10-725/36-725: Convex Optimization **Fall 2016**

Lecture 1: September 14

Lecturer: Ryan Tibshirani *Scribes: Brendan McVeigh, Cristobal De La Maza*

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

1.1 Gradient descent algorithm

The simplest algorithm to solve unconstraint smooth optimization problems is gradient descent. Lets consider function $f(x)$ is convex and differentiable with $dom(f) = R^n$. Denote the optimal criterion value by $f^* = \min xf(x)$, and a solution by x^*. The optimization program will take then the form:

$$\underset{X}{\text{minimize}} \quad \text{f(x)}$$

The gradient descent algorithm will solve this problem as follows:

1. choose initial point $x^{(0)} \in R^n$
2. $x^{(k)} = x^{(k-1)} - t^k \cdot \nabla f(x^{(k-1)})$
3. Repeat for $k = 1, 2, 3, ...$
4. Stop at some point

Think about this method as repeatedly going downhill. The negative gradient is going in the direction that decreases the optimization criterion. Thus, we will stop as some point close to the minimum solution independent on the starting point. The later is valid only for convex functions. In non-convex functions, depending on the starting point different local minima could be achieved.

Further, we can interpret gradient descent via a quadratic approximation. Suppose we are at point x, and we make a second order Taylor expansion of function $f(y)$.

$$f(y) \approx f(x) + \nabla f(x)^T \cdot (y - x) + \frac{1}{2}\nabla^2 f(x) \left\|y - x\right\|_2^2$$

Replacing, $\nabla^2 f(x) = \frac{1}{t} I$ and thus assuming a proximity term to x equal to $\frac{1}{2t} \left\|y - x\right\|_2^2$ with weight $\frac{1}{2t}$, and a linear approximation to f as $f(x) + \nabla f(x)^T \cdot (y - x)$, we have:

$$f(y) \approx f(x) + \nabla f(x)^T \cdot (y - x) + \frac{1}{2t} \left\|y - x\right\|_2^2$$

Gradient descent will choose the next point $y = x^+$ to minimize the quadratic approximation by taking the gradient of $f(y)$ equal to zero:

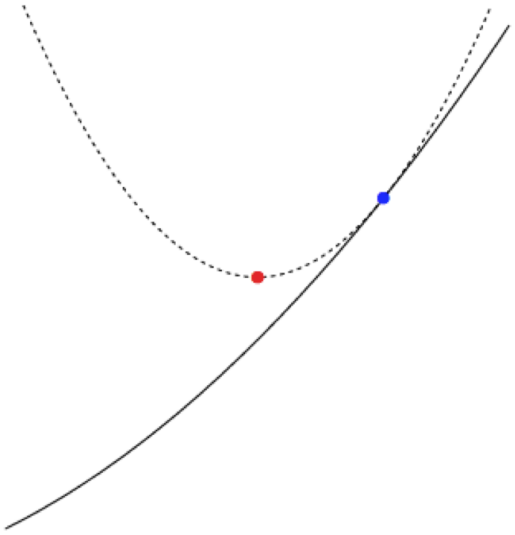


Figure 1.1: Gradient descent algorithm: red dot is x and blue dot x^+

$$x^+ = x - t\nabla f(x)$$

Depending on how close the next step should be to the current state x depends on weight $\frac{1}{2t}$ of the proximity term. If t is small, the weight of the proximity term is large and steps will be small. We can observe this process in the following figure:

$$x^+ = \underset{y}{\text{argmin}} \quad f(x) + \nabla f(x)^T \cdot (y - x) + \frac{1}{2t} \|y - x\|_2^2$$

1.2 Step size selection

1.2.1 Fixed step size

The simplest approach to find the step size, is to fix t in all iterations. If we choose $t_k = t$, $\forall k = 1, 2, 3, ..$ the function can diverge if t is too big. Consider $f(x) = (10x_1^2 + x_2^2)/2$, gradient descent after 8 steps:

As we can observe in the contour plot of Figure 1.2 A. The minimum is never achieved and at some point steps lead to increases in the function. On contrary, if steps size t is too small, convergence will be slow. Figure 1.2 B shows gradient descent for the same example with small step size after 100 steps.

As updates get closer to the minimum, the effective step $t\nabla f(x)$ gets small as the gradient $\nabla f(x)$ approaches zero and thus step direction will shrink by default and slowed down the process. The algorithm converges nicely when t is "just right" (e.g. found by trial and error). Figure 1.2 C shows convergence for the same example, with "just right" step size after 40 steps. Convergence analysis later will give us a precise idea of "just right".

