$$

Therefore

$$\begin{aligned}(x^*)^{(i)} &= \frac{1}{\exp\left(\log\sum_{i=1}^n e^{-g^{(i)}}\right) e^{g^{(i)}}} \\ &= \frac{e^{-g^{(i)}}}{\sum_{i=1}^n e^{-g^{(i)}}}.\end{aligned}$$

□

Note that the choice of $V(\cdot,\cdot)$ in Proposition 2 is also called the entropy prox-function.

1.5 Projection-free Algorithms

In previous section we mentioned projected gradient and accelerated gradient descent as examples of projection-based methods. These methods require projection as subproblems and specially the AGD with complexity $\mathcal{O}(\sqrt{1/\epsilon})$ is an efficient algorithm. However, the projection in subproblems of these algorithms are sometimes problematic. They are not always efficiently solvable. Algorithms called projection-free algorithms are useful in such expensive cases.

Similar to previous section, we mention some of the projection free algorithms and discuss their complexities. The methods that we are going describe are *conditional gradient* and *conditional gradient sliding*. We will compare their complexity with themselves and also with projection-based algorithm that we had in the last section.

1.5.1 Conditional Gradient Algorithm

Conditional gradient (CG) algorithm, also known as Frank-Wolfe method, is one of the earliest projection-free first-order algorithms for solving convex programming problems. It was initially developed by Frank and Wolfe in 1956 [5]. The algorithm of CG is described in Algorithm 3.

As we can observe in Algorithm 3, the CG method solves the projection subproblem (1.29) of AG approximately over the feasible set $\mathcal{X}$. Regarding the CG algorithm we should mention a few remarks. First, in CG method the assumption of compactness of $\mathcal{X}$ is important, because otherwise the subproblem (1.46) might become unbounded. Second, the solution to (1.46) is not necessarily unique and there might exist multiple solutions. Third, it is better to avoid setting $\gamma_k \equiv 1$. As an

example, consider the problem with $f(x) = x^2$ and $\mathcal{X} = [-1, 1]$. Setting $x_0 = z_0 = 1$ where $\gamma_k \equiv 1$ imply that the CG method has $x_1 = x_3 = \cdots = 1$ and $x_0 = x_2 = \cdots = -1$ as its outputs.

Algorithm 3 Conditional Gradient Algorithm

Choose $z_0 \in \mathcal{X}$ and set $x_0 = z_0$.
for $k = 1, \cdots, N$ **do**

$$z_k = (1 - \gamma_k) y_{k-1} + \gamma_k x_{k-1} \tag{1.45}$$
$$x_k \in \operatorname*{Argmin}_{x \in \mathcal{X}} \langle \nabla f(z_k), x \rangle \tag{1.46}$$
$$y_k = (1 - \gamma_k) y_{k-1} + \gamma_k x_k \tag{1.47}$$

end for
Output x_N.

We describe the convergence result [5] of the CG method in Algorithm 3. We first state a simple technical result that will be used in the analysis of the algorithm.

Lemma 1. *Let $w_t \in (0, 1]$, $t = 1, 2, \cdots$, be given. Also let us denote*

$$W_t := \begin{cases} 1, & t = 1, \\ (1 - w_t) W_{t-1}, & t \geq 2. \end{cases} \tag{1.48}$$

Suppose that $W_t > 0$ for all $t \geq 2$ and that the sequence $\{\delta_t\}_{t \geq 0}$ satisfies

$$\delta_t \leq (1 - w_t) \delta_{t-1} + B_t, \quad t = 1, 2, \cdots. \tag{1.49}$$

Then for any $1 \leq l \leq k$, we have

$$\delta_k \leq W_k \left(\frac{1 - w_l}{W_l} \delta_{l-1} + \sum_{i=l}^{k} \frac{B_i}{W_i} \right). \tag{1.50}$$

Proof. Dividing both sides of (1.49) by W_t, we obtain

$$\frac{\delta_1}{W_1} \leq \frac{(1 - w_1) \delta_0}{W_1} + \frac{B_1}{W_1}$$

and

$$\frac{\delta_i}{W_i} \leq \frac{(1-w_i)\delta_{i-1}}{W_i} + \frac{B_i}{W_i} = \frac{\delta_{i-1}}{W_{i-1}} + \frac{B_i}{W_i} \quad \forall i \geq 2.$$

The result then immediately follows by summing up the above inequalities for $i = 1 \cdots, k$ and rearranging the terms. □

In the following theorem and corollary the notation below will be used:

$$\Gamma_k = \begin{cases} 1 & \text{if } k = 1 \\ (1-\gamma_k)\Gamma_{k-1} & \text{if } k \geq 2. \end{cases} \tag{1.51}$$

Theorem 4. *For parameters $\gamma_k \in (0,1)$ in Algorithm we have*

$$f(y_k) - f^* \leq \frac{LD^2\Gamma_k}{2} \sum_{i=1}^{k} \frac{\gamma_i^2}{\Gamma_i}. \tag{1.52}$$

Proof. Fist, we can write (1.47) as $y_k = y_{k-1} + \gamma_k(x_k - y_{k-1})$, so

$$y_k - y_{k-1} = \gamma_k(x_k - y_{k-1}). \tag{1.53}$$

Also, from (1.45) and (1.45) we observe that

$$\begin{aligned} y_k - z_k &= y_k - y_{k-1} + y_{k-1} - z_k \\ &\overset{(1.53)}{=} \gamma_k(x_k - y_{k-1}) + y_{k-1} - z_k \\ &= \gamma_k \left[(x_k - x) + (x - z_k) + (z_k - y_{k-1})\right] + y_{k-1} - z_k \\ &= \gamma_k \left((x - z_k) + (x_k - x)\right) + (1-\gamma_k)(y_{k-1} - z_k). \end{aligned} \tag{1.54}$$

Since f is a smooth convex function from Corollary (1) and also from (1.47) we have

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L}{2} \|y_k - z_k\|^2. \tag{1.55}$$

Now from (1.54), for any $x \in X$ we have

$$f(y_k) \leq (1-\gamma_k) \left[f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle\right]$$

$$
\begin{aligned}
&+ \gamma_k \left[f(z_k) + \langle \nabla f(z_k), x - z_k \rangle + \langle \nabla f(z_k), x_k - x \rangle \right] \\
&+ \frac{L}{2} \left\| y_k - z_k \right\|^2 \\
&= (1 - \gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle) \\
&+ \gamma_k (f(z_k) + \langle \nabla f(z_k), x - z_k \rangle) \\
&+ \gamma_k \langle \nabla f(z_k), x_k - x \rangle + \frac{L}{2} \left\| y_k - z_k \right\|^2 .
\end{aligned}
$$

Here we note that since x_k is an optimal solution to the subproblem (1.46), then by optimality condition we have

$$\langle \nabla f(z_k), x_k - x \rangle \leq 0,$$

and also since f is a convex function we have

$$f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle \leq f(y_{k-1})$$

and

$$f(z_k) + \langle \nabla f(z_k), x - z_k \rangle \leq f(x).$$

In addition, from (1.45) and (1.47) we have $y_k - z_k = \gamma_k(x_k - x_{k-1})$. Summarizing all these and using (1.3) we obtain

$$f(y_k) \leq (1 - \gamma_k) f(y_{k-1}) + \gamma_k f(x) + \frac{L\gamma_k^2}{2} D^2,$$

or equivalently,

$$f(y_k) - f(x) \leq (1 - \gamma_k)(f(y_{k-1}) - f(x)) + \frac{L\gamma_k^2}{2} D^2.$$

Dividing both sides of the above inequality by Γ_k for $k \geq 2$ we obtain

$$\frac{f(y_k) - f(x)}{\Gamma_k} \leq \frac{(1 - \gamma_k)}{\Gamma_k} (f(y_{k-1}) - f(x)) + \frac{L\gamma_k^2}{2\Gamma_k} D^2.$$

Also, for $k = 1$ we have

$$f(y_1) - f(x) \leq \frac{LD^2\gamma_1^2}{2}.$$

Summing up from 1 to k we conclude that

$$\frac{1}{\Gamma_k}(f(y_k) - f(x)) \leq \frac{LD^2}{2}\sum_{i=1}^{k}\frac{\gamma_i^2}{\Gamma_i}.$$

□

Note that the sequence of parameters γ_k in CG is conceptual and clearly there might be many choices to set these parameters properly to get the best convergence result for this algorithm. In Corollary 3 below we provide a setting for this parameter and prove the convergence rate corresponding to our setting.

Corollary 3. *In Algorithm 3, if we set the parameter* $\gamma_k = 2/(k+1)$*, then*

$$f(y_k) - f^* \leq \frac{2LD^2}{k+1}. \tag{1.56}$$

Proof. With γ_k defined in assumption we have

$$\Gamma_k = \frac{2}{k(k+1)}. \tag{1.57}$$

Using (1.57) we obtain

$$\sum_{i=1}^{k}\frac{\gamma_i^2}{\Gamma_i} = \sum_{i=1}^{k}\frac{4}{(i+1)^2} \times \frac{i(i+1)}{2} = \sum_{i=1}^{k}\frac{2i}{i+1} = 2(\sum_{i=1}^{k}1 - \sum_{i=1}^{k}\frac{1}{i+1}) \leq 2k.$$

Therefore,

$$\begin{aligned} f(y_k) - f^* &\leq \frac{LD^2}{2} \times \frac{2}{k(k+1)} \times 2k \\ &= \frac{2LD^2}{k+1}. \end{aligned}$$

□

Corollary 3 shows that the CG method computes an ϵ-solution to the problem (1.1) in $\mathcal{O}(LD^2/\epsilon)$ iterations. This means that in order to compute an ϵ solution, CG requires more evaluations of ∇f than AGD method, which only requires $\mathcal{O}(\sqrt{1/\epsilon})$ evaluations. This drawback is resolved in *conditional gradient sliding* method [8]. In particular, conditional gradient sliding method requires $\mathcal{O}(\sqrt{1/\epsilon})$ evaluations of $\nabla f(\cdot)$ and $\mathcal{O}(1/\epsilon)$ evaluations for linear optimization problems of form (1.46). we will discuss the conditional gradient sliding method in later sections.

According to [6, 9] the number of evaluations of linear optimization problems of form (1.46) can not be improved from the lower complexity bound $\mathcal{O}(1/\epsilon)$. Also, it should be noted that the CG does not require knowledge on the Lipschitz constant L, the norm $\|.\|$, and diameter D. In particular, if there exists a norm $\|.\|$ that yields the smallest possible value of LD^2, then the convergence result (1.56) will follow such smallest value. In other words, the CG is a first-order method that would automatically adapt to the best possible geometric properties of the problem.

1.5.2 Conditional Gradient Sliding Algorithm

In this section we describe an algorithm and its analysis by [8]. The goal of the *conditional gradient sliding* (CGS) [8] method is to present a new linear optimization based convex programming method which can skip the computation for the gradient of f from time to time when performing Linear optimization over the feasible region $\mathcal{X}$. The basic scheme of this method is obtained by applying the CG method to solve the projection subproblems existing in the AGD approximately. By properly specifying the accuracy for solving these subproblems, we will show that the resulting CGS method can achieve the optimal bounds on the number of calls to the first-order and linear optimization oracles for solving problem (1.1). The development of CGS method, in spirit, is similar to the gradient sliding algorithm developed by Lan in [7] for solving a class of composite optimization problems. However, the gradient sliding algorithm in [7] requires us to perform projection over the feasible set $\mathcal{X}$ and targets to solve convex programming problems with a general nonsmooth term in objective function. The CGS method is formally described in Algorithm 4.

Algorithm 4 The Conditional Gradient Sliding Algorithm

Initial point $x_0 \in \mathcal{X}$ and iteration limit N.

