Lectures on Greedy-type Algorithms in Convex Optimization

prepared for the Machine Learning Summer School 2015

University of Texas, Austin

by Robert M. Freund, MIT

©2015 Massachusetts Institute of Technology. All rights reserved.

January 13, 2015
1 Motivation: Binary Classification and Convex Optimization

Suppose we are given \( m \) pixilated digital images of faces; some of the \( m \) faces are female faces and the others are male faces. For each \( i = 1, \ldots, m \) let \( a^i \in \mathbb{R}^n \) denote the digital image of face \( i \), where each of the \( n \) coefficients \((a^i)_j, j = 1, \ldots, n\), corresponds to a pixel. We also have the classification \( y_i \in \{-1, +1\} \) for each face \( i \), where \( y_i = -1 \) corresponds to a male face and \( y_i = +1 \) corresponds to a female face, for \( i = 1, \ldots, m \). We would like to compute a linear decision rule, comprised of a vector \( \lambda \in \mathbb{R}^n \), that will satisfy \( \lambda^T a^i > 0 \) if \( y_i = +1 \), and \( \lambda^T a^i < 0 \) if \( y_i = -1 \), for all \( i = 1, \ldots, m \). If we can compute \( \lambda \) that satisfies these conditions, we can then use \( \lambda \) to establish a prediction procedure to declare the gender of any other digital face image as follows: given a face image \( c \in \mathbb{R}^n \), we will declare whether \( c \) is the image of a male or female face as follows:

- if \( \lambda^T c > 0 \), then we declare that \( c \) is a female face image, and
- if \( \lambda^T c < 0 \), then we declare that \( c \) is a male face image.

The accuracy of this prediction procedure is a nontrivial matter. We will use convex optimization to try to compute an accurate linear predictor model \( \lambda \).

Much more generally, suppose we are given training data \( a^1, \ldots, a^m \in \mathbb{R}^n \) and \( y_i \in \{-1, +1\} \) for \( i = 1, \ldots, m \) for which:

- if \( y_i = +1 \), then data record \( i \) has property “P”, and
- if \( y_i = -1 \), then data record \( i \) does not have property “P”.

We would like to use the \( m \) training data records to develop a linear rule \( \lambda \) that can be used to predict whether or not other points \( c \) might or might not have property P. In particular, we seek a vector \( \lambda \) for which:

- if \( y_i = +1 \), then \( \lambda^T a^i > 0 \), and
- if \( y_i = -1 \), then \( \lambda^T a^i < 0 \).

We will then use \( \lambda \) to predict whether or not any other point \( c \) has property P or not. If we are given another vector \( c \), we will declare whether \( c \) has property P or not as follows:

- if \( \lambda^T c > 0 \), then we declare that \( c \) has property P, and
• if $\lambda^T c < 0$, then we declare that $c$ does not have property P.

Let us now simplify our notation a bit. Note that our criteria for selecting $\lambda$ can be re-written as:

$$y_i (a^i)^T \lambda > 0, \quad i = 1, \ldots, m.$$  

We say that the training data is separable if there exists $\lambda$ for which the above strict inequalities are satisfied. 

Let us write this in matrix form. Define the $m \times n$ matrix $A$ as follows:

$$A_{ij} := y_i (a^i)_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$  

The $i^{th}$ row of $A$ is simply the vector $a^i$ multiplied by $y_i$. Then we seek a vector $\lambda$ for which

$$A \lambda > 0.$$  

The above strict inequality is satisfiable if and only if the training data is separable. Of course, there is no guarantee that the data points will be separable, in which case we say that the training data is not separable. If the training data is not separable, we may seek to compute $\lambda$ for which “most” of the rows of $A \lambda$ are positive, and for which the negative components of $A \lambda$ are “not too negative.” We will write this rather informally as:

$$A \lambda \gtrapprox 0;$$  

there are several ways to make the above notions mathematically precise. For now it suffices to keep these ideas informal.

1.1 Separable Training Data, and the Maximum Margin Optimization Problem

When the training data is separable, we seek a vector $\lambda$ for which

$$A \lambda > 0.$$  

It makes reasonable sense to want to find $\lambda$ for which each of the inner products $(a^i)^T \lambda = (A \lambda)_i$ is very positive, so that the decision rule will work “very well” on the training data. We are thus motivated to make the smallest component of $(A \lambda)$, namely $\min_{i \in \{1, \ldots, m\}} (A \lambda)_i$, as large as possible. This is written as:

$$\max_{\lambda} \min_{i = 1, \ldots, m} (A \lambda)_i.$$
The quantity \( \min_{i=1,\ldots,m} (A\lambda)_i \) is called the \textit{margin} of \( \lambda \), so this optimization problem is maximizing the margin. One aspect of the margin is that it is homogeneous in \( \lambda \), which is to say that if we scale \( \lambda \) by a factor of \( \theta > 0 \) then the margin, which is the objective function above, scales by \( \theta \) as well. Since we can scale \( \theta\lambda \) arbitrarily large yielding an arbitrarily large margin, we impose a normalization criterion, and ours will be that \( \|\lambda\|_1 := \sum_{j=1}^n |\lambda_j| \leq 1 \). Our optimization problem now is:

\[
\text{MMP:} \quad \maximize_{\lambda} \quad \min_{i=1,\ldots,m} (A\lambda)_i \\
\text{s.t.} \quad \|\lambda\|_1 \leq 1 \\
\lambda \in \mathbb{R}^n .
\]

Notice that MMP (for Maximum Margin Problem) will have a positive optimal objective function value if and only if the training data is separable, that is, if and only if there exists \( \lambda \) for which \( A\lambda > 0 \).

### 1.2 Non-Separable Training Data, and the Exponential Loss and Logistic Loss Optimization Problems

Let us now presume that the training data is not separable, as is the case in most practical applications. Consider the following optimization problem:

\[
\text{LELP :} \quad L^*_{\exp} := \minimize_{\lambda} \quad L_{\exp}(\lambda) := \ln \left( \frac{1}{m} \sum_{i=1}^m \exp \left( (-A\lambda)_i \right) \right) \\
\text{s.t.} \quad \lambda \in \mathbb{R}^n .
\]

The objective function \( L_{\exp}(\lambda) \) of LELP is called the \textit{log-exponential loss function}. This function is a monotone decreasing function of \( (A\lambda)_i \) for \( i = 1,\ldots,m \); therefore minimizing this function will encourage larger values of \( (A\lambda)_i \) for \( i = 1,\ldots,m \).

