EE 381V: Large Scale Optimization **Fall 2014**

Lecture 5 — September 11

Lecturer: Sujay Sanghavi *Scribe: Murat Kocaoglu, Abishek Sankararaman, Rajat Sen*

5.1 Recap: Gradient Descent using Exact Line Search

In the last class we discussed the gradient descent for solving an unconstrained convex optimization problem. If we are trying to minimize a function $f(x)$, then the descent step of a general gradient descent algorithm is of the form,

$$x^+ = x - \eta \nabla f(x) \tag{5.1}$$

where x^+ is the next iterate. Under the assumptions of strong convexity (Equation 5.2) , we discussed that $\eta = \frac{1}{M}$ guarantees a linear convergence (log - log linear) rate with $c = 1 - \frac{m}{M}$.

$$mI \preceq \nabla^2 f(x) \preceq MI \tag{5.2}$$

The issue with this approach is the value of M is unknown in general. Therefore, we resorted to *Exact Line Search*. In exact line search, we optimize for the value of η once we obtain a descent direction, in each iteration. In other words, at each iteration we solve the one-dimensional optimization problem,

$$\eta_{opt} = \arg\min_{\eta} g(\eta) := f(x - \eta \Delta f(x)) \tag{5.3}$$

5.2 Gradient Descent using Backtracking Line Search

Backtracking line search is another way to compute the step size to be taken in each iteration of the *gradient descent* method. The best step size to choose at each iteration can be obtained through *exact line search* which involves solving a one dimensional optimization problem. However, it may be computationally inefficient to solve an optimization problem at each iteration. Thus, a natural approximation is to choose a step size which is not necessarily the best possible, but one that is reasonably good, i.e., produce 'enough' decrease in the objective f at each iteration. One such way of ensuring that the decrease in the function value is 'sufficient' in successive iterations without having to solve an optimization problem is by the method of *backtracking line search*.

Backtracking Line Search (BTLS) is defined by two parameters α, β with $0 < \alpha < 0.5$ and $0 < \beta < 1$. A step size η is defined to be '*good*' at a point x if

$$f(x - \eta \nabla f(x)) \leq f(x) - \alpha \eta ||\nabla f(x)||^2 \tag{5.4}$$

The way to find such a good step size η in any iteration is by first starting at large initial test step size η_0 (usually taken 1) and check if this step size satisfies (5.4). If it is satisfied, then one proceeds with this step size, else reduces the step size by a factor of β and tests it again. The algorithm is formally described below.

Algorithm 1 Gradient Descent with Backtracking Line Search

1: $x \leftarrow \mathbf{0}$
2: $\eta_0 \leftarrow 1$
3: iterations $\leftarrow 1$
4: $N \leftarrow$ *number of iterations*
5: **for** iterations $\leq N$ **do**
6: $\quad \eta \leftarrow \eta_0$
7: $\quad$ **while** $f(x - \eta \nabla f(x)) > f(x) - \alpha\eta ||\nabla f(x)||^2$ **do**
8: $\quad\quad \eta \leftarrow \beta\eta$
9: $\quad$ **end while**
10: $\quad x \leftarrow x - \eta \nabla f(x)$
11: $\quad$ iterations $\leftarrow$ iterations $+1$
12: **end for**
13: **return** x and $f(x)$

For BTLS to be a viable option, we need to ensure that

1. The while loop in Algorithm 1 terminates, i.e., there always exists a good $\eta > 0$ for every x.
2. The final step size η at each iteration (at line 9) should not be too small to have a significant descent.

It is shown that BTLS ensures the above conditions in the following section.

5.2.1 Analysis of Gradient Descent using BTLS

For the analysis, we assume that the function f is strictly convex, i.e., (5.2) is satisfied and f is differentiable at every point.

Claim: For any strictly convex function f, $\eta \leq \frac{1}{M}$ satisfies (5.4) for all x.