Let $\beta_k \in \mathbb{R}^n_{++}$, $\gamma_k \in [0,1]$, and $\eta_k \in \mathbb{R}_+$, $k = 1, 3 \cdots$, be given and set $y_0 = x_0$.

for $k = 1, \ldots, N$ **do**

$$z_k = (1-\gamma_k) y_{k-1} - \gamma_k x_{k-1} \tag{1.58}$$

$$x_k = \mathrm{CndG}(f'(z_k), x_{k-1}, \beta_k, \eta_k), \tag{1.59}$$

$$y_k = (1-\gamma_k) y_{k-1} + \gamma_k x_k \tag{1.60}$$

end for

procedure $u^+ = \mathrm{CndG}(g, u, \beta, \eta)$

1. Set $u_1 = u$ and $t = 1$.
2. Let v_t be the optimal solution for the subproblem of

$$V_{g,u,\beta}(u_t) := \max_{x \in \mathcal{X}} \langle g + \beta(u_t - u), u_t - x \rangle \tag{1.61}$$

3. If $V_{g,u,\beta}(u_t) \le \eta$, set $u^+ = u_t$ and terminate the procedure.
4. Set $u_{t+1} = (1-\alpha_t) u_t + \alpha_t v_t$, with

$$\alpha_t = \min \left\{ 1, \frac{\langle \beta(u - u_t) - g, v_t - u_t \rangle}{\beta \left\| v_t - u_t \right\|^2} \right\} \tag{1.62}$$

5. Set $t \leftarrow t + 1$ and go to step 2.

end procedure

Clearly, the most crucial step of the CGS method is to update the search point x_k by calling the CndG procedure in (1.59). Denoting $f(x) := \langle g, x \rangle + \beta \left\| x - u \right\|^2 / 2$, the CndG can be viewed as a specialized version of the classical CndG method applied to $\min_{x \in \mathcal{X}} f(x)$. In particular, it can be easily seen that $V_{g,u,\beta}(u_t)$ in (1.61) is equivalent to $\max_{x \in \mathcal{X}} \langle f'(u_t), u_t - x \rangle$, which is often called the Wolfe gap, and the CndG procedure terminates whenever $V_{g,u,\beta}(u_t)$ is smaller than the specified tolerance η. In fact, this procedure is slightly simpler than the generic CndG method in that the

selection of α_t in (1.62) explicitly solves

$$\alpha_t = \arg\min_{\alpha\in[0,1]} f((1-\alpha)u_t + \alpha v_t). \tag{1.63}$$

It should be pointed out that (1.62) was initially suggested by Frank and Wolfe to specify the stepsizes for the CndG method through the minimization of an upper quadratic approximation of $f(\cdot)$ at x_k [5, 3, 2]. In view of above discussion, we can easily see that x_k obtained in (1.59) is an approximate solution for the projection subproblem

$$\min_{x\in\mathcal{X}} \left\{ f(x) \ := \langle f'(z_k), x\rangle + \frac{\beta_k}{2}\left\|x - x_{k-1}\right\|^2 \right\} \tag{1.64}$$

such that

$$\langle f'(x_k), x_k - x\rangle = \langle f'(z_k) + \beta_k(x_k - x_{k-1}), x_k - x\rangle \le \eta_k, \quad \forall x \in \mathcal{X} \tag{1.65}$$

for some $\eta_k \ge 0$.

Clearly, problem (1.64) is equivalent to $\min_{x\in\mathcal{X}} \beta_k/2 \left\|x - x_{k-1} + f'(z_k)/\beta_k\right\|^2$, after completing the square, and it admits explicit solutions in some special cases, e.g., when $\mathcal{X}$ is standard Euclidean ball. However, here the focus is on the case where (1.64) is solved iteratively by calling the linear optimization oracle.

Before analyzing the convergence rate of CGS we add a few comments about this method. First, as an special case of CGS if we limit the number of inner iterations in CndG procedure of CGS we get the Algorithm 5. Note that (1.68) in Algorithm 5 is equivalent to

$$x_k \in \operatorname*{Argmin}_{x\in\mathcal{X}} \langle \nabla f(z_k), x\rangle \tag{1.66}$$

and (1.66) is exactly the subprobem of CG algorithm. Therefore, limiting the number of inner iterations of CGS leads to the CG algorithm.

Algorithm 5 The Conditional Gradient Sliding Algorithm with One Inner Iteration

Initial point $x_0 \in \mathcal{X}$ and iteration limit N.

Let $\beta_k \in \mathbb{R}^n_{++}$, $\gamma_k \in [0,1]$, and $\eta_k \in \mathbb{R}_+$, $k = 1, 3 \cdots$, be given and set $y_0 = x_0$.

for $k = 1, \ldots, N$ **do**

$$z_k = (1-\gamma_k) y_{k-1} - \gamma_k x_{k-1} \tag{1.67}$$

$$x_k \in \operatorname*{Argmax}_{x \in \mathcal{X}} \langle \nabla f(z_k), x_{k-1} - x \rangle, \tag{1.68}$$

$$y_k = (1-\gamma_k) y_{k-1} + \gamma_k x_k \tag{1.69}$$

end for

Output y_N.

Second, similarly to the AGD method, the above CGS method maintains the updating of three intertwined sequences, namely, $\{x_k\}$, $\{y_k\}$, and $\{z_k\}$, in each iteration. The main difference between CGS and the original AGD exists in the computation of x_k. More specifically, x_k in the original AG method is set to the exact solution of (1.64) (i.e., $\eta_k = 0$ in (1.65)), while the subproblem in (1.64) is only solved approximately for the CGS method (i.e., $\eta_k > 0$ in (1.65)).

Third, we say that an inner iteration of the CGS method occurs whenever the index t in the CndG procedure increments by 1. Accordingly, an outer iteration of CGS occurs whenever k increases by 1. While we need to call the first-order oracle to compute the gradient $f'(z_k)$ in each outer iteration, the gradient $f'_k(p_t)$ used in the CndG subroutine is given explicitly by $f'(z_k)+\beta_k(p-x_{k-1})$. Hence, the main cost per each inner iteration of the CGS method is to call the linear optimization oracle to solve the linear optimization problem in (1.61). As a result, the total number of outer and inner iterations performed by the CGS algorithm is equivalent to the total number of calls to the first order and linear optimization oracles, respectively.

Fourth, observe that the above CGS method is conceptual only since we have not yet specified a few parameters, including $\{\beta_k\}$, $\{\gamma_k\}$, and η_k, used in this algorithm. We will come back to this issue after establishing some important convergence properties for the above generic CGS algorithm.

We describe the convergence analysis of CGS method in [8]. For the sake of convergence analysis of CGS we need to mention the following lemma.

Lemma 2. *Let $\{\lambda_i\}$ and $\{a_i\}$ be sequences of nonnegative real numbers. Then for a fixed k,*

1- If the sequence $\{\lambda_i\}$ is a decreasing sequence, then

$$\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) \leq \lambda_0 a_0.$$

2- If the sequence $\{\lambda_i\}$ is an increasing sequence, then

$$\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) \leq \lambda_k \max_{0 \leq t \leq k} a_t.$$

Proof. In order to prove part 1 we have

$$\begin{aligned}
\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) &= -\sum_{i=1}^{k} \lambda_i (a_i - a_{i-1}) \\
&= -\sum_{i=1}^{k} \lambda_i a_i + \sum_{i=1}^{k} (\lambda_i - \lambda_{i-1}) a_{i-1} + \sum_{i=1}^{k} \lambda_{i-1} a_{i-1} \\
&= -\lambda_k a_k - \sum_{i=1}^{k-1} \lambda_i a_i + \sum_{i=1}^{k} (\lambda_i - \lambda_{i-1}) a_{i-1} + \lambda_0 a_0 + \sum_{i=1}^{k-1} \lambda_i a_i \\
&= \lambda_0 a_0 - \lambda_k a_k - \sum_{i=1}^{k} (\lambda_{i-1} - \lambda_i) a_{i-1} \\
&\leq \lambda_0 a_0.
\end{aligned}$$

Where the last inequality holds because $\{\lambda_i\}$ is decreasing.

To prove the second part we have

$$\begin{aligned}
\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) &= -\lambda_k a_l + \lambda_0 a_0 + \sum_{i=1}^{k} (\lambda_i - \lambda_{i-1}) a_{i-1} \\
&\leq \lambda_0 a_0 + \sum_{i=1}^{k} (\lambda_i - \lambda_{i-1}) \max_{0 \leq t \leq k} a_t \\
&= \lambda_0 a_0 + (\lambda_k - \lambda_0) \max_{0 \leq t \leq k} a_t \\
&\leq \lambda_k \max_{0 \leq t \leq k} a_t.
\end{aligned}$$

□

Theorem 5 describes the main convergence properties of the above CGS method. More specifically, Theorem 5(a) and (b) show the convergence of the AG method when the projection subproblem is approximately solved according to (1.65), while Theorem 5(c) states the convergence of the CndG procedure by using the Wolfe gap as the termination criterion. Hence, part (c) is included here mainly for the sake of completeness. It should be noted, however, that the analysis provided in part (c) is more specialized to problem (1.64).

Observe that the following quantity will be used in the convergence analysis of the CGS algorithm:

$$\Gamma_k := \begin{cases} 1, & k = 1, \\ \Gamma_{k-1}(1-\gamma_k), & k \geq 2. \end{cases} \tag{1.70}$$

Theorem 5. *Let Γ_k be defined in* (1.70). *Suppose that $\{\beta_k\}$ and $\{\gamma_k\}$ in the CGS algorithm satisfy*

$$\gamma_1 = 1 \quad \textit{and} \quad L\gamma_k \leq \beta_k, \quad k \geq 1. \tag{1.71}$$

(a) If

$$\frac{\beta_k\gamma_k}{\Gamma_k} \geq \frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}}, \quad k \geq 2, \tag{1.72}$$

then for any $x \in \mathcal{X}$ and $k \geq 1$,

$$f(y_k) - f(x^*) \leq \frac{\beta_k\gamma_k}{2} D_{\mathcal{X}}^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i\gamma_i}{\Gamma_i}, \tag{1.73}$$

where x^* is an arbitrary optimal solution of (1.1) *and $D_{\mathcal{X}}$ is defined in* (1.3).

(b) If

$$\frac{\beta_k\gamma_k}{\Gamma_k} \leq \frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}}, \quad k \geq 2, \tag{1.74}$$

then for any $x \in \mathcal{X}$ and $k \geq 1$,

$$f(y_k) - f(x^*) \leq \frac{\beta_1\Gamma_k}{2} \|x_0 - x^*\|^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i\gamma_i}{\Gamma_i}, \tag{1.75}$$

(c) *Under the assumptions either in part (a) or (b), the number of inner iterations performed at the kth outer iteration can be bounded by*

$$T := \left\lceil \frac{6\beta_k D_{\mathcal{X}}^2}{\eta_k} \right\rceil \quad \forall k \geq 1. \tag{1.76}$$

Proof. To prove part (a) note that by (1.58) and (1.60) we have $y_k - z_k = \gamma_k(x_k - x_{k-1})$. Also, from (1.60) we have

$$\begin{aligned} y_k - z_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_k - z_k \\ &= (y_{k-1} - z_k) + \gamma_k(x_k - y_{k-1}) \\ &= (1-\gamma_k)(y_{k-1} - z_k) + \gamma_k(x_k - z_k) \end{aligned} \tag{1.77}$$

Using this and also (1.10) we have

$$\begin{aligned} f(y_k) &\leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L}{2}\|y_k - z_k\|^2 \\ &= (1-\gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle) \\ &\quad + \gamma_k(f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle) + \frac{L\gamma_k^2}{2}\|x_k - x_{k-1}\|^2 \\ &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k(f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle) + \frac{\beta_k\gamma_k}{2}\|x_k - x_{k-1}\|^2 \end{aligned} \tag{1.78}$$

where the last inequality follows from convexity of f and (1.71). Also note that from the optimality condition (1.65) we have

$$\langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \leq \eta_k \qquad \forall x \in \mathcal{X},$$

and so

$$\langle x_k - x_{k-1}, x_k - x \rangle \leq \frac{\eta_k}{\beta_k} + \frac{1}{\beta_k}\langle f'(z_k), x - x_k \rangle \qquad \forall x \in \mathcal{X}. \tag{1.79}$$

Also, note that

$$\begin{aligned} \frac{1}{2}\|x_{k-1} - x\|^2 &= \frac{1}{2}\|(x_{k-1} - x_k) + (x_k - x)\|^2 \\ &= \frac{1}{2}\|x_k - x_{k-1}\|^2 + \langle x_{k-1} - x_k, x_k - x \rangle + \frac{1}{2}\|x_k - x\|^2, \end{aligned}$$

which implies that

$$\begin{aligned}\frac{1}{2}\left\|x_k - x_{k-1}\right\|^2 &= \frac{1}{2}\left\|x_{k-1} - x\right\|^2 - \langle x_{k-1} - x, x_k - x\rangle - \frac{1}{2}\left\|x_k - x\right\|^2 \\ &\leq \frac{1}{2}\left\|x_{k-1} - x\right\|^2 + \frac{1}{\beta_k}\langle f'(z_k), x - x_k\rangle - \frac{1}{2}\left\|x_k - x\right\|^2 + \frac{\eta_k}{\beta_k}.\end{aligned}$$

Hence,

$$\frac{\beta_k\gamma_k}{2}\left\|x_k - x_{k-1}\right\|^2 \leq \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \gamma_k\langle\nabla f(z_k), x - x_k\rangle + \eta_k\gamma_k. \tag{1.80}$$

Combining (1.78), (1.80) and (1.79) we obtain

$$\begin{aligned}f(y_k) &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k(f(z_k) + \langle\nabla f(z_k), x_k - z_k\rangle + \langle\nabla f(z_k), x - x_k\rangle) \\ &\quad + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k \\ &= (1-\gamma_k)f(y_{k-1}) + \gamma_k(f(z_k) + \langle\nabla f(z_k), x - z_k\rangle) \\ &\quad + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k \\ &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k f(x) + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k,\end{aligned} \tag{1.81}$$

where the last inequality is from convexity of f. Subtracting $f(x)$ from both sides of above inequality gives

$$f(y_k) - f(x) \leq (1-\gamma_k)(f(y_{k-1}) - f(x)) + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k \quad \forall x \in \mathcal{X}.$$

Now using Lemma 1,

$$\begin{aligned}f(y_k) - f(x) &\leq \frac{\Gamma_k(1-\gamma_1)}{\Gamma_1}\left[f(y_0) - f(x)\right] \\ &\quad + \Gamma_k\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{2\Gamma_i}\left(\left\|x_{i-1} - x\right\|^2 - \left\|x_i - x\right\|^2\right) + \Gamma_k\sum_{i=1}^{k}\frac{\eta_i\gamma_i}{\Gamma_i}.\end{aligned} \tag{1.82}$$

Note that $\gamma_1 = 1$ and since from the assumption $\{{}^{\beta_k\gamma_k}/_{\Gamma_k}\}$ is increasing; then from Lemma 2

$$\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{2\Gamma_i}\left(\left\|x_{i-1} - x\right\|^2 - \left\|x_i - x\right\|^2\right) \leq \frac{\beta_k\gamma_k}{\Gamma_k}D_{\mathcal{X}}^2. \tag{1.83}$$

Hence, we have

$$f(y_k) - f(x) \leq \frac{\beta_k \gamma_k}{2} D_{\mathcal{X}}^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i \gamma_i}{\Gamma_i}.$$

which completes the proof of part (a).

