Optimization Methods		Date:	*03/04/2018*
Instructor: *Sujit Prakash Gujar*		Scribes:	Aaron Jacob Varghese Divyani Sharma Anirudh Dahiya

Lecture 21: Conjugate Gradient Descent

1 Recap

In the last lecture we covered:

Conjugate Gradient Method:

It is a method useful for optimization of both linear and non-linear systems.It is an iterative algorithm used for very large systems where it is not practical to solve with a direct method.Equation should be of the form:

$$min f(x) = \frac{x^T Q x}{2} - b^T x$$

where $x \in R^n$ and Q is Symmetric and positive definite matrix.

The algorithm is as below

Algorithm 1: Conjugate Gradient Method

```
1  $x_0 \leftarrow 0$
2  $k \leftarrow 0$
3  $r_0 \leftarrow b - Qx_0 = b$
4  if $r_0 == 0$ STOP
5  $d_0 \leftarrow r_0$
6  $\alpha_k \leftarrow \frac{r_k^T d_k}{d_k^T Q d_k}$
7  $x_{k+1} \leftarrow x_k + \alpha_k d_k$
8  if $r_{k+1} = b - Qx_{k+1} = 0$ STOP
9  $\beta_k \leftarrow -\frac{r_{k+1}^T Q d_k}{d_k^T Q d_k}$
10 $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$
11 $k \leftarrow k + 1$
12 goto 6
```

Lemma 1: If $d_1, ..., d_n$ are conjugate with each other, then they are linearly independent.

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

Lemma 2: If $d_1, ..., d_n$ are Q-conjugate directions, then they form a basis.

2 Conjugate Gradient Descent

2.1 Lemma 1

Lemma : **If d_0, d_1, ... d_{n-1} be the Q-conjugate directions, then**

$$r_{k+1}^T\ d_i = 0\ \ \forall\ i = 0,1,2,...,k \tag{1}$$

Consider r_1 i.e. $i = 0$

$$r_1 = b \text{ - } Qx_1$$

$$r_1 = r_0 \text{ - } Q(x_0 + \alpha_0 d_0)$$

Since $x_0 = 0$

$$r_1 = r_0 \text{ - } Q(\alpha_0 d_0)$$

$$r_1^T\ d_0 = r_0^T\ d_0 \text{ - } \alpha_0\ d_0^T Q d_0 \tag{2}$$

$$\alpha_0 = \frac{r_0^T d_0}{d_0^T Q d_0}$$

Hence

$$r_1^T d_0 = 0$$

We need to prove by induction

$$r_{k+1}^T\ d_i = r_k^T\ d_i \text{ - } \alpha_k\ d_k^T Q d_i = 0 \tag{3}$$

With induction hypothesis as

$$r_k^T d_i = 0\ \forall\ i,\ i < k$$

Consider two cases

- $i < k$

  $$r_k^T d_i = 0 \text{ (i.e. the hypothesis)}$$

  $$d_k^T Q d_i = 0 \text{ (this is given)}$$

  Hence

  $$r_{k+1}^T d_i = 0$$

- $i = k$

$$r_{k+1}^T \ d_k = r_k^T \ d_k \text{ - } \alpha_k \ d_k^T Q d_k$$

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

Hence

$$r_{k+1}^T d_k = 0$$

2.2 Lemma 2

Lemma : If r_k denotes the residue in the k^{th} iteration of the Conjugate Gradient Descent Method, then, $\forall \ k$

$$r_{k+1}^T r_j = 0 \qquad\qquad \forall \ j = 0, 1, 2, ...k$$

We have

$$d_j = r_j + \beta_k d_{j-1}$$

which can be rewritten as

$$r_j = d_j - \beta_k d_{j-1}$$

Pre-multiplying with r_{k+1}^T on both sides

$$r_{k+1}^T r_j = r_{k+1}^T d_j - r_{k+1}^T \beta_k d_{j-1}$$

According to lemma 1 in 2.1

$$r_{k+1}^T d_j = 0$$

$$r_{k+1}^T \beta_k d_{j-1} = 0$$

Hence

$$r_{k+1}^T r_j = 0$$

2.3 Theorem

Let $d_0, d_1, ..., d_{n-1}$ be the directions generated in Conjugate gradient method.The theorem states that they are Q-Conjugate, i.e. linearly independent.

$$d_i^T Q d_j = 0 \qquad\qquad \text{if } i \neq j$$

Proof(Using Induction) :

Step 1 : w.k.t, directions are chosen orthagonal to each other,

$$d_0^T Q d_1 = 0$$

Step 2 :

Assume $d_0, d_1, ..., d_k$ are Q-conjugate, then $d_0, d_1, ..., d_k, d_{k+1}$ are also Q-conjugate.

we know that,

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

then,

$$\begin{aligned} d_{k+1} Q d_k &= (r_{k+1} + \beta_k d_k)^T Q d_k \\ &= r_{k+1} Q d_k + \beta_k d_k^T Q d_k \\ &= r_{k+1} Q d_k + \frac{-rk+1^T Q d_k}{d_k^T Q d_k} . d_k^T Q d_k \\ &= 0 \end{aligned}$$

