10-725: Optimization **Fall 2013**

Lecture 10: October 1

Lecturer: Barnabas Poczos/Ryan Tibshirani *Scribes: Jennifer King, Yitong Zhou*

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

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

10.1 Conjugate Direction Methods

Goal: We want to accelerate steepest descent without needing to use the inverse of the Hessian (as Newtons method does).

The methods were originally developed to solve the following problem:

$$\min_{x \in R^n} \frac{1}{2} x^T Q x - b^T x \Leftrightarrow Qx = b, x \in R^n$$

where $Q \succ 0$ and is symmetric. We can say the vectors $\{d_1, ..., d_k\}$ are Q-orthogonal (Q-conjugate directions) if and only if

$$d_i^T Q d_j = 0 \ \forall i \neq j$$

In the previous lecture we saw a proof of the following Lemma:

Lemma 10.1 *If $Q \succ 0$ and $\{d_1, ..., d_k\}$ are Q-conjugate then they are linearly independent.*

Using this lemma we can write:

$$x^* = \alpha_0 d_0 + ... + \alpha_k d_k$$

where

$$\alpha_i = \frac{d_i^T b}{d_i^T Q d_i}$$

The conjugate direction theorem means now we can do iterations to solve for x^*:

$$x_{k+1} = x_k + a_k d_k$$

$$a_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$$

that $x^* = x_n \implies$ convergence in n steps. So we can say that given $\{d_1 ... d_k\}$ Q-conjugate direction vectors we can solve $f(x) = \frac{1}{2} x^T Q x - b^T x$ without the need for the Hessian inverse.

10.2 Conjugate Gradient Method

Now we want to know how to generate the Q-conjugate directions: $\{d_1, ..., d_k\}$. The following describes the algorithm for doing this in the quadratic case ($f(x) = \frac{1}{2}x^TQx - b^Tx$ and $g(x) = \nabla f(x) = Qx - b$):

- Let $x_0 \in \mathbb{R}^n$ be an arbitrary start point.
- $d_0 = -g_0 = b - Qx_0$
- Iterate:
  1. $\alpha_k = \frac{-g_k^T d_k}{d_k^T Q d_k}$
  2. $x_{k+1} = x_k + \alpha_k d_k$
  3. $g_{k+1} = Qx_k - b$
  4. $\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$
  5. $d_{k+1} = -g_{k+1} + \beta_k d_k$

Note that unlike Newton's method, no matrix inversion is necessary.

Theorem 10.2 *Conjugate Gradient Theorem - The conjugate gradient algorithm is a conjugate direction method. The following properties hold:*

1. $span\,(g_0, ..., g_k) = span\left(g_0, Q^1 g_0, ..., Q^k g_0\right)$
2. $span\,(d_0, ..., d_k) = span\left(g_0, Q^1 g_0, ..., Q^k g_0\right)$
3. $d_k^T Q d_i = 0 \forall i \neq k$
4. $\alpha_k = \frac{g_k^T g_k}{d_k^T Q d_k}$
5. $\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$

10.2.1 Extension to non-quadratic case

Our goal is to modify the algorithm so that it can be generalized to non-quadratic functions. If we use line search instead to update α_k, then based on the above properties only g_k depends on Q. When dealing with non-quadratic functions, we can substitute in the gradient of our function:

$$g_k = \nabla f(x_k)$$

If we do not wish to use line search to find α_k we can use the Hessian of our function for Q:

$$Q_k = \nabla^2 f(x_k)$$

Now we can write a generalized algorithm:

- Step 1. $x_0 \in \mathbb{R}^n, g_0 = \nabla(x_0), d_0 = -g_0$
- Step 2. Iterate n times:

1. $\alpha_k = \frac{-g_k^T d_k}{d_k^T \nabla^2 f(x_k) d_k}$
2. $x_{k+1} = x_k + \alpha_k d_k$
3. $g_{k+1} = \nabla f(x_{k+1})$
4. $\beta_k = \frac{g_{k+1}^T \nabla^2 f(x_k) d_k}{d_k^T \nabla^2 f(x_k) d_k}$
5. $d_{k+1} = -g_{k+1} + \beta_k d_k$

- Step 3. Go back to Step 1. and use x_n as the new x_0

This generalized algorithm exhibits the following properties:

- No line search is required
- For the quadratic problem, convergence is achieved in a finite number of steps.
- The Hessian must be evaluated at every step.
- Not globally convergent.

