10-725/36-725: Convex Optimization **Fall 2016**

Lecture 23: November 21

Lecturer: Ryan Tibshirani *Scribes: Yifan Sun, Ananya Kumar, Xin Lu*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

This lecture's notes covers coordinate descent and screening rules.

Introduction

Coordinate descent is a very simple technique that can be surprisingly, efficient, scalable compare to some pretty sophisticated methods we have learned. Appropriately it is also called coordinatewise minimization.

Motivation

To motivate the objective function we would like to deal with using coordinate descent, let's consider these questions first:

Q1: Does $f(x + \delta e_i) \geq f(x)$ for all $\delta, i \implies f(x) = min_z f(z)$ (Here $e_i = (0, ..., 1, ...0)$, the i-th standard basis vector) always hold ? In other words, given convex, differentiable $f : R^n \to R$, if we are at a point x such that $f(x)$ is minimized along each coordinate axis, then have we found a global minimizer?

The answer is yes. Here is the proof:

$$f(x + \delta e_i) \geq f(x) \implies \frac{\partial f}{\partial x_i}(x) = 0$$

, which means

$$\nabla f(x) = (\frac{\partial f}{\partial x_1}(x), ..., \frac{\partial f}{\partial x_n}(x)) = 0$$

. Then we get $f(x) = min_z f(z)$.

Q2: Same question, but f is convex, not differentiable ?

The answer is no. In figure 23.1 we can see that the whatever the cross-point goes any direction along the axis, the criterion value will increase.

Q3: Same question again, but now $f(x) = g(x) + \sum_{i=1}^{n} h_i(x_i)$, where $g(x)$ is convex, differentiable and each h_i is just convex (Here the nonsmooth part is called separable) ?

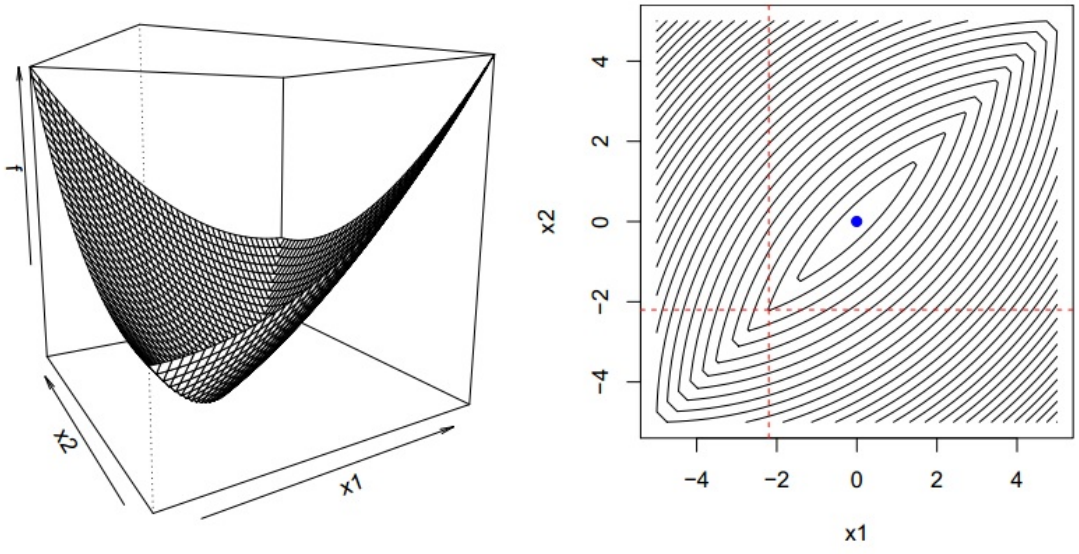

Figure 23.1: Counterexample of q2.

