10-725: Optimization **Fall 2012**

Lecture 25: November 27

Lecturer: Ryan Tibshirani *Scribes: Matt Wytock, Supreeth Achar*

25.1 A Taxonomy of Optimization Methods

In class, the general methods for minimizing function we have looked at so far are

- First Order Methods - Gradient descent, acceleration etc.
- Newton's Method
- Dual Methods - Dual Gradient Methods
- Interior Point Methods

These four classes of methods form the basis for most of optimization. From a statistics and machine learning point of view there are a couple of important specialised methods for optimization. They aren't as broadly applicable as the general methods listed above, but when they can be used they are very useful. These specialized methods include

- Coordinate Descent
- Path Algorithms

These notes look at coordinate descent, a method that can solve some classes of convex minimization problems efficiently and is highly scalable.

25.2 History and motivation

Why is it used?

- Very simple and easy to implement
- Careful implementations can attain state-of-the-art
- Scalable, e.g. don't need to keep data in memory

As we will see in the discussion for pathwise coordinate descent for lasso, it also allows us to ignore the majority of coordinates in lasso since they will be zero. Also, there are examples of coordinate descent being applied to non-convex penalties which in itself is pretty rare, see for example [MFH2011].

The idea first appeared in Fu (1998), and again in Daubechies et al. (2004), but was inexplicably ignored until three papers around 2007, and Friedman et al. (2007) really sparked interest.

25.3 Coordinate-wise Minimization

25.3.1 Convex, differentiable functions

Consider a convex, differentiable function $f : \mathcal{R}^n \rightarrow \mathcal{R}$. If at a point $\tilde{x}$, if $f(x)$ at $\tilde{x}$ is minimized along each coordinate axis, that is if we move any distance in either direction along any coordinate axis starting from the point $\tilde{x}$, the value of f will be greater than or equal to the value at f. That is

$$f(\tilde{x} + d \cdot e_i) \geq f(\tilde{x})$$

where d is any real number and e_i is the i^{th} standard basis vector.

Assertion: If f is convex and continuous, then $\tilde{x}$ is a minimiser for the function f. $f(\tilde{x}) = \min_x f(x)$.
Proof: For a convex, continuous function the gradient is zero at the minima. The individual components of the gradient give the directions to move along each axis to reduce the function value. When all these components are zero, then moving along any coordinate axis will not decrease the function value and hence we are also at $\tilde{x}$.

25.3.2 Convex, non differentiable functions

In general, for a convex non differentiable function, coordinate wise minimization does not yield the global minimizer. For example, consider a convex function with isocontours like those shown in figure 25.3.2. At of the places where the function derivative is not smooth (like the place where the two red lines meet), the function is minimized along both coordinate axes, but these points are not minima of the function.

25.3.3 Convex, non differentiable decomposable function

However, if the non differentiable part of the function can be decomposed into a sum of functions over each coordinate $f(x) = g(x) + \sum_{i=1}^{n} h_i(x_i)$ where $g(x)$ is convex and smooth and each h_i is convex, then the coordinate wise minimizer is the true minima of the function.

Proof: Consider any x

$$\begin{aligned} f(x) - f(\tilde{x}) &\geq \nabla g(x)^T (x - \tilde{x}) + \sum_{i=1}^{n} [h_i(x_i) - h_i(\tilde{x}_i)] && (25.1) \\ &= \sum_{i=1}^{n} [\nabla_i g(\tilde{x})(x_i - \tilde{x}_i) + h_i(x_i) - h_i(\tilde{x}_i)] && (25.2) \\ &\geq 0 && (25.3) \end{aligned}$$

Since $f(\tilde{x})$ is always less or equal to $f(x)$, $\tilde{x}$ must be a global minimizer of f.