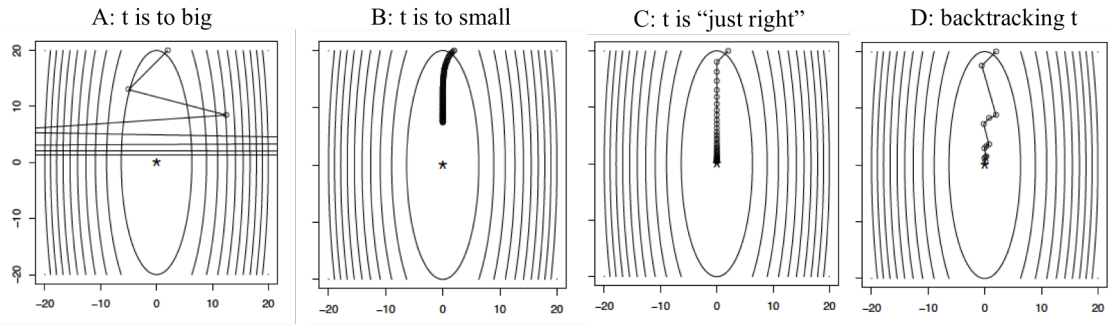


Figure 1.2: Step size different scenarios

1.2.2 Backtracking line search

One way to adaptively choose the step size is to use backtracking line search. The algorithm can be describe as follows:

1. First fix parameters $0 < \beta < 1$ and $0 < \alpha < 1$
2. At each iteration, start with $t = t_{init}$ (i.e. $t_{init} = 1$)
3. While, $f(x = x - t\nabla f(x)) > f(x) - \alpha t \|\nabla f(x)\|_2^2$ shrink $t = \beta \cdot t$
4. Else, perform gradient descent update $x^+ = x - t\nabla f(x)$

The algorithm is simple and tends to work well in practice. Further simplification just take $\alpha = \frac{1}{2}$. Lets assume current state is point x, and a step direction $\Delta x = -\nabla f(x)$. We would like to find x^+ such that $f(x) \geq f(x^+)$.

By convexity, the tangent line $f(x) - t\nabla f(x)^T \Delta x$ is always lower than $f(x)$. Thus, before making a comparison we adjust this value by fraction α, and then compare progress with $f(x) - \alpha t \nabla f(x)^T \Delta x$.

If the value of the function in the proposed step $f(x - t\nabla f(x))$ is to big, we adjust by a factor β and repeat until we find a value of f(x^+) that is lower or equal than our benchmark.

If the criterion is meet, we update our next value to $x^+ = x - t\nabla f(x)$. For the same example shown previously, we can observe in Figure 1.2 D that backtracking picks up roughly the "just right" step size (12 outer steps, 40 steps total). Here $\alpha = \beta = 0.5$. Figure 1.3 describes the algorithm.

1.2.3 Exact line search

Could also choose step to do the best we can along direction of negative gradient, called exact line search:

$$t = \underset{s \geq 0}{\text{argmin}} \quad f(x - s\nabla f(x))$$

Usually is not possible to do this minimization exactly. Approximations to exact line search are often not as efficient as backtracking, and it's usually not worth it. It is much simpler to use backtracking and the gain of exact line search in practice is not large.

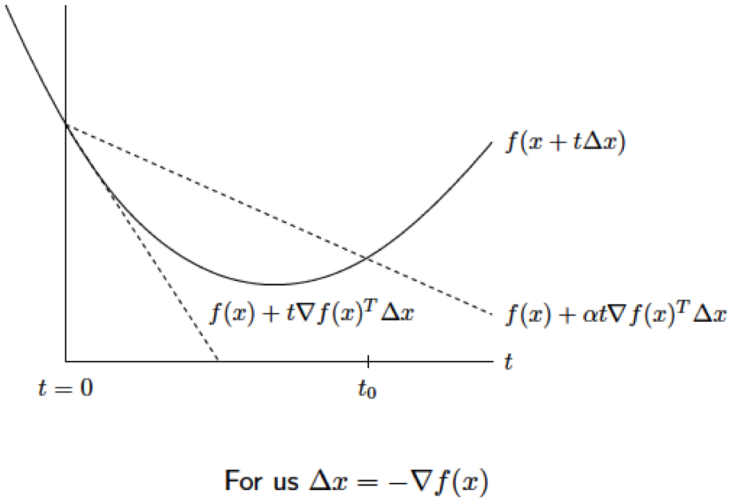


Figure 1.3: Backtracking line search

1.3 Convergence Analysis

1.3.1 Standard Gradient Descent

Theorem 1.1 *Assume that f convex and differentiable, with $dom(f) = \mathbb{R}^n$ and ∇f is Lipschitz continuous with constant $L > 0$. Then gradient descent with fixed step size $t < 1/L$ satisfies*