10.2.2 Fletcher-Reeves Method

The Fletcher-Reeves method makes two modifications to the algorithm listed above.

1. $x_{k+1} = \alpha_k d_k$ where $\alpha_k = \arg\min_\alpha f(x_k + \alpha d_k)$
2. compute $g_{k+1} = \nabla f(x_{k+1})$
3. if $k < n-1$ then we compute: $d_{k+1} = -g_{k+1} + \beta_k d_k$, and $\beta_k = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$ (equivalent to Hessian way)

These modifications introduce the need for a line search. However, the Hessian is no longer needed.

10.2.3 Polak-Ribiere Method

The Polak-Ribiere method uses the same search to find α_k as the Fletcher-Reeves method, but modifies the β_k update rule:

$$\beta_k = \frac{(g_{k+1} - g_k)^T g_{k+1}}{g_k^T g_k}$$

There is not theoretical proof behind this, but in application it has shown to work better. The intuition behind why is that it uses more moments. Under some conditions, the convergence rate that has been shown is the following:

$$\|x_{k+n} - x^*\| \leq c\|x_k - x^*\|^2$$

In other words, after n steps the error will have a quadratic rate. Additionally, under some conditions, line search methods are globally convergent.

10.3 Quasi-Newton Methods

As we noted earlier, one big problem with Newtons method is that it requires the inverse of the Hessian:

$$x_{k+1} = x_k - \alpha_k \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$$

The goal behind quasi-newton methods is to develop approximations of that inverse. Then we can modify the update rule for Newtons method to the following:

$$x_{k+1} = x_k - \alpha_k S_k \nabla f(x_k)$$

Where $\alpha_k > 0$ and S_k is the approximation of the inverse. We note that if $S_k = I$ then this update is the gradient descent update. If $S_k = \left[\nabla^2 f(x_k)\right]^{-1}$ then this update is exactly Newtons update.

Lemma 10.3 $S_k \succ 0 \implies$ *modified Newton step is a descent direction.*

Proof: $\nabla f(x_k)^T (x_{k+1} - x_k) = \nabla f(x_k)^T (-\alpha_k S_k \nabla f(x_k) < 0$ ■

Lets look at the quadratic function: $f(x) = \frac{1}{2} x^T Q x - b^T x$. We can say:

$$g_k = Q x_k - b = \nabla f(x_k)$$

$$x_{k+1} = x_k - \alpha_k S_k g_k$$

Lemma 10.4

$$\alpha_k = \min_\alpha f(x_k - \alpha S_k g_k)$$

$$\implies \alpha_k = \frac{g_k^T S_k g_k}{g_k^T S_k Q S_k g_k}$$

Proof:

$$\begin{aligned}
f(x_{k+1}) &= \frac{1}{2} x_{k+1}^T Q x_{k+1} - b^T x_{k+1} \\
&= \frac{1}{2} (x_k - \alpha_k S_k g_k)^T Q (x_k - \alpha_k S_k g_k) - b^T (x_k - \alpha_k S_k g_k) \\
0 &= \frac{\partial f(x_{k+1})}{\partial \alpha_k} \\
&= -g_k^T S_k Q (x_k - \alpha_k S_k g_k) + b^T S_k g_k \\
g_k^T S_k Q x_k - g_k^T S_k b &= \alpha_k g_k^T S_k Q S_k g_k \\
g_k^T S_k (Q x_k - b) &= \alpha_k g_k^T S_k Q S_k g_k \\
g_k^T S_k g_K &= \alpha_k g_k^T S_k Q S_k g_k \\
\alpha_k &= \frac{g_k^T S_k g_k}{g_k^T S_k Q S_k g_k}
\end{aligned}$$

■

Examining the convergence rate for the quadratic case we see the error is:

$$\epsilon(x_k) = \frac{1}{2} (x_k - x^*)^T Q (x_k - x^*)$$

Then at every step k:

$$\epsilon(x_{k+1}) \leq \left(\frac{B_k - b_k}{B_k + b_k}\right)^2 \epsilon(x_k)$$

where B_k and b_k are the largest and smallest eigenvalues for $S_k Q$. We note that S_k^{-1} close to $Q \implies$ fast convergence because B_k will be close to b_k.