To prove part (b), from (1.83) and Lemma 2 the assumption we have

$$\sum_{i=1}^{k} \frac{\beta_i \gamma_i}{2\Gamma_i} \left(\|x_{i-1} - x\|^2 - \|x_i - x\|^2 \right) \leq \beta_1 \|x_0 - x\|^2$$

Therefore, by (1.82) for any $x \in \mathcal{X}$ we have

$$f(y_k) - f(x) \leq \frac{\Gamma_k}{2} \beta_1 \|x_0 - x\|^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i \gamma_i}{\Gamma_i},$$

Which is true for $x = x^*$, and this completes the proof of part (b).

To prove part (c) let us denote $\phi \equiv \phi_k := \langle f'(z_k), x \rangle + \beta_k/2 \|x - x_{k-1}\|^2$ and $\phi^* \equiv \min_{x \in \mathcal{X}} \phi(x)$. Also let us denote

$$\lambda_t := \frac{2}{t} \quad \text{and} \quad \Lambda_t = \frac{2}{t(t-1)}, \tag{1.84}$$

which implies that

$$\Lambda_{t+1} = \Lambda_t (1 - \lambda_{t+1}) \quad \forall t \geq 2. \tag{1.85}$$

Let us define $\bar{u}_{t+1} := (1 - \lambda_{t+1}) u_t + \lambda_{t+1} v_t$. Clearly we have $\bar{u}_{t+1} - u_t + \lambda_{t+1}(v_t - u_t)$. Observe that $u_{t+1} = (1 - \alpha_t) u_t + \alpha_t v_t$ and α_t is an optimal solution to (1.63) and hence $\phi(u_{t+1}) \leq \phi(\bar{u}_{t+1})$.

Using this observation, (1.10), and the fact that ϕ has Lipschitz continuous gradients, we have

$$\begin{aligned}
\phi(u_{t+1}) &\leq \phi(\bar{u}_{t+1}) \\
&\leq \phi(u_t) + \langle \phi'(u_t), \bar{u}_{t+1} - u_t \rangle + \frac{\beta}{2} \|\bar{u}_{t+1} - u_t\|^2 \\
&= \phi(u_t) + \lambda_{t+1} \langle \phi'(u_t), v_t - u_t \rangle + \frac{\beta}{2} \lambda_{t+1}^2 \|v_t - u_t\|^2 \\
&= \phi(u_t) - \lambda_{t+1}\phi(u_t) + \lambda_{t+1} (\phi(u_t) + \langle \phi'(u_t), v_t - u_t \rangle) + \frac{\beta}{2} \lambda_{t+1}^2 \|v_t - u_t\|^2 \\
&\leq (1 - \lambda_{t+1})\phi(u_t) + \lambda_{t+1} (\phi(u_t) + \langle \phi'(u_t), x - u_t \rangle) + \frac{\beta}{2} \lambda_{t+1}^2 \|v_t - u_t\|^2 \\
&\leq (1 - \lambda_{t+1})\phi(u_t) + \lambda_{t+1}\phi(x) + \frac{\beta}{2} \lambda_{t+1}^2 \|v_t - u_t\|^2 .
\end{aligned} \tag{1.86}$$

Subtracting $\phi(x)$ from both sides implies that

$$\phi(u_{t+1}) - \phi(x) \leq (1 - \lambda_{t+1})(\phi(u_t) - \phi(x)) + \frac{\beta}{2} \lambda_{t+1}^2 \|v_t - u_t\|^2 \quad \forall x \in \mathcal{X}.$$

By Lemma 1, for any $x \in \mathcal{X}$ and $t \geq 1$

$$\begin{aligned}
\phi(u_{t+1}) - \phi(x) &\leq \Lambda_{t+1} \left(\frac{1 - \lambda_2}{\Lambda_1} (\phi(1) - \phi(x)) \right) + \sum_{i=2}^{t+1} \frac{\beta \lambda_i^2}{2\Lambda_i} \|v_{i-1} - u_{i-1}\|^2 \\
&= \Lambda_{t+1} \beta \sum_{i=1}^{t} \frac{i}{i+1} \|v_i - u_i\|^2 \\
&\leq \frac{2\beta D_{\mathcal{X}}^2}{t+1}
\end{aligned} \tag{1.87}$$

Now, let the gap function $V_{g,u,\beta}$ be defined in (1.61). Also let us denote $\Delta_j = \phi(u_j) - \phi^*$. It then follow from (1.61), and (1.86) that for any $j = 1, \cdots, t$,

$$\phi(u_{j+1}) \leq \phi(u_j) + \lambda_{j+1} \langle \phi'(u_j), v_j - u_j \rangle + \frac{\beta \lambda_{j+1}^2}{2} \|v_j - u_j\|^2 .$$

Hence,

$$\lambda_{j+1} \langle \phi'(u_j), u_j - v_j \rangle \leq \phi(u_j) - \phi(u_{j+1}) + \frac{\beta \lambda_{j+1}^2}{2} \|v_j - u_j\|^2 ,$$

which implies that

$$\lambda_{j+1} V_{g,u,\beta}(u_j) \leq \phi(u_j) - \phi(u_{j+1}) + \frac{\beta \lambda_{j+1}^2}{2} \|v_j - u_j\|^2$$

$$= \Delta_j - \Delta_{j+1} + \frac{\beta\lambda_{j+1}^2}{2}\|v_j - u_j\|^2.$$

Dividing both sides of above inequality by Λ_{j+1} and summing up the resulting inequalities, we obtain

$$\begin{aligned}
\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}}V_{g,u,\beta}(u_j) &\le \sum_{j=1}^{t}\frac{\Delta_j - \Delta_{j+1}}{\Lambda_{j+1}} + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2 \\
&= -\frac{1}{\Lambda_{t+1}}\Delta_{t+1} + \sum_{j=2}^{t}\left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j}\right)\Delta_j + \Delta_1 + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2 \\
&\le \sum_{j=2}^{t}\left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j}\right)\Delta_j + \Delta_1 + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2 \\
&\le \sum_{j=1}^{t} j\Delta_j + \beta\sum_{j=1}^{t}\frac{j}{j+1}D_{\mathcal{X}}^2 \\
&\le \sum_{j=1}^{t} j\Delta_j + t\beta D_{\mathcal{X}}^2,
\end{aligned}$$

where the last inequality follow from the definition of λ_t and Λ_t in (1.84). Using the above inequality and the bound on Δ_j given in (1.87), we conclude that

$$\min_{j=1,\ldots,t} V_{g,u,\beta}(u_j)\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}} \le \sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}}V_{g,u,\beta}(u_j) \le \sum_{j=1}^{t} j\frac{2\beta D_{\mathcal{X}}^2}{j} + t\beta D_{\mathcal{X}}^2 = 3t\beta D_{\mathcal{X}}^2.$$

Since $\sum_{j=1}^{t}\lambda_{j+1}/\Lambda_{j+1} = t(t+1)/2$, then

$$\min_{j=1,\ldots,t} V_{g,u,\beta}(u_j)\left(\frac{t(t+1)}{2}\right) \le 3t\beta D_{\mathcal{X}}^2,$$

Therefore,

$$\min_{j=1,\ldots,t} V_{g,u,\beta}(u_j) \le \frac{6\beta D_{\mathcal{X}}^2}{t+1},$$

which implies part (c). □

Clearly, there exist various options to specify the parameters $\{\beta_k\}$, $\{\gamma_k\}$, and $\{\eta_k\}$ so as to guarantee the convergence of the CGS method. In the following corollaries, we provide two different parameter settings for $\{\beta_k\}$, $\{\gamma_k\}$, and $\{\eta_k\}$, which lead to optimal complexity bounds on the total number of calls to the first-order and linear optimization oracles for smooth convex optimization.

Corollary 4. *If $\{\beta_k\}$, $\{\gamma_k\}$, and $\{\eta_k\}$ in the CGS method are set to*

$$\beta_k = \frac{3L}{k+1}, \quad \gamma_k = \frac{3}{k+2} \quad \textit{and} \quad \eta_k = \frac{LD_{\mathcal{X}}^2}{k(k+1)}, \qquad \forall k \geq 1, \tag{1.88}$$

then for any $k \geq 1$,

$$f(y_k) - f(x^*) \leq \frac{15LD_{\mathcal{X}}^2}{(k+1)(k+2)}. \tag{1.89}$$

As a consequence, the total number of calls to the first-order and linear optimization oracles performed by the CGS method for finding an ϵ-solution of (1.1) can be bounded by $\mathcal{O}(\sqrt{LD_{\mathcal{X}}^2/\epsilon})$ and $\mathcal{O}(LD_{\mathcal{X}}^2/\epsilon)$, respectively.

Proof. It can be easily seen from (1.88) and (1.71) holds. Also note that by (1.88), we have

$$\Gamma_k = \frac{6}{k(k+1)(k+2)} \tag{1.90}$$

and

$$\frac{\beta\gamma_k}{\Gamma_k} = \frac{9L}{(k+1)(k+2)} \cdot \frac{k(k+1)(k+2)}{6} = \frac{3Lk}{2},$$

which implies that (1.72) is satisfied. It then follows from Theorem 5(a), (1.88), and (1.90) that

$$f(y_k) - f(x^*) \leq \frac{9LD_{\mathcal{X}}^2}{2(k+1)(k+2)} + \frac{6}{k(k+1)(k+2)} \sum_{i=1}^{k} \frac{\eta_i\gamma_i}{\Gamma_i} = \frac{15LD_{\mathcal{X}}^2}{2(k+1)(k+2)},$$

which implies that the total number of outer iterations performed by the CGS method for finding an ϵ-solution can be bounded by $N = \sqrt{15LD_{\mathcal{X}}^2/2\epsilon}$. Moreover, it follows from the bound in (1.76) and (1.88) that the total number of inner iterations can be bounded by

$$\sum_{k=1}^{N} T_k \leq \sum_{k=1}^{N} \left(\frac{6\beta D_{\mathcal{X}}^2}{\eta_k} + 1 \right) = 18 \sum_{k=1}^{N} k + N = 9N^2 + 10N,$$

which implies that the total number of inner iterations is bounded by $\mathcal{O}(LD_{\mathcal{X}}^2/\epsilon)$.

□

Observe that in the above result, the number of calls to the linear optimization oracle is

not improvable in terms of their dependence on ϵ, L and $D_{\mathcal{X}}$ for linear optimization-based convex programming methods [6]. Similarly, the number of calls to the FO oracle is also optimal in terms of its dependence on ϵ and L [10]. It should be noted, however, that we can potentially improve the latter bound in terms of its dependence on $D_{\mathcal{X}}$. Indeed, by using a different parameter setting, we show in Corollary 5 a slightly improved bound on the number of calls to the first-order oracle which only depends on the distance from the initial point to the set of optimal solutions, rather than the diameter $D_{\mathcal{X}}$. This result will play an important role for the analysis of the CGS method for solving strongly convex problems. The disadvantage of using this parameter setting is that we need to fix the number of iterations N in advance.