$$f(x^{(k)}) - f^* \leq \frac{||x^{(0)} - x^*||_2^2}{2tk}$$

From this we can see that

$$\epsilon = \frac{||x^{(0)} - x^*||_2^2}{2tk} \Longrightarrow k = \frac{||x^{(0)} - x^*||_2^2}{2t\epsilon}$$

Hence, $O(1/\epsilon)$ iterations are required for $f(x^{(k)}) - f^* \leq \epsilon$. This is also stated as gradient descent having a convergence rate of $O(1/k)$

Proof: By assumption ∇f is Lipschitz with constant L which implies

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2}||y - x||_2^2 \quad \forall\, x, y \tag{1.1}$$

so we can upper bound the function by a quadratic .

Suppose we are at a x in gradient descent iterations, go to $x^+ = x - t\nabla f(x)$. Evaluating the inequality in

1.1 at $y = x^+$ we find that

$$\begin{aligned}
f(x^+) &\leq f(x) + \nabla f(x)^T(x^+ - x) + \frac{L}{2}||x^+ - x||_2^2 \\
&= f(x) + \nabla f(x)^T(x - t\nabla f(x) - x) + \frac{L}{2}||x - t\nabla f(x) - x||_2^2 \\
&= f(x) - t\nabla f(x)^T(\nabla f(x)) + \frac{L}{2}||t\nabla f(x)||_2^2 \\
&= f(x) - t||\nabla f(x)||_2^2 + \frac{Lt^2}{2}||\nabla f(x)||_2^2 \\
&= f(x) - t(1 - \frac{Lt}{2})||\nabla f(x)||_2^2 \\
&\leq f(x) - \frac{t}{2}||\nabla f(x)||_2^2
\end{aligned}$$

Where the last line follows because $t < 1/L$ and hence, $Lt/2 < 1/2$. Thus, we have shown that

$$f(x^+) \leq f(x) - \frac{t}{2}||\nabla f(x)||_2^2 \tag{1.2}$$

or that $f(x^+) < f(x)$ showing descent.

Since f is convex the first order characterization holds and hence

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \forall\, x, y \in \text{dom}(f)$$

Rearranging and setting $y = x^*$ yields

$$f(x) \leq f(x^*) + \nabla f(x)^T(x - x^*) \tag{1.3}$$

Combining this with 1.2 we have

$$\begin{aligned}
f(x^+) &\leq f(x) - \frac{t}{2}||\nabla f(x)||_2^2 \\
&\leq f(x^*) + \nabla f(x)^T(x - x^*) - \frac{t}{2}||\nabla f(x)||_2^2 \\
&= f(x^*) + \frac{1}{2t}\Big(||x - x^*||_2^2 - ||x - x^*||_2^2 - t^2||\nabla f(x)||_2^2 + 2t\nabla f(x)^T(x - x^*)\Big) \\
&= f(x^*) + \frac{1}{2t}\Big(||x - x^*||_2^2 - (x - x^*)^T(x - x^*) - t^2\nabla f(x)^T\nabla f(x) + 2t\nabla f(x)^T(x - x^*)\Big) \\
&= f(x^*) + \frac{1}{2t}\Big(||x - x^*||_2^2 - [(x - x^*)^T(x - x^*) + t^2\nabla f(x)^T\nabla f(x) - 2t\nabla f(x)^T(x - x^*)]\Big) \\
&= f(x^*) + \frac{1}{2t}\Big(||x - x^*||_2^2 - [(x - t\nabla f(x)^T - x^*)^T(x - t\nabla f(x)^T - x^*)]\Big) \\
&= f(x^*) + \frac{1}{2t}\Big(||x - x^*||_2^2 - ||x - t\nabla f(x)^T - x^*||_2^2\Big) \\
&= f(x^*) + \frac{1}{2t}\Big(||x - x^*||_2^2 - ||x^+ - x^*||_2^2\Big)
\end{aligned}$$

Where the last step follows because $x^+ = x - t\nabla f(x)$. Applying this result to a step i we find that

$$f(x^{(i)}) - f(x^*) \leq \frac{1}{2t}\Big(||x^{(i-1)} - x^*||_2^2 - ||x^{(i)} - x^*||_2^2\Big) \tag{1.4}$$