Another loss function that is often used is the \textit{logistic loss function} \( L_{\logit}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R} \) which is defined by:

\[
L_{\logit}(\lambda) := \frac{1}{m} \sum_{i=1}^m \ln (1 + \exp ((-A\lambda)_i)) ,
\]

which has very appealing statistical properties in addition to some very special optimization properties. This function is also a monotone decreasing function of \( (A\lambda)_i \) for \( i = 1,\ldots,m \).
The logistic loss optimization problem is that of minimizing the logistic loss function $L_{\logit}(\cdot)$ and is formulated as:

$$\text{LOGIT : } L^*_{\logit} := \min_{\lambda} \ L_{\logit}(\lambda) := \frac{1}{m} \sum_{i=1}^{m} \ln \left( 1 + \exp \left( -A\lambda_i \right) \right)$$

s.t. $\lambda \in \mathbb{R}^n$.

1.3 Computing Sparse Solutions, and Optimization Algorithms for Binary Classification

The above optimization problems are illustrative of optimization problems that arise quite naturally in machine learning. Let $\lambda^*$ be an optimal (or a nearly optimal) solution to one of the optimization problems MMP, LELP, or LOGIT. Let $\|\lambda^*\|_0$ denote the number of non-zero components of $\lambda^*$. In very many contexts it is important that $\|\lambda^*\|_0$ be small, that is, $\lambda^*$ is a sparse vector ($\lambda^*$ has very few non-zero components). This might be for computational reasons – to manage computations and memory especially when $n \gg 0$ is huge-scale. But in most instances the sparsity of $\lambda^*$ arises from the notion that the solution ought to depend on only a very few underlying features (components), etc. And in the presence of potential over-fitting of the solution based on the training data, experience in many contexts has shown that sparse solutions usually are more accurate at prediction on out-of-sample data. Therefore in addition to optimization of the loss functions in MMP, LELP, and/or LOGIT, we also seek a solution that is sparse in the sense that $\|\lambda^*\|_0$ is relatively small. We could try to enforce the sparsity requirement by adding a constraint of the type “$\|\lambda^*\|_0 \leq r$” to the optimization problem, for some small value of $r$. However, this direct approach to tackling the sparsity issue renders the resulting problem very hard – indeed NP-complete in the language of theoretical computer science. Instead, we will see that certain algorithms, that are usually referred to as “greedy” methods, are by their design able to encourage or guarantee sparse solutions.

In these notes we discuss three basic types of “greedy” algorithms in convex optimization, namely:

1. **Greedy Coordinate Descent.** Coordinate descent methods are methods for solving convex optimization problems that adjust one coordinate at each iteration. These methods were first developed in the early days of optimization as ways to keep computation and memory costs low. Today such methods have experienced a huge resurgence in interest due to their ability to produce sparse solutions, that is, solutions that use very few coordinates. See Beck and Tetruashvili [1] and Nesterov [9] for two excellent modern treatments of these methods.

2. **Frank-Wolfe Method.** The Frank-Wolfe method is originally due to Marguerite Frank and Philip Wolfe [5], and is also known as the “conditional gradient” method, see Levitin and Polyak [8]. Some older references for the method include Polyak [10],
Demayanov and Rubinov [3], and Dunn and Harshbarger [4]. More modern references include Jaggi [7] and Freund and Grigas [6]. See Clarkson [2] especially for a treatment that is more focused on computer science applications.

3. **Randomized Coordinate Descent.** The treatment of randomized coordinate descent that is developed in these notes is based very specifically on Richtárik and Takáč [11], which itself is motivated from the modern treatment of this topic by Nesterov [9].
2 Greedy Coordinate Descent Method for Unconstrained Smooth Optimization

2.1 Problem Setting and Basics for Coordinate Descent Method

Our problem of interest is

\[ P: \text{minimize}_x f(x) \]

\[ \text{s.t.} \quad x \in \mathbb{R}^n, \]

where \( f(\cdot) \) is a differentiable convex function defined on \( \mathbb{R}^n \). We denote the optimal objective value of \( P \) by \( f^* \), and let \( x^* \) denote an optimal solution of \( P \), when such a solution exists.

Let \( \nabla f(x) \) denote the gradient of \( f(\cdot) \) at \( x \), and recall the gradient inequality:

\[ f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \text{for all } x, y \in \mathbb{R}^n. \]  

We will measure distances between points using the \( \ell_1 \) norm \( \|x\|_1 := \sum_{j=1}^n |x_j| \).

We will measure the size of the gradients with the \( \ell_\infty \) norm \( \|g\|_\infty := \max_j \{ |g_j| \} \).

We assume that \( f(\cdot) \) has a Lipschitz gradient. Given our use of the above norms, this means that there is a constant \( L \) for which:

\[ \|\nabla f(y) - \nabla f(x)\|_\infty \leq L \|y - x\|_1 \quad \text{for all } x, y \in \mathbb{R}^n. \]  

Let \( B_1(c, R) \) denote the \( \ell_1 \) ball centered at \( c \) with radius \( R \), namely:

\[ B_1(c, R) := \{ x \in \mathbb{R}^n : \|x - c\|_1 \leq R \}. \]

Let \( e^i \) denote the \( i \)th unit vector in \( \mathbb{R}^n \), namely \( e^i = (0, \ldots, 0, 1, 0, \ldots, 0)^T \) where the 1 appears in the \( i \)th coefficient.

Algorithm 1 presents the greedy coordinate descent method for solving (P).

