Optimization Methods		Date:	*27-03-2018*
Instructor: *Sujit Prakash Gujar*		Scribes:	Rashmi Kethireddy Mayank Modi Tirth Maniar

Lecture 20: Quadratic Programming - Conjugate Gradient Algorithm

1 Recap

1.1 Line search methods

The rate at which an algorithm converges to minimum is given by α which is the step size. One way is to find gradient of $g(\alpha_k) = f(x_k + \alpha_k d_k)$ and equate to zero to find α_k which minimizes the function and maximizes the reduction. Analytically it is not possible always to find the appropriate α_k. So, we go for line searching algorithms to maximize step size.

Line search methods

1. Quadratic Interpolation: This method tries to fit a quadratic polynomial between 3 points a,b,c satisfying the conditions $a < b < c$ s.t. $g(a) > g(b)$ and $g(b) < g(c)$. The local minima of $g(\alpha)$ lies between the points a, c.
2. Cubic Interpolation: Similar to Quadratic Interpolation with cubic order and with only 2 points a,b. These points should satisfy the conditions $a < b$, s.t. $g'(a) < 0$ and either of following is true:

   (i) $g'(b) > 0$.

   (ii) $g(b) \leq g(a)$.
3. Golden Section Method
4. Armijo rule : Armoji rule ensures sufficient reduction in function value. A step size α_k is choose such that it satisfies Arjimo's rule :

$$g(\alpha_k) = f(x_k + \alpha_k d_k) \leq f(x_1) + c_1 \alpha_k d_k^T \triangledown f(x_k)$$

5. Armijo-Wolfe rule: Another condition together with Armijo rule is enforced to ensures convergence of the gradient of the function to 0. This is also called as the curvature condition.

$$d_k^T \triangledown f(x_k + \alpha_k d_k) \geq c_2 d_k^T \triangledown f(x_k)$$

2 Conjugate Gradient Method

If the function is quadratic, steepest descent method converges very slowly while the newton method converges in only one step. But, newton method has instability as it involves computing an inverse of an hessian matrix in each step. Matrix inversion is in general a slow as well as an unstable operation. So, a new approach *conjugate gradient* method which converges faster and overcome the challenges of newton method.

Conjugate gradient method was proposed by Magnus Hestenes and Eduard Stiefel in 1952 to solve quadratic equations of the form:

$$\min \quad \frac{1}{2} x^T Q x - b^T x + c$$

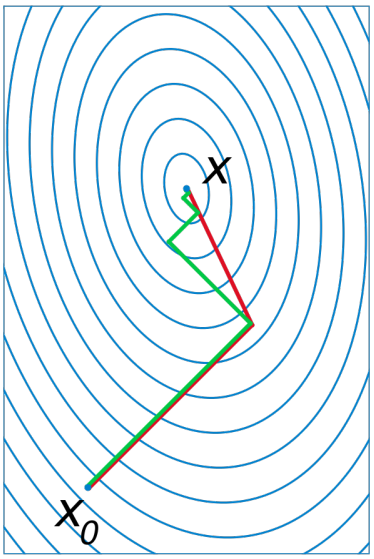

Figure 1: A comparison of the convergence of gradient descent with optimal step size (in green) and conjugate vector (in red) for minimizing a quadratic function associated with a given linear system.

$$\text{subject to } \ x \in \mathbb{R}^n$$

where Q is symmetric and positive-definite. Solving it by equating gradient to zero.

$$\nabla f(x) = 0$$

$$\implies Qx = b$$

computing x^* from $Qx = b$ is computationally unstable as it has to compute Q^{-1}. The next approach is cholesky decomposition but Q may be very sparse and solving sparse matrix using cholesky decomposition is very slow. If n is very large cholesky decomposition might become more slower. The goal in conjugate gradient method is to solve optimization problem much faster, as many real life applications such as search engines, the results should be calculated in less than a milli-second.

Definition 2.1. *Conjugate Directions: u,v are called conjugate with respect to a real, symmetric matrix A if* $u^T Av = 0$.

If A is symmetric and positive-definite then $< u, v >_A =< Au, v >=< u, Au >=< u^T Av >$. u,v are conjugate if they are orthogonal to each other with respect to the above inner product.