Corollary 5. *Suppose that there exist an estimate $D_0 \geq \|x_0 - x^*\|$ and that the outer iteration limit $N \geq 1$ is given. If*

$$\beta_k = \frac{2L}{k}, \quad \gamma_k = \frac{2}{k+1}, \quad \eta_k = \frac{2LD_0^2}{Nk} \tag{1.91}$$

for any $k \geq 1$, then

$$f(y_N) - f(x^*) \leq \frac{6LD_0^2}{N(N+1)}. \tag{1.92}$$

As a consequence, the total number of calls to the first-order and linear optimization oracles performed by the CGS method for finding an ϵ-solution of (1.1)*, respectively, can be bound by*

$$\mathcal{O}\left(D_0\sqrt{\frac{L}{\epsilon}}\right) \tag{1.93}$$

and

$$\mathcal{O}\left(\frac{LD_X^2}{\epsilon} + D_0\sqrt{\frac{L}{\epsilon}}\right). \tag{1.94}$$

Proof. It can be easily seen from the definition of k in (1.91) and γ_k in (1.70) that

$$\Gamma_k = \frac{2}{k(k+1)}. \tag{1.95}$$

Using the previous identity and (1.91), we have $\beta_k\gamma_k/\Gamma_k = 2L$, which implies that (1.74) holds. It

then follows from (1.75), (1.91), and (1.95) that

$$f(y_N) - f(x^*) \leq \Gamma_N \left(LD_0^2 + \sum_{i=1}^{N} \frac{\eta_i \gamma_i}{\Gamma_i} \right) = \Gamma_N \left(LD_0^2 + \sum_{i=1}^{N} i\eta_i \right) = \frac{6LD_0}{N(N+1)}.$$

Moreover, it follows from the bound in (1.76) and (1.91) that the total number of inner iterations can be bounded by

$$\sum_{k=1}^{N} T_k \leq \sum_{k=1}^{N} \left(\frac{6\beta_k D_{\mathcal{X}}^2}{\eta_k} + 1 \right) = \frac{6N^2 D_{\mathcal{X}}^2}{D_0^2} + N.$$

The complexity bounds in (1.93) and (1.94) then immediately follow from the previous two inequalities. □

We end this section by summarizing and comparing the convergence result and also requirements of AGD, CG and CGS. As we proved in previous sections, in order to compute an ϵ-solution AGD requires $\mathcal{O}(\sqrt{L^*(D_{\mathcal{X}}^*)^2/\epsilon})$ gradient evaluations where L^* and $D_{\mathcal{X}}^*$ are the true values of Lipschitz constant and the diameter of $\mathcal{X}$, respectively. This number of evaluations is significantly smaller than the $\mathcal{O}(LD^2/\epsilon)$ evaluations of CG. However, AGD requires the solution to the projection subproblem in each iteration of its algorithm which is not always efficiently solvable. This can be a drawback for AGD method. CGS, on the other hand, resolves the requirement of projection calculation in AGD and also the complexity of total number of gradient evaluations in CG. This observation is summarized in Table 1.1. Note that $\Pi_{\mathcal{X}}(\cdot)$ in Table 1.1 is the projection function where (1.29) and (1.20) the subproblem of AGD and GD, respectively, are examples of this function.

	AGD	CG	CGS
Subproblem	$\Pi_{\mathcal{X}}(\cdot)$	$\min_{x\in\mathcal{X}}\langle \cdot, x\rangle$	$\min_{x\in\mathcal{X}}\langle \cdot, x\rangle$
Number of subproblem computations	$\sqrt{LD_{\mathcal{X}}^2/\varepsilon}$	$L^*(D_{\mathcal{X}}^*)^2/\varepsilon$	$L^*(D_{\mathcal{X}}^*)^2/\varepsilon$
Number of gradient evaluations	$\sqrt{LD_{\mathcal{X}}^2/\varepsilon}$	$L^*(D_{\mathcal{X}}^*)^2/\varepsilon$	$\sqrt{L^*(D_{\mathcal{X}}^*)^2/\varepsilon}$

Table 1.1: Comparing the complexity of algorithms AGD, CG and CGS

However, CGS still requires $L^*(D_{\mathcal{X}}^*)^2/\varepsilon$ number of solutions to linear optimization problem that cannot be improved according to [6, 9]. Another drawback of CGS is its requirement to the parameter L^*. This drawback is resolved in *CGS with line search* (CGS-ls) method that we will

propose it in next chapter.

1.5.3 Examples of LP subproblems

As we can see in previous section, projection-free algorithms such as CG and CGS have a linear programming problem (LP) over the feasible set of the main problem that are needed to be solved on each iteration or each inner iteration. It is reasonable to have an analytic and closed form solution for such subproblems instead of directly asking solvers to find the optimal solution or optimal value. In this section we have a few examples of closed form solutions for some LPs over common feasible regions.

1.5.3.1 LP over simplex

In this example we find a closed form solution for the LP over the simplex.

Proposition 3. *The optimal objective to the problem*

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^T x \\ s.t \quad & \sum_{i=1}^{n} x^{(i)} = 1 \\ & x \geq 0 \end{aligned} \tag{1.96}$$

is $\min\{c^{(1)}, \cdots, c^{(n)}\}$*, where* $c^{(i)}$*,* $i = 1, \cdots, n$ *are the elements of cost vector* c*.*

Proof. If we write the dual of the problem (1.96), we have

$$\begin{aligned} D = \max_{y\in\mathbb{R}} \quad & y \\ \text{s.t.} \quad & y \leq c^{(1)} \\ & \vdots \\ & y \leq c^{(n)} \\ & y \text{ free} \end{aligned}$$

The solution to the dual D is clearly $y^* = \min\{c^{(1)}, \cdots, c^{(n)}\}$, and by LP strong duality we have $c^T x = y$. Then the primal optimal solution is given by $x^* = e_{y^*}$, where e_{y^*} is the vector of zeros except for y^*-th element. □

Proposition 3 states that, to find the LP solution over a simplex, we just need to find index k such that $c^{(k)} = \min\{c^{(1)}, \cdots, c^{(n)}\}$ and then the optimal solutions will be $x^* = e_k$. Note that this can also be observed from that fact that all extreme points of a simplex are $e_i,\ i = 1, \cdots, n$.

1.5.3.2 LP over SVM dual constraints

In this example we find a closed form solution for the LP over the feasible set in (1.19). This problem is discussed in Section 1.2.

Proposition 4. *Consider the linear programming problem*

$$\begin{aligned} \min_{u,v \in \mathbb{R}^k} \quad & a^T u + b^T v \\ s.t. \quad & e^T u - e^T v = 0 \\ & 0 \le u^{(i)} \le \sigma \quad i = 1, \cdots, k \\ & 0 \le v^{(i)} \le \sigma \quad i = 1, \cdots, k \end{aligned} \tag{1.97}$$

where $a, b \in \mathbb{R}^k,\ \sigma \in \mathbb{R},\ e$ *is vector of ones in* $\mathbb{R}^k$ *and* $a^{(1)} \le a^{(2)} \le \cdots \le a^{(k)}$ *and* $b^{(1)} \le b^{(2)} \le \cdots \le b^{(k)}$. *Then* (1.97) *has an optimal solution* $w^* = (u^*, v^*)^T \in \mathbb{R}^{2k}$ *so that* $w^* \in \{0, \sigma\}^{2k}$. *Also, if there exist a number,* $0 < m \le k$, *so that* $\sum_{i=1}^{m}(a^{(i)} + b^{(i)}) < 0$ *and*

$$\sum_{i=1}^{m}(a^{(i)} + b^{(i)}) = \min\left\{\sum_{i=1}^{j}(a^{(i)} + b^{(i)}) \ : \ j = 1, \cdots, k\right\},$$

then the optimal solution is $w^* = (u^*, v^*)^T \in \mathbb{R}^{2k}$ *where* u^* *and* v^* *are as* $u^{(i)} = v^{(i)} = \sigma$ *for* $i = 1, \cdots, m$ *and* $u^{(j)} = v^{(j)} = 0$ *for* $j = m+1, \cdots, k$. *If no such* m *exists then* $\mathbf{0} \in \mathbb{R}^{2k}$ *is the optimal solution.*

Proof. Let S be the feasible set of the (1.97). Then S is clearly a polytope and there exists an optimal solution $w^* = (u^*, v^*)^T$ so that $w^* \in \text{ext}\{S\}$. Also, (1.97) has $4k+1$ constraints in which $2k$ of them are active at w^*. Since the equality constraint is already active then $2k-1$ constraints out of $2k$ remaining inequality constraints should be active. This implies that $w^* \in \{0, \sigma\}^{2k-1} \times \mathbb{R}$. Without loss of generality, suppose that $u^* \in \{0, \sigma\}^k$ and $v^* \in \{0, \sigma\}^{k-1} \times \mathbb{R}$. Then there exists a

positive integer value, t, so that

$$\sum_{i=1}^{k}(u^*)^{(i)} = t\sigma.$$

However, in order to stay feasible from the equality constraint of (1.97) we have $\sum_{i=1}^{k}(u^*)^{(i)} = \sum_{i=1}^{k}(v^*)^{(i)}$ or $\sum_{i=1}^{k}(v^*)^{(i)} = t\sigma$. But since $v \in \{0,\sigma\}^{k-1} \times \mathbb{R}$ we must have $v \in \{0,\sigma\}^k$. Therefore, the optimal solution $w^* \in \{0,\sigma\}^{2k}$.

Now let $I = \{i : u^{(i)} = \sigma\}$ and $J = \{j : v^{(j)} = \sigma\}$. Then $|I| = |J| = t$ for some $t \in \mathbb{Z}_+$ and from the objective function we have

$$\begin{aligned} a^T u + b^T v &= \sum_{i=1}^{k} a^{(i)}u^{(i)} + \sum_{j=1}^{k} b^{(j)}v^{(j)} \\ &= \sigma \sum_{i\in I} a^{(i)} + \sigma \sum_{j\in J} b^{(i)} \\ &= \sigma(\sum_{i\in I} a^{(i)} + \sum_{j\in J} b^{(i)}) \\ &\geq \sigma \sum_{i=1}^{t}(a^{(i)} + b^{(i)}). \end{aligned}$$

This implies that if for any $t = 1, \cdots, k$ we have $\sum_{i=1}^{t}(a^{(i)}+b^{(i)}) > 0$ then $t = 0$ or zero is the optimal solution and $w^* = \mathbf{0}$. If otherwise there exist a number $m \in \{1, \cdots, k\}$ so that $\sum_{i=1}^{m}(a^{(i)} + b^{(i)}) < 0$ and

$$\sum_{i=1}^{m}(a^{(i)} + b^{(i)}) = \min\left\{\sum_{i=1}^{j}(a^{(i)} + b^{(i)}) : j = 1, \cdots, k\right\}$$

then $\sigma\sum_{i=1}^{m}(a^{(i)} + b^{(i)})$ is the optimal value. In this case $w^* = (u^*, v^*)^T$ where $(u^*)^{(i)} = (v^*)^{(i)} = \sigma$ for $i = 1, \cdots, m$ and $(u^*)^{(j)} = (v^*)^{(j)} = 0$ for $j = m+1, \cdots, k$

□

Note that the feasible set of (1.97) looks different from that in (1.19). However, if the SVM problem associated to (1.19) is balanced, namely, the samples belonging to the two sets are the same, then the feasible set of (1.97) and (1.19) are the same (with $k = m/2$ and $\sigma = C$).

1.5.3.3 LP over spectrahedron

As we mentioned before, a spectrahedron is the set of all positive semi-definite matrices with trace one. First of all, note that if $A \in \mathbb{R}^{n\times n}$, then the solution to the problem

$$\begin{aligned} \min \quad & x^T A x \\ \text{s.t} \quad & x^T x = 1 \end{aligned} \tag{1.98}$$

is the smallest eigenvalue of A. The reason is that if using spectral theorem we decompose A as $A = U^T \Lambda U$, where U is the orthogonal matrix of eigen vectors of A, and Λ is the diagonal matrix of eigenvalues of A, then we have

$$x^T U^T \Lambda U x = (Ux)^T \lambda (Ux) = y^T \Lambda y.$$

Here $y = Ux$, and since $y^T y = (Ux)^T (Ux) = x^T U^T U x = x^T x = 1$, then the problem (1.98) will change to

$$\begin{aligned} \min \quad & y^T \Lambda y \\ \text{s.t} \quad & y^T y = 1 \end{aligned}$$

and the solution to this problem is the smallest value on the diagonal of Λ which is the smallest eigenvalue of A. Now, let us define $X = xx^T$. Then the objective function in (1.98) can be written as $x^T A x = \text{trace}(Axx^T) = \|AX\|_F^2$. Also, $x^T x = \text{trace}(xx^T) = \text{trace}(X) = 1$, where $X = x^T x$ implies that X has rank 1 and is positive semidefinite. So, as a relaxation of (1.98) to rank one matrices, we have the following problem

$$\begin{aligned} \min \quad & \|AX\|_F^2 \\ \text{s.t} \quad & \text{trace}(X) = 1 \\ & X \succcurlyeq 0. \end{aligned} \tag{1.99}$$

Here we show that (1.98) and (1.99) have the same solution.

Theorem 6. *Let* $S = \{xx^T \ : \ x \in \mathbb{R}^n, \ x^T x = 1\}$, *then*

$$Conv(S) = \{X \ : \ X \in \mathcal{S}^n, \ \text{trace}(X) = 1, \ X \succcurlyeq 0\}.$$

Moreover, if $v \in S$ *then* v *is an extreme point of* $Conv(S)$.