Suppose that Algorithm 1 is initiated at \( x^0 \leftarrow 0 \). Then due to the update rule in Step (3.) of the algorithm, it will hold that \( \|x^{k+1}\|_0 \leq \|x^k\|_0 + 1 \), whereby it holds by induction that \( \|x^k\|_0 \leq k \) for all \( k \). This sparsity guarantee is particularly useful in the case of huge-scale problems, where \( n \gg 0 \) might be on the order of \( 10^6 \) or larger.
Algorithm 1 Greedy Coordinate Descent Method

**Initialize.** Initialize at \( x^0, i \leftarrow 0 \)

At iteration \( i \):

1. **Compute Gradient.** Compute \( \nabla f(x^i) \)

2. **Compute Coordinate.** Compute:
   
   \[
   j_i \in \arg \max_{j \in \{1, \ldots, n\}} \{ |\nabla f(x^k)_j| \}, \text{ and} \]
   
   \[
   s_i \leftarrow \text{sgn}(\nabla f(x^i)_j_i) \]

3. **Compute Step-Size and New Point.** Compute:
   
   \[
   \alpha_i \leftarrow \frac{|\nabla f(x^i)_j_i|}{L} \]
   
   \[
   x^{i+1} \leftarrow x^i - \alpha_is_ie_j_i
   \]

### 2.2 Analysis and Complexity of the Greedy Coordinate Descent Method

We start with the following:

**Proposition 1.** If \( f(\cdot) \) has Lipschitz gradient with constant \( L \), then

\[
f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2} \|y - x\|_1^2 \]

for all \( x, y \in \mathbb{R}^n \).

**Proof.** We use the fundamental theorem of calculus applied in a multivariate setting. We have

\[
f(y) = f(x) + \int_0^1 \nabla f(x + t(y - x))^T(y - x) dt
\]

\[
= f(x) + \nabla f(x)^T(y - x) + \int_0^1 [\nabla f(x + t(y - x)) - \nabla f(x)]^T (y - x) dt
\]

\[
\leq f(x) + \nabla f(x)^T(y - x) + \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\|_\infty \|y - x\|_1 dt
\]

\[
\leq f(x) + \nabla f(x)^T(y - x) + \int_0^1 tL \|y - x\|_1^2 dt
\]

\[
= f(x) + \nabla f(x)^T(y - x) + \frac{L}{2} \|y - x\|_1^2 . \quad \Box
\]
We will also use the following property of a nonnegative series, whose proof is given at the end of this subsection.

**Proposition 2.** Suppose there is a constant $C > 0$ for which the nonnegative series $\{a_i\}$ satisfies
\[ a_{i+1} \leq a_i - \frac{a_i^2}{C} \quad \text{for all } i \geq 0 . \]

Then it holds that
\[ a_k \leq \frac{1}{a_0 + \frac{k}{C}} \quad \text{for all } k \geq 0 . \]

Let $S_0$ denote the set of all $x$ whose objective value is at most $f(x^0)$, namely $S_0 := \{ x \in \mathbb{R}^n : f(x) \leq f(x^0) \}$, and let $S^*$ denote the set of optimal solutions of (1), namely $S^* := \{ x \in \mathbb{R}^n : f(x) = f^* \}$. Now let Dist$_0$ denote the largest distance of points in $S_0$ to the set of optimal solutions $S^*$:
\[ \text{Dist}_0 := \max_{x \in S_0} \left\{ \min_{x^* \in S^*} \|x - x^*\|_1 \right\} . \] (4)

Our main algorithmic analysis result for the greedy coordinate descent method is:

**Theorem 1.** Let $\{x^k\}$ be generated according to the greedy coordinate descent method (Algorithm 1). Then for all $k \geq 0$ it holds that:
\[ f(x^k) - f^* \leq \frac{1}{f(x^0) - f^* + \frac{k}{2L(\text{Dist}_0)^2}} \leq \frac{2L(\text{Dist}_0)^2}{k} . \] (5)

**Proof:** From Proposition 1 we have for each $i \geq 0$:
\[ f(x^{i+1}) \leq f(x^i) + \nabla f(x^i)^T(x^{i+1} - x^i) + \frac{L}{2}\|x^{i+1} - x^i\|_1^2 \]
\[ = f(x^i) - a_i \|\nabla f(x^i)\|_1 + \frac{L}{2}a_i^2 \]
\[ = f(x^i) - \frac{1}{2L} \|\nabla f(x^i)\|_\infty^2 \]
\[ = f(x^i) - \frac{1}{2L} \|\nabla f(x^i)\|_\infty^2 . \] (6)

This shows that the values $f(x^i)$ are decreasing and in particular $f(x^i) \leq f(x^0)$, whereby $x^i \in S_0$ for all $i \geq 0$. Therefore it follows from (4) that there exists $x^* \in S^*$ for which $\|x^i - x^*\|_1 \leq \text{Dist}_0$, and from the gradient inequality for the convex function $f(\cdot)$ it holds
that
\[ f^* = f(x^*) \geq f(x^i) + \nabla f(x^i)^T (x^* - x^i) \quad \text{(from Gradient Inequality)} \]
\[ \geq f(x^i) - \|\nabla f(x^i)\|_\infty \|x^* - x^i\|_1 \quad \text{(from Norm Inequality)} \]
\[ \geq f(x^i) - \|\nabla f(x^i)\|_\infty \text{Dist}_0, \quad \text{(from above discussion)} \]
and rearranging the above yields \( \|\nabla f(x^i)\|_\infty \geq \frac{f(x^i) - f^*}{\text{Dist}_0} \). Substituting this inequality into (6) and subtracting \( f^* \) from both sides yields:
\[ f(x^{i+1}) - f^* \leq f(x^i) - f^* - \frac{(f(x^i) - f^*)^2}{2\text{Dist}_0^2}. \]
Define \( a_i := f(x^i) - f^* \), and it follows that the nonnegative series \( \{a_i\} \) satisfies \( a_{i+1} \leq a_i - \frac{a_i^2}{2\text{Dist}_0^2} \). It then follows from Proposition 2 using \( C = 2L(\text{Dist}_0)^2 \) that
\[ a_k \leq \frac{1}{a_0 + \frac{k}{2\text{Dist}_0^2}}, \]
which establishes that:
\[ f(x^k) - f^* \leq \frac{1}{\frac{1}{f(x^i) - f^*} + \frac{k}{2L(\text{Dist}_0)^2}}. \]

**Proof of Proposition 2:** Notice that the conclusion of the proposition holds for \( k = 0 \). By induction, suppose that the conclusion is true for some \( k \geq 0 \).