Lemma 2.1. *If* d_1,d_2,...d_n *are Q-conjugate with respect to each other, then they are linearly independent.*

Proof. To prove linear independence between vectors $d_1,\ldots,d_n$. We need to prove $\sum_{i=1}^{n} \alpha_i d_i = 0 \implies \alpha_i = 0$.

$$\text{Let, } \sum_{i=1}^{n} \alpha_i d_i = 0$$

Multiply $d_j^T Q$ on both sides

$$(d_j^T Q) \sum_{i=1}^{n} \alpha_i d_i = 0$$

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

$$\therefore \alpha_j d_j^T Q d_j = 0$$

$\because Q$ is positive definite, $d_j^T Q d_j > 0$

$$\implies \alpha_j = 0, \forall j$$

Hence proved.

□

Definition 2.2. *If d_1,d_2,...,d_n are Q-conjugate directions, then they form a basis.*

Since, d_1,...,d_n are linearly independent vectors spanned over $\mathbb{R}^n$

$$x^* = \sum_{i=1}^{n} \alpha_i d_i$$

Multiplying with Q on both sides

$$\therefore Qx^* = \sum_{i=1}^{n} \alpha_i Q d_i$$

$$\therefore b = \sum_{i=1}^{n} \alpha_i Q d_i$$

Multiplying with d_j^T on both sides

$$\therefore d_j^T b = \sum_{i=1}^{n} \alpha_j d_j^T Q d_i$$

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

$$d_j^T b = \alpha_j d_j^T Q d_j$$

$$\implies \alpha_j = \frac{d_j^T b}{d_j^T Q d_j}$$

α_k is the step size in each iteration. Start with some direction d_0 and update x^* in each iteration with a new direction d_k. By n^{th} iteration, it will reach an optimal solution.

We want to find x such that $Qx = b$. If our guess at k^{th} iteration is x_k, we define residue $r_k = b - Qx_k$. Observe that, $r_k = -\nabla f(x_k)$.

$$r_k = b - Qx_k = -\nabla f(x_k)$$

$$g(\alpha_k) = f(x_k + \alpha_k d_k)$$

To get maximum reduction in r_k choose α_k such that $g'(\alpha) = 0$.

$$\nabla f(x_k + \alpha_k d_k)^T d_k = 0$$

$$[Q(x_k + \alpha_k d_k) - b]^T d_k = 0$$

$$x_k^T Q d_k + \alpha_k d_k^T Q d_k = b^T d_k$$

$\because x_k = x_0 + \alpha_0 d_0 + \alpha_1 d_1 + \cdots + \alpha_{k-1} d_{k-1}$ and lets assume we start with $x_0 = 0$ and $(\alpha_0 d_0 + \alpha_1 d_1 + \cdots + \alpha_{k-1} d_{k-1})^T Q d_k = 0$

$$\alpha_k d_k^T Q d_k = b^T d_k$$

$$\implies \alpha_k = \frac{b^T d_k}{d_k^T Q d_k}$$

$\because r_k = b - Qx_k$, α_k can also be written as

$$\alpha_k = \frac{r_k^T d_k}{d_k^T Q d_k}$$

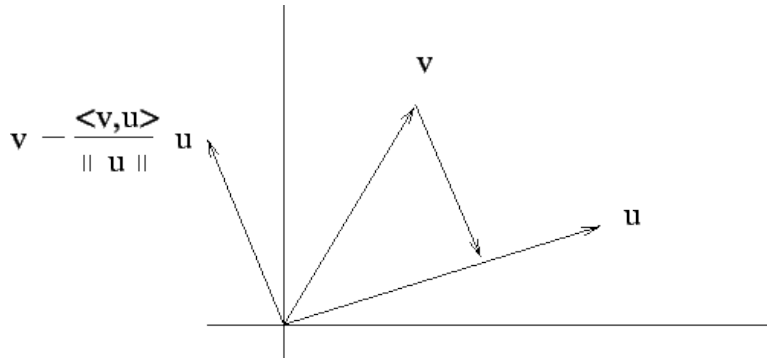

Figure 2: Gram-Schmidt orthogonal process (source : NPTEL)

Definition 2.3. *Gram-Schmidt Orthogonalisation : Let V be a finite dimensional inner product space. Suppose $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ is a linearly independent subset of V. Then the Gram-Schmidt orthogonalisation process uses the vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n$ to construct new vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ such that $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$ for $i \neq j$, $\|\mathbf{v}_i\| = 1$*

Based on Gram-Schmidt orthogonal basis, d_{k+1} is computed by subtracting vector projection of r_{k+1} on to unit vector $\frac{Qd_k}{d_k^T Q d_k}$. The updation rule for d_{k+1} is as follows.