Proof. Let us denote $\bar{S} = \{X \ : \ X \in \mathcal{S}^n, \ \text{trace}(X) = 1, \ X \succcurlyeq 0\}$. First, we show that $Conv(S) \subset \bar{S}$. To show this let $X \in Conv(S)$. Then $X = \sum_{i=1}^n \lambda_i x^{(i)}(x^{(i)})^T$, where $\lambda^{(i)} \geq 0, \ i = 1 \cdots, k$ and $\sum_{i=1}^n \lambda^{(i)} = 1$. Since $\lambda^{(i)} \geq 0$, and $x^{(i)}(x^{(i)})^T$ are rank one positive semidefinite matrices for all $i = 1 \cdots, n$, then X is positive semidefinite too. Also,

$$\text{trace}(X) = \text{trace}(\sum_{i=1}^k \lambda^{(i)} x^{(i)}(x^{(i)})^T) = \sum_{i=1}^k \lambda^{(i)}(x^{(i)})^T x^{(i)} = \sum_{i=1}^k \lambda^{(i)} = 1.$$

This implies that $X \in \bar{S}$ and so $Conv(S) \subset \bar{S}$.

Now let $X \in \bar{S}$. Then X being positive semidefinite implies that using spectral theorem and eigenvalue decomposition, we get

$$X = U^T \Lambda U = \sum_{i=1}^n \lambda^{(i)} v^{(i)}(v^{(i)})^T,$$

where $\lambda^{(i)} \geq 0, \ i = 1, \cdots, n$ are the eigenvalues of X and $v^{(i)}$ is normalized eigen vector corresponding to $\lambda^{(i)}$ for $i = 1, \cdots, n$. Since $\text{trace}(X) = 1$ we have $\sum_{i=1}^n \lambda^{(i)} = 1$. This means that $\sum_{i=1}^n \lambda^{(i)} v^{(i)}(v^{(i)})^T$ is a convex combination of $v^{(i)}(v^{(i)})^T$ where by definition $v^{(i)}(v^{(i)})^T \in S$. Therefore, $X \in Conv(S)$ and then $\bar{S} \subset Conv(S)$.

First part of this theorem clearly implies that any point of set S is an extreme point of $Conv(S)$. Also note that any element of $Conv(S)$ with rank higher than one can be written as a nontrivial convex combination elements in S and so is no longer an extreme point. □

Observe that Theorem 6 show that even though we relaxed he problem (1.98) to (1.99) they will have the same optimal solutions. This means that the solution to (1.99) which is a LP over Spectrahedron is the smallest eigenvalue of matrix A, say λ, and the optimal value is vv^T where v is the eigen vector corresponding to λ.

1.5.3.4 LP over Birkhoff Polytope

The Birkhoff polytope is defined as

$$\mathcal{B} = \{X \in \mathbb{R}^{m \times n} \ : \ X \text{ is doubly stochastic with non-negative elements}\}.$$

In other words, the Birkhoff is the set of all $m \times n$ matrices such that both columns and rows sum to one. The LP over Birkhoff polytope is define as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c^{(ij)} x^{(ij)} \\
\text{s.t.} \quad & \sum_{i=1}^{m} x^{(ij)} = 1 \quad j = 1, \cdots, n \\
& \sum_{j=1}^{n} x^{(ij)} = 1 \quad i = 1, \cdots, m \\
& 0 \leq x^{(ij)} \leq 1 \quad i = 1, \cdots, m, \ j = 1, \cdots, n
\end{aligned}$$

and this is the relaxed LP assignment problem and can be solved with special LP techniques.

1.5.3.5 LP over the set of Hamiltonian cycles on degree n

The LP over the set of Hamiltonian cycles on degree n can be formulated as traveling salesman problem.

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} c^{(ij)} x^{(ij)} \\
\text{s.t.} \quad & \sum_{\substack{i=1 \\ i \neq j}}^{m} x^{(ij)} = 1 \quad j = 1, \cdots, n \\
& \sum_{\substack{j=1 \\ i \neq j}}^{n} x^{(ij)} = 1 \quad i = 1, \cdots, m \\
& u^{(i)} - u^{(j)} + n x_{ij} \leq n - 1 \quad 2 \leq i \neq j \leq n \\
& 0 \leq x^{(ij)} \leq 1 \quad 1 \leq i \leq m, \ 1 \leq j \leq n
\end{aligned}$$

where the last inequality constraint eliminates the subtours which are cycles but not Hamiltonian.

Chapter 2

CGS With Line Search

In Chapter 1 we introduced the CGS algorithm (see Algorithm 4) and also mentioned the complexity and convergence results under different parameter set up. In the k^{th} outer iteration (k^{th} gradient evaluation) of CGS the parameters β_k and η_k depend on other constants such as the Lipschitz constant L, the diameter $D_{\mathcal{X}}$ of the feasible set $\mathcal{X}$ and also the maximum number of outer iteration N. Although in theory these constants do not change the convergence rate of the algorithm with proper setting up of parameters, in practice we might have issue in finding these constants. For example, for large $\mathcal{X}$ finding the the Lipschitz constant L might be expensive in terms of computing and CPU-time consuming. This is a draw back of CGS in practice.

One of the most common methods in optimization is the back tracking line search [1]. Line search involves starting with an estimate of the corresponding constant and continue the iterations of the algorithm while some conditions are satisfied. This guessed value of constant will be increased iteratively if the specific condition is violated.

In this chapter we utilize the line search approach for the Lipschitz constant L. Algorithm 6 is the generic procedure of CGS with line search (CGS-ls). It starts with a first guess of L, that is L_0 in the algorithm and while the condition (2.1) is satisfied the algorithm iterates with the same value of L_k at iteration k; once (2.1) is violated this value will be increased by a multiple of 2 until the condition holds.

Algorithm 6 A CGS-ls algorithm

Initial point $y_0 \in \mathcal{X}$. Set $x_0 = y_0$.

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

Find $L_k > 0$ such that

$$f(y_k) \leq f(z_k) + \langle f'(z_k), y_k - z_k \rangle + \frac{L_k}{2} \|y_k - z_k\|^2 + \frac{\epsilon}{2}\gamma_k \tag{2.1}$$

where

$$\gamma_k = \begin{cases} 1 & k = 1 \\ \text{Positive solution to } \Gamma_k = \Gamma_{k-1}(1-\gamma_k) & k \geq 2 \end{cases} \tag{2.2}$$

where Γ_k depends on L_k and γ_k.

$$z_k = (1-\gamma_k)y_{k-1} - \gamma_k x_{k-1} \tag{2.3}$$

$$x_k \approx \arg\min_{x \in \mathcal{X}} \langle f'(z_k), x \rangle + \frac{\beta_k}{2} \|x - x_{k-1}\|^2 \tag{2.4}$$

$$y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k \tag{2.5}$$

Set

$$\ell_k(x) := \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} \left[f(y_i) + \langle f(z_i), x - z_i \rangle \right]. \tag{2.6}$$

Stop if

$$f(y_k) - \min_{x \in \mathcal{X}} \ell_k(x) \leq \epsilon, \tag{2.7}$$

or equivalently,

$$\max_{x \in \mathcal{X}} f(y_k) - \ell_k(x) \leq \epsilon. \tag{2.8}$$

end for

Output y_N.

Note that the approximate solution to x_k in (2.4), the approximate solution to the projection problem, is given by the same inner iterations as CGS Algorithm 4 inner iterations depending on parameters $\{\beta_k\}$ and $\{\eta_k\}$. Namely, $x_k = \text{CndG}(f'(z_k), x_{k-1}, \beta_k, \eta_k)$. Indeed, if we consider the

optimality condition

$$\langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \leq \eta_k,$$

for $\eta_k = 0$ we have the exact solution to the projection subproblem which leads to the Nesterov's AG method and for $\eta_k > 0$, x_k is solved with accuracy η_k. Also, the L_k in Algorithm 6 can be the Lipschitz constant or any other amount that (2.1) hold. In addition, we can observe that the above method is conceptual only since we have not yet specified the parameters $\{\beta_k\}$, $\{\gamma_k\}$ and η_k that are used in the algorithm. Before setting these parameters we first establish some convergence properties for the Algorithm 6.

We begin the proof of the convergence of the above algorithm by first showing some technical results that will be used in the analysis of the Algorithm 6.

Lemma 3. *For a given k we have*

$$\Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} = 1. \tag{2.9}$$

Proof. Let $1 \leq i \leq k$ then by (2.2) we have

$$\begin{aligned} \frac{1}{\Gamma_i} &= \frac{1-\gamma_i}{\Gamma_i} + \frac{\gamma_i}{\Gamma_i} \\ &= \frac{1}{\Gamma_{i-1}} + \frac{\gamma_i}{\Gamma_i}. \end{aligned}$$

Now summing up both sides of above inequality for $i = 2, \cdots, k$ we get

$$\sum_{i=2}^{k} \frac{1}{\Gamma_i} - \frac{1}{\Gamma_{i-1}} = \sum_{i=2}^{k} \frac{\gamma_i}{\Gamma_i}.$$

Hence,

$$\frac{1}{\Gamma_k} - 1 = \sum_{i=2}^{k} \frac{\gamma_i}{\Gamma_i}.$$

This and also the fact that $\gamma_1 = 1$ imply the (2.9).

□

Lemma 4. *If* (2.7) *or* (2.8) *hold, then*

$$f(y_k) - f^* \leq \epsilon.$$

Proof. From the definition of $\ell_k(x)$ in (2.6), Lemma 3 and also the convexity of f for any $x \in \mathcal{X}$ we have

$$\begin{aligned}
\ell_k(x) &= \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} \left[f(y_i) + \langle f(z_i), x - z_i \rangle \right] \\
&\leq \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} f(x) \\
&= f(x).
\end{aligned}$$

This implies that $\min_{x \in \mathcal{X}} \ell_k(x) \leq f^*$ and therefore,

$$f(y_k) - f^* \leq f(y_k) - \min_{x \in \mathcal{X}} \ell_k(x) \leq \epsilon.$$

□

Theorem 7 below describes the main convergence property of the Algorithm 6.

Theorem 7. *In Algorithm 6 if* $\beta_k \geq L_k \gamma_k$*, then*

$$\begin{aligned}
f(y_k) - \ell_k(x) \leq \frac{\epsilon}{2} &+ \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \beta_i}{2\Gamma_i} \left(\|x - x_{i-1}\|^2 - \|x - x)i\|^2 \right) \\
&+ \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \eta_i}{\Gamma_i} - \sum_{i=1}^{k} \frac{\gamma_i}{2\Gamma_i} \left(\beta_i - L_i \gamma_i \right) \|x_i - x_{i-1}\|^2 .
\end{aligned}$$

Proof. We have

$$\begin{aligned}
\frac{\ell_k(x)}{\Gamma_k} &\overset{(2.6)}{=} \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} \left[f(z_i) + \langle \nabla f(z_i), x - x_i \rangle + \langle \nabla f(z_i), x_i - z_i \rangle \right] \\
&= \sum_{i=1}^{k} \frac{1}{\Gamma_i} \left[\gamma_i f(z_i) + \gamma_i \langle \nabla f(z_i), x - x_i \rangle + \langle \nabla f(z_i), \gamma_i (x_i - z_i) \rangle \right].
\end{aligned}$$

Here

$$\begin{aligned}\gamma_i(x_i - z_i) &\overset{(2.5)}{=} y_i - (1-\gamma_i)y_{i-1} - \gamma_i z_i \\ &= (y_i - z_i) - (1-\gamma_i)(y_{i-1} - z_i).\end{aligned}$$

So,

$$\begin{aligned}\frac{\ell_k(x)}{\Gamma_k} = \sum_{i=1}^{k} \frac{1}{\Gamma_i}[&\gamma_i f(z_i) + \langle \nabla f(z_i), y_i - z_i\rangle \\ &- (1-\gamma_i)\left(f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i\rangle\right) + \gamma_i \langle \nabla f(z_i), x - x_i\rangle].\end{aligned}$$

Note that from (2.3) and (2.5) we have $y_i - z_i = \gamma_i(x_i - x_{i-1})$ and

$$f(z_i) + \langle \nabla f(z_i), y_i - z_i\rangle \geq f(y_i) - \frac{L_i}{2}\|y_i - z_i\|^2 - \frac{\epsilon}{2}\gamma_i.$$

Also, from convexity of f we have

$$-\left(f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i\rangle\right) \geq -f(y_{i-1}),$$

and from (2.1) we have

$$\gamma_i \langle \nabla f(z_i), x - x_i\rangle \geq -\frac{\gamma_i\beta_i}{2}\left(\|x - x_{i-1}\|^2 - \|x_i - x_{i-1}\|^2 - \|x - x_i\|^2\right) - \gamma_i\eta_i.$$