Then:
\[ a_{k+1} \leq a_k - \frac{a_k^2}{C} \]
\[ \leq a_k - \frac{a_k a_{k+1}}{C}, \]
and collecting terms and rearranging yields:
\[ a_{k+1} \left(1 + \frac{a_k}{C}\right) \leq a_k. \]
Taking reciprocals and rearranging yields:
\[ \frac{1}{a_{k+1}} \geq \frac{1}{a_k} \left(1 + \frac{a_k}{C}\right) \]
\[ = \frac{1}{a_k} + \frac{1}{C} \]
\[ \geq \frac{1}{a_0} + \frac{k}{C} + \frac{1}{C} \]
\[ = \frac{1}{a_0} + \frac{k+1}{C}, \]
10
which rearranges to yield:

\[ a_{k+1} \leq \frac{1}{a_0 + \frac{k+1}{c}}, \]

completing the proof.

\[ \square \]

2.3 Comments and Extensions

1. How might the method and the analysis be modified if \( L \) is not explicitly known?

2. How might the method and the analysis be modified if one can efficiently do a line-search to compute:

\[ \alpha_i := \arg \min_\alpha \left\{ f(x^i - \alpha s_i e^j) \right\}, \]

instead of assigning the value \( \alpha_i \leftarrow \frac{|\nabla f(x^i)_h|}{L} \) at each iteration?
3 The Frank-Wolfe Method for Constrained Smooth Optimization

Let us now consider the following *constrained* optimization problem:

\[
P: \text{minimize}_{x} \ f(x) \\
\text{s.t.} \quad x \in S,
\]

where \( f(x) \) is a differentiable convex function on \( S \subseteq \mathbb{R}^n \), and \( S \) is a closed and bounded convex set. Let \( f^* \) denote the optimal objective value of \( P \). We assume that \( f(\cdot) \) has a Lipschitz gradient on \( S \), which means there exists \( L \) for which:

\[
\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\| \quad \text{for all} \ x, y \in S. \quad (7)
\]

(Throughout this section the norm is the Euclidean norm \( \|v\| = \sqrt{v^T v} \).)

We now describe the Frank-Wolfe method for solving \( P \). This method is based on the premise that the set \( S \) is well-suited for linear optimization. This means that either \( S \) is itself a system of linear inequalities \( S = \{x \mid Ax \leq b\} \) over which linear optimization is an easy task, or more generally that the problem:

\[
LO_c: \text{minimize}_{x} \ c^T x \\
\text{s.t.} \quad x \in S
\]

is easy to solve for any given objective function vector \( c \). This being the case, suppose that we have a given iterate value \( x^k \in S \). The linearization of the function \( f(x) \) at \( x = x^k \) is:

\[
z_1(x) := f(x^k) + \nabla f(x^k)^T(x - x^k),
\]

which is the first-order Taylor expansion of \( f(\cdot) \) at \( x^k \). Since we can easily do linear optimization on \( S \), let us solve:

\[
LP: \text{minimize}_{x} \ f(x^k) + \nabla f(x^k)^T(x - x^k) \\
\text{s.t.} \quad x \in S,
\]
which of course is equivalent to:

\[ LP : \text{ minimize}_x \quad \nabla f(x^k)^T x \]
\[ \text{s.t.} \quad x \in S . \]

Let \( \tilde{x}^k \) denote the optimal solution to this problem. The Frank-Wolfe method proceeds by choosing the next iterate as \( x^{k+1} \leftarrow x^k + \alpha(\tilde{x}^k - x^k) \) for some step-size \( \alpha \). Since \( S \) is a convex set and both \( x^k \) and \( \tilde{x}^k \) are contained in \( S \), then \( x^k + \alpha(\tilde{x}^k - x^k) \in S \) for all \( \alpha \in [0, 1] \). Let \( \bar{\alpha}_k \in [0, 1] \) denote the step-size at iteration \( k \) of the method. We have:

\[ x^{k+1} \leftarrow x^k + \bar{\alpha}_k(\tilde{x}^k - x^k) \quad \text{for some } \bar{\alpha}_k \in [0, 1] . \]

The value of \( \bar{\alpha}_k \) can be determined in a number of different ways. One way to set the step-size \( \bar{\alpha}_k \) is according to some pre-determined rule, such as the following often-used rule:

\[ \bar{\alpha}_k = \frac{2}{k + 2} . \]

(We will see shortly that this rule leads to a very good computational bound on the convergence of the Frank-Wolfe method.) Another way is to choose \( \bar{\alpha}_k \) by performing a line-search of \( f(\cdot) \) over the interval \( \alpha \in [0, 1] \). That is, we might determine \( \bar{\alpha}_k \) as the optimal solution to the following line-search problem:

\[ \bar{\alpha}_k \leftarrow \arg \min_{\alpha} \quad f(x^k + \alpha(\tilde{x}^k - x^k)) \]
\[ \text{s.t.} \quad 0 \leq \alpha \leq 1 . \]

Algorithm 2 presents a formal statement of the Frank-Wolfe method just described.

Let us now see how the Frank-Wolfe method can be used to generate sparse solutions in the context of certain structured problems. Suppose our problem of interest is the constrained minimization problem:

\[ P_r : \text{ minimize}_x \quad f(x) \]
\[ \text{s.t.} \quad \|x\|_1 \leq r , \]

where \( f(\cdot) \) is a differentiable convex loss function associated with a machine learning binary classification problem, such as the log-exponential loss function:

\[ L_{\text{exp}}(x) := \ln \left( \frac{1}{m} \sum_{i=1}^{m} \exp((-Ax)_i) \right) \]
Algorithm 2 Frank-Wolfe Method for minimizing $f(x)$ over $x \in S$

Initialize at $x^0 \in S$, $k \leftarrow 0$.

