

Subgradient Methods

Stephen Boyd and Almir Mutapcic
Notes for EE364b, Stanford University, Winter 2006-07

January 23, 2007

Contents

1	Introduction	2
2	Basic subgradient method	2
2.1	Negative subgradient update	2
2.2	Step size rules	3
2.3	Convergence results	4
3	Convergence proof	4
3.1	Assumptions	4
3.2	Some basic inequalities	5
3.3	A bound on the suboptimality bound	7
3.4	A stopping criterion	8
3.5	Numerical example	8
4	Alternating projections	9
4.1	Optimal step size choice when f^* is known	9
4.2	Finding a point in the intersection of convex sets	11
4.3	Solving convex inequalities	14
4.4	Positive semidefinite matrix completion	15
5	Projected subgradient method	16
5.1	Numerical example	18
6	Projected subgradient for dual problem	18
6.1	Numerical example	20
7	Subgradient method for constrained optimization	21
7.1	Numerical example	24
8	Speeding up subgradient methods	24

1 Introduction

The *subgradient method* is a very simple algorithm for minimizing a nondifferentiable convex function. The method looks very much like the ordinary gradient method for differentiable functions, but with several notable exceptions:

- The subgradient method applies directly to *nondifferentiable* f .
- The step lengths are not chosen via a line search, as in the ordinary gradient method. In the most common cases, the step lengths are fixed ahead of time.
- Unlike the ordinary gradient method, the subgradient method is *not* a descent method; the function value can (and often does) increase.

The subgradient method is readily extended to handle problems with constraints.

Subgradient methods can be *much* slower than interior-point methods (or Newton's method in the unconstrained case). In particular, they are first-order methods; their performance depends very much on the problem scaling and conditioning. (In contrast, Newton and interior-point methods are second-order methods, not affected by problem scaling.)

However, subgradient methods do have some advantages over interior-point and Newton methods. They can be immediately applied to a far wider variety of problems than interior-point or Newton methods. The memory requirement of subgradient methods can be much smaller than an interior-point or Newton method, which means it can be used for extremely large problems for which interior-point or Newton methods cannot be used. Moreover, by combining the subgradient method with primal or dual decomposition techniques, it is sometimes possible to develop a simple distributed algorithm for a problem. In any case, subgradient methods are well worth knowing about.

The subgradient method was originally developed by Shor and others in the Soviet Union in the 1960s and 1970s. A basic reference on subgradient methods is his book [Sho85]; a very clear discussion can be found in chapter 5 of Polyak's book [Pol87]. Bertsekas [Ber99] is another good reference on the subgradient method, in particular, on how to combine it with primal and dual decomposition. Other book treatments of the topic are in Ruszczyński [Rus06, §7.1], Nesterov [Nes04, Chap. 3], Akgul [Akg84], Yudin and Nemirovski [NY83], Censor and Zenios [CZ97], and Shor [Sho98, Chap. 2]. Some interesting recent research papers on subgradient methods are [NB01] and [Nes05].

2 Basic subgradient method

2.1 Negative subgradient update

We start with the unconstrained case, where the goal is to minimize $f : \mathbf{R}^n \rightarrow \mathbf{R}$, which is convex and has domain \mathbf{R}^n (for now). To do this, the subgradient method uses the simple iteration

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}.$$

Here $x^{(k)}$ is the k th iterate, $g^{(k)}$ is *any* subgradient of f at $x^{(k)}$, and $\alpha_k > 0$ is the k th step size. Thus, at each iteration of the subgradient method, we take a step in the direction of a negative subgradient.

Recall that a subgradient of f at x is any vector g that satisfies the inequality $f(y) \geq f(x) + g^T(y - x)$ for all y . When f is differentiable, the only possible choice for $g^{(k)}$ is $\nabla f(x^{(k)})$, and the subgradient method then reduces to the gradient method (except, as we'll see below, for the choice of step size). The set of subgradients of f at x is the subdifferential of f at x , denoted $\partial f(x)$. So the condition that $g^{(k)}$ be a subgradient of f at $x^{(k)}$ can be written $g^{(k)} \in \partial f(x^{(k)})$.

It can happen that $-g^{(k)}$ is not a descent direction for f at $x^{(k)}$, *i.e.*, $f'(x; -g^{(k)}) > 0$. In such cases we always have $f(x^{(k+1)}) > f(x^{(k)})$. Even when $-g^{(k)}$ is a descent direction at $x^{(k)}$, the step size can be such $f(x^{(k+1)}) > f(x^{(k)})$. In other words, an iteration of the subgradient method can increase the objective function.

Since the subgradient method is not a descent method, it is common to keep track of the best point found so far, *i.e.*, the one with smallest function value. At each step, we set

$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)}, f(x^{(k)})\},$$

and set $i_{\text{best}}^{(k)} = k$ if $f(x^{(k)}) = f_{\text{best}}^{(k)}$, *i.e.*, if $x^{(k)}$ is the best point found so far. (In a descent method there is no need to do this, since the current point is always the best one so far.) Then we have

$$f_{\text{best}}^{(k)} = \min\{f(x^{(1)}), \dots, f(x^{(k)})\},$$

i.e., the best objective value found in k iterations. Since $f_{\text{best}}^{(k)}$ is decreasing, it has a limit (which can be $-\infty$).

2.2 Step size rules

In the subgradient method the step size selection is very different from the standard gradient method. Many different types of step size rules are used. We'll start with five basic step size rules.

- *Constant step size.* $\alpha_k = \alpha$ is a positive constant, independent of k .
- *Constant step length.* $\alpha_k = \gamma / \|g^{(k)}\|_2$, where $\gamma > 0$. This means that $\|x^{(k+1)} - x^{(k)}\|_2 = \gamma$.
- *Square summable but not summable.* The step sizes satisfy

$$\alpha_k \geq 0, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

One typical example is $\alpha_k = a/(b + k)$, where $a > 0$ and $b \geq 0$.

- *Nonsummable diminishing.* The step sizes satisfy

$$\alpha_k \geq 0, \quad \lim_{k \rightarrow \infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

Step sizes that satisfy this condition are called *diminishing step size rules*. A typical example is $\alpha_k = a/\sqrt{k}$, where $a > 0$.

- *Nonsummable diminishing step lengths.* The step sizes are chosen as $\alpha_k = \gamma_k/\|g^{(k)}\|_2$, where

$$\gamma_k \geq 0, \quad \lim_{k \rightarrow \infty} \gamma_k = 0, \quad \sum_{k=1}^{\infty} \gamma_k = \infty.$$

There are still other choices, and many variations on these choices. In §4.1 we will encounter another step size rule that requires knowledge of the optimal value f^* .

The most interesting feature of these choices is that *they are determined before the algorithm is run; they do not depend on any data computed during the algorithm*. This is very different from the step size rules found in standard descent methods, which very much depend on the current point and search direction.

2.3 Convergence results

There are many results on convergence of the subgradient method. For constant step size and constant step length, the subgradient algorithm is guaranteed to converge to within some range of the optimal value, *i.e.*, we have

$$\lim_{k \rightarrow \infty} f_{\text{best}}^{(k)} - f^* < \epsilon,$$

where f^* denotes the optimal value of the problem, *i.e.*, $f^* = \inf_x f(x)$. (This implies that the subgradient method finds an ϵ -suboptimal point within a finite number of steps.) The number ϵ is a function of the step size parameter h , and decreases with it.

For the diminishing step size and step length rules (and therefore also the square summable but not summable step size rule), the algorithm is guaranteed to converge to the optimal value, *i.e.*, we have $\lim_{k \rightarrow \infty} f(x^{(k)}) = f^*$. It's remarkable that such a simple algorithm can be used to minimize any convex function for which you can compute a subgradient at each point. We'll also see that the convergence proof is also simple.

When the function f is differentiable, we can say a bit more about the convergence. In this case, the subgradient method with constant step size yields convergence to the optimal value, provided the parameter α is small enough.

3 Convergence proof

3.1 Assumptions

Here we give a proof of some typical convergence results for the subgradient method. We assume that there is a minimizer of f , say x^* . We also make one other assumption on f :

We will assume that the norm of the subgradients is bounded, *i.e.*, there is a G such that $\|g^{(k)}\|_2 \leq G$ for all k . This will be the case if, for example, f satisfies the Lipschitz condition

$$|f(u) - f(v)| \leq G\|u - v\|_2,$$

for all u, v , because then $\|g\|_2 \leq G$ for any $g \in \partial f(x)$, and any x . In fact, some versions of the subgradient method (*e.g.*, diminishing nonsummable step lengths) work when this assumption doesn't hold; see [Sho85] or [Pol87].

We'll also assume that a number R is known that satisfies $R \geq \|x^{(1)} - x^*\|_2$. We can interpret R as an upper bound on $\mathbf{dist}(x^{(1)}, X^*)$, the distance of the initial point to the optimal set.

3.2 Some basic inequalities

For the standard gradient descent method, the convergence proof is based on the function value decreasing at each step. In the subgradient method, the key quantity is not the function value (which often increases); it is the *Euclidean distance to the optimal set*.