The answer is yes. Here is the proof: since $g(x)$ is convex, differentiable, for any y, we have

$$f(y)-f(x) = g(y)+\sum_{i=1}^{n} h_i(y_i)-[g(x)+\sum_{i=1}^{n} h_i(x_i)] \geq \nabla g(x)^T(y-x)+\sum_{i=1}^{n}[h_i(y_i)-h_i(x_i)] = \sum_{i=1}^{n}(\nabla_i g(x)(y_i-x_i)+h_i(y_i)-h_i(x_i))$$

. We now want to proof

$$\nabla_i g(x)(y_i - x_i) + h_i(y_i) - h_i(x_i) \geq 0$$

. Consider

$$f_i(x_i) = g(x_i; x_{-i}) + h_i(x_i)$$

(here x_{-i} means all x_j, $j \neq i$ is constant now).

$$f(x + \delta e_i) \geq f(x) \implies 0 \in \partial f_i(x_i) = \nabla_i g(x) + \partial h_i(x_i) \implies \nabla_i g(x) \in -\partial h_i(x_i)$$

, then by definition of subgradient:

$$h_i(y_i) \geq h_i(x_i) - \nabla_i g(x)(y_i - x_i)$$

. Thus, we can conclude that for any y, $f(y) - f(x) \geq 0$.

Update Rule

Q3 suggests that for $f(x) = g(x) + \sum_{i=1}^{n} h_i(x_i)$, where $g(x)$ is convex, differentiable and each h_i is just convex, we can use coordinate descent to find a minimizer: start with some initial guess $x^{(0)}$, and repeat:

$$x_1^{(k)} \in \arg\min_{x_1} f(x_1, x_2^{(k-1)}, ..., x_n^{(k-1)})$$

$$x_2^{(k)} \in \arg\min_{x_2} f(x_1^{(k)}, x_2, ..., x_n^{(k-1)})$$

...

$$x_n^{(k)} \in \arg\min_{x_n} f(x_1^{(k)}, x_2^{(k)}, ..., x_n)$$

for $k = 1, 2, 3....$

Here is several things worth to notice:

- The order of cycle through coordinates is arbitrary, we can use any permutation of $1, 2, ..., n$. If only we visit linear number of updates x_i before going to update x_j (eg. update $2n$ times, but cannot be n^2), the algorithm can converge.
- We can replace individual coordinates with blocks of coordinates in everywhere. We will see the example in Box-constrained QP.
- "One-at-a-time" update scheme is critical, and "all-at-once" scheme does not necessarily converge. In other words, after solving for $x_i^{(k)}$, we use its new value from then on.
- Coordinate descent can be the analogy for solving linear systems: Gauss-Seidel versus Jacobi method. We can see the example in Linear Regression.

Example

Linear Regression

Given $y \in R^n$, and $X \in R^{n \times p}$ with columns $X_1, ..., X_n$, consider linear regression:

$$\min_{\beta} \frac{1}{2}||y - X\beta||_2^2$$

We can perform coordinate descent by repeatedly minimize over β_i for $i = 1, 2, ..., p, 1, 2, ...$ Here β_i can be gotten by solving:

$$0 = \nabla_i f(\beta) = X_i^T(X\beta - y) = X_i^T(X_i\beta_i + X_{-i}\beta_{-i} - y)$$

$$\implies \beta_i = \frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i}$$

We can easily observe that this process is exactly solving the Gauss-Seidel system $X^T X\beta = X^T y$.

We now compare the coordinate descent with gradient descent for linear regression: The result is in figure 23.2.

Someone may have such natural question for the comparison: Is it fair to compare 1 cycle of coordinate descent to 1 iteration of gradient descent? The answer is yes, since they are both $O(np)$ in each iteration:

For gradient descent, in each iteration, we compute $\beta^+ = \beta + tX^T(y - X\beta)$, these matrix multiplication results in $O(np)$ flops.