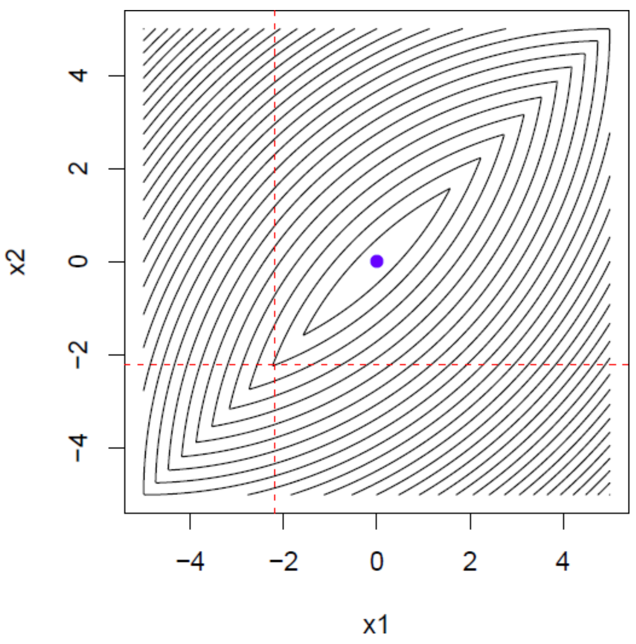

Figure 25.1: These are the isocontours of 2D convex, non differentiable function. At the point where the two red lines meet, the function value can not be reduced by moving in either direction along either coordinate axis, but we are not at a minima of the function.

25.4 The Coordinate Descent Algorithm

A convex, separable function of the form $f(x) = g(x) + \sum h_i(x_i)$ can be minimized by finding the coordinate wise minimizer. This point can be found from an arbitrary starting location by repeatedly cycling through coordinates and finding the value of each coordinate that minimizes the function while keeping the other coordinates fixed.

Algorithm 1 Coordinate Descent Algorithm

```
Initialize with guess $x = [x_1, x_2, ..., x_n]^T$
repeat
  for all $j$ in $1, 2, ..., n$ do
    $x_j \leftarrow \underset{x_j}{\text{argmin}} f(x)$
  end for
until convergence
```

Points to note:

- The sequence of values taken on by x over iterations, $x^{(k)}$ has a subsequence converging to the optimal x^*
- $f(x^{(k)})$ converges to f^*
- the coordinates can be cycled through in any arbitrary order
- instead of minimizing over individual coordinates, any block of coordinates can be minimized over instead

- however, the minimizations can not be performed in parallel, the result of one minimization must be used as the starting point for the next. The"all-at-once" scheme is not guaranteed to converge.

For more details on the conditions for convergence and a proof refer to [T01].

25.5 Applications of Coordinate Descent

Here are some problems that can be solved using coordinate descent

25.5.1 Linear Regression

Consider the standard linear regression problem. Find x to minimize $f(x) = \frac{1}{2}||y - Ax||^2$, with $y \in \mathcal{R}^n$ and $A \in \mathcal{R}^{n \times p}$. Let $A_1, A_2, ..., A_p$ denote the columns of A. To minimize over a single coordinate x_i in x with all over values in x fixed we have

$$\begin{aligned} \nabla_i f(x) &= 0 \\ &= A_i^T(Ax - y) \\ &= A_i^T(A_i x_i + A_{-i}x_{-i} - y) \end{aligned}$$

Where A_{-i} is the matrix obtained by deleting column i from A. This equation can be solved for x_i.

$$x_i = \frac{A_i^T(y - A_{-i}x_{-i})}{A_i^T A_i} = \frac{A_i^T r}{||A_i||^2} + x_i^{old}$$

Where $r = y - Ax$. Coordinate descent can be performed by cycling through all values of i and solving the above equation for each element x_i. Computationally, each coordinate descent update takes $\mathcal{O}(n)$ time and so running through a full cycle over all the coordinates will take $\mathcal{O}(np)$ time. This is the same amount of time as one iteration of gradient descent. Coordinate descent is often faster than gradient descent for linear regression problems (see figure 25.5.1), ie it converges to the correct solution in fewer iterations. In fact, it is even faster than accelerated gradients. Accelerated gradient descent is an optimal first order method, but coordinate descent uses more than first order information.