Thus,

$$\begin{aligned}\sum_{i=1}^{k} f(x^{(i)}) - f(x^*) &\leq \sum_{i=1}^{k} \frac{1}{2t}\Big(||x^{(i-1)} - x^*||_2^2 - ||x^{(i)} - x^*||_2^2\Big) \\ &= \frac{1}{2t}\Big(||x^{(0)} - x^*||_2^2 - ||x^{(k)} - x^*||_2^2\Big) \\ &\leq \frac{1}{2t}\Big(||x^{(0)} - x^*||_2^2\Big)\end{aligned}$$

The last step follows because this is a telescoping sum where the second term for each $i-1$ cancels with the first term for each i. Applying the result from 1.2 we note that

$$\frac{1}{k}\sum_{i=1}^{k} f(x^{(i)}) - f(x^*) \geq \frac{1}{k}\sum_{i=1}^{k} f(x^{(k)}) - f(x^*) = f(x^{(k)}) - f(x^*)$$

Combining these yields our desired result

$$f(x^{(k)}) - f(x^*) \leq \frac{||x^{(0)} - x^*||_2^2}{2tk}$$

■

1.3.2 Extensions and Special Cases of Gradient Descent

Theorem 1.2 *Assume that f convex and differentiable, with $dom(f) = \mathbb{R}^n$ and ∇f is Lipschitz continuous with constant $L > 0$. Then gradient descent with backtracking line search satisfies*

$$f(x^{(k)}) - f^* \leq \frac{||x^{(0)} - x^*||_2^2}{2t_{\min}k}$$

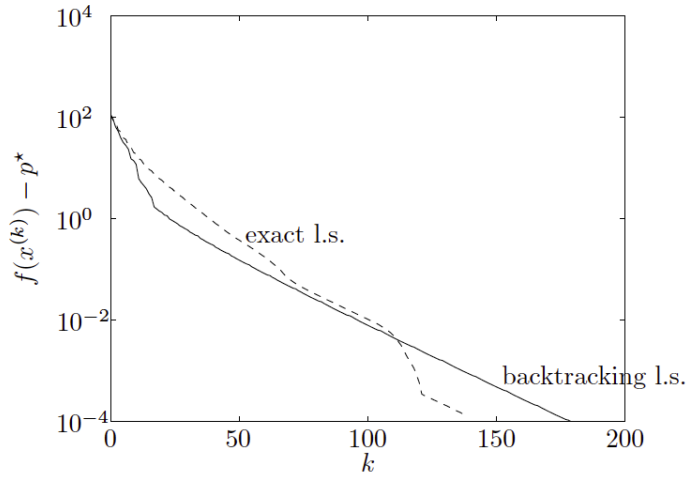
where $t_{\min} = \min\{1, \beta/L\}$

Theorem 1.3 *Assume that f strongly convex with $m > 0$ and differentiable, with $dom(f) = \mathbb{R}^n$ and ∇f is Lipschitz continuous with constant $L > 0$. Then gradient descent with fixed step size $t < 2/(m + L)$ or with backtracking line search satisfies*

$$f(x^{(k)}) - f^* \leq c^k \frac{L}{2}||x^{(0)} - x^*||_2^2$$

where $0 < c < 1$

Adding the strong convexity constraint means that the function is bounded below by a quadratic in addition to being bounded above by a quadratic from the Lipschitz condition. Thus, gradient descent is able to achieve a much faster rate of convergence. Convergence is $O(c^k)$ or exponentially fast. This is often refereed to as linear convergence since it appears linear when plotted on a log scale as shown in Figure 1.4. The constant c depends on the condition number L/m where a larger value for the condition number implies a slower rate of convergence.



(From B & V page 487)

Figure 1.4: Linear Convergence

1.3.3 Practical Concerns

Stopping Rule: $\nabla f(x^*) = 0$ so in practice the algorithm is terminated when $||\nabla f(x)||_2$ is small. If f is strongly convex with parameter m then stop when $||\nabla f(x)||_2 \leq \sqrt{2m\epsilon} \Longrightarrow f(x) - f(x^*) \leq \epsilon$

Pros and Cons of gradient descent

- Pro: concept is simple and each iterations is (relatively) cheap
- Pro: fast for well-conditioned strongly convex f
- Con: Can be slow when f is not strongly convex and well-conditioned
- Con: f must be differentiable, and calculating the derivative must be computationally feasible.