For coordinate descent, in each iteration, for one coordinate update, we compute

$$\beta_i = \frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i} = \frac{X_i^T(y - X_i\beta_i)}{X_i^T X_i} + \frac{X_i^T X_i\beta_i}{X_i^T X_i} = \frac{X_i^T r}{||X_i||_2^2} + \beta_i$$

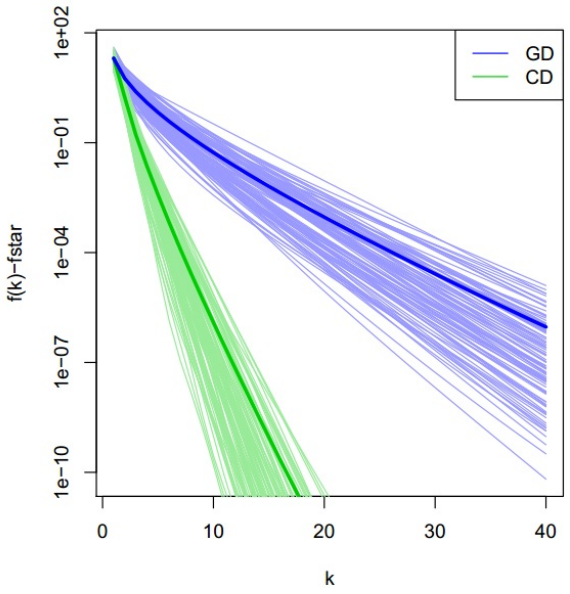

Figure 23.2: linear regression: 100 instances with $n = 100$, $p = 20$.

, where $r = y - X\beta$. Thus, each coordinate costs $O(n)$ flops ($O(n)$ to update r, $O(n)$ to compute $X_i^T r$). Each iteration has p coordinate operation, so costs $O(np)$, which is the same as gradient descent.

We now add accelerated gradient descent into the comparison, the results shows in figure 23.3. Note that this doesn't contradict with the optimality of accelerated gradient descent, as coordinate descent uses more information than first-order method.

Lasso Regression

Given $y \in R^n$, and $X \in R^{n \times p}$ with columns $X_1, ..., X_n$, consider lasso regression:

$$\min_\beta \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_1$$

We can perform coordinate descent by repeatedly minimize over β_i for $i = 1, 2, ..., p, 1, 2, ...$ Here β_i can be gotten by solving:

$$0 = X_i^T(X_i\beta_i + X_{-i}\beta_{-i} - y) + \lambda s_i$$

, where $s_i \in \partial|\beta_i|$. Then by using soft-thresholding we get

$$\beta_i = S_{\lambda/||X_i||_2^2}\frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i}$$

.

Figure 23.4 shows proximal gradient vs coordinate descent for lasso regression. The coordinate gradient descent here is comparable to gradient descent for the same reason that they both share $O(np)$ flops in each iteration.

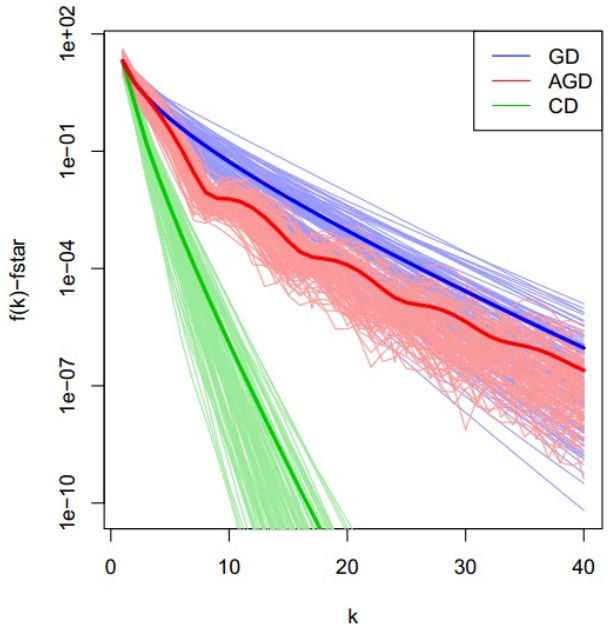

Figure 23.3: Same example, but now with accelerated gradient descent for comparison.