Recall that x^* is a point that minimizes f , *i.e.*, it is an arbitrary optimal point. We have

$$\begin{aligned} \|x^{(k+1)} - x^*\|_2^2 &= \|x^{(k)} - \alpha_k g^{(k)} - x^*\|_2^2 \\ &= \|x^{(k)} - x^*\|_2^2 - 2\alpha_k g^{(k)T}(x^{(k)} - x^*) + \alpha_k^2 \|g^{(k)}\|_2^2 \\ &\leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^*) + \alpha_k^2 \|g^{(k)}\|_2^2, \end{aligned}$$

where $f^* = f(x^*)$. The last line follows from the definition of subgradient, which gives

$$f(x^*) \geq f(x^{(k)}) + g^{(k)T}(x^* - x^{(k)}).$$

Applying the inequality above recursively, we have

$$\|x^{(k+1)} - x^*\|_2^2 \leq \|x^{(1)} - x^*\|_2^2 - 2 \sum_{i=1}^k \alpha_i (f(x^{(i)}) - f^*) + \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2.$$

Using $\|x^{(k+1)} - x^*\|_2^2 \geq 0$ and $\|x^{(1)} - x^*\|_2 \leq R$ we have

$$2 \sum_{i=1}^k \alpha_i (f(x^{(i)}) - f^*) \leq R^2 + \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2. \quad (1)$$

Combining this with

$$\sum_{i=1}^k \alpha_i (f(x^{(i)}) - f^*) \geq \left(\sum_{i=1}^k \alpha_i \right) \min_{i=1, \dots, k} (f(x^{(i)}) - f^*) = \left(\sum_{i=1}^k \alpha_i \right) (f_{\text{best}}^{(k)} - f^*),$$

we have the inequality

$$f_{\text{best}}^{(k)} - f^* = \min_{i=1, \dots, k} f(x^{(i)}) - f^* \leq \frac{R^2 + \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2}{2 \sum_{i=1}^k \alpha_i}. \quad (2)$$

Finally, using the assumption $\|g^{(k)}\|_2 \leq G$, we obtain the basic inequality

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i}. \quad (3)$$

From this inequality we can read off various convergence results.

Constant step size. When $\alpha_k = \alpha$, we have

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + G^2 \alpha^2 k}{2 \alpha k}.$$

The righthand side converges to $G^2 \alpha / 2$ as $k \rightarrow \infty$. Thus, for the subgradient method with fixed step size α , $f_{\text{best}}^{(k)}$ converges to within $G^2 \alpha / 2$ of optimal. We also find that $f(x^{(k)}) - f^* \leq G^2 \alpha$ within at most $R^2 / (G^2 \alpha^2)$ steps.

Constant step length. With $\alpha_k = \gamma / \|g^{(k)}\|_2$, the inequality (2) becomes

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + \gamma^2 k}{2 \sum_{i=1}^k \alpha_i} \leq \frac{R^2 + \gamma^2 k}{2 \gamma k / G},$$

using $\alpha_i \geq \gamma / G$. The righthand side converges to $G \gamma / 2$ as $k \rightarrow \infty$, so in this case the subgradient method converges to within $G \gamma / 2$ of optimal.

Square summable but not summable. Now suppose

$$\|\alpha\|_2^2 = \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha_k = \infty.$$

Then we have

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + G^2 \|\alpha\|_2^2}{2 \sum_{i=1}^k \alpha_i},$$

which converges to zero as $k \rightarrow \infty$, since the numerator converges to $R^2 + G^2 \|\alpha\|_2^2$, and the denominator grows without bound. Thus, the subgradient method converges (in the sense $f_{\text{best}}^{(k)} \rightarrow f^*$).

Diminishing step size rule. If the sequence α_k converges to zero and is nonsummable, then the righthand side of the inequality (3) converges to zero, which implies the subgradient method converges. To show this, let $\epsilon > 0$. Then there exists an integer N_1 such that $\alpha_i \leq \epsilon / G^2$ for all $i > N_1$. There also exists an integer N_2 such that

$$\sum_{i=1}^{N_2} \alpha_i \geq \frac{1}{\epsilon} \left(R^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2 \right),$$

since $\sum_{i=1}^{\infty} \alpha_i = \infty$. Let $N = \max\{N_1, N_2\}$. Then for $k > N$, we have

$$\begin{aligned} \frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i} &\leq \frac{R^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2}{2 \sum_{i=1}^k \alpha_i} + \frac{G^2 \sum_{i=N_1+1}^k \alpha_i^2}{2 \sum_{i=1}^{N_1} \alpha_i + 2 \sum_{i=N_1+1}^k \alpha_i} \\ &\leq \frac{R^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2}{(2/\epsilon) \left(R^2 + G^2 \sum_{i=1}^{N_1} \alpha_i^2 \right)} + \frac{G^2 \sum_{i=N_1+1}^k (\epsilon \alpha_i / G^2)}{2 \sum_{i=N_1+1}^k \alpha_i} \\ &= \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon. \end{aligned}$$

Nonsummable diminishing step lengths. Finally, suppose that $\alpha_k = \gamma_k / \|g^{(k)}\|_2$, with γ_k nonsummable and converging to zero. The inequality (2) becomes

$$f_{\text{best}}^{(k)} - f^* \leq \frac{R^2 + \sum_{i=1}^k \gamma_k^2}{2 \sum_{i=1}^k \alpha_i} \leq \frac{R^2 + \sum_{i=1}^k \gamma_k^2}{(2/G) \sum_{i=1}^k \gamma_i},$$

which converges to zero as $k \rightarrow \infty$.

3.3 A bound on the suboptimality bound

It's interesting to ask the question, what sequence of step sizes minimizes the righthand side of (3)? In other words, how do we choose positive $\alpha_1, \dots, \alpha_k$ so that

$$\frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{2 \sum_{i=1}^k \alpha_i}$$

(which is an upper bound on $f_{\text{best}}^{(k)} - f^*$) is minimized? This is a convex and symmetric function of $\alpha_1, \dots, \alpha_k$, so we conclude the optimal occurs when all α_i are equal (to, say, α). This reduces our suboptimality bound to

$$\frac{R^2 + G^2 k \alpha^2}{2k\alpha}$$

which is minimized by $\alpha = (R/G)/\sqrt{k}$.

In other words, the choice of $\alpha_1, \dots, \alpha_k$ that minimizes the suboptimality bound (3) is given by

$$\alpha_i = (R/G)/\sqrt{k}, \quad i = 1, \dots, k.$$

This choice of constant step size yields the suboptimality bound

$$f_{\text{best}}^{(k)} - f^* \leq RG/\sqrt{k}.$$

Put another way, we can say that for *any* choice of step sizes, the suboptimality bound (3) must be at least as large as RG/\sqrt{k} . If we use (3) as our stopping criterion, then the number of steps to achieve a guaranteed accuracy of ϵ will be at least $(RG/\epsilon)^2$, no matter what step sizes we use. (It will be this number if we use the step size $\alpha_k = (R/G)/\sqrt{k}$.)

Note that RG has a simple interpretation as an initial bound on $f(x^{(1)}) - f^*$, based on $\|x^{(1)} - x^*\|_2 \leq R$ and the Lipschitz constant G for f . Thus $(RG)/\epsilon$ is the ratio of initial uncertainty in f^* to final uncertainty in f^* . If we square this number, we get the minimum number of steps it will take to achieve this reduction in uncertainty. This tells us that the subgradient method is going to be very slow, if we use (3) as our stopping criterion. To reduce the initial uncertainty by a factor of 1000, say, it will require at least 10^6 iterations.

3.4 A stopping criterion

We can use (1) to find a lower bound on f^* that is sharper than the lower bounds (2) and (3), and can be used as a stopping criterion. Re-arranging (1) and using $R \geq \|x^{(1)} - x^*\|_2$. we get

$$f^* \geq l_k = \frac{2 \sum_{i=1}^k \alpha_i f(x^{(i)}) - R^2 - \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2}{2 \sum_{i=1}^k \alpha_i}, \quad (4)$$

which can be computed after the k th step. The sequence l_1, l_2, \dots need not increase, so we can keep track of the best lower bound on f^* found so far,

$$l_{\text{best}}^{(k)} = \max\{l_1, \dots, l_k\}.$$

We can terminate the algorithm when $f_{\text{best}}^{(k)} - l_{\text{best}}^{(k)}$ is smaller than some threshold.

This bound is better than (3), and doesn't depend on G , but it too goes to zero very slowly. For this reason, the subgradient method is usually used without any formal stopping criterion.

3.5 Numerical example

We consider the problem of minimizing a piecewise linear function:

$$\text{minimize } f(x) = \max_{i=1, \dots, m} (a_i^T x + b_i),$$

with variable $x \in \mathbf{R}^n$. Of course this problem is readily (and efficiently) solved via linear programming.

Finding a subgradient of f is easy: given x , we first find an index j for which

$$a_j^T x + b_j = \max_{i=1, \dots, m} (a_i^T x + b_i).$$

Then we can take as subgradient $g = a_j$. We can take $G = \max_{i=1, \dots, m} \|a_i\|_2$.

