Chapter 6

Coordinate Descent Methods

Some of the earliest approaches for multivariable optimization proceeded by optimizing with respect to one variable at a time, and cycling repeatedly through the full set of variables. This approach, known as coordinate descent (CD) has a certain intuitive appeal: It replace the difficult problem of minimizing with respect to many variables with a sequence of simpler scalar optimization problems. There are many variants and extensions of the basic CD approach that have gone in and out of style over the years. Nowadays there is considerable interest, driven largely by the usefulness of CD methods in data analysis problems.

The kth of coordinate descent applied to a function $f : \mathbb{R}^n \to \mathbb{R}$ chooses some index $i_k \in \{1, 2, \ldots, n\}$, and takes a step of the form

$$x^{k+1} \leftarrow x^k + \gamma_k e_{i_k}, \tag{6.1}$$

where e_{i_k} is the i_k unit vector and γ_k is the step. In one CD variant, also known as the Gauss-Seidel method), γ_k is chosen to minimize f along direction e_{i_k}, that is,

$$\gamma_k := \arg\min_{\gamma} f(x^k + \gamma e_{i_k}).$$

In the most popular CD variants, γ_k is chosen to be a multiple of the negative partial gradient of with respect to x_{i_k} (denoted by $\nabla_{i_k} f$), that is,

$$x^{k+1} \leftarrow x^k - \alpha_k \nabla_{i_k} f(x^k) e_{i_k}, \tag{6.2}$$

for some $\alpha_k > 0$. Different variants of CD are distinguished by different techniques for choosing i_k and α_k. The common theme is that all are descent methods, that is, they ensure that $f(x^{k+1}) < f(x^k)$ for all k.

6.1 Coordinate Descent in Machine Learning

In deciding whether CD is a plausible approach for minimizing f, relative to alternative approaches such as the gradient methods of Chapters 3 and 4, we need to consider carefully the properties and structure of f. We note first that CD methods almost invariably make use of partial gradient information about f at each step. That is, if CD chooses to update component i_k of x at the current iteration k, it needs information about the gradient (or subgradient) of f with respect to

the component x_{i_k} to help decide how far to move along the i_k coordinate direction. If the cost of computing this single component of gradient information is not much less that computing the full gradient at x^k, there is no good reason to use CD; we are better off applying an algorithm that exploits more full the full-gradient information. However, there are important applications in machine learning and other areas, in which it is much less expensive to compute a single gradient component than a full gradient. Two such examples follow.

6.1.1 Coordinate Descent for Empirical Risk Minimization

Consider the objective that arises in regularized regression, classification, and ERM problems:

$$f(x) = \frac{1}{m}\sum_{j=1}^{m} \phi_j(A_{j\cdot}^T x) + \lambda \sum_{i=1}^{n} \Omega_i(x_i),$$

where each ϕ_j is a convex loss, $A_{j\cdot}$ denotes the jth row of the $m \times n$ matrix A, the functions Ω_i, $i = 1, 2, \ldots, n$ are convex component-wise regularization functions, and $\lambda \geq 0$ is a regularization parameter. (We assume that the functions ϕ_j and Ω_i are all differentiable, for now.) Although it is expensive to compute "from scratch" the ith component of the gradient ($\nabla_i f$), it is easy to maintain and update information from one iteration of CD to the next that make this task easy and inexpensive. The trick is to maintain in storage the vector $g = Ax$ for the current x, along with the scalars $\nabla\phi_j(g_j)$, $j = 1, 2, \ldots, m$. Using this information, the ith element of the gradient can be obtained from the following formula:

$$\nabla_i f(x) = \sum_{j=1}^{m} A_{j,i} \nabla\phi_j(g_j) + \lambda \nabla\Omega_i(x_i),$$

where $A_{j,i}$ denotes the (j, i) element of the matrix A. However, we note that the terms in the summation need be evaluated only for those indices j for which $A_{j,i}$ is nonzero, that is,

$$\nabla_i f(x) = \sum_{j:A_{i,j}\neq 0} A_{j,i} \nabla\phi_j(g_j) + \lambda \nabla\Omega_i(x_i).$$