Therefore,

$$\begin{aligned}\frac{\ell_k(x)}{\Gamma_k} &\geq \sum_{i=1}^{k} \frac{1}{\Gamma_i}\left[f(y_i) - \frac{L_i\gamma_i^2}{2}\|x_i - x_{i-1}\|^2 - \frac{\epsilon}{2}\gamma_i - (1-\gamma_i)f(y_{i-1})\right. \\ &\qquad \left. - \frac{\gamma_i\beta_i}{2}\left(\|x - x_{i-1}\|^2 - \|x_i - x_{i-1}\|^2 - \|x - x_i\|^2\right) - \gamma_i\eta_i\right] \\ &= \sum_{i=1}^{k} \frac{1}{\Gamma_i} f(y_i) - \frac{1-\gamma_i}{\Gamma_i} f(y_{i-1}) - \sum_{i=1}^{k} \frac{\gamma_i\beta_i}{2\Gamma_i}\left(\|x - x_{i-1}\|^2 - \|x - x_i\|^2\right) \\ &\quad - \frac{\epsilon}{2}\sum_{i=1}^{k}\frac{\gamma_i}{\Gamma_i} - \sum_{i=1}^{k}\frac{\gamma_i}{2\Gamma_i}(L_i\gamma_i - \beta_i)\|x_i - x_{i-1}\|^2 - \sum_{i=1}^{k}\frac{\gamma_i\eta_i}{\Gamma_i}.\end{aligned}$$

Noting that

$$\sum_{i=1}^{k} \frac{1}{\Gamma_i} f(y_i) - \frac{1-\gamma_i}{\Gamma_i} f(y_{i-1}) \overset{(2.2)}{=} \frac{f(y_k)}{\Gamma_k},$$

and also using Lemma 3 we have

$$\begin{aligned}\frac{\ell_k(x)}{\Gamma_k} \geq \frac{f(y_k)}{\Gamma_k} - \sum_{i=1}^{k} \frac{\gamma\beta_i}{2\Gamma_i} \left(\|x - x_{i-1}\|^2 - \|x - x_i\|^2 \right) \\ - \frac{\epsilon}{2\Gamma_k} - \sum_{i=1}^{k} \frac{\gamma_i}{2\Gamma_i} (L_i\gamma_i - \beta_i) \|x_i - x_{i-1}\|^2 - \sum_{i=1}^{k} \frac{\gamma_i\eta_i}{\Gamma_i},\end{aligned}$$

which implies that

$$\begin{aligned}f(y_k) - \ell_k(x) \leq \frac{\epsilon}{2} + \Gamma_k \sum_{\gamma_i\beta_i}^{2\Gamma_i} \left(\|x - x_{i-1}\|^2 - \|x - x)i\|^2 \right) \\ + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i\eta_i}{\Gamma_i} - \sum_{i=1}^{k} \frac{\gamma_i}{2\Gamma_i} (\beta_i - L_i\gamma_i) \|x_i - x_{i-1}\|^2 .\end{aligned}$$

□

As we can see in Theorem 7 and the Algorithm 6 the parameters $\{\beta_k\}, \{\gamma_k\}$ and $\{\eta_k\}$ are needed to be specified in a proper way to have the desired convergence result. Clearly, there are many options to choose these parameters to guarantee the convergence of CGS-ls. In our next step we and in corollaries 6 and 7 we provide two different parameter settings for $\{\beta_k\}, \{\gamma_k\}$ and $\{\eta_k\}$ which lead to optimal complexity bounds on total number of gradient evaluations and also the total number of calls to the linear optimizations oracle.

Corollary 6. *If we set*

$$\beta_k = \frac{2L_k}{k}, \quad \gamma_k = \frac{2}{k+1}, \quad \Gamma_k = \frac{2}{k(k+1)}, \quad \eta_k = \frac{2L_k D_{\mathcal{X}}^2}{Nk},$$

where N is the max number of iteration and is bounded by

$$N \geq \sqrt{\frac{2MD_{\mathcal{X}}^2}{\epsilon}},$$

where M is the true value of Lipschitz constant, then for all $x \in \mathcal{X}$

$$f(y_k) - \ell_k(x) \leq \frac{\epsilon}{2} + 3\Gamma_k M D_{\mathcal{X}}^2.$$

In particular, the total number of $\nabla f(\cdot)$ evaluations and linear objective optimization computations are bounded by

$$\sqrt{\frac{12MD_{\mathcal{X}}^2}{\epsilon}} \quad \textit{and} \quad \left(\frac{72MD_{\mathcal{X}}^2}{\epsilon} + \sqrt{\frac{12MD_{\mathcal{X}}^2}{\epsilon}}\right),$$

respectively.

Proof. Applying the parameter setting to Theorem 7 we have

$$\begin{aligned} f(y_k) - \ell_k(x) &\leq \frac{\epsilon}{2} + \Gamma_k \left(MD_{\mathcal{X}}^2 + 2MD_{\mathcal{X}}^2\right) \\ &= \frac{\epsilon}{2} + 3\Gamma_k M D_{\mathcal{X}}^2. \end{aligned}$$

In particular, (2.7) is satisfied if

$$3\Gamma_k M D_{\mathcal{X}}^2 \leq \frac{\epsilon}{2},$$

i.e.

$$\frac{6MD_{\mathcal{X}}^2}{k(k+1)} \leq \frac{\epsilon}{2}, \quad \text{or} \quad k \geq \sqrt{\frac{12MD_{\mathcal{X}}^2}{\epsilon}}.$$

Now by Proposition 5 part (c),

$$\begin{aligned} \sum_{k=1}^{N} T_k &\leq \sum_{k=1}^{N} \frac{6\beta_k D_{\mathcal{X}}^2}{\eta_k} + 1 \\ &= 6N^2 + N \\ &\leq \frac{72MD_{\mathcal{X}}^2}{\epsilon} + \sqrt{\frac{12MD_{\mathcal{X}}^2}{\epsilon}}. \end{aligned}$$

□

Corollary 7. *As we set of the parameters in the Algorithm 6 as*

$$\beta_k = L_k\gamma_k, \quad \Gamma_k = L_k\gamma_k^2 \quad \textit{and} \quad \eta_k = \frac{L_k\gamma_k D_{\mathcal{X}}^2}{N}, \tag{2.10}$$

where N is the number of outer iterations, we have,

$$f(y_k) - f^* \leq \frac{12L}{(k-1)^2} D_{\mathcal{X}}^2. \tag{2.11}$$

In other words, the number of gradient evaluations in Algorithm 6 is bounded by

$$\mathcal{O}\left(\sqrt{\frac{LD_{\mathcal{X}}^2}{\epsilon}}\right). \tag{2.12}$$

In addition, the number of LP evaluations in the Algorithm 6 is bounded by

$$\mathcal{O}\left(\frac{LD_{\mathcal{X}}^2}{\epsilon}\right). \tag{2.13}$$

Proof. Clearly, with parameter set up in (2.10) all conditions (1.71) of Theorem 5 are satisfied and from part (a) of this theorem with K the total number of iterations we have

$$\begin{aligned} f(y_k) - f^* &\leq \frac{\beta_k\gamma_k}{2} D_{\mathcal{X}}^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i\gamma_i}{\Gamma_i} \\ &\leq \frac{L_k\gamma_k^2 D_{\mathcal{X}}^2}{2} + L_k\gamma_k^2 \sum_{i=1}^{k} \frac{L_i\gamma_i^2 D_{\mathcal{X}}^2}{KL_i\gamma_i^2} \\ &= \frac{1}{2} L_k\gamma_k^2 D_{\mathcal{X}}^2 + l_k\gamma_k^2 D_{\mathcal{X}}^2 \\ &= \frac{3}{2}\Gamma_k D_{\mathcal{X}}^2. \end{aligned} \tag{2.14}$$

But for $k > 1$ we have $\Gamma_k = \Gamma_{k-1}(1 - \gamma_k)$. This implies that

$$\frac{1}{\Gamma_{k-1}} = \frac{1}{\Gamma_k} - \frac{\gamma_k}{\Gamma_k} \qquad \forall k > 1.$$

Therefore,

$$\sqrt{\frac{1}{\Gamma_k}} - \sqrt{\frac{1}{\Gamma_{k-1}}} = \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}}}{\sqrt{\frac{1}{\Gamma_k}} + \sqrt{\frac{1}{\Gamma_{k-1}}}} = \frac{\frac{\gamma_k}{\Gamma_k}}{\sqrt{\frac{1}{\Gamma_k}} + \sqrt{\frac{1}{\Gamma_{k-1}}}}.$$

Note that $\Gamma_k = (1-\gamma_k)\Gamma_{k-1} \leq \Gamma_{k-1}$ and $\Gamma_k = L_k\gamma_k^2$. Hence,

$$\sqrt{\frac{1}{\Gamma_k}} - \sqrt{\frac{1}{\Gamma_{k-1}}} \geq \frac{\frac{\gamma_k}{\Gamma_k}}{\sqrt{\frac{1}{\Gamma_k}} + \sqrt{\frac{1}{\Gamma_k}}} = \frac{\gamma_k}{2\sqrt{\Gamma_k}} = \frac{1}{2\sqrt{L_k}} \geq \frac{1}{2\sqrt{2}L}.$$

So performing inductively we get

$$\sqrt{\frac{1}{\Gamma_k}} - \sqrt{\frac{1}{\Gamma_{k-1}}} \geq \frac{k-1}{2\sqrt{2}L},$$

then

$$\sqrt{\frac{1}{\Gamma_k}} \geq \frac{k-1}{2\sqrt{2}L},$$

which implies that

$$\Gamma_k \leq \frac{8L}{(k-1)^2}.$$

Therefore, from (2.14) we have

$$\begin{aligned} f(y_k) - f^* &\leq \frac{3}{2} \cdot \frac{8L}{(k-1)^2} \cdot D_{\mathcal{X}}^2 \\ &= \frac{12L}{(k-1)^2} D_{\mathcal{X}}^2. \end{aligned} \tag{2.15}$$

Also, from part (c) of Theorem 5 and definition of η_k we have

$$\begin{aligned} \sum_{k=1}^{N} T_k &= \sum_{k=1}^{N} \frac{6\beta_k D_{\mathcal{X}}^2}{\eta_k} \\ &= \sum_{k=1}^{N} \frac{6NL_k\gamma_k D_{\mathcal{X}}^2}{L_k\gamma_k D_{\mathcal{X}}^2} \\ &= 6N^2. \end{aligned} \tag{2.16}$$

Therefore from (2.15) and (2.16) w get that the total number of iterations it takes for the CGS algorithm with line search to get an ϵ-certificate is bounded by $\mathcal{O}\left(\sqrt{LD_{\mathcal{X}}^2/\epsilon}\right)$ and the number of linear programming evaluations is bounded by $\mathcal{O}\left(LD_{\mathcal{X}}^2/\epsilon\right)$. □

An observation that we can have from corollaries 6 and 7 is that in both type of settings

of parameter we require the knowledge of N, the maximum number of outer iterations or the total number of gradient evaluations. However, we can improve the bound on number of calls to linear optimization oracle or the number of inner iterations in terms of its dependence on N. In corollary 8 we have a new setting for $\{\beta_k\}, \{\gamma_k\}$ and $\{\eta_k\}$ that lead to our desired improvement.

Corollary 8. *Let*

$$\beta_k = L_k\gamma_k, \qquad \Gamma_k = L_k\gamma_k^3, \qquad \textit{and} \qquad \eta_k = \frac{cL_k\gamma_k D_{\mathcal{X}}^2}{k}$$

for constant $c > 0$ and $L_1 = tM$, where $t \in (0, 1)$. Then (2.7) is satisfied when

$$k \geq \frac{17D_{\mathcal{X}}}{\sqrt[6]{t}}\sqrt{\frac{(c+1)M}{\epsilon}}.$$

Proof. Since $\Gamma_k = L_k\gamma_k^3 = (1-\gamma_k)\gamma_{k-1}$ and $L_1 \leq L_k \leq 2M$ we have

$$\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}} = \frac{\gamma_k}{\Gamma_k}.$$

Then

$$\begin{aligned}
\sqrt[3]{\frac{1}{\Gamma_k}} - \sqrt[3]{\frac{1}{\Gamma_{k-1}}} &= \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}}}{\sqrt[3]{\frac{1}{\Gamma_k^2}} + \sqrt[3]{\frac{1}{\Gamma_k\Gamma_{k-1}}} + \sqrt[3]{\frac{1}{\Gamma_{k-1}^2}}} \\
&\geq \frac{\frac{\gamma_k}{\Gamma_k}}{3\sqrt[3]{\frac{1}{\Gamma_k^2}}} \\
&= \frac{\gamma_k}{3\sqrt[3]{\Gamma_k}} \\
&\geq \frac{\gamma_k}{3\sqrt[3]{L_k\gamma_k^3}} \\
&\geq \frac{1}{3\sqrt[3]{2M}}
\end{aligned}$$

Summing up both sides of above inequality we obtain

$$\frac{1}{\sqrt[3]{\Gamma_k}} - \frac{1}{\sqrt[3]{\Gamma_1}} \geq \frac{k-1}{3\sqrt[3]{2M}},$$

which implies that

$$\Gamma_k \leq \frac{54M}{(k-1)^3}. \tag{2.17}$$

In addition, since $\Gamma_k = L_k \gamma_k^3$ we have

$$\frac{1}{L_k} = \frac{\gamma_k^3}{\Gamma_k},$$

and since $L_1 \leq L_k \leq 2M$ we have

$$\frac{1}{2M} \leq \frac{1}{L_k} \leq \frac{1}{L_1}.$$

Combining these two inequalities give that

$$\frac{1}{\sqrt[3]{2M}} \leq \frac{\gamma_k}{\sqrt[3]{\Gamma_k}} \leq \frac{1}{\sqrt[3]{L_1}}.$$