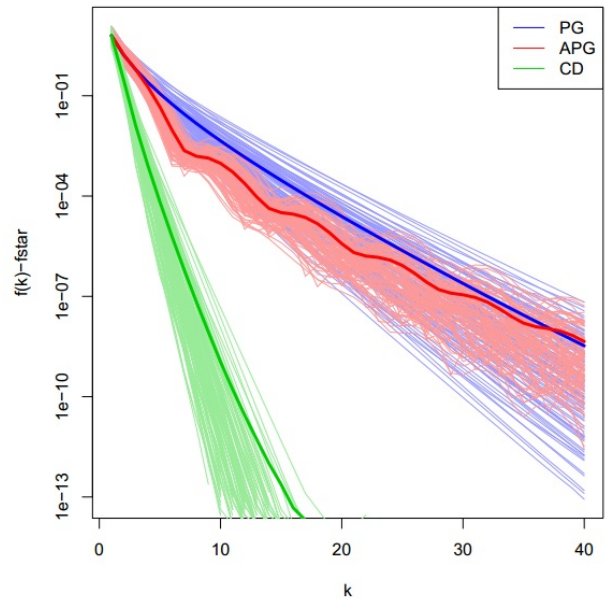

Figure 23.4: lasso regression: 100 instances with $n = 100$, $p = 20$.

Box-constrained QP

Given $b \in \mathbb{R}^n$ and $Q \in \mathbb{S}^n_+$, consider the following box-constrained quadratic programming:

$$\begin{aligned} \min_x \quad & \tfrac{1}{2} x^T Q x + b^T x \\ \text{subject to} \quad & l \le x \le u \end{aligned}$$

First we note that the constrained are coordinate-wise separable:

$$I\{l \le x \le u\} = \sum_{i=1}^n I\{l_i \le x_i \le u_i\}$$

This tells us we can use the coordinate descent to sequentially minimize the function over different coordinate, the minimization is given by:

$$x_i = T_{[l_i,u_i]}\left(\frac{b_i - \sum_{j \ne i} Q_{i,j} x_j}{Q_{ii}}\right)$$

where $T_{[l_i,u_i]}$ is the truncation (projection) operator onto $[l_i, u_i]$:

$$T_{[l_i,u_i]}(z) = \begin{cases} u_i, & \text{if z¿}u_i \\ z, & \text{if } l_i \le z \le u_i \\ l_i, & \text{if z¡}l_i \end{cases}$$

Support Vector Machine

Coordinate descent can be applied to SVM dual problem:

$$\begin{aligned} \min_\alpha \quad & \tfrac{1}{2} \alpha^T \tilde{X} \tilde{X}^T \alpha - \mathbf{1}^T \alpha \\ \text{subject to} \quad & 0 \le \alpha \le C\mathbf{1} \\ & \alpha^T y = 0 \end{aligned}$$

Note that the constraint $\alpha^T y = 0$ is not coordinate-wise separable hence vanilla coordinate descent cannot be used here. However, Platt proposed a sequential minimal optimization (SMO) algorithm, which is blockwise coordinate descent in blocks of 2. Instead of cycling over blocks, it chooses the next block greedily. The SMO algorithm utilizes the complementary slackness condition:

$$\begin{aligned} \alpha_i(1 - \zeta_i - (\tilde{X}\beta)_i - y_i\beta_0) = 0, \qquad & \text{i=1,...n} \\ (C - \alpha_i)\zeta_i = 0 \qquad & \text{i=1,...n} \end{aligned}$$

where β_0, β, ζ_i are primal intercepts, coefficients and slack variables. $\beta = \tilde{X}^T \alpha$ and β_0 is calculated using any i s.t. $0 < \alpha_i < C$,ζ_i is also calculated from the above two conditions. The SMO repeats the following steps:

- Choose α_i, α_j that do not satisfy the complementary slackness condition greedily.
- Minimize over α_i and α_j exactly, keeping other variables fixed. Using equality constraint $\alpha^T y = 0$ to reduce the problem into minimizing a quadratic function over one variable on an interval.