Proof: Since f is strictly convex, for all x, y

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{M}{2} ||y - x||^2 \tag{5.5}$$

Substituting $y = x - \eta \nabla f(x)$ in (5.5), we get

$$f(x - \eta \nabla f(x)) \leq f(x) - \langle \nabla f(x), -\eta \nabla f(x) \rangle + \frac{M}{2} ||\eta \nabla f(x)||^2 \tag{5.6}$$

$$\leq f(x) - \eta \left(1 - \frac{\eta M}{2}\right) ||\nabla f(x)||^2 \tag{5.7}$$

Thus, one can see that, for any $\eta \leq \frac{1}{M}$,

$$f(x - \eta \nabla f(x)) \leq f(x) - \frac{\eta}{2} ||\nabla f(x)||^2 \tag{5.8}$$

$$\leq f(x) - \eta \alpha ||\nabla f(x)||^2 \tag{5.9}$$

$$\tag{5.10}$$

for any $\alpha < 0.5$

□

This result answers both points we needed to ensure the viability of running BTLS. All values of $\eta \in \left(0, \frac{1}{M}\right]$ necessarily satisfy (5.4), which addresses the first point. Furthermore, it is easy to see that the step size in line 9 of Algorithm 1 is lower bounded by $\frac{\beta}{M}$. Hence, the step size used in any iteration can be bounded away from zero as $\eta \geq \min(1, \frac{\beta}{M})$.

Theorem 5.1. *If f is a strictly convex function, then successive iterations of gradient descent using BTLS satisfy*

$$f(x^+) - f^* \leq c(f(x) - f^*)$$

with

$$c = \left(1 - 2m\alpha \min\left(1, \frac{\beta}{M}\right)\right) \in (0, 1),$$

where x^+ denotes the next iterate and x the current iterate.

Proof: From the condition on the while loop in Algorithm 1,

$$f(x^+) \leq f(x) - \alpha\eta ||\nabla f(x)||^2 \tag{5.11}$$

Since we know η is lower bounded by $\min(1, \frac{\beta}{M})$,

$$f(x^+) \leq f(x) - \alpha \min(1, \frac{\beta}{M}) ||\nabla f(x)||^2 \tag{5.12}$$

$$f(x^+) - f^* \leq f(x) - f^* - \alpha \min(1, \frac{\beta}{M}) ||\nabla f(x)||^2 \tag{5.13}$$

Recall that from strong convexity

$$||\nabla f(x)||^2 \geq 2m(f(x) - f^*) \tag{5.14}$$

Substituting (5.14) in (5.13), we get the theorem □

The above theorem guarantees a linear convergence (log - log linear) with rate c.

5.2.2 Comparison of BTLS and Exact Line Search

We numerically compare the performance of BTLS and Exact Line search. The objective function to minimize is chosen to be

$$f(x) = e^{(a^T x - 0.5)} + e^{(b^T x - 0.1)} + e^{(c^T x - 0.1)} \tag{5.15}$$

where $a^T = [1\ 2], b^T = [1\ -3]$ and $c^T = [-1\ 0]$. The optimal f^* (see Fig. 5.1) for this function and the optimal point x^* are computed separately using CVX.

The initial starting point for both the algorithms was set to $x_0^T = [2\ 1]$.