So,

$$\begin{aligned}
\frac{1}{\sqrt[3]{\Gamma_k}} - \frac{1}{\sqrt[3]{\Gamma_{k-1}}} &= \frac{\frac{\gamma_k}{\Gamma_k}}{\sqrt[3]{\frac{1}{\Gamma_k^2}} + \sqrt[3]{\frac{1}{\Gamma_k \Gamma_{k-1}}} + \sqrt[3]{\frac{1}{\Gamma_{k-1}^2}}} \\
&= \frac{\sqrt[3]{\frac{1}{\Gamma_k^2}} \frac{\gamma_k}{\sqrt[3]{\Gamma_k}}}{\sqrt[3]{\frac{1}{\Gamma_k^2}} + \sqrt[3]{\frac{1}{\Gamma_k \Gamma_{k-1}}} + \sqrt[3]{\frac{1}{\Gamma_{k-1}^2}}} \\
&\leq \frac{\gamma_k}{\sqrt[3]{\Gamma_k}} \\
&\leq \frac{1}{\sqrt[3]{L_1}}.
\end{aligned}$$

We have $L_1 = \Gamma_1$. Summing up above inequality we get

$$\frac{1}{\sqrt[3]{\Gamma_k}} \leq \frac{k-1}{\sqrt[3]{L_1}} + \frac{1}{\sqrt[3]{L_1}} = \frac{k}{\sqrt[3]{L_1}}$$

Therefore,

$$\Gamma_k \geq \frac{L_1}{k^3}$$

or

$$L_k \gamma_k^3 \geq \frac{L_1}{K^3}$$

which implies that

$$\frac{1}{\gamma_k^3} \leq \frac{L_k k^3}{L_1}$$

or

$$\frac{1}{\gamma_k} \leq k \sqrt[3]{\frac{2M}{L_1}}. \tag{2.18}$$

Hence, using (2.17), (2.18) and Theorem 7 we have

$$\begin{aligned}
f(y_k) - \ell_k(x) &\leq \frac{\epsilon}{2} + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \beta_i}{2\Gamma_i} \left(\|x - x_{i-1}\|^2 - \|x - x)i\|^2 \right) \\
&\quad + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \eta_i}{\Gamma_i} - \sum_{i=1}^{k} \frac{\gamma_i}{2\Gamma_i} \left(\beta_i - L_i \gamma_i\right) \|x_i - x_{i-1}\|^2 \\
&= \frac{\epsilon}{2} + \Gamma_k \left(\sum_{i=1}^{k} \frac{\gamma_i \beta_i}{2\Gamma_i} \left(\|x - x_{i-1}\|^2 - \|x - x)i\|^2 \right) + \sum_{i=1}^{k} \frac{\gamma_i \eta_i}{\Gamma_i} \right) \\
&\leq \frac{\epsilon}{2} + \Gamma_k \left(\frac{D_{\mathcal{X}}^2}{\gamma_k} + c \sum_{i=1}^{k} \frac{D_{\mathcal{X}}^2}{i \gamma_i} \right) \\
&\leq \frac{\epsilon}{2} + D_{\mathcal{X}}^2 \Gamma_k \left(\frac{1}{\gamma_k} + c \sum_{i=1}^{k} \sqrt[3]{\frac{2M}{L_1}} \right) \\
&= \frac{\epsilon}{2} + D_{\mathcal{X}}^2 \left(\frac{\Gamma_k}{\gamma_k} + ck \sqrt[3]{\frac{2M}{L_1}} \Gamma_k \right) \\
&\leq \frac{\epsilon}{2} + D_{\mathcal{X}}^2 \left(\frac{54M}{(k-1)^3} k \sqrt[3]{\frac{2M}{L_1}} + ck \sqrt[3]{\frac{2M}{L_1}} \frac{54M}{(k-1)^3} \right) \\
&= \frac{\epsilon}{2} + D_{\mathcal{X}}^2 \frac{(c+1)108Mk \sqrt[3]{\frac{2M}{L_1}}}{(k-1)^3}
\end{aligned}$$

So, to have

$$\frac{(c+1)108 D_{\mathcal{X}}^2 M k \sqrt[3]{\frac{2M}{L_1}}}{(k-1)^3} \leq \frac{\epsilon}{2} \tag{2.19}$$

we need

$$\frac{1}{k^2} \leq \frac{k}{(k-1)^3} \leq \frac{\epsilon}{2} \frac{\sqrt[3]{\frac{L_1}{2M}}}{108 D_{\mathcal{X}}^2 M} = \frac{\epsilon \sqrt[3]{L_1}}{(c+1)216 D_{\mathcal{X}}^2 \sqrt[3]{2M^4}}$$

or

$$k \geq \sqrt{\frac{(c+1)216 D_{\mathcal{X}}^2 \sqrt[3]{2M^4}}{\epsilon \sqrt[3]{L_1}}} = \frac{D_X \sqrt{216 \sqrt[3]{2}} \sqrt[3]{M^2} \sqrt{c+1}}{\sqrt{\epsilon} \sqrt[6]{L_1}}. \tag{2.20}$$

Therefore, if $L_1 = cM$ for some $c \in (0, 1)$ and

$$k \geq \frac{17 D_{\mathcal{X}}}{\sqrt[6]{c}} \sqrt{\frac{(c+1)M}{\epsilon}}.$$

then (2.20) and therefore (2.19) will also hold. □

Note that even when L_1 is significantly smaller than the true Lipschitz constant M, in order to get an epsilon solution, the increase in number of iterations is not significant. For example, for $t = 0.01$ the number of outer iterations is approximately bounded by

$$17\sqrt{c+1} D_{\mathcal{X}} \sqrt{\frac{5M}{\epsilon}}.$$

We finish this chapter by mentioning the fact that the constant c in Corollary 8 needs to be tuned for best practical performance.

Chapter 3

Experimental results

In this chapter we present the experimental results showing the performance of CGS-ls compared to CG and CGS. To this goal we are interested in approximately solving the dual problem of the SVM. Recall that, the dual of the soft-margin SVM is a quadratic programming problem of the form

$$f^* := -\min_{x\in\mathcal{X}} f(x) := \frac{1}{2}\sum_{j=1}^{m}\sum_{i=1}^{n} x_i x_j b_i b_j \langle X_i, X_j\rangle - \sum_{i=1}^{n} x_i \tag{3.1}$$

where

$$\mathcal{X} = \left\{ x \in \mathbb{R}^n \ : \ \sum_{i=1}^{n} x_i b_i = 0,\ 0 \le x_i \le C,\ b_j, C \in \mathbb{R},\ i = 1,\dots,n,\ j = 1,\dots,m \right\}. \tag{3.2}$$

Here $C \in \mathbb{R}$ is the regularization parameter and must be chosen properly.

As we mentioned in previous chapters the algorithms CG, CGS, and CGS-ls derive linear approximate solutions to the quadratic programming problem (3.1) iteratively until an ϵ-solution is provided. The linear approximations that appear in these algorithms as subproblem is the following LP:

$$\begin{aligned} \max_{x\in\mathbb{R}^n} \quad & -\sum_{j=1}^{m}\sum_{i=1}^{n} b_i b_j \langle X_i, X_j\rangle x_j + \mathbf{e} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i b_i = 0 \\ & 0 \le x_i \le C \qquad i = 1, \ldots, n. \end{aligned} \tag{3.3}$$

This problem is equivalent to

$$\begin{aligned} -\min_{x\in\mathbb{R}^n} \quad & \sum_{j=1}^{m}\sum_{i=1}^{n} b_i b_j \langle X_i, X_j\rangle x_j - \mathbf{e} \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_i b_i = 0 \\ & 0 \le x_i \le C \qquad i = 1, \ldots, n. \end{aligned} \tag{3.4}$$

For balanced SVM (the two classes in the training dataset have the same number of samples), the above LP has explicit solution (see our derivation in Section 1.5.3.2).

In next two sections we experiment the SVM classification on two different data sets. The first data set is the set of uniformly chosen random points in some subsets of $\mathbb{R}^2$, and the second is the MNIST hand-written digits data set. The goal is to examine and compare the performance of algorithms, in terms of both objective function value and accuracy. The accuracy is basically the percentage of test data points that are classified correctly.

3.1 Binary classification of 2D data set

In this section to have a better intuition of the classification we try to classify two types of subsets in $\mathbb{R}^2$ to be able to visualize the line that separates the two data set. The first type of subsets of $\mathbb{R}^2$ is the first and third orthants and the second type is two unit balls.

3.1.1 Boxes in first and third Orthants

As an example we consider two simple sets of uniformly chosen random points in boxes of length 10 in first and third orthants of $\mathbb{R}^2$ as training data sets in which there are 500 random points

in each box and 1000 points in total. While the data sets are relatively small the three algorithms CG, CGS, and CGS-ls have similar performance and after a few iterations they have an accuracy more than 98 percent in classifying the data sets. In Figures 3.1, 3.2, 3.3, 3.4 and 3.5 we can see the results of classification and the accuracy over iterations using these three algorithms. Note that the parameter $C = 1$ is used for these data sets as the regularization parameter in (3.2).

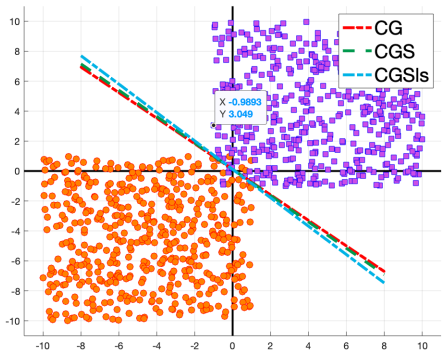

Figure 3.1: Classifying 2D data sets in two orthants

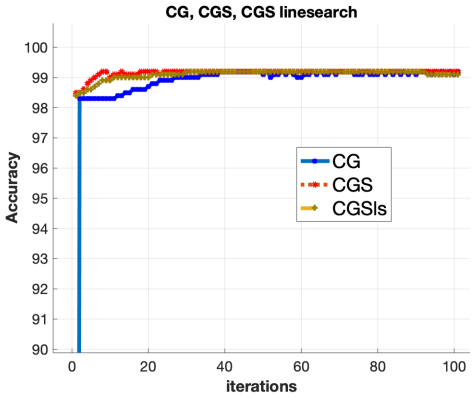

Figure 3.2: accuracy of algorithms classifying data sets in two orthants

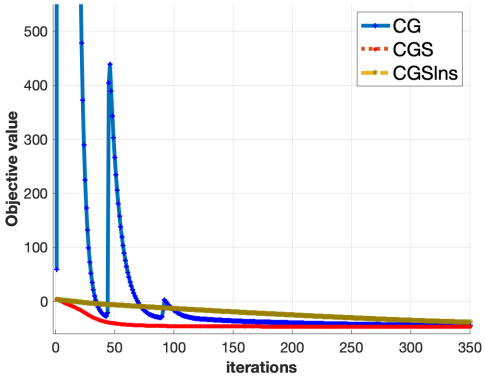

Figure 3.3: iteration versus objective value in classifying data sets in two orthants

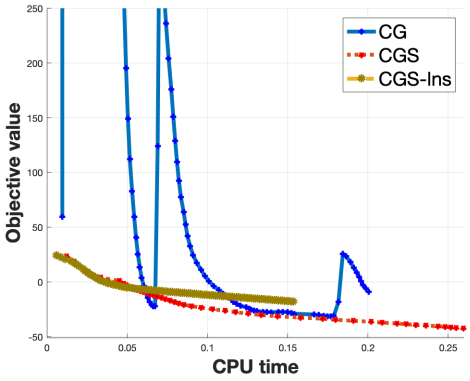

Figure 3.4: CPU-time versus objective value in classifying data sets in two orthants

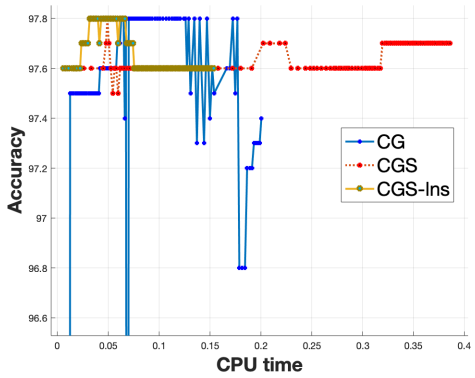

Figure 3.5: CPU-time versus accuracy in classifying data sets in two orthants

3.1.2 Two unit balls

As another example we consider two sets of uniformly chosen random points in two 1-balls of radius one with centers $(0, 0)$ and $(1, 1)$ for the training data sets in which there are 500 points in each disc and 1000 random points in total. In a same way but with different seed the two test set with 1000 random points are chosen. After a very small number of iteration all three algorithms CG, CGS, and CGS-ls give a classification with more than 89 percent accuracy. Although this accuracy is 10 percent lower than the accuracy we get in binary classification of two boxes, but we should consider that higher intersection of the two discs in comparison with the two boxes. Note that the parameter $C = 0.01$ is used for these data sets as regularization parameter. In figures 3.6, 3.7, 3.8, 3.9 and 3.10 we can see the results of classification and the accuracy over iterations using these three algorithms.