Step 3 :

w.k.t, $\quad r_{j+1} = b - Qx_{j+1}$ also, $r_j = b - Qx_j$

then, $\quad r_{j+1} - r_j = -Q(x_{j+1} - x_j) = -\alpha_j Q d_j$

which gives, $\quad Qd_j = \dfrac{r_{j+1} - r_j}{\alpha_j} \qquad \alpha_j \neq 0$

Now,

$$\begin{aligned} d_{k+1}^T Q d_j &= \frac{d_{k+1}^T (r_{j+1} - r_j)}{-\alpha_j} \qquad \forall j < k \\ &= \frac{(r_{k+1} + \beta_k d_k)^T (r_{j+1} - r_j)}{-\alpha_j} \\ &= \frac{r_{k+1}^T r_{j+1} - r_{k+1}^T r_j + \beta_k^T d_k^T r_{j+1} - \beta_k^T d_k^T r_j}{-\alpha_j} = 0 \end{aligned}$$

from Lemma 2.1, first and second terms equates to zero and from lemma 2.2, remaining terms equates to zero.

Note : The Conjugate Gradient can be used if and only if the function is a Quadratic and Q is Symmetric Real Positive Definite.

2.4 Quasi-Newton Method

Previously we've learned Newton's method, where we update x by,

$$x_{k+1} = x_k - H_k^{-1} \nabla f(x_k)$$

where H is $\nabla^2 f(x)$. A computational drawback of Newtons method is the need to compute H_k^{-1} . To avoid the computation, the quasi-Newton methods use an approximation to H_k^{-1} in place of the true inverse.So,the update equation becomes,

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

where B_k is an n x n positive definite matrix and α_k is a positive search parameter. α_k should satisfy Armijo-Wolfe conditions.

To approximate the value of B_k in the next iteration, we use 2 type of update rules:

1. Rank 1 update rule:

$$B_{k+1} = B_k + U_k$$

where, $\qquad U_k = a_k y_k y_k^T$

where,$a_k \in R$ and $y_k \in R^n$

Notice, $rank(y_k y_k^T) = 1$, hence name rank 1.

Observe that if B_k is symmetric, then so is B_{k+1}.

Unfortunately, the rank one correction algorithm is not very satisfactory for several reasons,B_{k+1} may not be positive definite and thus d_{k+1} may not be a descent direction,or there may be numerical problems in evaluationg B_{k+1}. if we use a rank two update, then B_{k+1} is guaranteed to be positive definite for all k.

2. Rank 2 update rule:

$$B_{k+1} = B_k + U_k + V_k$$

Here both U_k and V_k are rank 1 matrices.

One of its example is BFGS.

2.5 Broyden, Fletcher, Goldfarb, and Shanno.

Here $H(x_0)$ is the Hessian matrix from Newton's method.

Step 1 :

$$B_0 = H(x_0)$$

$$d_0 = B_0^{-1} f(x_0)$$

$$k = 0$$

Step 2 : α_k satisfying armija wolfe condition.

Step 3 :

$$s_k = \alpha_k d_k$$

$$x_{k+1} = x_k + \alpha_k B_k^{-1} f(x_k)$$

$$y_{k+1} = f(x_{k+1}) - f(x_k)$$

$$B_{k+1} = B_k + \frac{r_k^T d_k}{d_k^T Q d_k}$$

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$$

This gives us the update rule for B_k. It can be computed efficiently, without temporary matrices, recognizing that B_k^{-1} is symmertic, and that $y_k^T B_k^{-1} y_k$ and $s_k^T y_k$ are scalar, using an expansion such as,

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(s_k^T y_k + y_k^T B_k^{-1} y_k)(s_k s_k^T)}{(s_k^T y_k)^2} - \frac{B_k^{-1} y_k s_k^{-1} + s_k y_k^T B_k^{-1}}{s_k^T y_k}$$

It's low memory implementation is called L-BFGS, which works with large N but is relatively slower.

2.6 Stochastic Gradient Descent

Suppose, $g_i(x)$ be the loss for ith data point. Now, the net loss to be minimised is:

$$min f(x) = \frac{1}{2} \sum_{i=1}^{m} g_i(x)^2$$

This gives,

$$\nabla f = \sum_{i=1}^{m} g_i \nabla g_i(x)$$

In Stochastic gradient descent, the true gradient is approximated by gradient at a single sample, thus giving the update equation as:

$$x_{k+1} = x_k + \alpha_k \nabla f_i(x)$$

$$x_{k+1} = x_k + \alpha_k g_i(x_k).\nabla g_i(x_k)$$

As the algorithm sweeps through the training set, it performs the above update for each training example. Several passes can be made over the training set until the algorithm converges. If this is done, the data can be shuffled for each pass to prevent cycles. Typical implementations may use an adaptive learning rate so that the algorithm converges.

Stochastic Gradient descent is known to run faster, but may lead to convergence at a suboptimal minima. Alternate approaches such as minibatch SGD avoid the problem of noise in learning by making updates for a minibatch of samples at each iteration, while being more efficient than full-batch methods.