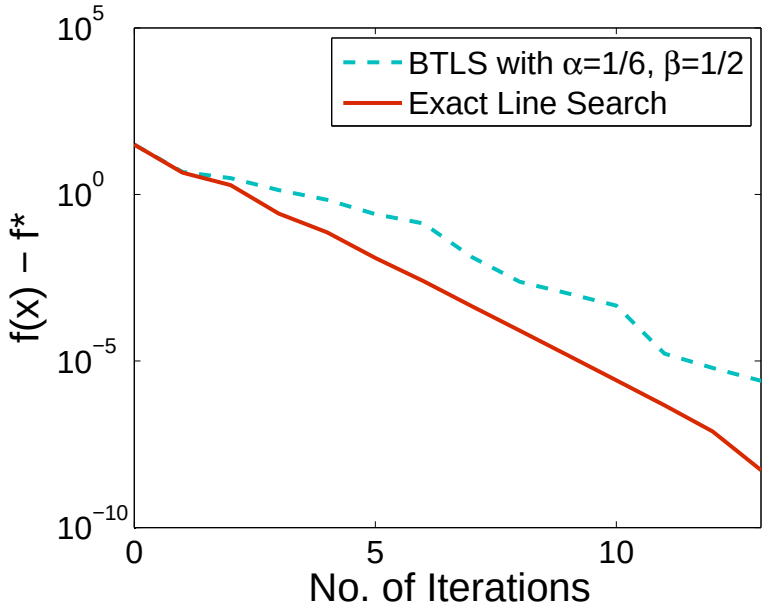

Figure 5.1. The plot comparing the error with iteration for BTLS and Exact Line Search

The performance of BTLS is almost as good as the Exact Line Search. Moreover, BTLS has a much better runtime than the exact line search. BTLS for the above example takes 0.017 seconds to terminate while the Exact Line search takes 28.95 seconds.

5.3 Introduction to Coordinate Descent

In the previous sections we discussed the convergence of various methods of gradient descent. There are two main practical concerns regarding the gradient descent:

(1) It may be cumbersome to calculate the gradient $\nabla f(x)$ at every iterate.

(2) The gradient descent methods do not take advantage of any additional information we may have regarding the function, for instance the Hessian $\nabla^2 f(x)$.

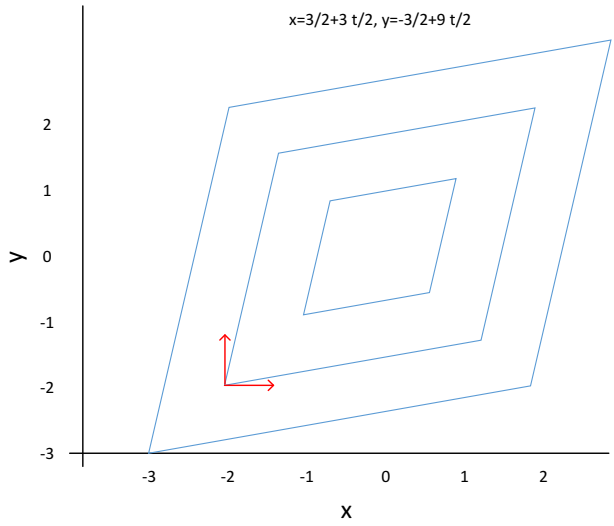

Figure 5.2. The illustration shows that coordinate descent may converge to a non-stationary point.

The latter point will be discussed in detail in the upcoming lectures. The former can be circumvented by using a method known as *coordinate descent*. This is a iterative descent algorithm where the descent vector at every stage is chosen to be one of the canonical basis vectors in $\mathbb{R}^n$. More precisely, at each iteration, we fix a coordinate i, and the descent is performed as,

$$x_j^+ = \begin{cases} x_j & \forall j \neq i \\ x_i - \eta \nabla_i f(x) & \text{if } j = i. \end{cases} \tag{5.16}$$

There are various ways in which the coordinate i can be chosen. It seems reasonable to choose the coordinate with the best descent direct, that is the one which maximizes $\nabla_i f(x)$. But this method forces us to calculate the whole gradient at the iterate, which we wanted to avoid in the first place. There are two intuitive ways of choosing the coordinate at each step,

(1) *Cyclic*: We choose the next coordinate at each iteration in a cyclic manner.

(2) *Random*: We sample the coordinate uniformly at random.

5.3.1 When does it converge ?

The coordinate descent method may not converge for every convex function that is differentiable. Recall that in order to prove the convergence of gradient descent we had to impose a *Lipschitz condition* on the function. Therefore, intuitively we should need a continuity constraint on $\nabla f(x)$ for the coordinate descent algorithm to converge.

We can see that for the function whose level sets are given in Fig. 5.2 the coordinate descent will not converge. This is because the gradient of this function is not continuous. With this motivation in mind, we can state the following about the convergence of coordinate descent.

"If f is a convex function and ∇f is continuous, then coordinate descent converges to a stationary point x^* such that $\nabla f(x^*) = 0$, provided that the level set of the initial point was compact."