When A is sparse, this computation can be performed cheaply, in $O(|A_{\cdot i}|)$ operations, where $A_{\cdot i}$ is the ith column of A. (The number of operations required to compute the full gradient would be proportional to the number of nonzeros in the full matrix A.) However, we still need to verify that the cost of updating the quantities $g_j := A_{j\cdot}^T x$ and $\nabla\phi_j(g_j)$, $j = 1, 2, \ldots, m$ following a step along the coordinate direction x_i is reasonable. Indeed, this is the case. If we update x by taking a step d_i along coordinate direction i, the update formulae for g_j are:

$$g_j \leftarrow g_j + A_{j,i} d_i, \; j = 1, 2, \ldots, m,$$

so it is necessary to update only those g_j (and $\nabla\phi_j(g_j)$) for which $A_{j,i} \neq 0$—a total workload of $O(|A_{i\cdot}|)$ operations. Thus, considering all possible choices of components $i = 1, 2, \ldots, n$, we see that the average cost per iteration of CD is about $O(|A|/n)$, where $|A|$ is the number of nonzeros in A, whereas the cost per iteration of a gradient method would be $O(|A|)$. It is this difference in complexity—a factor of $1/n$ difference between the iteration costs—that makes CD potentially appealing relative to gradient methods.

Note that the least-squares problem $\min \frac{1}{2m}\|A^T x - b\|_2^2$ is special case of this example. (We see this by defining $\phi_j(g_j) = \frac{1}{2}(g_j - b_j)^2$.)

6.2 Objective functions arising from graphs

Many optimization can be written as a linear combination of functions that only involve pairs of variables coupled due to some graph structure. For example, problems in image segmentation might couple adjacent pixels. In topic modeling, terms that appear in the same document may be coupled.

Consider an undirected graph $G = (V, E)$ where the edges $(j, l) \in E$ connect two vertices j and l from $V = \{1, 2, \ldots, n\}$. Suppose our objective has the following form, where each component x_i of the variable $x \in \mathbb{R}^n$ is associated with vertex i:

$$f(x) = \sum_{(j,l)\in E} f_{j,l}(x_j, x_l) + \lambda \sum_{j=1}^{n} \Omega_j(x_j),$$

where f_{jl} (for all $(j, l) \in E$) and the regularization functions Ω_j (for $j = 1, 2, \ldots, n$) are all differentiable. Evaluation of the functiion f and the full gradient ∇f would be an $O(|E|)$ operation (if we assume that evaluation of each f_{jl} and ∇f_{jl} is $O(1)$). To implement a CD method efficiently, we could store the values of f_{jl} and ∇f_{jl} at the current x, for all $(j, l) \in E$. To compute the ith gradient component $\nabla_i f(x)$, we need to sum components from the terms $\nabla f_{jl}(x)$ for which $j = i$ or $l = i$ (at a total cost proportional to the number of edges incident on vertex i) and evaluate the term $\nabla\Omega_i(x_i)$. In updating the values of f_{jl} and ∇f_{jl} after the step in x_i, we need again only change those components for which $j = i$ or $l = i$. The "expected" cost of one CD iteration is thus $O(|E|/n)$. We see once again the desired $1/n$ relationship between the cost per iteration of CD and the cost per iteration of a gradient method.

6.3 Coordinate Descent for Smooth Convex Functions

We again develop most of the ideas with reference to the familiar smooth convex minimization problem defined by

$$\min_{x\in\mathbb{R}^n} f(x), \tag{6.3}$$

where f is smooth and convex, with modulus of convexity μ and a bound L on the Lipschitz constant of the gradient for all points x in some region of interest; see (2.18) and (2.7). We showed in Lemmas 2.3 and 2.9 that, in the case of f twice continuously differentiable, these conditions are a consequence of uniform bounds on the eigenvalues of the Hessian (2.9), that is, $\mu I \preceq \nabla^2 f(x) \preceq LI$. Because the variants we consider here are mostly descent methods, it is enough to restrict our attention in these definitions to an open neighborhood $\mathcal{O}^0$ of the level set of f for the starting point x^0, which is $\mathcal{L}^0 := \{x \mid f(x) \le f(x^0)\}$.