10.3.1 Classical Modified Newton

The classical modified newtons method uses the Hessian at x_0 to approximate the Hessian at every step:

$$\nabla^2 f(x_k) = \nabla^2 f(x_0) \ , \forall k$$

10.3.2 Constructing the Inverse of the Hessian

The following sections describe various methods for constructing an approximation to the inverse of the Hessian. First we introduce the following notation which will be used throughout the remainder of the section:

$$g_{k+1} = \nabla f(x_{k+1})$$

$$p_k = x_{k+1} - x_k$$

$$Q(x_k) = \nabla^2 f(x_k)$$

$$q_k = g_{k+1} - g_k = \nabla f(x_{k+1}) - \nabla f(x_k) \approx Q(x_k) p_k$$

For the quadratic case $g_k = \nabla f(x_k) = Qx_k - b$ and $q_k = g_{k+1} - g_k = Q(x_{k+1} - x_k) = Qp_k$. If we let $H = Q^{-1}$ then:

$$[q_0, q_1, ..., q_{n-1}] = Q\,[p_0, ..., p_{n-1}]$$

$$\implies Q = [q_0, ..., q_{n-1}]\,[p_0, ..., p_{n-1}]^{-1}$$

$$\implies H\,[q_0, ..., q_{n-1}] = [p_0, ..., p_{n-1}]$$

Here $\{p_0, ..., p_{n-1}\}$ are linearly independent, and $\{q_0, ..., q_{n-1}\}$ are known. Thus we can say that we can construct H_k which approximates H based on data from the first k steps.

10.3.2.1 Symmetric Rank One Correction (SR1)

We want:

$$H_{k+1}\,[q_0 ... q_k] = [p_0 ... p_k]$$

Let

$$H_{k+1} = H_k + a_k z_k z_k^T$$

where $a_k \in \mathbb{R}$ and $z_k \in \mathbb{R}^n$. Putting these two equations for H_{k+1} together we get the following theorem:

Theorem 10.5

$$H_{k+1} = H_k + \frac{(p_k - H_k q_k)(p_k - H_k q_k)^T}{q_k^T (p_k - H_k q_k)}$$

Proof: Given $[p_0...p_k] = H_{k+1}[q_0...q_k]$ and $H_{k+1} = H_k + a_k z_k z_k^T$:

$$p_k = H_{k+1}q_k = (H_k + a_k z_k z_k^T)q_k = H_k q_k + a_k z_k z_k^T q_k$$

$$p_k - H_k q_k = a_k z_k z_k^T q_k$$

$$(p_k - H_k q_k)(p_k - H_k q_k)^T = a_k^2 (z_k z_k^T q_k q_k^T z_k z_k^T)$$

$$\frac{(p_k - H_k q_k)(p_k - H_k q_k)^T}{a_k} = a_k z_k (z_k^T q_k)^2 z_k^T$$

$$\frac{(p_k - H_k q_k)(p_k - H_k q_k)^T}{a_k (z_k^T q_k)^2} = a_k z_k z_k^T$$

So we can say:

$$H_{k+1} = H_k + \frac{(p_k - H_k q_k)(p_k - H_k q_k)^T}{a_k (z_k^T q_k)^2}$$

Now lets look at:

$$q_k^T p_k = q_k^T H_{k+1} q_k = q_k^T (H_k + a_k z_k z_k^T) q_k$$

$$= q_k^T H_k q_k + a_k q_k^T z_k z_k^T q_k$$

$$= q_k^T H_k q_k + a_k (q_k^T z_k)^2$$

$$a_k (q_k^T z_k)^2 = q_k^T (p_k - H_k q_k)$$

Putting the two together we get:

$$H_{k+1} = H_k + \frac{(p_k - H_k q_k)(p_k - H_k q_k)^T}{q_k^T (p_k - H_k q_k)}$$