At iteration $k$:

1. Compute $\nabla f(x^k)$, and then solve linear optimization problem:
   \[
   \tilde{x}^k \leftarrow \arg\min_{x \in S} \{f(x^k) + \nabla f(x^k)^T (x - x^k)\}.
   \]

2. Set $x^{k+1} \leftarrow x^k + \bar{\alpha}_k (\tilde{x}^k - x^k)$, where $\bar{\alpha}_k \in [0, 1]$.

or the logistic loss function:
\[
L_{\text{logit}}(x) := \frac{1}{m} \sum_{i=1}^{m} \ln (1 + \exp ((-Ax)_i))
\]
or perhaps some other context-specific loss function. Here the feasible region is described by the $\ell_1$ norm constraint $\|x\|_1 \leq r$, namely $\sum_{j=1}^{n} |x_j| \leq r$. At each iteration of the Frank-Wolfe method, the algorithm computes $\tilde{x}^k \leftarrow \arg\min_{\|x\|_1 \leq r} \{f(x^k) + \nabla f(x^k)^T (x - x^k)\}$. It is straightforward to see that the $\tilde{x}^k$ can always be chosen as $\tilde{x}^k \leftarrow r \cdot s_i e_j$ where:
\[
j_i \in \arg \max_{j \in \{1, \ldots, n\}} \{|\nabla f(x^k)_j|\} \quad \text{and} \quad s_i \leftarrow \text{sgn}(\nabla f(x^k)_i).
\]
Suppose that Algorithm 2 is initiated at $x^0 = 0$, which is feasible for $P_r$. Then due to the update rule in Step (2.) of the algorithm, it holds by induction that $x^k$ will be a convex combination of at most $k + 1$ of the vectors $\pm re^1, \ldots, \pm re^n$. Therefore after $k$ iterations $x^k$ will have at most $k$ non-zeroes, namely, $\|x^k\|_0 \leq k$. This structural guarantee is particularly useful in the case of huge-scale problems, where $n \gg 0$ might be on the order of $10^6$ or larger.

3.1 Analysis and Complexity of the Frank-Wolfe Method

Before stating the main algorithmic analysis result for the Frank-Wolfe method, we first present two important preliminary results. The first result states that at each iteration $k$ of the Frank-Wolfe method, the solution of the linear optimization problem leads to a valid lower bound on the optimal value $f^*$ of the optimization problem $P$.

**Proposition 3.** At iteration $k$ of the Frank-Wolfe method (Algorithm 2), it holds that
\[
f^* \geq f(x^k) + \nabla f(x^k)^T (\tilde{x}^k - x^k).
\]
Proof: From the gradient inequality for convex functions one has:

\[ f(x) \geq f(x^k) + \nabla f(x^k)^T (x - x^k) \quad \text{for any } x \in S. \]

Therefore:

\[ f^* = \min_{x \in S} f(x) \geq \min_{x \in S} f(x^k) + \nabla f(x^k)^T (x - x^k) = f(x^k) + \nabla f(x^k)^T (\tilde{x}^k - x^k). \]

We will also use the following property of certain types of nonnegative series', whose proof is given at the end of this subsection.

**Proposition 4.** Suppose there is a constant \( C > 0 \) for which the nonnegative series \( \{a_i\} \) satisfies

\[ a_{i+1} \leq a_i \left(1 - \frac{2}{i+2}\right) + \frac{C}{(i+2)^2} \quad \text{for all } i \geq 0. \]

Then it holds that

\[ a_k \leq \frac{C}{k+2} \quad \text{for all } k \geq 1. \]

Let \( \text{Diam}(S) \) denote the largest distance between any two points in \( S \), namely:

\[ \text{Diam}(S) := \max_{x,y \in S} \{ \|x - y\| \}. \quad (8) \]

Our main algorithmic analysis result for the Frank-Wolfe method is:

**Theorem 2.** Let \( \{x^k\} \) be generated according to the Frank-Wolfe method (Algorithm 2) using the step-size rule

\[ \bar{\alpha}_k = \frac{2}{k+2}. \]

Then for all \( k \geq 1 \) it holds that:

\[ f(x^k) - f^* \leq \frac{2L(\text{Diam}(S))^2}{k+2}. \quad (9) \]

Proof: For every iteration \( k \geq 0 \) it holds that:
\[
f(x^{k+1}) \leq f(x^k) + \nabla f(x^k)^T (x^{k+1} - x^k) + \frac{L}{2} \|x^{k+1} - x^k\|^2 \quad \text{(from Prop. 1)}
\]
\[
= f(x^k) + \bar{\alpha}_k \nabla f(x^k)^T (\bar{x}^k - x^k) + \frac{L\bar{\alpha}_k^2}{2} \|\bar{x}^k - x^k\|^2
\]
\[
\leq f(x^k) + \bar{\alpha}_k \nabla f(x^k)^T (\bar{x}^k - x^k) + \frac{L\bar{\alpha}_k^2}{2} (\text{Diam}(S))^2
\]
\[
= (1 - \bar{\alpha}_k) f(x^k) + \bar{\alpha}_k (f(x^k) + \nabla f(x^k)^T (\bar{x}^k - x^k)) + \frac{L\bar{\alpha}_k^2}{2} (\text{Diam}(S))^2
\]
\[
\leq (1 - \bar{\alpha}_k) f(x^k) + \bar{\alpha}_k f^* + \frac{L\bar{\alpha}_k^2}{2} (\text{Diam}(S))^2. \quad \text{(from Prop. 3)}
\]

Subtracting \( f^* \) from each side and rearranging the right-hand side, one obtains:
\[
f(x^{k+1}) - f^* \leq (f(x^k) - f^*)(1 - \bar{\alpha}_k) + \frac{L\bar{\alpha}_k^2}{2} (\text{Diam}(S))^2,
\]

and substituting the value of \( \bar{\alpha}_k = \frac{2}{k+2} \) we arrive at:
\[
f(x^{k+1}) - f^* \leq (f(x^k) - f^*)(1 - \frac{2}{k+2}) + \frac{2L(\text{Diam}(S))^2}{(k+2)^2}.
\]

Define \( a_k := f(x^k) - f^* \), and define \( C := 2L(\text{Diam}(S))^2 \), and notice that the above inequality is:
\[
a_{k+1} \leq a_k \left(1 - \frac{2}{k+2}\right) + \frac{C}{(k+2)^2} \quad \text{for all } k \geq 0.
\]