We introduce other partial Lipschitz constants for the function f and its gradient. We define each componentwise Lipschitz constant L_i, $i = 1, 2, \ldots, n$ to satisfy the bound

$$|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \le L_i|\alpha|, \quad i = 1, 2, \ldots, n, \tag{6.4}$$

for all x, α such that $x \in \mathcal{O}^0$ and $x + \alpha e_i \in \mathcal{O}^0$, while we define

$$L_{\max} := \max_{i=1,2,\ldots,n} L_i. \tag{6.5}$$

The "restricted" Lipschitz constant L_{res} satisfies

$$\|\nabla f(x_i + \alpha e_i) - \nabla f(x_i)\| \leq L_{\text{res}}|\alpha|, \quad i = 1, 2, \ldots, n. \tag{6.6}$$

Note that for f twice continuously differentiable, we have that

$$[\nabla^2 f(x)]_{ii} \leq L_i, \quad \|[\nabla^2 f(x)]_{\cdot i}\| \leq L_{\text{res}}, \quad \text{for all } x \in \mathcal{O}^o \text{ and } i = 1, 2, \ldots, n,$$

where $A_{\cdot i}$ denotes the ith column of the matrix A.

These Lipschitz constants play important roles both in implementing variants of CD and it analyzing its convergence rates, especially in comparing these rates with those of full-gradient methods. We can obtain some bounds on the difference between L and $L_{\max}$ by considering the convex quadratic function $f(x) = (1/2)x^T Ax$ where A is symmetric positive semidefinite. We have that $L = \|A\|_2 = \lambda_{\max}(A)$, while from the definition of $L_{\max}$ we have $L_{\max} = \max_{i=1,2,\ldots,n} A_{ii}$. It is clear from definition of matrix norm that

$$L \geq \|Ae_i\|/\|e_i\| = \sqrt{\sum_{j=1}^{n} A_{ji}^2} \geq A_{ii},$$

so by taking the max of both sides, we have $L \geq L_{\max}$. (Equality holds for any nonnegative diagonal matrix.) On the other hand, we have by the relationship between trace and sum of eigenvalues (A.3) that

$$L = \lambda_{\max}(A) \leq \sum_{i=1}^{n} \lambda_i(A) = \sum_{i=1}^{n} A_{ii} \leq nL_{\max}.$$

(Equality holds for the matrix $A = ee^T$, where $e = (1, 1, \ldots, 1)^T$. Thus, we have

$$L_{\max} \leq L \leq nL_{\max}. \tag{6.7}$$

6.3.1 Stochastic CD

In the basic stochastic coordinate descent (SCD) approach, the index i_k to be updated is selected uniformly at random from $\{1, 2, \ldots, n\}$, and the iterations have the form (6.2) for some $\alpha_k > 0$. For "short-step" methods, in which α_k is determined by the Lipschitz constants rather than by an exact minimization or line-search process, we show that sublinear convergence rates can be attained for convex functions and linear convergence rates for strongly convex functions ($\mu > 0$ in (2.18)). Later, we dicuss how this rate relates to the rates obtained in Chapter 3 for full-gradient, steepest descent methods.

For precision, we make the following assumption for the remainder of this section. We make use here of the level set $\mathcal{L}^0$ and its open neighborhood $\mathcal{O}^0$ defined above.

Assumption 1. *The function f is convex and uniformly Lipschitz continuously differentiable on the set $\mathcal{O}^0$ defined above, and attains its minimum on a set $\mathcal{S}$. There is a finite positive number R_0 for which the following bound is satisfied:*

$$\max_{x^* \in \mathcal{S}} \max_{x \in \mathcal{L}^0} \|x - x^*\| \leq R_0.$$

In the analysis that follows, we denote expectation with respect to a single random index i_k by $E_{i_k}(\cdot)$, while $E(\cdot)$ denotes expectation with respect to all random variables $i_0, i_1, i_2, \ldots$.

We prove a convergence result for the randomized algorithm, for the simple steplength choice $\alpha_k \equiv 1/L_{\max}$.