We illustrate the subgradient method with a specific problem instance with $n = 20$ variables and $m = 100$ terms, with problem data a_i and b_i generated from a unit normal distribution. We start with $x^{(1)} = 0$. There is no simple way to find a justifiable value for R (*i.e.*, a value of R for which we can prove that $\|x^{(1)} - x^*\|_2 \leq R$ holds) so we take $R = 10$. For our particular problem instance, it turns out that $\|x^{(1)} - x^*\|_2 = 0.91$, where we computed an optimal point and the optimal value $f^* \approx 1.1$ using linear programming.

We first consider the constant step length rule $\alpha_k = \gamma/\|g^{(k)}\|_2$. Figure 1 shows convergence of $f_{\text{best}}^{(k)} - f^*$ for $\gamma = 0.05$, $\gamma = 0.01$, and $\gamma = 0.005$. The figure reveals a trade-off: larger γ gives faster convergence, but larger final suboptimality. Non-monotonicity of $f(x^{(k)})$ is clearly seen in figure 2, which shows $f^{(k)} - f^*$ for the first 100 iterations.

To illustrate the subgradient method with some diminishing step size rules, we consider the nonsummable diminishing step size rule $\alpha_k = 0.1/\sqrt{k}$, and the square summable but not summable step rule $\alpha_k = 1/k$. The convergence for these step size rules is plotted in figure 3.

These plots are fairly typical: The subgradient method is very slow. But what do you expect from an algorithm that is just a few lines of code, has no line search, and uses any subgradient? (And has a convergence proof that is also just a few lines long.) One of its advantages, apart from simplicity, is robustness. We'll see this very clearly when we study the stochastic subgradient method.

4 Alternating projections

In this section we describe a step length choice due to Polyak, and use it to derive some versions of the alternating projections method for finding a point in the intersection of convex sets.

4.1 Optimal step size choice when f^* is known

Polyak [Pol87] suggests a step size that can be used when the optimal value f^* is known, and is in some sense optimal. (You might imagine that f^* is rarely known, but we will see that's not the case.) The step size is

$$\alpha_k = \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2}. \quad (5)$$

To motivate this step size, imagine that

$$f(x^{(k)} - \alpha g^{(k)}) \approx f(x^{(k)}) + g^{(k)T} (x^{(k)} - \alpha g^{(k)} - x^{(k)}) = f(x^{(k)}) - \alpha g^{(k)T} g^{(k)}.$$

(This would be the case if α were small, and $g^{(k)} = \nabla f(x^{(k)})$.) Replacing the lefthand side with f^* and solving for α gives the step length above.

We can give another simple motivation for the step length (5). The subgradient method starts from the basic inequality

$$\|x^{(k+1)} - x^*\|_2^2 \leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k(f(x^{(k)}) - f^*) + \alpha_k^2\|g^{(k)}\|_2^2.$$

The step size (5) minimizes the righthand side.

To analyze convergence, we substitute the step size (5) into (1), to get

$$2 \sum_{i=1}^k \frac{(f(x^{(i)}) - f^*)^2}{\|g^{(i)}\|_2^2} \leq R^2 + \sum_{i=1}^k \frac{(f(x^{(i)}) - f^*)^2}{\|g^{(i)}\|_2^2},$$

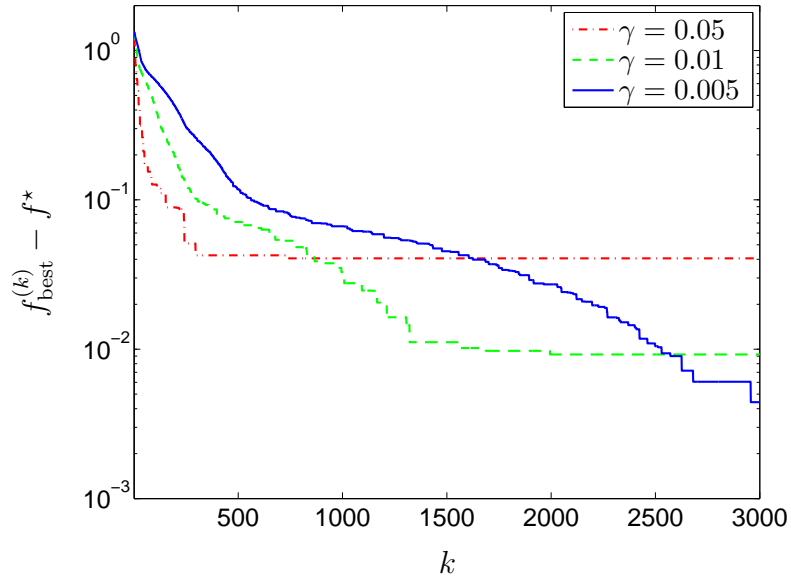


Figure 1: The value of $f_{\text{best}}^{(k)} - f^*$ versus iteration number k , for the subgradient method with constant step length γ .

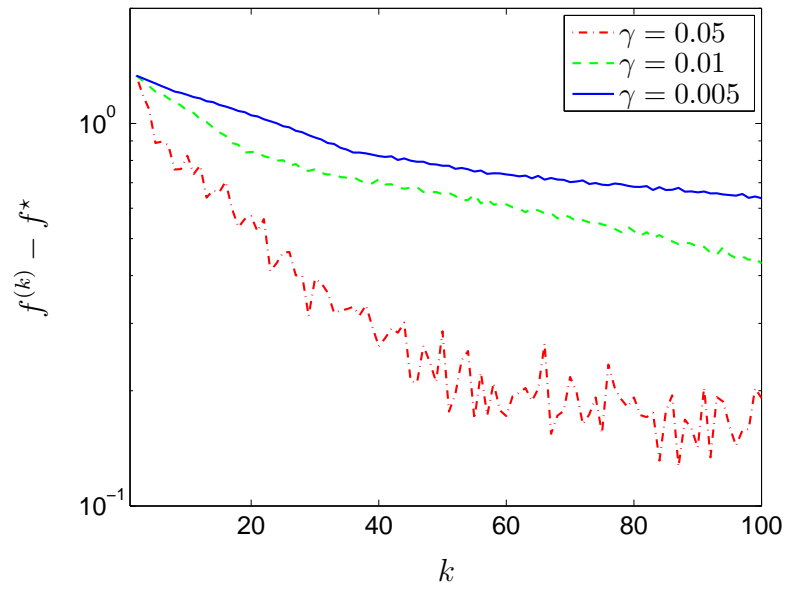


Figure 2: The value of $f^{(k)} - f^*$ versus iteration number k , for the subgradient method with constant step length γ .

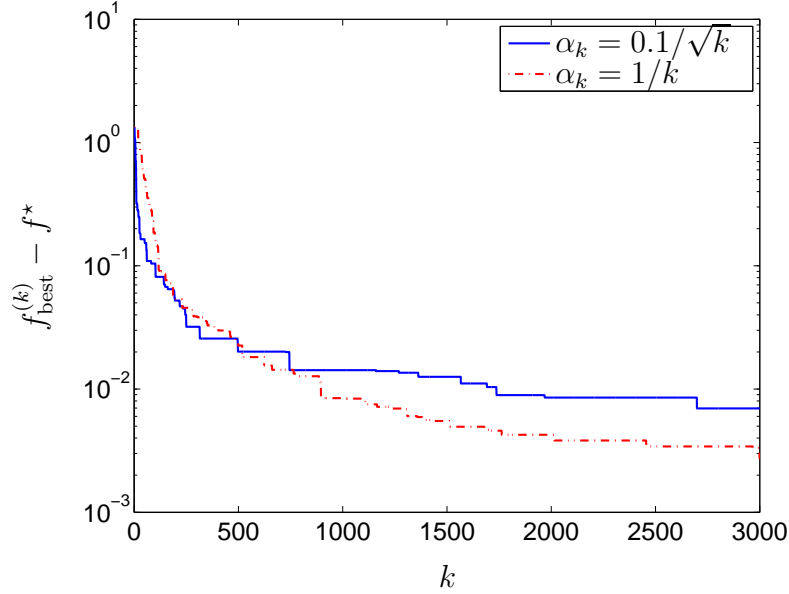


Figure 3: The value of $f_{\text{best}}^{(k)} - f^*$ versus iteration number k , for the subgradient method with diminishing step rule $\alpha_k = 0.1/\sqrt{k}$, and square summable step size rule $\alpha_k = 1/k$.

so

$$\sum_{i=1}^k \frac{(f(x^{(i)}) - f^*)^2}{\|g^{(i)}\|_2^2} \leq R^2.$$

Using $\|g^{(i)}\|_2 \leq G$ we get

$$\sum_{i=1}^k (f(x^{(i)}) - f^*)^2 \leq R^2 G^2.$$

We conclude that $f(x^{(k)}) \rightarrow f^*$. The number of steps needed before we can guarantee suboptimality ϵ is $k = (RG/\epsilon)^2$, which is optimal from our analysis above.