$$\beta_k = -\frac{r_{k+1}^T Q d_k}{d_k^T Q d_k}$$

$$d_{k+1} = r_{k+1} + \beta_k d_k$$

2.1 Conjugate Gradient Algorithm

1. $k = 0$,$x_0 = 0$
   $r_0 = -\nabla f(x_0) = b - Qx_0 = b$
   If $r_0 = 0$ then STOP
2. $d_0 = r_0$
3. $\alpha_k = \frac{r_k^T d_k}{d_k^T Q d_k}$
4. $x_{k+1} = x_k + \alpha_k d_k$
5. If $r_{k+1} = b - Q_{k+1} = 0$, STOP
6. $\beta_k = \frac{-r_{k+1}^T Q d_k}{d_k^T Q d_k}$
7. $d_{k+1} = r_{k+1} + \beta_k d_k$
8. $k = k + 1$, go to step 3

Will it loop infinitely? No, It takes maximum of n iterations as it can generate only n Q-conjugate directions for an n-dimensional space.

2.2 Example Code in MATLAB

```
function[x] = conjgrad(A, b, x)
    r = b − A ∗ x;
    p = r;
    rsold = r′ ∗ r;
    for i = 1 : length(b)
        Ap = A ∗ p;
        alpha = rsold/(p′ ∗ Ap);
        x = x + alpha ∗ p;
        r = r − alpha ∗ Ap;
        rsnew = r′ ∗ r;
        if sqrt(rsnew) < 1e−10
            break;
        end
        p = r + (rsnew/rsold) ∗ p;
        rsold = rsnew;
    end
end
```

2.3 Example 1

Consider quadratic minimization problem
Minimize $f(x, y) = x^2 + \frac{y^2}{2} - x - y$
To solve it using conjugate gradient method, represent f(x,y) in the form of $\frac{1}{2}x^TQx - b^Tx + c$

$$f(x,y) = \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} 1 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$

where $Q = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$, $X = \begin{bmatrix} x \\ y \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
Since, Q is symmetric,positive definite matrix we can apply conjugate gradient method to find optimal value.

Iteration 1:

Step 1: $k = 0$, x_0=0
$r_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Step 2: $d_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

Step 3: $\alpha_0 = \frac{r_0^T d_0}{d_0^T Q d_0}$
$= \frac{2}{3}$

Step 4: $x_1 = x_0 + \alpha_0 d_0$
$= \begin{bmatrix} \frac{2}{3} \\ \\ \frac{2}{3} \end{bmatrix}$

Step 5: $r_1 = b - Qx_1$
$= \begin{bmatrix} -\frac{1}{3} \\ \\ \frac{1}{3} \end{bmatrix}$

Step 6: $\beta_0 = \frac{-r_1^T Q d_0}{d_0^T Q d_0}$
$= \frac{1}{9}$

Step 7: $d_1 = r_1 + \beta_0 d_0$
$= \begin{bmatrix} -\frac{2}{9} \\ \\ \frac{4}{9} \end{bmatrix}$

Iteration 2:

Step 3: $\alpha_1 = \frac{r_1^T d_1}{d_1^T Q d_1}$
$= \frac{3}{4}$

Step 4: $x_2 = x_1 + \alpha_1 d_1$
$= \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$

Step 5: $r_2 = b - Qx_2$
$= \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

stop as residue $r = 0$ at $x = \begin{bmatrix} \frac{1}{2} \\ 1 \end{bmatrix}$

Note : Generally, conjugate gradient method reaches optimal solution in n steps where n is the dimension of the variable. But, this method can only be used if function is quadratic and Q is symmetric and positive definite.

2.4 Example 2

Note : A closer analysis of the algorithm shows that r_i is orthonormal to r_j , i.e. $r_i^T r_j = 0$, for $i \neq j$. And d_i is Q-orthogonal to d_j, i.e. $d_i^T Q d_j \neq 0$, for $i \neq j$. This can be regarded that as the algorithm progresses, d_i and r_i span the same Krylov subspace. Where r_i form the orthogonal basis with respect to standard inner product, and d_i form the orthogonal basis with respect to inner product induced by Q.

Consider the linear system $Qx = b$ given by

$$Qx = \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

we will perform two steps of the conjugate gradient method beginning with the guess

$$x_0 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

in order to find an approximate solution to the system.