Theorem 6.1. *Suppose that Assumption 1 holds, that each index i_k in the iteration (6.2) is selected uniformly at random from $\{1, 2, \ldots, n\}$, and that $\alpha_k \equiv 1/L_{\max}$. Then for all $k > 0$ we have*

$$E(f(x^k)) - f^* \leq \frac{2nL_{\max}R_0^2}{k}. \tag{6.8}$$

When $\mu > 0$ in (2.18), we have in addition that

$$E\left(f(x^k)\right) - f^* \leq \left(1 - \frac{\mu}{nL_{\max}}\right)^k (f(x^0) - f^*). \tag{6.9}$$

Proof. By application of Taylor's theorem, and using (6.4) and (6.5), we have

$$\begin{aligned} f(x^{k+1}) &= f\left(x^k - \alpha_k \nabla_{i_k} f(x^k) e_{i_k}\right) \\ &\leq f(x^k) - \alpha_k [\nabla_{i_k} f(x^k)]^2 + \frac{1}{2}\alpha_k^2 L_{i_k} [\nabla_{i_k} f(x^k)]^2 \\ &\leq f(x^k) - \alpha_k \left(1 - \frac{L_{\max}}{2}\alpha_k\right) [\nabla_{i_k} f(x^k)]^2 \\ &= f(x^k) - \frac{1}{2L_{\max}} [\nabla_{i_k} f(x^k)]^2, \end{aligned} \tag{6.10}$$

where we substituted the choice $\alpha_k = 1/L_{\max}$ in the last equality. Taking the expectation of both sides of this expression over the random index i_k, we have

$$\begin{aligned} E_{i_k} f(x^{k+1}) &\leq f(x^k) - \frac{1}{2L_{\max}} \frac{1}{n} \sum_{i=1}^{m} [\nabla_i f(x^k)]^2 \\ &= f(x^k) - \frac{1}{2nL_{\max}} \|\nabla f(x^k)\|^2. \end{aligned} \tag{6.11}$$

(We used here the facts that x^k does not depend on i_k, and that i_k was chosen from among $\{1, 2, \ldots, n\}$ with equal probability.) We now subtract $f(x^*)$ from both sides this expression, take expectation of both sides with respect to *all* random variables $i_0, i_1, \ldots$, and use the notation

$$\phi_k := E(f(x^k)) - f^*. \tag{6.12}$$

to obtain

$$\phi_{k+1} \leq \phi_k - \frac{1}{2nL_{\max}} E\left(\|\nabla f(x^k)\|^2\right) \leq \phi_k - \frac{1}{2nL_{\max}} \left[E(\|\nabla f(x^k)\|)\right]^2. \tag{6.13}$$

(We used Jensen's Inequality in the second inequality.) By convexity of f we have for any $x^* \in \mathcal{S}$ that

$$f(x^k) - f^* \leq \nabla f(x^k)^T (x^k - x^*) \leq \|\nabla f(x^k)\| \|x^k - x^*\| \leq R_0 \|\nabla f(x^k)\|,$$

where the final inequality is obtained from Assumption 1, because $f(x^k) \leq f(x^0)$, so that $x^k \in \mathcal{L}^0$. By taking expectations of both sides, we have

$$E(\|\nabla f(x^k)\|) \geq \frac{1}{R_0}\phi_k.$$

When we substitute this bound into (6.13), and rearrange, we obtain

$$\phi_k - \phi_{k+1} \geq \frac{1}{2nL_{\max}}\frac{1}{R_0^2}\phi_k^2.$$

We thus have

$$\frac{1}{\phi_{k+1}} - \frac{1}{\phi_k} = \frac{\phi_k - \phi_{k+1}}{\phi_k\phi_{k+1}} \geq \frac{\phi_k - \phi_{k+1}}{\phi_k^2} \geq \frac{1}{2nL_{\max}R_0^2}.$$

By applying this formula recursively, we obtain

$$\frac{1}{\phi_k} \geq \frac{1}{\phi_0} + \frac{k}{2nL_{\max}R_0^2} \geq \frac{k}{2nL_{\max}R_0^2},$$

so that (6.8) holds, as claimed.

In the case of f strongly convex with modulus $\mu > 0$, we have by taking the minimum of both sides with respect to y in (2.18), and setting $x = x^k$, that

$$f^* \geq f(x^k) - \frac{1}{2\mu}\|\nabla f(x^k)\|^2.$$

By using this expression to bound $\|\nabla f(x^k)\|^2$ in (6.13), we obtain

$$\phi_{k+1} \leq \phi_k - \frac{\mu}{nL_{\max}}\phi_k = \left(1 - \frac{\mu}{nL_{\max}}\right)\phi_k.$$

Recursive application of this formula leads to (6.9). □

Note that the same convergence expressions can be obtained for more refined choices of steplength α_k, by making minor adjustments to the logic in (6.10). For example, the choice $\alpha_k = 1/L_{i_k}$ leads to the same bounds (6.8) and (6.9). The same bounds hold too when α_k is the exact minimizer of f along the coordinate search direction; we modify the logic in (6.10) for this case by taking the minimum of all expressions with respect to α_k, and use the fact that $\alpha_k = 1/L_{\max}$ is in general a suboptimal choice.