Figure 4 shows the progress of the subgradient method with Polyak's step size for the piecewise linear example from §3.5. Of course this isn't fair, since we don't know f^* before solving the problem. (However, we can estimate f^* .) But this plot shows that even with this unfair advantage in choosing step lengths, the subgradient method is pretty slow.

4.2 Finding a point in the intersection of convex sets

Suppose we want to find a point in

$$C = C_1 \cap \cdots \cap C_m,$$

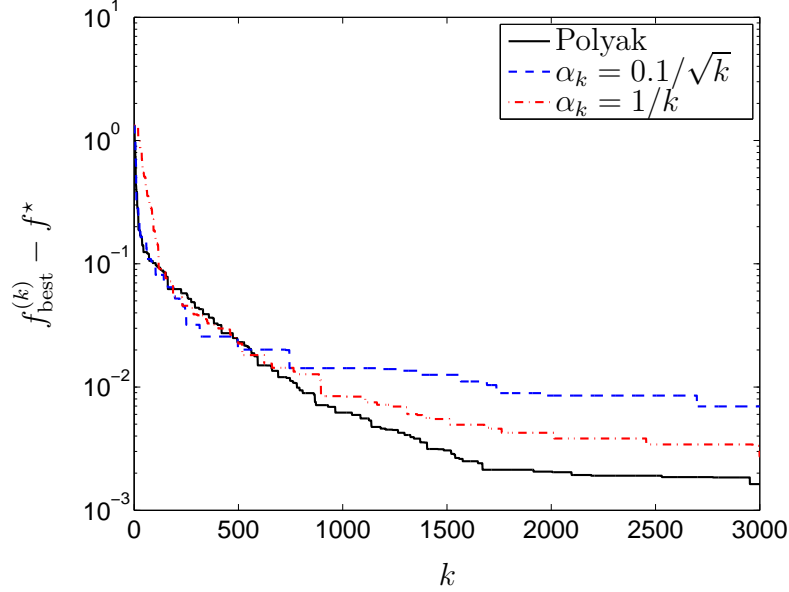


Figure 4: The value of $f_{\text{best}}^{(k)} - f^*$ versus iteration number k , for the subgradient method with Polyak's step size (solid black line) and the subgradient methods with diminishing step sizes considered in the previous example (dashed lines).

where $C_1, \dots, C_m \subseteq \mathbf{R}^n$ are closed and convex, and we assume that C is nonempty. We can do this by minimizing the function

$$f(x) = \max\{\mathbf{dist}(x, C_1), \dots, \mathbf{dist}(x, C_m)\},$$

which is convex, and has minimum value $f^* = 0$ (since C is nonempty).

We first explain how to find a subgradient g of f at x . If $f(x) = 0$, we can take $g = 0$ (which in any case means we are done). Otherwise find an index j such that $\mathbf{dist}(x, C_j) = f(x)$, *i.e.*, find a set that has maximum distance to x . A subgradient of f is

$$g = \nabla \mathbf{dist}(x, C_j) = \frac{x - P_{C_j}(x)}{\|x - P_{C_j}(x)\|_2},$$

where P_{C_j} is Euclidean projection onto C_j . Note that $\|g\|_2 = 1$, so we can take $G = 1$.

The subgradient algorithm update, with step size rule (5), and assuming that the index j is one for which $x^{(k)}$ has maximum distance to C_j , is given by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \alpha_k g^{(k)} \\ &= x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - P_{C_j}(x^{(k)})}{\|x^{(k)} - P_{C_j}(x^{(k)})\|_2} \\ &= P_{C_j}(x^{(k)}). \end{aligned}$$

Here we use $\|g^{(k)}\|_2 = 1$ and $f^* = 0$ in the second line, and

$$f(x^{(k)}) = \mathbf{dist}(x^{(k)}, C_j) = \|x^{(k)} - P_{C_j}(x^{(k)})\|_2$$

in the third line.

The algorithm is very simple: at each step, we simply project the current point onto the farthest set. This is an extension of the famous *alternating projections* algorithm. (When there are just two sets, then at each step you project the current point onto the *other* set. Thus the projections simply alternate.)

We are only guaranteed that $f(x^{(k)}) \rightarrow f^* = 0$. In other words, a subsequence of our points approaches a point in C ; we are not guaranteed to actually find a point in C (except in the limit). This can be addressed several ways. One way is to run the algorithm using closed sets $\tilde{C}_i \subseteq \mathbf{int} C_i$, so that $x^{(k)} \rightarrow \tilde{C} = \tilde{C}_1 \cap \dots \cap \tilde{C}_m$. Then we are guaranteed that $x^{(k)} \in C$ for some (finite) k .

Another method is to do *over-projection* at each step. Suppose we know the intersection of the sets contains a Euclidean ball of radius ϵ . Its center is a point that is ϵ -deep in all the sets. Then we can over project by ϵ , which roughly speaking means we project the current point to the farthest set, and then keep moving a distance ϵ :

$$x^{(k+1)} = P_{C_j}(x^{(k)}) - \epsilon \frac{x^{(k)} - P_{C_j}(x^{(k)})}{\|x^{(k)} - P_{C_j}(x^{(k)})\|_2}.$$

Alternating projections is usually (but not always) applied when projection onto the sets is simple. This is the case, for example, for the following sets.

- Affine set.
- Nonnegative orthant.
- A halfspace or slab.
- A box, *e.g.*, unit ball in ℓ_∞ .
- Unit simplex.
- A Euclidean ball.
- An ellipsoid (there is no closed-form expression for the projection, but it can be computed very quickly.)
- A second-order cone.
- Cone of positive semidefinite matrices.
- Spectral norm matrix ball.

Alternating projections can be used, of course, in cases where a bit more computation is needed to compute the Euclidean projection, *e.g.*, for a polyhedron (which can be done by solving a QP).

4.3 Solving convex inequalities

We want to find a point that satisfies $f_i(x) \leq 0$, $i = 1, \dots, m$. (We assume we can find a subgradient of each function, at any point.)

To solve this set of convex inequalities, we can minimize the unconstrained function $f(x) = \max_i f_i(x)$ using the subgradient method. If the set of inequalities is strictly feasible, then f^* is negative, and in a finite number of steps we'll find a point with $f(x) \leq 0$, *i.e.*, a feasible point.

We can also use the step size that uses knowledge of the optimal value, applied to the function

$$f(x) = \max\{f_1(x), \dots, f_m(x), -\epsilon\},$$

where $\epsilon > 0$ is a tolerance. Assuming there exists a point with $f_i(x) \leq -\epsilon$, we can use the step length

$$\alpha = \frac{f(x) + \epsilon}{\|g\|_2^2}. \quad (6)$$

We can give a simple interpretation of this step length, taking the case $\epsilon = 0$ for simplicity. Suppose the current point is x , and that $f_i(x) = f(x) > 0$, with $g \in \partial f_i(x)$. Let x^* be any point with $f_i(x^*) \leq 0$. Then we have

$$0 \geq f_i(x^*) \geq f_i(x) + g^T(x^* - x),$$

i.e., x^* is in the halfspace

$$\mathcal{H} = \{z \mid 0 \geq f_i(x) + g^T(z - x)\}.$$

The subgradient update at x , using Polyak's step length, is just projection of x onto the halfspace \mathcal{H} .

As an example we consider finding a point $x \in \mathbf{R}^n$ that satisfies a set of linear inequalities $a_i^T x \leq b_i$, $i = 1, \dots, m$. With $\epsilon = 0$, the subgradient method is very simple: at each step, we find the most violated inequality. Then we project the current point onto the set (halfspace) of points that satisfy this particular inequality:

$$x^{(k+1)} = x^{(k)} - \frac{a_i^T x - b_i}{\|a_i\|_2^2} a_i,$$

where i is the index of the most violated inequality at $x^{(k)}$.

We take a problem instance with $n = 100$ variables and $m = 1000$ inequalities, and randomly generate the data, making sure that the set of inequalities is feasible. We use the step size rule (6) with three different values of ϵ . Figure 5 shows the convergence for $\epsilon = 0$, $\epsilon = 0.01$, and $\epsilon = 0.1$. (We terminate the algorithm when we find a point that satisfies the inequalities.)

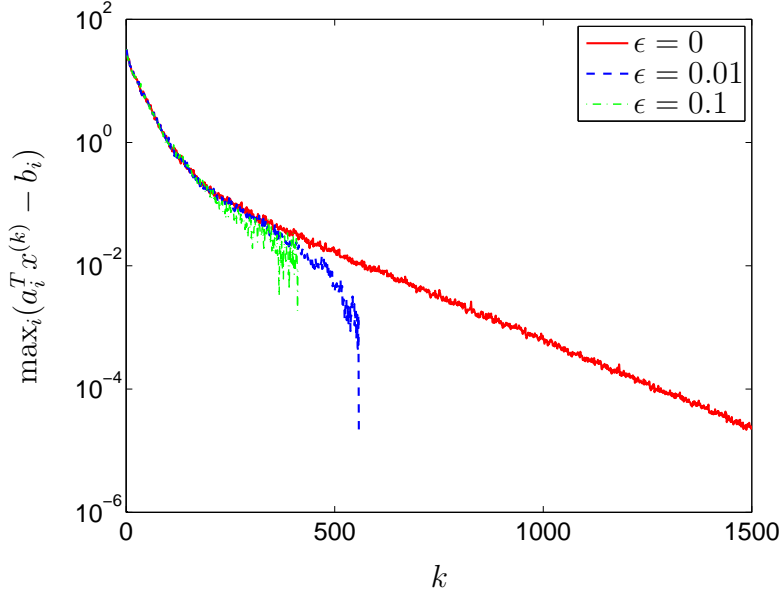


Figure 5: Convergence of the maximum violation for the linear feasibility problem, where we use the subgradient method with Polyak’s step size and three different values of tolerance ϵ .