■

We can now construct the SR1 algorithm:

1. $\alpha_k = \arg\min_\alpha f(x_k - \alpha H_k g_k)$
2. $g_k = \nabla f(x_k)$
3. $x_{k+1} = x_k - \alpha_k H_k g_k$
4. $g_{k+1} = \nabla f(x_{k+1})$
5. $p_k = x_{k+1} - x_k$
6. $q_k = g_{k+1} - g_k$
7. $H_{k+1} = H_k + \frac{(p_k - H_k q_k)(p_k - H_k q_k)^T}{q_k^T (p_k - H_k q_k)}$

We note two potential problems with the update rule in step 7:

- The algorithm can become unstable if $g_k^T (p_k - H_k q_k)$ is close to 0.
- There is no guarantee that H_k will be positive definite.

10.3.2.2 Davidson-Fletcher-Powell Method (DFP)

The DFP method attempts to refine the method above and provide a way to guarantee H_k is positive definite at each iteration. The algorithm is as follows:

- Initialize $H_0 \in \mathbb{R}^{n \times m}$ to be symmetric and positive definite.
- Iterate:
  1. $d_k = -H_k g_k$
  2. $\alpha_k = \arg\min_\alpha f(x_k + \alpha d_k)$
  3. $x_{k+1} = x_k + \alpha_k d_k$
  4. $p_k = \alpha_k d_k$
  5. $g_{k+1} = \nabla f(x_{k+1})$
  6. $H_{k+1} = H_k + \frac{p_k p_k^T}{p_k^T q_k} - \frac{H_k q_k q_k^T H_k}{q_k^T H q_k}$

Now we can say that H_k is positive definite at every iteration. Additionally, if f is quadratic with positive definite Hessian Q then $p_i Q p_j = 0$ for all $0 \leq i \leq j \leq k$. Thus $H_{k+1} Q p_j = p_j$ for $0 \leq j \leq k$. Additionally it can be shown that $H_n = Q^{-1}$ for some n.

10.3.2.3 Broyden-Fletcher-Goldfarb-Shanno Method (BFGS)

If we switch p_k and q_k in the update for H provided by the DFP method we get an update rule for the Hessian Q:

$$Q_{k+1} = Q_k + \frac{q_k q_k^T}{q_k^T p_k} - \frac{Q_k p_k p_k^T Q_k}{p_k^T Q_k q_k}$$

However, we want the inverse. And with Sherman-Morrison formula:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^t A^{-1} u}$$

we apply it twice and get:

$$H_{k+1} = H_k + \left(1 + \frac{q_k^T H_k q_k}{p_k^T q_k}\right) \frac{p_k p_k^T}{p_k^T q_k} - \frac{p_k q_k^T H_k + H_k q_k p_k^T}{q_k^T p_k}$$

Then the BFGS algorithm is the following:

- Initialize $H_0 \in \mathbb{R}^{nxn}$ to a symmetric, positive definite matrix.
- Initialize $x_0 \in \mathbb{R}^n, k = 0, g_k = \nabla(f(x_k))$.
- Iterate
  1. $d_k = -H_k g_k$
  2. $\alpha_k = \arg\min_{\alpha > 0} f(x_k + \alpha d_k)$
  3. $x_{k+1} = x_k + \alpha_k d_k$
  4. $p_k = \alpha_k d_k$

5. $g_{k+1} = \nabla f(x_{k+1})$
6. $q_k = g_{k+1} - g_k$
7. $H_{k+1} = H_k + \left(1 + \frac{q_k^T H_k q_k}{p_k^T q_k}\right) \frac{p_k p_k^T}{p_k^T q_k} - \frac{p_k q_k^T H_k + H_k q_k p_k^T}{q_k^T p_k}$

In practice this works better than DFP.