25.5.2 Lasso Regression

For $L1$ regularized version of the regression problem $f(x) = \frac{1}{2}||y - Ax||^2 + \lambda||x||_1$. The regularizer is non smooth, but it is separable - $||x||_1 = \sum |x_i|$. Following the same approach of minimizing over x_i while keeping all other coordinates fixed we get the following update rule

$$x_i = S_{\lambda/||A_i||^2}\left(\frac{A_i^T\,(y - A_{-i}x_{-i})}{A_i^T A_i}\right)$$

where S_α is the soft thresholding function with parameter α.

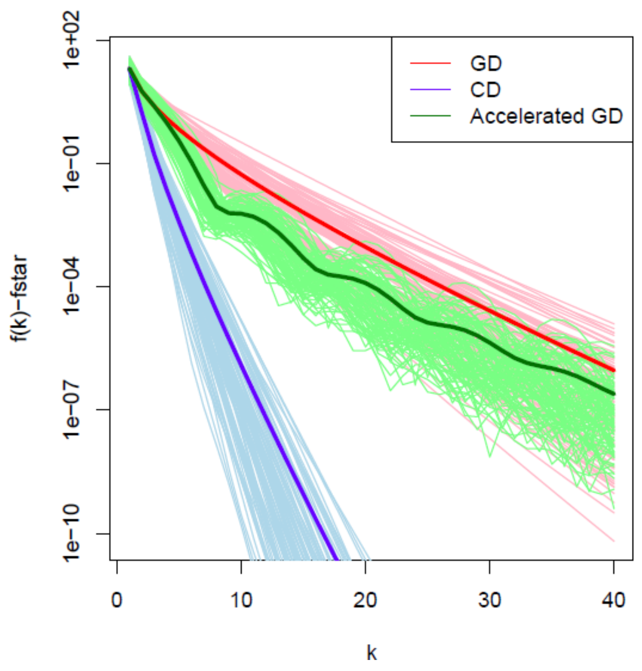

Figure 25.2: A comparison of coordinate descent (blue), gradient descent (red) and accelerated descent (green) on 100 random instances of a linear regression problems ($n = 100$, $p = 20$).

25.5.3 Box Constrained Regression

We want to perform linear regression given the constraint that for all i $-s \le x_i \le s$.

$$\begin{aligned} \min_{x \in \mathcal{R}^n} \quad & \frac{1}{2}||y - Ax||^2 \\ \text{subject to} \quad & ||x||_\infty \le s \end{aligned}$$

By minimizing over x_i while keeping all other coordinates fixed we get the update rule

$$x_i = T_s\left(\frac{A_i^T\left(y - A_{-i}x_{-i}\right)}{A_i^T A_i}\right)$$

T_s is the truncating operator whose value is the same as the input between $-s$ and $+s$, equal to s for inputs greater than s and equal to $-s$ for inputs less than $-s$.

25.5.4 Support Vector Machines

The dual formulation of the support vector machine optimization problem is

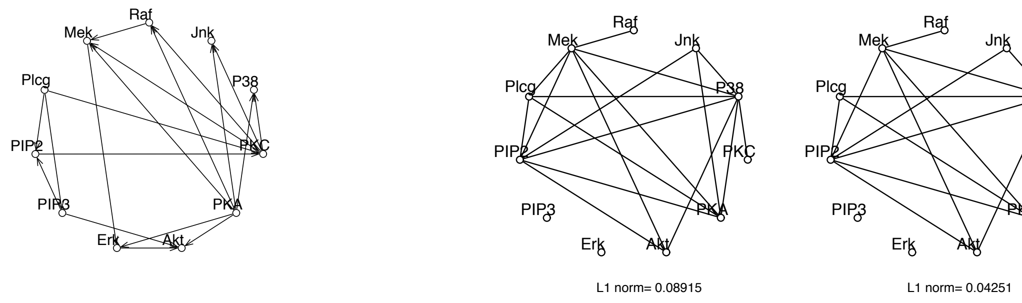

Figure 25.3: Graphical lasso example, the true graph (left) and estimated graphs (right).