1.3.4 Bound on Convergence Rate

Gradient descent belongs to a class of methods know as first-order methods, iterative methods with updates $x^{(k)}$ in

$$x^{(0)} + \text{span}\{\nabla f(x^{(0)}), \nabla f(x^{(1)}), \ldots, \nabla f(x^{(k-1)})\}$$

Theorem 1.4 (Nesterov) *For any $k \leq (n-1)/2$ and any starting point $x^{(0)}$, there is a function f in the problem class such that any first-order method satisfies*

$$f(x^{(k)} - f^* \geq \frac{3L||x^{(0)} - x^*||_2^2}{32(k+1)^2}$$

Thus convergence rates $O(1/k^2)$, or $O(1/\sqrt{\epsilon})$ may be (and are) possible.

1.4 Extensions of Gradient Descent

1.4.1 Gradient boosting

Gradient boosting is basically a version of gradient descent that is forced to work with trees. Lets assume we have observations $y = (y_1, ..., y_n) \in R^n$ and predictor measurements $x_i \in R^p, i = 1, ..., n$. We want to construct a flexible (nonlinear) model based on our predictors.

Lets define the weighted sum of trees as:

$$u_i = \sum_{j=1}^{m} \beta_j \cdot T_j(x_i), i = 1, ..., n \tag{1.5}$$

Each tree T_j takes as input a predictor measurements x_i and outputs a prediction. To solve this problem lets first pick a loss function L that reflects the setting (e.g. for continuos y we could take $L(y_i, u_i) = (y_i - u_i)^2$). We want to solve

$$\underset{\beta}{\text{minimize}} \quad \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{L}\Big(\mathrm{y_i}, \sum_{\mathrm{j}=1}^{\mathrm{M}} \beta_{\mathrm{j}} \cdot \mathrm{T_j(x_i)}\Big) \tag{1.6}$$

First think about the optimization as $min_u f(u)$, subject to u coming from trees. The algorithm proceeds as follows:

Start with an initial model (e.g. fit a single tree $u^{(0)} = T_0$) and repeat:

1. Compute negative gradient d at latest prediction $u^{(k-1)}$,

$$d_i = -\Big[\frac{\partial L(y_i, u_i)}{\partial u_i}\Big]\Big|_{u_i = u_i^{(k-1)}}, i = 1, ..., n \tag{1.7}$$

2. Find a tree T_k that is close to a,i.e. according to:

$$\underset{trees T}{\text{minimize}} \quad \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{d_i} - \mathrm{T(x_i)})^2$$

   This is not hard to (approximately) solve for a single tree.

3. Compute step size α_k, and update our prediction:

$$u^{(k)} = u^{(k-1)} + \alpha_k \cdot T_k \tag{1.8}$$

   Note that predictions are weighted sum of tress as desired.

1.4.2 Stochastic gradient descent

Suppose we want to minimize the sum of functions

$$\min_x \sum_{i=1}^m f_i(x)$$

Then since $\nabla \sum_{i=1}^m f_i(x) = \sum_{i=1}^m \nabla f_i(x)$ gradient descent would sum the gradients of all of the functions when

$$x^{(k)} = x^{(k-1)} - t_k \cdot \sum_{i=1}^m \nabla f_i(x^{(k-1)}), \qquad k = 1, 2, \ldots$$

Stochastic gradient descent instead looks at each component individually

$$x^{(k)} = x^{(k-1)} - t_k \nabla f_{i_k}(x^{(k-1)}), \qquad k = 1, 2, \ldots$$

Where $i_k \in \{1, \ldots, m\}$. The i_k can be chosen in two different ways

- *Cyclic rule* choose $i_k = 1, 2, \ldots, m, 1, 2, \ldots$
- *Randomized rule* choose $i_k \in \{1, \ldots, m\}$ uniformly at random

In practice the randomize rule is used more frequently in practice. What is the difference between stochastic and the usual (batch) gradient descent? Computationally m steps in stochastic gradient descent is approximately equivalent to one batch step. After m steps using the cyclic rule the function has moved to

$$x^{(k+m)} = x^{(k)} - t \sum_{i=1}^m \nabla f_i(x^{(k+i-1)})$$

Whereas for batch gradient descent, as we have seen previously it is

$$x^{(k+m)} = x^{(k)} - t \sum_{i=1}^m f_i(x^{(k)})$$

Difference between these terms is

$$\sum_{i=1}^m \nabla f_i(x^{(k+i-1)}) - f_i(x^{(k)})$$

These should be close as long as the f_is don't move around too much. In practice stochastic gradient seems to do well when it is far away from the optimum.