Many further developments on coordinate descent for SVMs have been made; e.g., a recent one is Hsiesh et al. (2008)

Coordinate descent in ML and Statistics

History:

- Idea appeared in Fu (1998), and again in Daubechies et al. (2004), but was inexplicably ignored. (Students of Robert Tibshirani)
- Three papers around 2007, especially Friedman et al. (2007), really sparked interest in statistics and ML communities (Work done with Robert Tibshirani)

Advantages of coordinate descent:

- Very simple and easy to implement
- Careful implementations can be near state-of-the-art
- Scalable, e.g., dont need to keep full data in memory

Examples of applications: lasso regression, lasso GLMs (under proximal Newton), SVMs, group lasso, graphical lasso (applied to the dual), additive modeling, matrix completion, regression with nonconvex penalties.

Pathwise coordinate descent for Lasso

Friedman et al. proposed the pathwise coordinate descent method for Lasso problem. The pathwise coordinate descent uses a similar method as central path, instead of solving the problem of given tuning parameter λ, it minimizing over a sequence of tuning parameters $\lambda_1 > \lambda_2 > ... > \lambda_r = \lambda$ and at for each λ_{k+1} it uses the solution from λ_k as a warm start. The algorithm takes two loops:

Outer loop(pathwise strategy):

- Compute the solution over a sequence $\lambda_1 > \lambda_2 > ... > \lambda_r = \lambda$ of tuning parameter values
- For tuning parameter value λ_k, initialize coordinate descent algorithm at the computed solution for λ_{k+1} (warm start)

Inner loop (active set strategy):

- Perform one coordinate cycle (or small number of cycles), and record active set A of coecients that are nonzero
- Cycle over coecients in A until convergence

- Check KKT conditions over all coecients; if not all satised, add oending coecients to A, go back one step

Notes:

- Even when the solution is desired at only one , the pathwise strategy (solving over $\lambda_1 > \lambda_2 > ...\lambda_r = \lambda$) is typically much more efficient than directly performing coordinate descent at λ, indeed this is what many existing packages will do.
- Active set strategy takes advantage of sparsity; e.g., for very large problems, coordinate descent for lasso is much faster than it is for ridge regression
- With these strategies in place (and a few more clever tricks), coordinate descent can be competitve with fastest algorithms for L1 penalized minimization problems
- Fast Fortran implement glmnet, easily linked to R or MATLAB

Other versions of coordinate descent

People may use coordinate descent as name for the following algorithm, which is different to what we have covered at the beginning of the class:

$$
\begin{aligned}
x_1^{(k)} &= x_1^{(k-1)} - t_{k,1} \cdot \nabla_1 f(x_1^{(k-1)}, x_2^{(k-1)}, x_3^{(k-1)}, ...x_n^{(k-1)}) \\
x_2^{(k)} &= x_2^{(k-1)} - t_{k,2} \cdot \nabla_2 f(x_1^{(k)}, x_2^{(k-1)}, x_3^{(k-1)}, ...x_n^{(k-1)}) \\
x_3^{(k)} &= x_3^{(k-1)} - t_{k,3} \cdot \nabla_3 f(x_1^{(k)}, x_2^{(k)}, x_3^{(k-1)}, ...x_n^{(k-1)}) \\
&... \\
x_n^{(k)} &= x_n^{(k-1)} - t_{k,n} \cdot \nabla_n f(x_1^{(k)}, x_2^{(k)}, x_3^{(k)}, ...x_n^{(k-1)})
\end{aligned}
$$

If $f = g + h$ where g is smooth and h is separable, the proximal version of the above algorithm is also called coordinate descent. These versions are generally easier to apply than the exact coordinate descent we talked about at the beginning of the class, but the version we covered is performs an exact minimization and hence makes more progress.