$$\begin{aligned} \min_{\alpha \in \mathcal{R}^n} \quad & \frac{1}{2}\alpha^T K \alpha - 1^T \alpha \\ \text{subject to} \quad & 0 \leq \alpha \leq C \\ & y^T \alpha = 0 \end{aligned}$$

The $y^T \alpha = 0$ constraint is non separable, but it can still be shown that a coordinate descent strategy called sequential minimal optimization (SMO) can solve the problem. The complementary slackness conditions for the primal are

$$\begin{aligned} \alpha_i[(Av)_i - y_i d - (1 - s_i)] = & \quad 0 \qquad i = 1, ..., n \\ (C - \alpha_i)s_i = & \quad 0 \qquad i = 1, ..., n \end{aligned}$$

where v is the primal coefficients, d is the primal intercept and s is a vector of slacks. SMO works over blocks of two coordinates at a time. Pairs of coordinates that do not satisfy the complementary slackness conditions are chosen and jointly optimized over while respecting the $y^T \alpha = 0$ constraint and the complementary slackness constraint pair for the two variables. This 2 element block minimization step reduces to minimizing a quadratic over a line segment. Instead of chosing coordinate pairs sequentially, SMO picks the next pair to optimize over greedily using heuristics.

For more information on the SMO method including choice of heuristics, block minimization and convergence proofs refer to [P98].

25.5.5 Graphical Lasso

This is the application of coordinate descent to a problem that is "pretty hard to solve" in general. Given a matrix $A \in \mathbb{R}^{n \times p}$ of n observations from a p-dimensional multivariate Gaussian, $\mathcal{N}(0, \Sigma)$, we want to estimate the unknown covariance matrix Σ.

For a Gaussian distribution, we have $\Sigma^{-1}_{ij} = 0$ if and only if variables are i and j are conditionally independent given the other variables. Note that in the language of graphical models, this is referred to as a Gaussian Markov random field and the problem of estimating the inverse covariance is equivalent to estimating the edges between the nodes in this graph. For an example from [FHT07], see Figure 25.3.

Under a zero-mean Guassian distribution, we can formulate the ℓ_1 penalized maximum likelihood problem as

$$\underset{\Theta \in \mathcal{R}^{p \times p}}{\text{minimize}} - \log \det \Theta + \operatorname{tr} S\Theta + \lambda \|\Theta\|_1$$

where S is the empirical covariance matrix, $S = \frac{1}{n} A^T A$. Its straightforward to see that that stationarity KKT conditions are given by

$$-\Theta^{-1} + S + \lambda \Gamma = 0$$

where $\Gamma_{ij} \in \partial |\Theta_{ij}|$. Let $W = \Theta^{-1}$ and then partition the matrix blockwise

$$W = \begin{bmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \qquad \Theta = \begin{bmatrix} \Theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \qquad S = \begin{bmatrix} S_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} \qquad \Gamma = \begin{bmatrix} \Gamma_{11} & \gamma_{12} \\ \gamma_{21} & \gamma_{22} \end{bmatrix}$$

where $W_{11} \in \mathbb{R}^{(p-1)\times(p-1)}$, $w_{12} \in \mathbb{R}^{(p-1)\times 1}$, $w_{21} \in \mathbb{R}^{1\times(p-1)}$, and $w_{22} \in \mathbb{R}$. This allows us to solve for a single row (or column—doesn't matter due to symmetry) while holding the others fixed. Note that for the diagonal elements of W, we have $W_{ii} = S_{ii} + \lambda$ because we have $\Theta_{ii} > 0$ at the solution. Therefore, we solve for w_{12}, and then cycle through the columns by holding the other ones fixed, etc.