Notice that the series \( \{a_k\} \) satisfies the condition of Proposition 4. It therefore follows from Proposition 4 that \( a_k \leq \frac{C}{k+2} \) for \( k \geq 1 \). Therefore:
\[
f(x^k) - f^* = a_k \leq \frac{C}{k+2} = \frac{2L(\text{Diam}(S))^2}{k+2}.
\]

\textbf{Proof of Proposition 4:} Notice that the conclusion of the proposition holds for \( k = 1 \), since using \( i = 0 \) we have:
\[
a_1 = a_{i+1} \leq a_i \left(1 - \frac{2}{i+2}\right) + \frac{C}{(i+2)^2}
\]
\[
= a_0 \left(1 - \frac{2}{0+2}\right) + \frac{C}{(0+2)^2}
\]
\[
= \frac{C}{4} \leq \frac{C}{3} = \frac{C}{k+2}.
\]
By induction, suppose that the conclusion is true for some $k \geq 0$, namely $a_k \leq \frac{C}{k+2}$. Then:

$$a_{k+1} \leq a_k \left(1 - \frac{2}{k+2}\right) + \frac{C}{(k+2)^2}$$

$$\leq \frac{C}{k+2} \left(1 - \frac{2}{k+2}\right) + \frac{C}{(k+2)^2}$$

$$= \frac{C(k+1)}{(k+2)^2}$$

$$< \frac{C}{(k+3)} ,$$

where the last inequality follows since $(k+1)(k+3) < (k+2)^2$. The result then follows by induction. \qed

### 3.2 Comments and Extensions

1. How might the Frank-Wolfe method and its analysis be modified if $L$ is not explicitly known?

2. How might the method and the analysis be modified if one can efficiently do a line-search to compute:
   $$\bar{\alpha}_k := \arg \min_{\alpha \in [0,1]} \left\{ f(x^k + \alpha(\bar{x}^k - x^k)) \right\} ,$$
   instead of assigning the value $\bar{\alpha}_k = \frac{2}{k+2}$ at each iteration?
4 Randomized Block-Coordinate Descent for Unconstrained Smooth Optimization

4.1 Randomized Block-Coordinate Descent Method

Herein we consider the following unconstrained smooth optimization problem:

$$f^* := \min_{x \in \mathbb{R}^N} f(x),$$  \hspace{1cm} (10)

where $f(\cdot)$ is convex and differentiable on $x \in \mathbb{R}^N$, and $x$ and/or $f(\cdot)$ are assumed to have special block structure. We model the block structure of the problem by decomposing the space $\mathbb{R}^N$ into $n$ subspaces as follows. Let $U \in \mathbb{R}^{N \times N}$ be a column permutation of the $N \times N$ identity matrix and further let $U = [U_1, U_2, \ldots, U_n]$ be a decomposition of $U$ into $n$ submatrices, with $U_i$ being of size $N \times N_i$, where $\sum_i N_i = N$. Clearly, any vector $x \in \mathbb{R}^N$ can be written uniquely as $x = \sum_{i=1}^n U_i x^{(i)}$, where $x^{(i)} = U_i^T x \in \mathbb{R}^{N_i}$. Notice that:

$$U_i^T U_j = \begin{cases} N_i \times N_i & \text{identity matrix} \quad \text{if } i = j, \\ N_i \times N_j & \text{zero matrix} \quad \text{otherwise} \end{cases}. \hspace{1cm} (11)$$

For simplicity we will write $x = (x^{(1)}, \ldots, x^{(n)})^T$. We equip $\mathbb{R}^{N_i}$ with the Euclidean norm:

$$\|t\|_{(i)} = \sqrt{\langle t, t \rangle} \quad \text{for } t \in \mathbb{R}^{N_i}, \hspace{1cm} (12)$$

where $\langle \cdot, \cdot \rangle$ is the standard Euclidean inner product. For $x = \sum_{i=1}^n U_i x^{(i)} \in \mathbb{R}^N$ we have

$$\|x\| := \sqrt{\sum_{i=1}^n \|x^{(i)}\|_{(i)}^2} = \sqrt{\sum_{i=1}^n \langle x^{(i)}, x^{(i)} \rangle}. \hspace{1cm} (13)$$

The $i$th block component of the gradient of $f(\cdot)$ is denoted by $\nabla_i f(x)$ and is formally defined by:

$$\nabla_i f(x) := U_i^T \nabla f(x) \in \mathbb{R}^{N_i} \quad \text{for } i = 1, \ldots, N. \hspace{1cm} (14)$$

We assume that the gradient of $f(\cdot)$ is block coordinate-wise Lipschitz, uniformly in $x$, with Lipschitz constant $L > 0$. This means that for all $x \in \mathbb{R}^N$, $i = 1, 2, \ldots, n$, and $t \in \mathbb{R}^{N_i}$, it holds that:

$$\|\nabla_i f(x + U_i t) - \nabla_i f(x)\|_{(i)} \leq L \|t\|_{(i)}. \hspace{1cm} (15)$$

Almost identical to Proposition 1, we have:

**Proposition 5.** If $f(\cdot)$ satisfies (15), then

$$f(x + U_i t) \leq f(x) + \langle \nabla_i f(x), t \rangle + \frac{L}{2} \|t\|_{(i)}^2, \quad \text{for } i = 1, \ldots, N.$$
Proof. Similar to the proof of Proposition 1, we use the fundamental theorem of calculus applied in a multivariate setting. We have:

\[
\begin{align*}
f(x + U_t) &= f(x) + \int_0^1 \langle \nabla f(x + \alpha U_t), U_t \rangle d\alpha \\
&= f(x) + \int_0^1 \langle \nabla f(x + \alpha U_t), t \rangle d\alpha \\
&= f(x) + \langle \nabla f(x), t \rangle + \int_0^1 \langle \nabla_i f(x + \alpha U_t) - \nabla_i f(x), t \rangle d\alpha \\
&\leq f(x) + \langle \nabla_i f(x), t \rangle + \int_0^1 \|\nabla_i f(x + \alpha U_t) - \nabla_i f(x)\|_{(i)} \|t\|_{(i)} d\alpha \\
&\leq f(x) + \langle \nabla_i f(x), t \rangle + \int_0^1 \alpha L \|t\|_{(i)} d\alpha \\
&= f(x) + \langle \nabla_i f(x), t \rangle + \frac{L}{2} \|t\|_{(i)}^2 .
\end{align*}
\]

We present and study the Randomized Block Coordinate Descent Method, shown here in Algorithm 3.