Solution

For reference, the exact solution is

$$x = \begin{bmatrix} \frac{1}{11} \\ \\ \frac{7}{11} \end{bmatrix} \approx \begin{bmatrix} 0.0909 \\ 0.6364 \end{bmatrix}$$

Step 1: To calculate the residual vector r_0 associated with x_0. This residual is computed from the formula $r_0 = b - Qx_0$, and in our case is equal to

$$r_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -8 \\ -3 \end{bmatrix}$$

Since this is the first iteration, we will use the residual vector r_0 as our initial search direction d_0; the method of selecting d_k will change in further iterations.

$$\therefore d_0 = \begin{bmatrix} -8 \\ -3 \end{bmatrix}$$

Step 2: Compute the scalar α_0 using the relationship.

$$\alpha_0 = \frac{r_0^T r_0}{d_0^T Q d_0} = \frac{\begin{bmatrix} -8 & -3 \end{bmatrix} \begin{bmatrix} -8 \\ -3 \end{bmatrix}}{\begin{bmatrix} -8 & -3 \end{bmatrix} \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -8 \\ -3 \end{bmatrix}} = \frac{73}{331}$$

Step 3: Compute x_1 using the formula

$$x_1 = x_0 + \alpha_0 d_0 = \begin{bmatrix} 2 & 1 \end{bmatrix} + \frac{73}{331} \begin{bmatrix} -8 \\ -3 \end{bmatrix} = \begin{bmatrix} 0.2356 \\ 0.3384 \end{bmatrix}$$

This result completes the first iteration, the result being an "improved" approximate solution to the system, x_1. We may now move on and compute the next residual vector r_1 using the formula

$$r_1 = b - Qx_1 = b - Qx_0 - \alpha_0 Q d_0 = r_0 - \alpha_0 Q d_0 = \begin{bmatrix} -8 & -3 \end{bmatrix} - \frac{73}{331} \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -8 \\ -3 \end{bmatrix} = \begin{bmatrix} -0.2810 \\ 0.7492 \end{bmatrix}$$

Step 4: Compute the scalar β_0 that will eventually be used to determine the next search definition d_1.

$$\beta_0 = -\frac{r_1^T Q d_0}{d_0^T Q d_0} = -\frac{\begin{bmatrix}-0.2810 & 0.7492\end{bmatrix}\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}\begin{bmatrix}-8\\ -3\end{bmatrix}}{\begin{bmatrix}-8 & -3\end{bmatrix}\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}\begin{bmatrix}-8\\ -3\end{bmatrix}} = 0.0088$$

Step 5: Using the scalar β_0, compute the next search direction d_1 using the relationship

$$d_1 = r_1 + \beta_0 d_0 = \begin{bmatrix}-2810\\ 0.7492\end{bmatrix} + 0.0088\begin{bmatrix}-8\\ -3\end{bmatrix} = \begin{bmatrix}-0.3511\\ 0.7299\end{bmatrix}$$

Step 6: Now compute the scalar α_1 suing out newly acquired d_1 suing the relationship

$$\alpha_1 = \frac{r_1^T r_1}{d_1^T Q d_1} = \frac{\begin{bmatrix}-0.2810 & 0.7492\end{bmatrix}\begin{bmatrix}-0.2810\\ 0.7492\end{bmatrix}}{\begin{bmatrix}-0.3511 & 0.7229\end{bmatrix}\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}\begin{bmatrix}-0.3511\\ 0.7229\end{bmatrix}} = 0.4122$$

Step 7: Finally, we find x_2 using the same method as that used to find x_1.

$$x_2 = x_1 + \alpha_1 d_1 = \begin{bmatrix}0.2356\\ 0.3384\end{bmatrix} + 0.4122\begin{bmatrix}-0.3511\\ 0.7229\end{bmatrix} = \begin{bmatrix}0.0909\\ 0.6364\end{bmatrix}$$

Result: The result, x_2, is a "better" approximation to the system's solution than x_1 and x_0. If exact arithmetic were to be used in this example instead of limited-precision, then the exact solution would theoretically have been reached after n=2 iterations, where n is the order of the system.

References

[1] Dimitri P.Bertsekas *Nonlinear Programming.* Massachusetts Institute of Technology, 1995.

[2] Wikipedia on Conjugate gradient method,
`https://en.wikipedia.org/wiki/Conjugate_gradient_method`

[3] NPTEL lecture notes on Gram-Schmidt,
`http://www.nptel.ac.in/courses/122104018/node49.html`

[4] Wikipedia on Krylov subspace,
`https://en.wikipedia.org/wiki/Krylov_subspace`