4.4 Positive semidefinite matrix completion

We use the subgradient method with step size (5) to solve the *positive semidefinite matrix completion problem* (see [BV04, exer. 4.47]). We briefly describe the problem. Suppose we have a matrix in \mathbf{S}^n with some of its entries (including all of its diagonal entries) fixed, and the others to be found. The goal is to find values for the other entries so that the (completed) matrix is positive semidefinite.

We use alternating projections onto the set of positive semidefinite matrices \mathbf{S}_+^n , and the set of matrices with the given fixed entries. (Projection is in the Frobenius norm sense.) The first projection can be found from the eigenvalue decomposition (see [BV04, §8.1.1]); for example, let $X = \sum_{i=1}^n \lambda_i q_i q_i^T$, then

$$P(X) = \sum_{i=1}^n \max\{0, \lambda_i\} q_i q_i^T.$$

The second projection is straightforward: we simply take the given matrix and set its fixed entries back to the given fixed values. Thus, the algorithm will alternate between eigenvalue decomposition and truncation, and re-setting the fixed entries back to their required values.

As a specific example we consider a randomly generated problem with a 50×50 matrix that is missing about half of its entries. The sparsity pattern of our particular matrix is shown in figure 6. We initialize $X^{(1)}$ by taking the unknown entries to be 0.

To track convergence of the algorithm, we plot the Frobenius norm of the difference

between the current matrix and its projection onto one of the sets, *i.e.*, $\|X^{(k+1)} - X^{(k)}\|_F$. In the case of the projection onto the set of positive semidefinite matrices, this value is the squareroot of the sum of the squares of the negative eigenvalues of $X^{(k)}$. In the case of the other projection, it is the squareroot of the sum of the squares of the adjustments made to the fixed entries of $X^{(k)}$. In each case, this distance gives an upper bound on the distance to the intersection of the two sets, *i.e.*, the distance to the nearest positive semidefinite completion. The plot is shown in figure 7. We can see that the unknown entries are converging to a positive semidefinite completion. By overprojecting onto \mathbf{S}_+^n , we could have found an actual positive semidefinite completion in a finite number of steps.

5 Projected subgradient method

One extension of the subgradient method is the *projected subgradient method*, which solves the constrained convex optimization problem

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && x \in \mathcal{C}, \end{aligned}$$

where \mathcal{C} is a convex set. The projected subgradient method is given by

$$x^{(k+1)} = P\left(x^{(k)} - \alpha_k g^{(k)}\right),$$

where P is (Euclidean) projection on \mathcal{C} , and $g^{(k)}$ is any subgradient of f at $x^{(k)}$. The step size rules described before can be used here, with similar convergence results. Note that $x^{(k)} \in \mathcal{C}$, *i.e.*, $x^{(k)}$ is feasible.

The convergence proofs for the subgradient method are readily extended to handle the projected subgradient method. Let $z^{(k+1)} = x^{(k)} - \alpha_k g^{(k)}$, *i.e.*, a standard subgradient update, before the projection back onto \mathcal{C} . As in the subgradient method, we have

$$\begin{aligned} \|z^{(k+1)} - x^*\|_2^2 &= \|x^{(k)} - \alpha_k g^{(k)} - x^*\|_2^2 \\ &= \|x^{(k)} - x^*\|_2^2 - 2\alpha_k g^{(k)T}(x^{(k)} - x^*) + \alpha_k^2 \|g^{(k)}\|_2^2 \\ &\leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^*) + \alpha_k^2 \|g^{(k)}\|_2^2. \end{aligned}$$

Now we observe that

$$\|x^{(k+1)} - x^*\|_2 = \|P(z^{(k+1)}) - x^*\|_2 \leq \|z^{(k+1)} - x^*\|_2,$$

i.e., when we project a point onto \mathcal{C} , we move closer to every point in \mathcal{C} , and in particular, any optimal point. Combining this with the inequality above we get

$$\|x^{(k+1)} - x^*\|_2^2 \leq \|x^{(k)} - x^*\|_2^2 - 2\alpha_k (f(x^{(k)}) - f^*) + \alpha_k^2 \|g^{(k)}\|_2^2,$$

and the proof proceeds exactly as in the ordinary subgradient method.

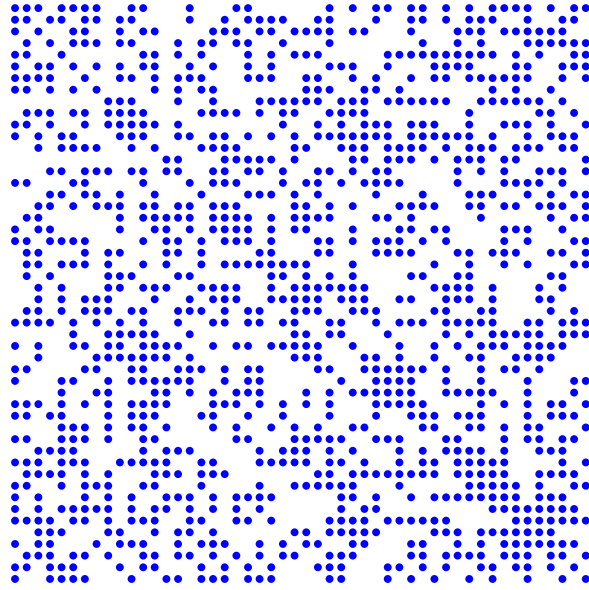


Figure 6: Sparsity pattern of the given matrix with blue entries corresponding to fixed values and white entries corresponding to missing values.

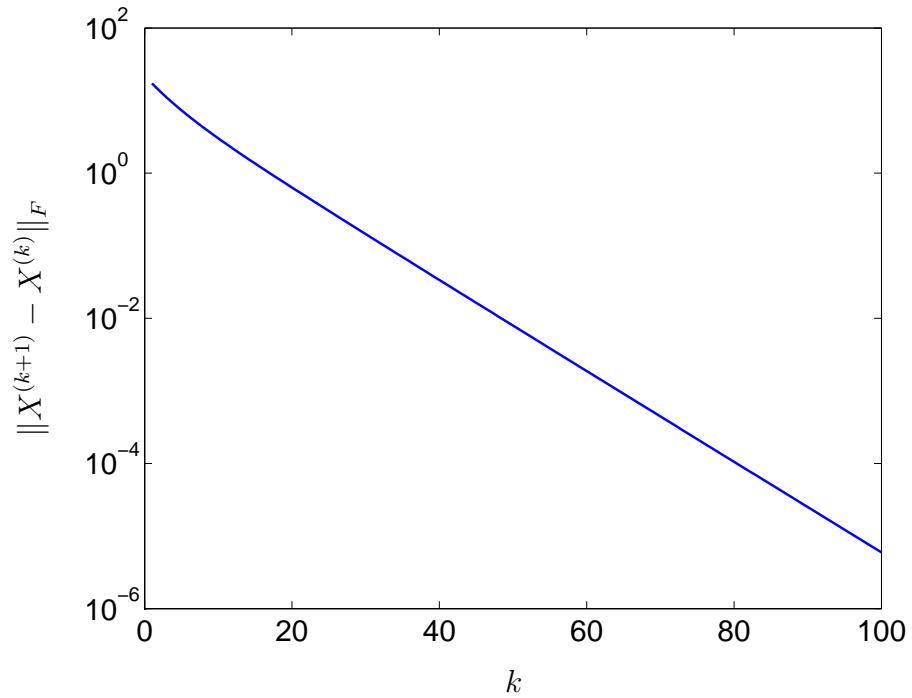


Figure 7: Convergence of the subgradient method for a matrix completion problem.

In some cases we can express the projected subgradient update in an alternative way. When \mathcal{C} is affine, *i.e.*, $\mathcal{C} = \{x \mid Ax = b\}$, where A is fat and full rank, the projection operator is affine, and given by

$$P(z) = z - A^T(AA^T)^{-1}(Az - b).$$

In this case, we can simplify the subgradient update to

$$x^{(k+1)} = x^{(k)} - \alpha_k(I - A^T(AA^T)^{-1}A)g^{(k)}, \quad (7)$$

where we use $Ax^{(k)} = b$. Thus, we simply project the current subgradient onto the nullspace of A , and then update as usual. The update (7) is not the same as the projected subgradient update when \mathcal{C} is not affine, because in this case the projection operator is not affine.