**Algorithm 3** Randomized Block Coordinate Descent Method

**Initialize.** Initialize at \(x_0 \in \mathbb{R}^N\), \(k \leftarrow 0\)

At iteration \(k\):

1. **Choose Random Block.** Choose \(i_k = i \in \{1, 2, \ldots, n\}\) according to the uniform distribution on \(\{1, 2, \ldots, n\}\): \(P(i_k = i) = 1/n\), for \(i = 1, \ldots, n\).

2. **Take Step in Chosen Block.** \(x_{k+1} = x_k - \frac{1}{L} U_i(\nabla_i f(x_k))\).

Let \(\mathcal{S}_0\) denote the set of all \(x\) whose objective value is at most \(f(x_0)\), namely \(\mathcal{S}_0 := \{x \in \mathbb{R}^N : f(x) \leq f(x_0)\}\), and let \(\mathcal{S}^*\) denote the set of optimal solutions of (10), namely, \(\mathcal{S}^* := \{x \in \mathbb{R}^N : f(x) = f^*\}\). Now let \(\text{Dist}_0\) denote the largest distance of points in \(\mathcal{S}_0\) to the set of optimal solutions \(\mathcal{S}^*\):

\[
\text{Dist}_0 := \max_{x \in \mathcal{S}_0} \left\{ \min_{x^* \in \mathcal{S}^*} \|x - x^*\| \right\} .
\]

(16)

The main computational guarantee for Algorithm 3 is presented in Theorem 3.

**Theorem 3.** Given the initial point \(x_0\), a target accuracy \(\varepsilon\) satisfying \(0 < \varepsilon < \min\{f(x_0) - f^*, 2nL(\text{Dist}_0)^2\}\) and an error tolerance \(\rho \in (0, 1)\), let the Randomized Block Coordinate Descent Method (Algorithm 3) be run for \(k\) iterations where:

\[
k \geq \frac{2nL(\text{Dist}_0)^2}{\varepsilon} \left(1 + \ln \left(\frac{1}{\rho}\right)\right) + 2 .
\]

(17)
Then
\[ P(f(x_k) \leq f^* + \varepsilon) \geq 1 - \rho. \]

We will prove this theorem in Section 4.2.

Suppose that Algorithm 3 is initiated at \( x^0 \leftarrow 0 \). Then due to the update rule in Step (2.) of the algorithm, it holds by induction that \( x^k \) will use at most \( k \) different blocks, for all \( k \). This sparsity guarantee is particularly useful in the case of huge-scale problems, where \( N \gg 0 \) might be on the order of \( 10^6 \) or larger.

### 4.2 Proof of Theorem 3

Let us first develop some straightforward properties of the iterate step of random block coordinate descent. Let \( x_k \) be the current iterate and let \( i \) denote the randomly chosen index at iteration \( k \). Then it holds that:

\[
\begin{align*}
f(x_{k+1}) &= f(x_k - \frac{1}{L}U_i(\nabla_i f(x_k))) \\
&\leq f(x_k) - \frac{1}{L} \langle \nabla_i f(x_k), \nabla_i f(x_k) \rangle + \frac{L}{2} \| \nabla_i f(x_k) \|_2^2 \quad \text{(from Prop. 5)} \\
&= f(x_k) - \frac{1}{2L} \| \nabla_i f(x_k) \|_2^2,
\end{align*}
\]

which shows in particular that the iterate values of \( f(x_k) \) are decreasing and therefore \( x_k \in S_0 \) for all \( k \geq 0 \).

We now provide a lower bound on \( \| \nabla f(x_k) \| \). Because \( x_k \in S_0 \) there exists \( x^* \in S^* \) that satisfies \( \| x^* - x_k \| \leq \text{Dist}_0 \). From the gradient inequality for convex functions we have:

\[
f^* = f(x^*) \geq f(x_k) + \langle \nabla f(x_k), x^* - x_k \rangle \geq f(x_k) - \| \nabla f(x_k) \|_2 \| x^* - x_k \| \geq f(x_k) - \| \nabla f(x_k) \| \text{Dist}_0,
\]

which rearranges to:

\[
\| \nabla f(x_k) \| \geq \frac{f(x_k) - f^*}{\text{Dist}_0}.
\]

We will also will need the following technical result, whose proof is deferred to the end of this section.

**Lemma 1.** Fix \( x_0 \in \mathbb{R}^N \) and let \( \{x_k\}_{k \geq 0} \) be a sequence of random vectors in \( \mathbb{R}^N \) with \( x_{k+1} \) depending only on \( x_k \). Let \( \phi : \mathbb{R}^N \rightarrow \mathbb{R} \) be a nonnegative function and define \( \xi_k = \phi(x_k) \). Lastly, choose accuracy level \( 0 < \varepsilon < \xi_0 \) and error tolerance \( \rho \in (0,1) \), and assume that the sequence of random variables \( \{\xi_k\}_{k \geq 0} \) is nonincreasing and has the following property:
(i) \( E[\xi_{k+1} \mid x_k] \leq \xi_k - \frac{\xi_k^2}{c_1} \), for all \( k \), where \( c_1 > 0 \) is a constant.

If \( \varepsilon < c_1 \) and

\[
K \geq \frac{c_1}{\varepsilon} \left( 1 + \ln \left( \frac{1}{\rho} \right) \right) + 2 - \frac{c_1}{\xi_0},
\]

then

\[
P(\xi_K \leq \varepsilon) \geq 1 - \rho.
\]

We are now ready to prove Theorem 3.