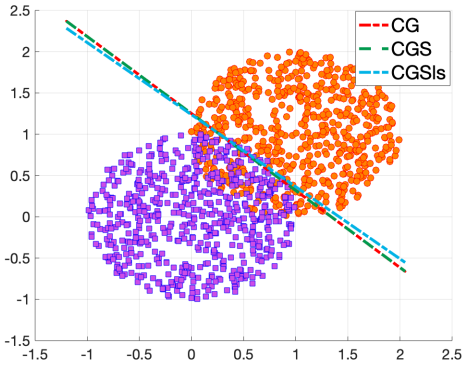

Figure 3.6: Classifying 2D data sets

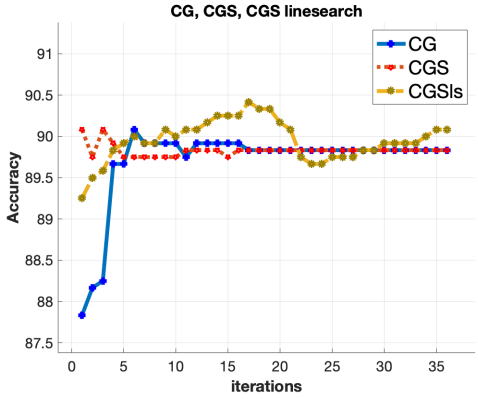

Figure 3.7: accuracy of algorithms

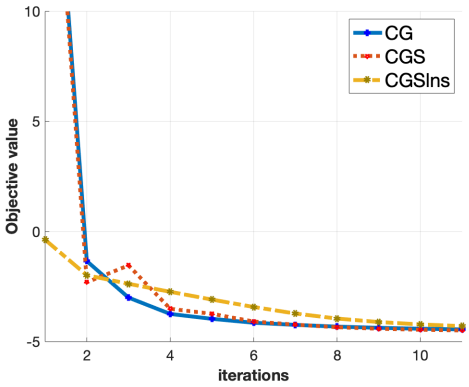

Figure 3.8: iteration versus objective value in classifying data sets in two unit balls

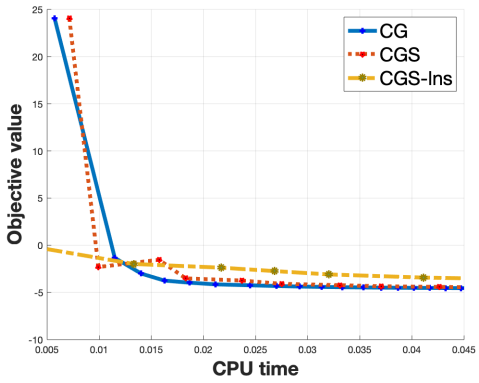

Figure 3.9: CPU-time versus objective value in classifying data sets in two unit balls

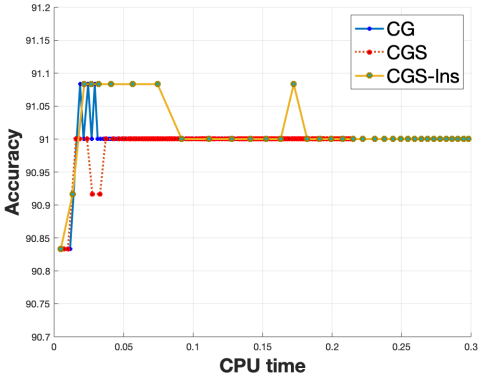

Figure 3.10: CPU-time versus accuracy in classifying data sets in two unit balls

3.2 Binary classification of MNIST hand-written digits

The data that we use here is from the MNIST database that is a large database of hand-written digits 0 through 9. The image of these digits are vectorized and the representation of each single image is a row vector of length 784. Each digit has different number of sample images and row vectors corresponding to images are stacked on each other creating a matrix with fixed number of columns and various number of rows. The whole database contains 60,000 training images and 10,000 testing images.

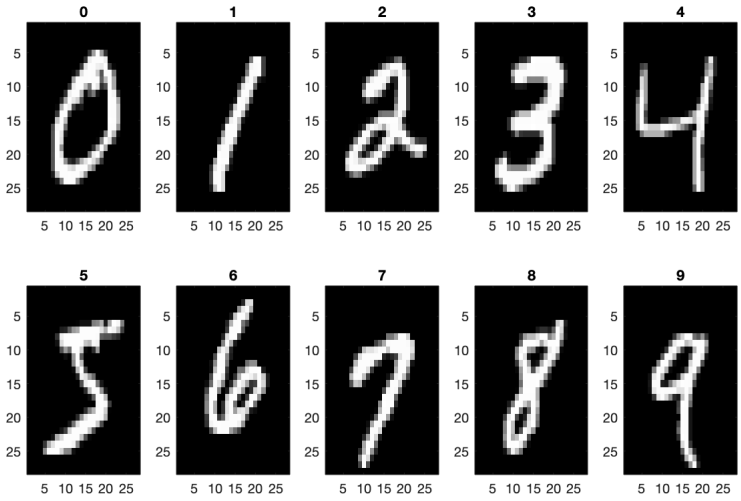

Figure 3.11: a sample of converted MNIST data set to images.

In this work we specifically separate the train set *"train3"* of size 6131×484 from train set *"train8"* of size 5851×784. Here *"train3"* and *"train8"* are hand-written digits of 3 and 8, respectively. The objective matrix of training data set is then of size 11982×784. Consequently, the test sets are *"test3"* of size 1010×784 and *"test8"* of size 974×784. The parameter $C = 1$ is used for these data sets as regularization parameter. As we mentioned in previous chapters CG is a parameter free algorithm but CGS and CGS-ls require the value of parameter *diameter* which based on the regularization parameter C the diameter approximately equals 1.5. The CGS algorithm also requires the parameter *Lipschitz constatnt* which approximately equals 2×10^5. For the CGS with linesearch, on the other hand, instead of the Lipschitz constant we have the first guess of 5000 as L_0.

We run the three algorithms for binary classification of the MNIST dataset without fixing

the CPU-time. The stopping criteria for the algorithms is the Wolfe gap, which is a lower bound of the objective value difference $f(x) - f^*$ for any feasible approximate solution x. In better words, we iterate the algorithms until the Wolfe gap becomes smaller than a tolerance that we already defined. The primary tolerance that is defined for this case is 10^{-3}.

In Figure 3.12 we compare the value of the objective value of (3.1) per iteration. We can observe that CGS-ls shows higher decrease in objective value per iteration in comparison with CG and CGS.

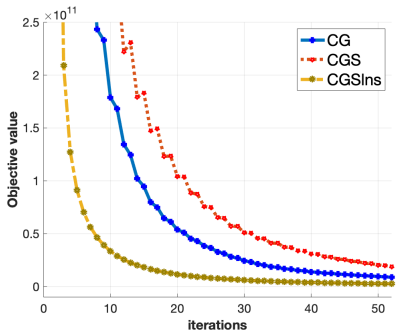

Figure 3.12: Iterations versus objective value.

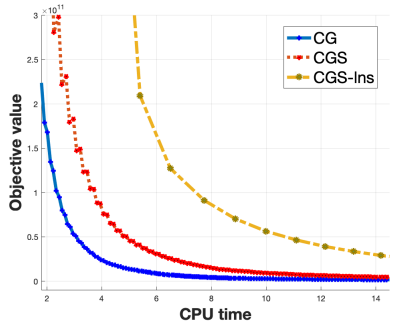

Figure 3.13: CPU time versus Objective value.

On the other hand, unlike CG, per outer iteration of CGS and CGS-ls there are various number of inner iterations. This leads these algorithms to consume higher CPU-time per gradient evaluation in comparison with CG. In addition, since the value of parameter L might change per outer iteration in CGS-ls we expected a weaker performance in this algorithm in terms of CPU time in comparison with CGS. Figure 3.13 illustrates this drawback in CPU-time clearly.

3.2.0.1 Iteration and CPU-time v.s Wolfe Gap and Accuracy

In Figures 3.14 and 3.16 we can see the performance of three algorithms in terms of Wolfe gap. Because of lower number of gradient evaluations and the existence of inner iterations we expect a faster decrease in Wolfe gap per iteration from CGS-ls in comparison with CG.

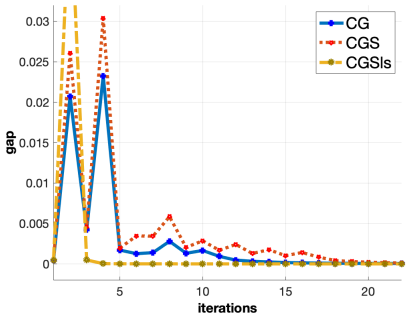

Figure 3.14: Iterations versus Wolfe gap.

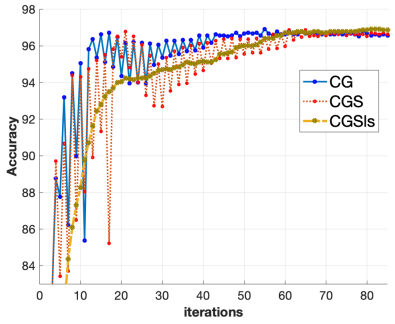

Figure 3.15: Iterations versus accuracy.

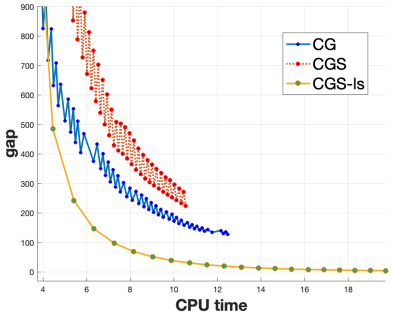

Figure 3.16: CPU-time versus Wolfe gap.

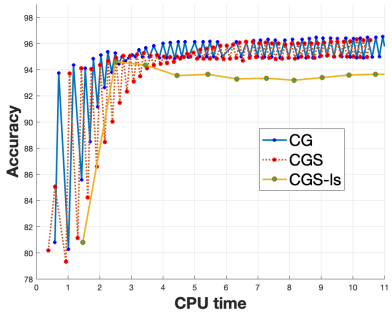

Figure 3.17: CPU-time versus accuracy.

In terms of accuracy per iteration and CPU-time, however, from Figures 3.15 and 3.17 we can see that the algorithm CGS-ls shows a weaker performance in first 60 iterations and after that the maximum accuracy of CG, CGS and CGS-ls after 60 iteration is around 97%.

To end this chapter we provide in Figure 3.18 of a few converted points in test sets test3 and test8 that are note classified correctly via the three algorithms. The first row of Figure 3.18 are points in test set test3 and the second row are points in test set test8 of MNIST data set.

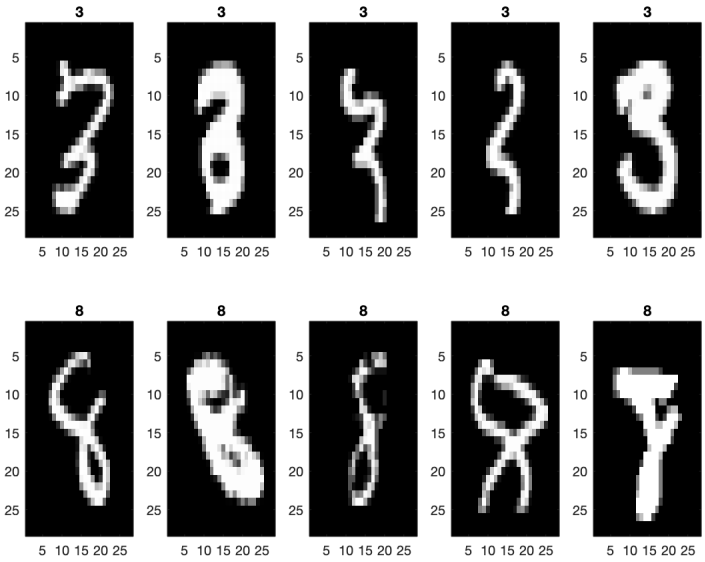

Figure 3.18: A sample of points that are not successfully classified with CG.

Bibliography

[1] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[2] Thomas Bonesky, Kristian Bredies, Dirk A Lorenz, and Peter Maass. A generalized conditional gradient method for nonlinear operator equations with sparsity constraints. *Inverse Problems*, 23(5):2041, 2007.

[3] Kristian Bredies and Dirk A Lorenz. Iterated hard shrinkage for minimization problems with sparsity constraints. *SIAM Journal on Scientific Computing*, 30(2):657–683, 2008.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[5] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[6] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint arXiv:1309.5550*, 2013.

[7] Guanghui Lan. Gradient sliding for composite optimization. *Mathematical Programming*, 159(1-2):201–235, 2016.

[8] Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016.

[9] Yu Nesterov. Complexity bounds for primal-dual methods minimizing the model of objective function. *Mathematical Programming*, 171(1-2):311–330, 2018.

[10] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.