5.1 Numerical example

We consider the least l_1 -norm problem

$$\begin{aligned} & \text{minimize} && \|x\|_1 \\ & \text{subject to} && Ax = b, \end{aligned} \quad (8)$$

where the variable is $x \in \mathbf{R}^n$, and the data are $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^m$. We assume that A is fat and full rank, *i.e.*, $m < n$ and $\mathbf{Rank} A = m$. Of course, this problem is readily solved using linear programming.

A subgradient of the objective at x is given by $g = \mathbf{sign}(x)$. Thus, the projected subgradient update is

$$x^{(k+1)} = x^{(k)} - \alpha_k(I - A^T(AA^T)^{-1}A)\mathbf{sign}(x^{(k)}).$$

We consider an instance of the problem (8) with $n = 1000$ and $m = 50$, with randomly generated A and b . We use the least-norm solution as the starting point, *i.e.*, $x^{(1)} = A^T(AA^T)^{-1}b$. In order to report $f_{\text{best}}^{(k)} - f^*$, we solve the problem using linear programming and obtain $f^* \approx 3.2$. Figure 8 shows the progress of the projected subgradient method with square summable step size rule $\alpha_k = 0.1/k$.

6 Projected subgradient for dual problem

One famous application of the projected subgradient method is to the dual problem. We start with the (convex) primal problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \end{aligned}$$

We'll assume, for simplicity, that for each $\lambda \succeq 0$, the Lagrangian

$$L(x, \lambda) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x)$$

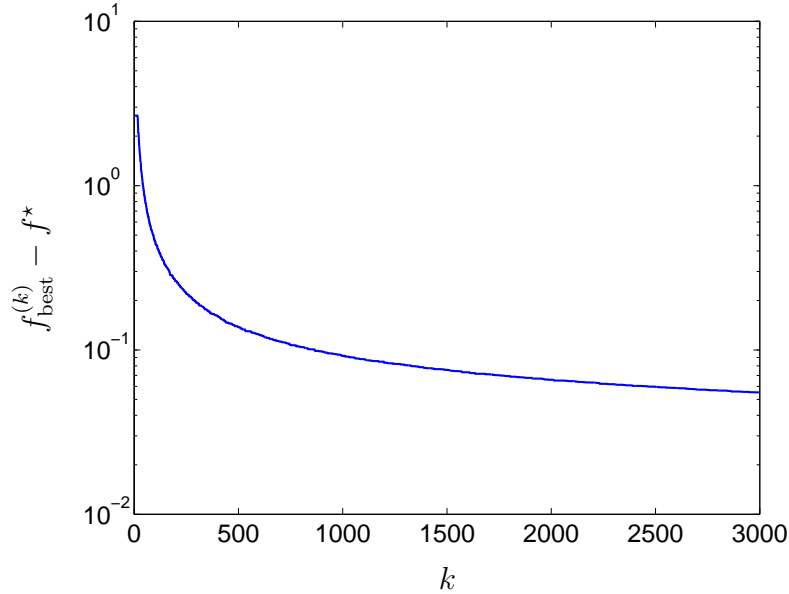


Figure 8: The value of $f_{\text{best}}^{(k)} - f^*$ versus iteration number k , for the subgradient method with square summable step size rule $\alpha_k = 0.1/k$.

has a unique minimizer over x , which we denote $x^*(\lambda)$. The dual function is then

$$g(\lambda) = \inf_x L(x, \lambda) = f_0(x^*(\lambda)) + \sum_{i=1}^m \lambda_i f_i(x^*(\lambda))$$

(for $\lambda \succeq 0$). The dual problem is

$$\begin{aligned} & \text{maximize} && g(\lambda) \\ & \text{subject to} && \lambda \succeq 0. \end{aligned}$$

We'll assume that Slater's condition holds (again, for simplicity), so we can solve the primal problem by finding an optimal point λ^* of the dual, and then taking $x^* = x^*(\lambda^*)$. (For a discussion of solving the primal problem via the dual, see [BV04, §5.5.5].)

We will solve the dual problem using the projected subgradient method,

$$\lambda^{(k+1)} = \left(\lambda^{(k)} - \alpha_k h \right)_+, \quad h \in \partial(-g)(\lambda^{(k)}).$$

Let's now work out a subgradient of the negative dual function. Since $-g$ is a supremum of a family of affine functions of λ , indexed by x , we can find a subgradient by finding one of these functions that achieves the supremum. But there is just one, and it is

$$-f_0(x^*(\lambda)) - \sum_{i=1}^m \lambda_i f_i(x^*(\lambda)),$$

which has gradient (with respect to λ)

$$h = -(f_1(x^*(\lambda)), \dots, f_m(x^*(\lambda))) \in \partial(-g)(\lambda).$$

(Our assumptions imply that $-g$ has only one element in its subdifferential, which means g is differentiable. Differentiability means that a small enough constant step size will yield convergence. In any case, the projected subgradient method can be used in cases where the dual is nondifferentiable.)

The projected subgradient method for the dual has the form

$$x^{(k)} = x^*(\lambda^{(k)}), \quad \lambda_i^{(k+1)} = \left(\lambda_i^{(k)} + \alpha_k f_i(x^{(k)}) \right)_+ \quad (9)$$

In this algorithm, the primal iterates $x^{(k)}$ are not feasible, but become feasible only in the limit. (Sometimes we can find a method for constructing a feasible, suboptimal $\tilde{x}^{(k)}$ from $x^{(k)}$.) The dual function values $g(\lambda^{(k)})$, as well as the primal function values $f_0(x^{(k)})$, converge to $f^* = f_0(x^*)$.

We can give a simple interpretation of the algorithm (9). We interpret λ_i as the price for a ‘resource’ with usage measured by $f_i(x)$. When we calculate $x^*(\lambda)$, we are finding the x that minimizes the total cost, *i.e.*, the objective plus the total bill (or revenue) for the resources used. The goal is to adjust the prices so that the resource usage is within budget (*i.e.*, $f_i(x) \leq 0$). At each step, we increase the price λ_i if resource i is over-utilized (*i.e.*, $f_i(x) > 0$), and we decrease the price λ_i if resource i is under-utilized (*i.e.*, $f_i(x) < 0$). But we never let prices get negative (which would encourage, rather than discourage, resource usage).

In general, there is no reason to solve the dual instead of the primal. But for specific problems there can be an advantage. We will see later that the projected subgradient dual algorithm (9) is, in some cases, a *decentralized* algorithm.

6.1 Numerical example

We consider the problem of minimizing a strictly convex quadratic function over the unit box:

$$\begin{aligned} & \text{minimize} && (1/2)x^T P x - q^T x \\ & \text{subject to} && x_i^2 \leq 1, \quad i = 1, \dots, n, \end{aligned}$$

where $P \succ 0$. The Lagrangian is

$$L(x, \lambda) = (1/2)x^T (P + \mathbf{diag}(2\lambda))x - q^T x - \mathbf{1}^T \lambda,$$

so $x^*(\lambda) = (P + \mathbf{diag}(2\lambda))^{-1}q$. The projected subgradient algorithm for the dual is

$$x^{(k)} = (P + \mathbf{diag}(2\lambda^{(k)}))^{-1}q, \quad \lambda_i^{(k+1)} = \left(\lambda_i^{(k)} + \alpha_k ((x^{(k)})^2 - 1) \right)_+.$$

The dual function is differentiable, so we can use a fixed size α (provided it is small enough).

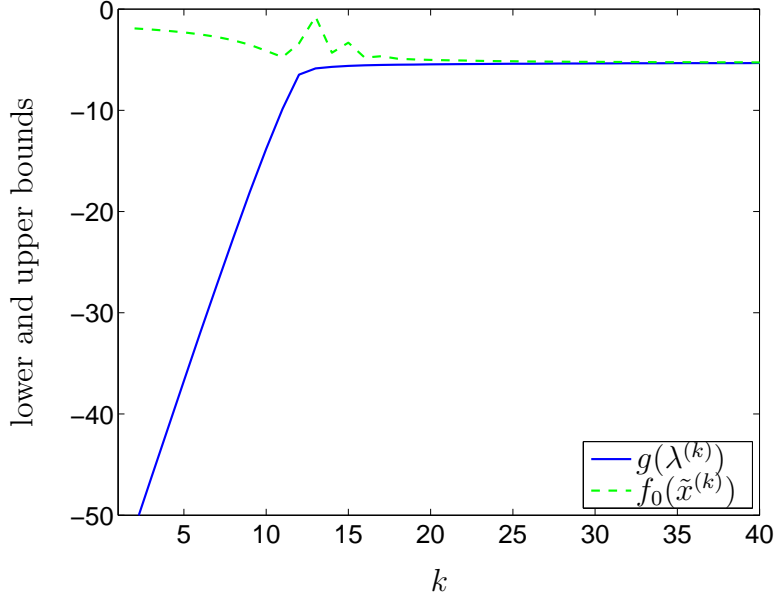


Figure 9: The values of the lower bound $g(\lambda^{(k)})$ and the upper bound $f_0(\tilde{x}^{(k)})$, versus the iteration number k . We use the fixed step size with $\alpha = 0.1$.