We can compare the convergence rates in Theorem 6.1 with the corresponding rates for full-gradient short-step methods from Sections 3.2. In comparing (6.8) with the corresponding result for full-gradient descent with constant steplength $\alpha_k = 1/L$ (where L is from (2.7)). The iteration

$$x^{k+1} = x^k - \frac{1}{L}\nabla f(x^k)$$

leads to a convergence expression

$$f(x^k) - f^* \leq \frac{2LR_0^2}{k} \tag{6.14}$$

(see, for example, [21]). Since, for problems of interest in this chapter, there is roughly a factor-of-n difference between one iteration of CD and one iteration of a full-gradient method, the bounds (6.14) and (6.8) would be comparable if L and $L_{\max}$ are approximately the same. The bounds (6.7) suggest that $L_{\max}$ can be significantly less than L for some problems, and by comparing the convergence expressions, we see that randomized CD may have an advantage in such cases.

A similar conclusion is reached when we compare the convergence rates on the strongly convex case. We have for steepest-descent with line search $\alpha \equiv 2/(L+\mu)$ (see Section 3.2) that

$$\|x_{k+1} - x^*\| \leq \left(1 - \frac{2}{(L/\mu)+1}\right) \|x_k - x^*\|. \tag{6.15}$$

Because of Lemma 3.4, the quantities $f(x_k) - f(x^*)$ and $\|x_k - x^*\|^2$ converge at similar rates, so we get a more apt comparison with (6.9) by squaring both sides of (6.15). By using the approximation $(1-\epsilon)^m \approx 1 - m\epsilon$ for any constants m and ϵ with $m\epsilon \ll 1$, we estimate that the rate constant for convergence of $\{f(x_k)\}$ in short-step steepest descent would be about

$$1 - \frac{4\mu}{L+\mu} \approx 1 - \frac{4\mu}{L}, \tag{6.16}$$

because we can assume that $L + \mu \approx L$ for all but the most well conditioned problems. Apart from the extra factor of 4 in (6.16), and the expected factor-of-n difference between the key terms, we note again that the main difference is the replacement of $L_{\max}$ in (6.9) by L in (6.16). Again, we note the possibility of a faster overall rate for CD when $L_{\max}$ is significantly less than L.

6.3.2 Cyclic CD

Cyclic variants of CD, where we update the coordinates in a fixed order, repeatedly cycling through them all until convergence, are perhaps the most intuitive form of the algorithm. The classical Gauss-Seidel method, popular also for linear systems of equations, has this form, with the steplengths chosen to minimize f exactly along each search direction. Other variants do not minimize exactly but rather take steps of the form (6.2), with α_k chosen according to estimates of the Lipschitz properties of the function, and other considerations.

The choice of index i_k in cyclic CD is as follows:

$$i_k = (k \bmod n) + 1, \quad k = 0, 1, 2, \ldots, \tag{6.17}$$

giving the sequence $1, 2, 3, \ldots, n, 1, 2, 3, \ldots, n, 1, 2, 3, \ldots$.

Surprisingly, results concerning the convergence of cyclic variants for smooth convex f have been proved only recently [1]. (Results for the special case of Gauss-Seidel applied to a convex quadratic f, and its important symmetric over-relaxation (SOR) variant, have been the subjects of research in the numerical linear algebra community for many years.) We describe a result with a flavor similar to Theorem 6.1. We assume a fixed steplength α is used at every iteration, where $\alpha \leq 1/L_{\max}$.