Considering the just $(1, 2)$ block of the KKT conditions we have

$$-w_{12} + s_{12} + \lambda \gamma_{12} = 0$$

Because

$$\begin{bmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} \Theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & 1 \end{bmatrix}$$

we have $w_{12} = -W_{11}\theta_{12}/\theta_{22}$, which we can subsitute back into the above equation to get

$$W_{11} \frac{\theta_{12}}{\theta_{22}} + s_{12} + \lambda \gamma_{12} = 0$$

But letting $x = \theta_{12}/\theta_{22}$ and noting that $\theta_{22} > 0$ at the solution, this can be written as

$$W_{12} x + s_{12} + \lambda \rho = 0$$

where $\rho \in \partial \|x\|_1$ which is exactly the KKT conditions for

$$\text{minimize}\, x^T W_{11} x + s_{12}^T x + \lambda \|x\|_1$$

which is a lasso problem that can be solved quickly via coordinate descent. So in summary we can solve the graphical lasso problem by applying coordinate descent to get a series of lasso problems.

Note that by reparameterizing in terms of W, and then solving in terms of blocks of W, these *do not* correspond to blocks Θ so this is not technically a coordinate descent algorithm. However, there is a nice result from Mazumder et al. [MH11] that shows that this is actually performing coordinate ascent in the dual problem so everything is okay.

25.6 Pathwise coordinate descent for lasso

Here we exploit sparsity in the in the problem to make the algorithm more efficient.

Outer loop:

- Compute the solution at sequence $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$

- For λ_k, initialize coordinate descenta lgorithm at the compute solution for λ_{k+1} (warm start)

Inner loop:

- Perform one (or a small number) of complete cycles and record the active set S of coefficients that are nonzero
- Cycle over coefficients S until convergence
- Check KKT conditions over all coefficients; if not all satisified, add offending coefficients to S and go back one step

With these strategies in place, coordinate descent is competitive with fastest algorithms for lasso problems. Freely available in the "glmnet" package in MATLAB and R.

25.7 No convergence rates

There is a dearth of theory for coordinate descent especially when it comes to convergence rates—theory is sorely needed. In general, we do not have global convergence rates. In recent work, Saha et al. [ST10] consider a restricted class of problems

$$f(x) = g(x) + \lambda\|x\|_1$$

with g convex. However, they assume that

- ∇g Lipschitz with constant $L > 0$ and $I - \nabla g/L$ montone increasing in each component
- there is a z such that $z \geq S_\lambda(z - \nabla g(z))$ or $z \leq S_\lambda(z - \nabla g(z))$

The first condition is "a little bit funny." A sufficient condition for this is that the off-diagonal elements of the Hessian of g at any x are nonpositive. The second condition follows from the first condition, as is shown in the paper.

They show that for coordinate descent starting at $x^{(0)} = z$ and generalized gradient descent starting at $y^{(0)} = z$ with step size $1/L$, we have

$$f(x^{(k)}) - f(x^\star) \leq f(y^{(k)}) - f(x^\star) \leq \frac{L\|x^{(0)} - x^\star\|^2}{2k}$$

In other words, coordinate descent dominates generalized gradient descent at every iteration. It's not clear if this fully explains the behavior that we see in practice or how big this class of problems is due to the assumptions.

References

[FHT07] J. Friedman, T. Hastie and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso."

[MH11] R. Mazumder and T. Hastie, "The graphical lasso: new insights and alternatives."

[MFH11] R. Mazumder, J. Friedman and T. Hastie, "SparseNet: coordinate descent with non-convex penalties"

[P98] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," *Advances in Kernel Methods*, pp. 185-208.

[ST10] A. Saha and A. Tewari, "On the finite time convergence of cyclc coordinate descent methods."

[T01] P. Tseng, "Convergence of Block Coordinate Descent Method for Nondifferentiable Minimization," *Journal of Optimization Theory and Applications*, Vol 109, Issue 3.