Proof. We first provide a lower bound on the expected decrease of the objective function during one iteration of the method:

\[
E[f(x_{k+1}) \mid x_k] - f(x_k) \leq \sum_{i=1}^{n} \frac{1}{n} \left[ -\frac{1}{2n} \| \nabla_i f(x_k) \|^2 \right],
\]

(from (18))

\[
= -\frac{1}{2nL} \| \nabla f(x_k) \|^2
\]

\[
\leq \frac{(f(x_k) - f^*)^2}{2nL(D_{\text{dist}})^2},
\]

(from (19))

which upon subtracting \( f^* \) from both sides rearranges to:

\[
E[f(x_{k+1}) - f^* \mid x_k] \leq f(x_k) - f^* - \frac{(f(x_k) - f^*)^2}{2nL(D_{\text{dist}})^2}.
\]

We now apply Lemma 1 with \( \xi_k = f(x_k) - f^* \) and \( c_1 = 2nL(D_{\text{dist}})^2 \) to obtain that \( P(f(x_k) \leq f^* + \varepsilon) = P(\xi_k \leq \varepsilon) \geq 1 - \rho \) holds for all \( k \) whose value is at least:

\[
\frac{c_1}{\varepsilon} \left( 1 + \ln \left( \frac{1}{\rho} \right) \right) + 2 - \frac{c_1}{\xi_0} \leq \frac{c_1}{\varepsilon} \left( 1 + \ln \left( \frac{1}{\rho} \right) \right) + 2 = \frac{2nL(D_{\text{dist}})^2}{\varepsilon} \left( 1 + \ln \left( \frac{1}{\rho} \right) \right) + 2.
\]

We end this section with the proof of Lemma 1.

Proof of Lemma 1: Define the sequence \( \{\xi^\varepsilon_k\}_{k \geq 0} \) by:

\[
\xi^\varepsilon_k = \begin{cases} 
\xi_k & \text{if } \xi_k \geq \varepsilon \\
0 & \text{otherwise}
\end{cases},
\]

and notice that \( \xi^\varepsilon_k > \varepsilon \iff \xi_k > \varepsilon \) and also \( \xi^\varepsilon_k \leq \xi_k \) for all \( k \). Therefore by the Markov inequality it holds that: \( P(\xi_k > \varepsilon) = P(\xi^\varepsilon_k > \varepsilon) \leq \frac{E[\xi^\varepsilon_k]}{\varepsilon} \), and hence it suffices to show that

\[
\theta_K \leq \varepsilon \rho.
\]
where $\theta_k := \mathbb{E}[\xi_k^2]$. Towards the proof of (23) we first prove the following two inequalities:

$$
\mathbb{E}[\xi_{k+1}^2 | x_k] \leq \xi_k^2 - \frac{(\xi_k^2)}{c_1}, \quad k \geq 0
$$

(24)

and

$$
\mathbb{E}[\xi_{k+1}^2 | x_k] \leq (1 - \frac{\epsilon}{c_1})\xi_k^2, \quad k \geq 0.
$$

(25)

We consider two cases, where the first case is $\xi_k \geq \epsilon$. Then $\xi_k^2 = \xi_k \geq \epsilon$ and $\xi_{k+1}^2 \leq \xi_k^2 = \xi_k$. Therefore it follows from the hypothesis (i) of the lemma that:

$$
\mathbb{E}[(\xi_k^2)] \leq \mathbb{E}[\xi_{k+1}^2 | x_k] \leq \xi_k - \frac{(\xi_k^2)}{c_1} \leq \xi_k - \frac{(\xi_k^2)}{c_1} = (1 - \frac{\epsilon}{c_1})\xi_k^2,
$$

which proves (24) and (25) in this case. Next consider the second case wherein $\xi_k < \epsilon$. In this case $\xi_k^2 = 0$ and also $\xi_{k+1}^2 = 0$ from which (24) and (25) follow simply since all terms therein are identically zero.

Next notice from the convexity of the function $t \mapsto t^2$ that $\mathbb{E}[(\xi_k^2)] \geq (\mathbb{E}[\xi_k^2])^2 = \theta_k^2$. By taking expectations in (24) and (25) and using the above inequality we obtain:

$$
\theta_{k+1} \leq \theta_k - \frac{\theta_k^2}{c_1}, \quad k \geq 0,
$$

(26)

and

$$
\theta_{k+1} \leq (1 - \frac{\epsilon}{c_1})\theta_k, \quad k \geq 0.
$$

(27)

Notice that the right-hand side of (26) is less than the right-hand side of (27) precisely when $\theta_k > \epsilon$. Since

$$
\frac{1}{\theta_{k+1}} - \frac{1}{\theta_k} = \frac{\theta_k - \theta_{k+1}}{\theta_{k+1} \theta_k} \geq \frac{\theta_k - \theta_{k+1}}{\theta_k^2} \geq \frac{1}{c_1},
$$

(where the last inequality above uses (26)), we have $\frac{1}{\theta_k} \geq \frac{1}{\theta_0} + \frac{k}{c_1} = \frac{1}{\theta_0} + \frac{k}{c_1}$. Therefore, if we let $k_1 = \frac{\epsilon}{\epsilon} - \frac{\epsilon}{\theta_0}$, we obtain $\theta_{k_1} \leq \epsilon$. Finally, letting $k_2 \geq \frac{\epsilon}{\epsilon} \ln \left(\frac{1}{\rho}\right)$, we obtain (23) as follows:

$$
\theta_K \leq \theta_{k_1+k_2} \leq (1 - \frac{\epsilon}{c_1})k_2\theta_{k_1} \leq \left((1 - \frac{\epsilon}{c_1})\frac{1}{\epsilon} \right)^{c_1 \ln \left(\frac{1}{\rho}\right)} \epsilon \leq \left(\epsilon^{-\frac{1}{c_1}} \right)^{c_1 \ln \left(\frac{1}{\rho}\right)} \epsilon = \epsilon \rho,
$$

where the first inequality above uses (20), the second inequality uses (27), and the third inequality uses the convexity the function $t \mapsto e^t$. 

\[\square\]
5 Computation Exercises

1. Binary Classification with a particular data set. This is under construction.

2. Binary Classification with a another data set. This is under construction.
References