Convergence analysis

Empirically, optimized implementations of coordinate descent often give state of the art algorithms for many problems. However, the theory for coordinate descent is still being developed. In particular, there are still many unanswered questions, for example whether the coordinate-wise descent and exact minimization strategies perform comparably. Below is a list of results on the convergence of coordinate descent. Warning: some references below treat coordinatewise minimization, some do not.

- Convergence of coordinatewise minimization for solving linear systems, the Gauss-Seidel method, is a classic topic. E.g., see Golub and van Loan (1996), or Ramdas (2014) for a modern twist that looks at randomized coordinate descent

- Nesterov (2010) considers randomized coordinate descent for smooth functions and shows that it achieves a rate $O(1/\epsilon)$ under a Lipschitz gradient condition, and a rate $O(\log(1/\epsilon))$ under strong convexity

- Richtarik and Takac (2011) extend and simplify these results, considering smooth plus separable functions, where now each coordinate descent update applies a prox operation

- Saha and Tewari (2013) consider minimizing l_1 regularized functions of the form $g(\beta) + \lambda\|\beta\|_1$, for smooth g, and study both cyclic coordinate descent and cyclic coordinatewise min. Under (very strange) conditions on g, they show both methods dominate proximal gradient descent in iteration progress

- Beck and Tetruashvili (2013) study cyclic coordinate descent for smooth functions in general. They show that it achieves a rate $O(1/\epsilon)$ under a Lipschitz gradient condition, and a rate $O(\log(1/\epsilon))$ under strong convexity. They also extend these results to a constrained setting with projections

- Nutini et al. (2015) analyze greedy coordinate descent (called Gauss-Southwell rule), and show it achieves a faster rate than randomized coordinate descent for certain problems

- Wright (2015) provides some unification and a great summary. Also covers parallel versions (even asynchronous ones)

Screening Rules Intuition

Consider the Lasso problem:

$$\min \frac{1}{2}||y - X\beta||_2^2 + \lambda|\beta|_1$$

If we knew that some of the β_is are 0 in the final solution, then we could get rid of the corresponding columns X_i. This would make optimization a lot faster and more memory efficient. In general, finding all the β_is that are 0 requires solving the lasso problem. However, we can use screening rules to quickly identify some of the β_is that *must* be 0 in the final solution.

Screening rules are not restricted to lasso, they are used in all kinds of problems to whittle down the active set.

SAFE Screening Rules

First, we define $\lambda_{\max}$:

$$\lambda_{\max} = ||X^T y||_\infty$$

Then we get the following rule:

$$|X_i^T y| < \lambda - ||X_i||_2||y||_2 \frac{\lambda_{\max} - \lambda}{\lambda_{\max}} \Rightarrow \beta_i = 0$$

Note that this is not an if and only if statement - it only identifies some of the β_is that are 0.

To prove this rule, we start from the dual of the lasso. The dual objective function of the lasso is:

$$g(u) = \frac{1}{2}||y||_2^2 - \frac{1}{2}||y - u||_2^2$$

In the dual, we want to minimize $g(u)$ subject to $||X^T u||_\infty \leq \lambda$.

We note that $u_0 = \frac{\lambda}{\lambda_{\max}} y$ is dual feasible. This implies that $\gamma = g(u_0)$ is a lower bound for the dual optimal value. So we can add the constraint $g(u) \geq \gamma$ to the dual problem, without changing the dual problem. In general, adding this constraint does not add much value, but in this case it gives us something useful.

For each i between 1 and p, we compute,

$$m_i = \max |X_i^T y| \text{ subject to } g(u) \geq \gamma$$

Then we get,

$$m_i < \lambda \Rightarrow |X_i^T \hat{u}| < \lambda$$

Using the stationarity KKT condition, this implies that $\beta_i = 0$.

It remains to compute m_i. Using another dual argument, we get that m_i can be computed exactly.

$$m_i = ||X_i||_2 \sqrt{||y||_2^2 - 2\gamma} + |X_i^T y|$$

Substituting in $\gamma = g(u_0)$, we get the SAFE rule.

References