Theorem 6.2. *Suppose that Assumption 1 holds, and that the iteration* (6.2) *is applied with the index i_k at iteration k chosen according to the cyclic ordering* (6.17) *and $\alpha_k \equiv \alpha \leq 1/L_{\max}$. Then for $k = n, 2n, 3n, \ldots$, we have*

$$f(x^k) - f^* \leq \frac{(4n/\alpha)(1 + nL^2\alpha^2)R_0^2}{k+8}. \tag{6.18}$$

When μ in the strong convexity condition (2.17) *is strictly positive, we have in addition for $k = n, 2n, 3n, \ldots$ that*

$$f(x^k) - f^* \leq \left(1 - \frac{\mu}{(2/\alpha)(1 + nL^2\alpha^2)}\right)^{k/n} (f(x^0) - f^*). \tag{6.19}$$

Proof. The result (6.18) follows from [1, Theorems 3.6 and 3.9] when we note that (i) each iteration of Algorithm BCGD in [1] corresponds to a "cycle" of n iterations of (6.2); (ii) we update coordinates rather than blocks, so that the parameter p in [1] is equal to n; (iii) we set $\bar{L}_{\max}$ and $\bar{L}_{\min}$ in [1] both to $1/\alpha$, which is greater than or equal to $L_{\max}$, as required by the proofs in this reference. □

The cyclic CD approach would seem to have an intuitive advantage over the full-gradient steepest descent method, if we compare a single cycle of cyclic CD to one step of steepest descent. Cyclic CD is making use of the most current gradient information whenever it takes a step along a coordinate direction, whereas steepest descent evaluates the moves along all n coordinates at the same value of x. This advantage is not reflected in the worst-case analysis of Theorem 6.2, however, which suggest slower convergence than full-gradient steepest descent, even when we assume that the cost per iteration differs by $O(n)$ between the two approaches (see details below). Indeed, the proof in [1] treats the cyclic CD method as a kind of perturbed steepest descent method, bounding the change in objective value over one cycle in terms of the gradient at the start of the cycle.

The bounds (6.18) and (6.19) are generally worse than the corresponding bounds (6.8) and (6.9) obtained for the randomized algorithm, as we explain in a moment. Computational comparisons between randomized and cyclic methods show similar performance on many problems, but as a comparison of the bounds suggests, cyclic methods perform worse (sometimes much worse) when the ratio $L/L_{\max}$ exceeds its lower bound on 1 significnatly. We note also that the bounds (6.18) and (6.19) are deterministic, whereas (6.8) and (6.9) are bounds on expected error.

We illustrate the results of Theorem 6.2 with three possible choices for α. Setting α to upper lower bound of $1/L_{\max}$, we have for (6.18) that

$$f(x^k) - f^* \leq \frac{4nL_{\max}(1 + nL^2/L_{\max}^2)R_0^2}{k+8}.$$

The numerator here is worse than the corresponding result (6.8) by a factor of approximately $2nL^2/L_{\max}^2 \in [2n, 2n^3]$, suggesting better performance for the randomized method, with a larger advantage on problems for which $L_{\max} \ll L$. If we set $\alpha = 1/L$ (a legal choice, since $L \geq L_{\max}$), (6.18) becomes

$$\frac{4n(n+1)LR_0^2}{k+8},$$

which is worse by a factor of approximately $2n^2$ than the bound (6.14) for the full-step gradient descent approach. For $\alpha = 1/(\sqrt{n}L)$, we obtain

$$\frac{8n^{3/2}LR_0^2}{k+8},$$

which still trails (6.14) by a factor of $4n^{3/2}$. If we take into account the factor-of-n difference in cost between iterations of CD and full-gradient methods for problems of interest, these differences shrink to factors of n and $n^{1/2}$, respectively.

6.3.3 Coordinate Sampling Without Replacement

An important variant of CD is a kind of hybrid of the randomized and cyclic approaches. As in the cyclic approach, we divide the computations into a cycles of n iterations each, where each within each cycle, every coordinate is updated exactly once. Unlike the cyclic approach, however, we reshuffle the coordinates at each cycle. (Equivalently, we can think of each cycle as sampling the coordinates from the set $\{1, 2, \ldots, n\}$ *without replacement.*) Unlike the fully randomized method, this variant "touches" each component exactly once per cycle, whereas in the randomized CD approach it is possible (though unlikely) that a given coordinate x_i will not be updated for arbitrarily many iterations.