The iterates $x^{(k)}$ are not feasible. But we can construct a nearby feasible $\tilde{x}^{(k)}$ as

$$\tilde{x}_i^{(k)} = \begin{cases} 1 & x_i^{(k)} > 1 \\ -1 & x_i^{(k)} < -1 \\ x_i^{(k)} & -1 \leq x_i^{(k)} \leq 1. \end{cases}$$

We consider an instance with $n = 50$. We start the algorithm with $\lambda^{(1)} = 1$, and use a fixed step size $\alpha = 0.1$. Figure 9 shows the convergence of $g(\lambda^{(k)})$ (a lower bound on the optimal value) and $f_0(\tilde{x}^{(k)})$ (an upper bound on the optimal value), versus iterations.

7 Subgradient method for constrained optimization

The subgradient algorithm can be extended to solve the inequality constrained problem

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m, \end{aligned}$$

where f_i are convex. The algorithm takes the same form:

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)},$$

where $\alpha_k > 0$ is a step size, and $g^{(k)}$ is a subgradient of the objective or one of the constraint functions at $x^{(k)}$. More specifically, we take

$$g^{(k)} \in \begin{cases} \partial f_0(x^{(k)}) & f_i(x^{(k)}) \leq 0, \quad i = 1, \dots, m, \\ \partial f_j(x^{(k)}) & f_j(x^{(k)}) > 0. \end{cases}$$

In other words: If the current point is feasible, we use an objective subgradient, as if the problem were unconstrained; If the current point is infeasible, we choose any violated constraint, and use a subgradient of the associated constraint function. (In the latter case, we can choose *any* of the violated constraints, if there is more than one.)

In this generalized version of the subgradient algorithm, the iterates can be (and often are) infeasible. In contrast, the iterates of the projected subgradient method (and of course, the basic subgradient algorithm) are always feasible.

As in the basic subgradient method, we keep track of the best (feasible) point found so far:

$$f_{\text{best}}^{(k)} = \min\{f_0(x^{(i)}) \mid x^{(i)} \text{ feasible}, i = 1, \dots, k\}.$$

(If none of the points $x^{(1)}, \dots, x^{(k)}$ is feasible, then $f_{\text{best}}^{(k)} = \infty$.)

We assume that Slater's condition holds, *i.e.*, the problem is strictly feasible: there is some point x^{sf} with $f_i(x^{\text{sf}}) < 0$, $i = 1, \dots, m$. We also assume that the problem has an optimal point x^* . We assume that there are numbers R and G with $\|x^{(1)} - x^*\|_2 \leq R$, $\|x^{(1)} - x^{\text{sf}}\|_2 \leq R$, and $\|g^{(k)}\|_2 \leq G$ for all k .

We'll establish convergence of the generalized subgradient method using diminishing non-summable α_k . (Similar results can be obtained for other step size rules.) We claim that $f_{\text{best}}^{(k)} \rightarrow f^*$ as $k \rightarrow \infty$. This implies in particular that we obtain a feasible iterate within some finite number of steps.

Assume that $f_{\text{best}}^{(k)} \rightarrow f^*$ does not occur. Then there exists some $\epsilon > 0$ so that $f_{\text{best}}^{(k)} \geq f^* + \epsilon$ for all k , which in turn means that $f(x^{(k)}) \geq f^* + \epsilon$ for all k for which $x^{(k)}$ is feasible. We'll show this leads to a contradiction.

We first find a point \tilde{x} and positive number μ that satisfy

$$f_0(\tilde{x}) \leq f^* + \epsilon/2, \quad f_1(\tilde{x}) \leq -\mu, \dots, f_m(\tilde{x}) \leq -\mu.$$

Such a point is $\epsilon/2$ -suboptimal, and also satisfies the constraints with a margin of μ . We will take $\tilde{x} = (1 - \theta)x^* + \theta x^{\text{sf}}$, where $\theta \in (0, 1)$. We have

$$f_0(\tilde{x}) \leq (1 - \theta)f^* + \theta f_0(x^{\text{sf}}),$$

so if we choose $\theta = \min\{1, (\epsilon/2)/(f_0(x^{\text{sf}}) - f^*)\}$, we have $f_0(\tilde{x}) \leq f^* + \epsilon/2$. We have

$$f_i(\tilde{x}) \leq (1 - \theta)f_i(x^*) + \theta f_i(x^{\text{sf}}) \leq \theta f_i(x^{\text{sf}}),$$

so we can take

$$\mu = -\theta \min_i f_i(x^{\text{sf}}).$$

Consider any index $i \in \{1, \dots, k\}$ for which $x^{(i)}$ is feasible. Then we have $g^{(i)} \in \partial f_0(x^{(i)})$, and also $f_0(x^{(i)}) \geq f^* + \epsilon$. Since \tilde{x} is $\epsilon/2$ -suboptimal, we have $f_0(x^{(i)}) - f_0(\tilde{x}) \geq \epsilon/2$. Therefore

$$\begin{aligned} \|x^{(i+1)} - \tilde{x}\|_2^2 &= \|x^{(i)} - \tilde{x}\|_2^2 - 2\alpha_i g^{(i)T}(x^{(i)} - \tilde{x}) + \alpha_i^2 \|g^{(i)}\|_2^2 \\ &\leq \|x^{(i)} - \tilde{x}\|_2^2 - 2\alpha_i (f_0(x^{(i)}) - f_0(\tilde{x})) + \alpha_i^2 \|g^{(i)}\|_2^2 \\ &\leq \|x^{(i)} - \tilde{x}\|_2^2 - \alpha_i \epsilon + \alpha_i^2 \|g^{(i)}\|_2^2. \end{aligned}$$

In the second line here we use the usual subgradient inequality

$$f_0(\tilde{x}) \geq f_0(x^{(i)}) + g^{(i)T}(\tilde{x} - x^{(i)}).$$

Now suppose that $i \in \{1, \dots, k\}$ is such that $x^{(i)}$ is infeasible, and that $g^{(i)} \in \partial f_p(x^{(i)})$, where $f_p(x^{(i)}) > 0$. Since $f_p(\tilde{x}) \leq -\mu$, we have $f_p(x^{(i)}) - f_p(\tilde{x}) \geq \mu$. Therefore

$$\begin{aligned} \|x^{(i+1)} - \tilde{x}\|_2^2 &= \|x^{(i)} - \tilde{x}\|_2^2 - 2\alpha_i g^{(i)T}(x^{(i)} - \tilde{x}) + \alpha_i^2 \|g^{(i)}\|_2^2 \\ &\leq \|x^{(i)} - \tilde{x}\|_2^2 - 2\alpha_i (f_p(x^{(i)}) - f_p(\tilde{x})) + \alpha_i^2 \|g^{(i)}\|_2^2 \\ &\leq \|x^{(i)} - \tilde{x}\|_2^2 - 2\alpha_i \mu + \alpha_i^2 \|g^{(i)}\|_2^2. \end{aligned}$$

Thus, for every iteration we have

$$\|x^{(i+1)} - \tilde{x}\|_2^2 \leq \|x^{(i)} - \tilde{x}\|_2^2 - \alpha_i \delta + \alpha_i^2 \|g^{(i)}\|_2^2,$$

where $\delta = \min\{\epsilon, 2\mu\} > 0$. Applying this inequality recursively for $i = 1, \dots, k$, we get

$$\|x^{(k+1)} - \tilde{x}\|_2^2 \leq \|x^{(1)} - \tilde{x}\|_2^2 - \delta \sum_{i=1}^k \alpha_i + \sum_{i=1}^k \alpha_i^2 \|g^{(i)}\|_2^2.$$

It follows that

$$\delta \sum_{i=1}^k \alpha_i \leq R^2 + G^2 \sum_{i=1}^k \alpha_i^2,$$

which cannot hold for large k since

$$\frac{R^2 + G^2 \sum_{i=1}^k \alpha_i^2}{\sum_{i=1}^k \alpha_i}$$

converges to zero as $k \rightarrow \infty$.

There are many variations on the basic step size rule. For example, when the current point is infeasible, we can use an over-projection step length, as we would when solving convex inequalities. If we know (or estimate) f^* , we can use Polyak's step length when the current point is feasible. Thus our step lengths are chosen as

$$\alpha_k = \begin{cases} (f_0(x^{(k)}) - f^*) / \|g^{(k)}\|_2^2 & x^{(k)} \text{ feasible} \\ (f_i(x^{(k)}) + \epsilon) / \|g^{(k)}\|_2^2 & x^{(k)} \text{ infeasible} \end{cases}$$

where ϵ is a small positive margin, and i is the index of the most violated inequality in the case when $x^{(k)}$ is infeasible.