The convergence properties proved in Theorem 6.2 continue to hold for this variant; the proofs in [1] need no modification. Curiously, however, computational experience shows that this variant avoids the poor behavior of the purely cyclic variant in cases for which the ratio $L/L_{\max}$ is large. In general, its performance is quite similar to that of "sampling with replacement" stochastic approach of Section 6.3.1.

Notes and References

Mention the Lee-Sidford versions of accelerated CD.

The proof of Theorem 6.1 is a simplified version of the analysis in Nesterov [22, Section 2].

Analysis of the cyclic method in Section 6.3.2 is drawn from [1].

The justification for using CD methods as opposed to full-gradient methods is perhaps seen best in asynchronous implementations on parallel computers. Multiple cores can of course share the workload of evaluating a full gradient, but there is inevitably a synchronization point—the computation ust wait for all cores to complete their share of the work before it can proceed with computing and taking the step. Asynchronous implementations of CD methods are extremely easy to design, especially for multicore, shared-memory computers in which all cores have access to a shared version of the variable x (and possibly other quantities involved in the evaluation of gradient information). Strong results about the convergence of asynchronous algorithms under weak assumptions were obtained in [3, Section 7.5]. Recently, Liu et al. [19], Liu and Wright [17] showed that convergence rates of the serial CD methods are largely inherited by multicore implementations provided that the number of cores is not too large.

Exercises

1. In the example of Section 6.1.1, assume that the objective function f is known at the current point x, along with the quantity $g = Ax$. Show that the cost of computing $f(x + gamma_i e_i)$ for some $i = 1, 2, \ldots, n$ is $O(|A_{\cdot i}|)$—the same order as the cost of updating the gradient ∇f. Show that a similar observation holds for the example in Section 6.2.

2. Consider the convex quadratic $f(x) = (1/2)x^T Ax$ with $A = ee^T$, where $e = (1, 1, \ldots, 1)^T$, for which $L = n$ and $L_{\max} = L_i = 1$ for $i = 1, 2, \ldots, n$. Show that any variant of CD with $\alpha = 1/L_{\max}$ or $\alpha = 1/L_i$ converges in one iteration. Show that steepest descent (with either exact line search or step length $\alpha = 1/L$) also converges in one step.

3. Implement the following variants of coordinate descent:

- Stochastic (the method of Section 6.3.1) with exact line search and with constant step length $1/L_{\max}$;
- Cyclic (Section 6.3.2) with exact line search and with constant step lengths $1/L_{\max}$, $1/L$, and $1/(\sqrt{n}L)$;
- "Sampling without replacement" (Section 6.3.3) with exact and with constant step length $1/L_{\max}$.

Compare the performance of these methods on convex quadratic problems $f(x) = \frac{1}{2}x^T Ax$, where A is an $n \times n$ positive semidefinite matrix constructed randomly in the manner described below. (Note that $x^* = 0$ with $f(x^*) = 0$.) Terminate when $f(x) \leq 10^{-6} f(x^0)$. Use a random starting point x^0 whose components are uniformly distributed in $[0, 1]$. Compute and print the values of L and $L_{\max}$ for each instance.

Test your code on the following matrices A.

(i) $A = Q^T D Q^T$, where Q is random orthogonal and D is a positve diagonal matrix whose here each diagonal D_{ii} has the form $10^{-\zeta_i}$, where each ζ_i is drawn uniformly i.i.d. from $[0, 1]$.

(ii) The same as in (i), but with each ζ_i drawn uniformly i.i.d. from $[0, 2]$.

(iii) Generate the matrix A as in (i), them replace it by $A + 5E$, where E is the $n \times n$ matrix whose components are all 1.

Discuss the relative performance of the methods on these different problems. How is your computational experience consistent (or inconsistent) with the convergence expressions obtained in Theorems 6.1 and 6.2?

4. Compare the linear convergence bounds (6.9) and (6.19) for the stochastic and cyclic variants of CD, for various choices of steplength in the cyclic method, including $\alpha = 1/L_{\max}$, $\alpha = 1/L$, and $\alpha = 1/(\sqrt{n}L)$. (In making these comparisons, note that for small ϵ we have $(1 - \epsilon)^{1/n} \approx 1 - \epsilon/n$.) Which of these choices of fixed step length α in the cyclic method is optimal, in the sense of approximately minimizing the factor on the right-hand side of (6.19)?