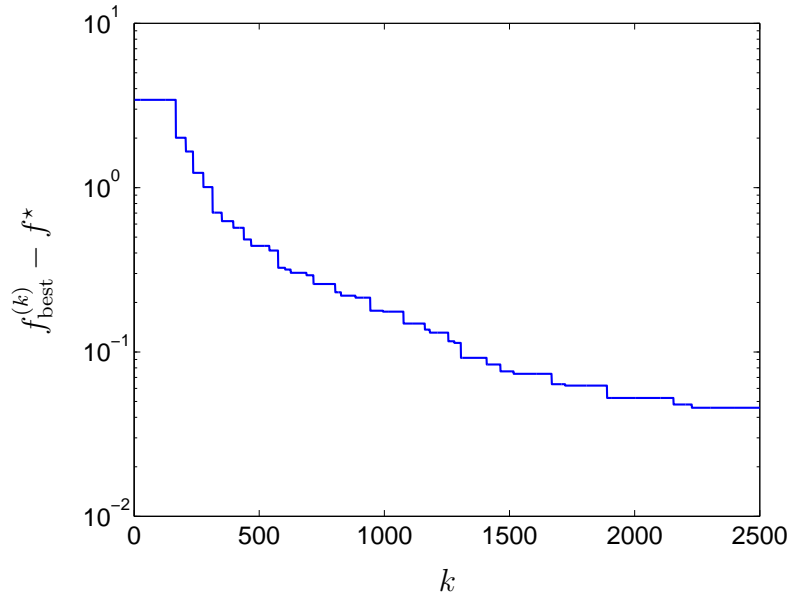


Figure 10: The value of $f_{\text{best}}^{(k)} - f^*$ versus the iteration number k . In this case, we use the square summable step size with $\alpha_k = 1/k$ for the optimality update.

7.1 Numerical example

We consider a linear program

$$\begin{aligned} & \text{minimize} && c^T x \\ & \text{subject to} && a_i^T x \leq b_i, \quad i = 1, \dots, m, \end{aligned} \tag{10}$$

with variable $x \in \mathbf{R}^n$. The objective and constraint functions are affine, and so have only one subgradient, independent of x . For the objective function we have $g = c$, and for the i th constraint we have $g_i = a_i$.

We solve an instance of the problem (10) with $n = 20$ and $m = 200$ using the subgradient method. In order to report $f_{\text{best}}^{(k)} - f^*$, we solve the LP using the interior-point methods and obtain $f^* \approx -3.4$. Figure 10 shows progress of the subgradient method, which uses the square summable step size with $\alpha_k = 1/k$ for the optimality update, and the step size (6) with $\epsilon = 10^{-3}$ for the feasibility update. The objective value only changes for the iterations when $x^{(k)}$ is feasible.

8 Speeding up subgradient methods

Several general approaches can be used to speed up subgradient methods. *Localization methods* such as cutting-plane and ellipsoid methods also require the evaluation of one subgradient per iteration, but require more computation to carry out the update. They are typically much

faster than subgradient methods. Some of these methods have real (non-heuristic) stopping criteria.

Another general approach is to base the update on some conic combination of previously evaluated subgradients. In *bundle methods*, the update direction is found as the least-norm convex combination of some ('bundle' of) previous subgradients. (This gives an approximation of the steepest descent direction.)

One general class of methods uses an update direction that is a conic combination of the current negative subgradient and the last search direction, as in

$$x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)} + \beta_k (x^{(k)} - x^{(k-1)})$$

where α_k and β_k are positive. (There are many other ways to express this update.) Such algorithms have state, whereas the basic subgradient method is stateless (except for the iteration number). We can interpret the second term as a memory term, or as a momentum term, in the algorithm. Polyak refers to some algorithms of this form as the *heavy ball method*. *Conjugate gradients* methods have a similar form.

We describe two examples of these types of methods, that use a known (or estimated) value of f^* to determine step lengths. Each has an update of the form

$$x^{(k+1)} = x^{(k)} - \alpha_k s^{(k)}, \quad \alpha_k = \frac{f(x^{(k)}) - f^*}{\|s^{(k)}\|_2^2},$$

where $s^{(k)}$ is a direction to be used in place of a subgradient. In the simple method, $s^{(k)}$ is just a filtered, or smoothed, version of the subgradients:

$$s^{(k)} = (1 - \beta)g^{(k)} + \beta s^{(k-1)},$$

where $0 \leq \beta < 1$ is a (constant) filter parameter that controls how much memory the algorithm has. When $\beta = 0$ we obtain the subgradient method with Polyak's step size.

A more sophisticated method for updating $s^{(k)}$ was proposed by Camerini, Fratta, and Maffioli [CFM75]. Their algorithm has the form

$$s^{(k)} = g^{(k)} + \beta_k s^{(k-1)}, \tag{11}$$

where

$$\beta_k = \max\{0, -\gamma_k (s^{(k-1)})^T g^{(k)} / \|s^{(k-1)}\|_2^2\}.$$

Here $\gamma_k \in [0, 2]$; they recommend using the constant value $\gamma_k = 1.5$.

They show that

$$\frac{(x^{(k)} - x^*)^T s^{(k)}}{\|s^{(k)}\|_2^2} \geq \frac{(x^{(k)} - x^*)^T g^{(k)}}{\|g^{(k)}\|_2^2},$$

i.e., the direction with modified update has a smaller angle towards the optimal set than the negative subgradient. (It follows that the convergence proofs for the subgradient algorithm work for this one as well.)

To illustrate these acceleration techniques, we consider again our piecewise-linear minimization example. We use the CFM algorithm and its simpler update rule given above for $\beta = 0$ and $\beta = 0.25$. Figure 11 shows the progress of these algorithms.

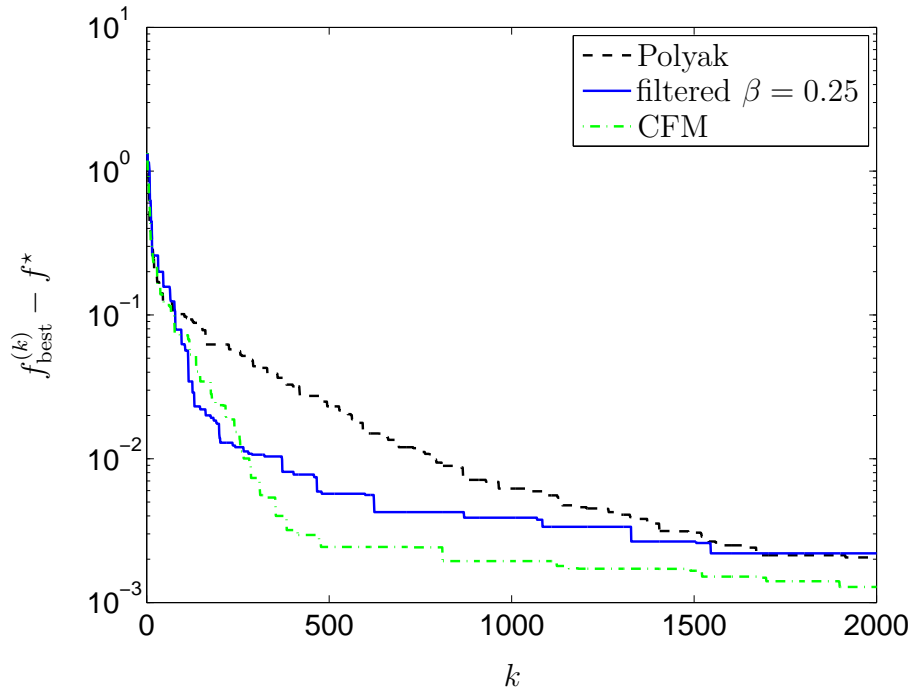


Figure 11: The value of $f_{\text{best}}^{(k)} - f^*$ versus iteration number k , for the subgradient method with two types of Polyak's step sizes, the original update when $\beta = 0$ (dashed black line) and a filtered update with $\beta = 0.25$ (solid blue line). The plot also shows the subgradient method with CFM step size (dash dotted green line).

Acknowledgments

We thank Lieven Vandenberghé and Lin Xiao, who helped with an earlier version of these notes.

References

- [Akg84] M. Akgül. *Topics in Relaxation and Ellipsoidal Methods*, volume 97 of *Research Notes in Mathematics*. Pitman, 1984.
- [Ber99] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, second edition, 1999.
- [BV04] S. Boyd and L. Vandenberghé. *Convex Optimization*. Cambridge University Press, 2004.
- [CFM75] P. Camerini, L. Fratta, and F. Maffioli. On improving relaxation methods by modifying gradient techniques. *Math. Programming Study*, 3:26–34, 1975.
- [CZ97] Y. Censor and S. Zenios. *Parallel Optimization*. Oxford University Press, 1997.
- [NB01] A. Nedić and D. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM J. on Optimization*, 12:109–138, 2001.
- [Nes04] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers, 2004.
- [Nes05] Y. Nesterov. Primal-dual subgradient methods for convex problems. CORE Discussion Paper #2005/67. Available at www.core.ucl.ac.be/services/psfiles/dp05/dp2005_67.pdf, 2005.
- [NY83] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience, 1983.
- [Pol87] B. Polyak. *Introduction to Optimization*. Optimization Software, Inc., 1987.
- [Rus06] A. Ruszczyński. *Nonlinear Optimization*. Princeton University Press, 2006.
- [Sho85] N. Shor. *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics. Springer, 1985.
- [Sho98] N. Shor. *Nondifferentiable Optimization and Polynomial Problems*. Nonconvex Optimization and its Applications. Kluwer, 1998.