Class #7

Tuesday, February 11, 2025 10:19 AM

$f \in C^1_L(\mathbb{R}^n) \Rightarrow$ Continuously differentiable functions with Lipschitz gradients

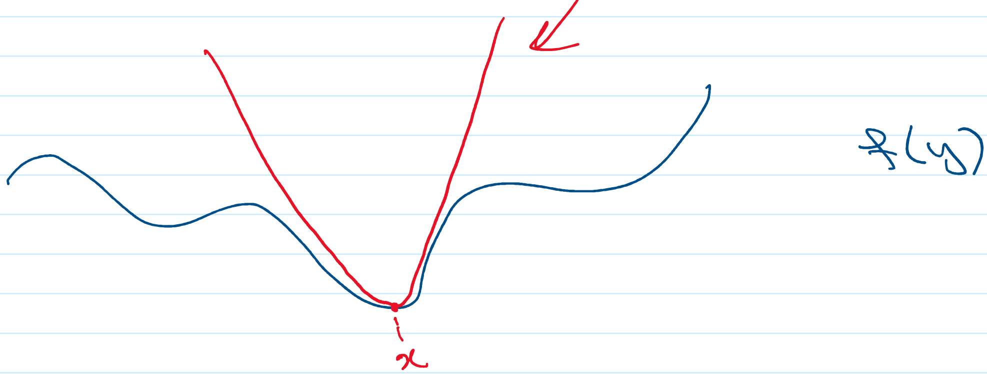
Lemma: Quadratic upper bound on $f \in C^1_L(\mathbb{R}^n)$

Let $f \in C^1_L(\mathbb{R}^n)$ and dom f is a convex set.

Then; $\forall\, x, y \in \text{dom } f$

quadratic function in y

$$f(y) \le f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|_2^2$$



Proof:

Define $g(t) = f(x + t(y-x))$ for $t \in [0,1]$

Since dom f is convex $(x + t(y-x))$ is in the dom f.

$\Rightarrow x + ty - tx = (1-t)x + ty$

Note: $g(1) = f(y)$

$g(0) = f(x)$

$$g'(t) = \nabla f(x + t(y-x))^T (y-x)$$

$$\boxed{g'(0) = \nabla f(x)^T (y-x)}$$

$$\boxed{g'(0) = \nabla f(x)^T (y-x)}$$

$$g'(t) - g'(0) = \nabla f(x + t(y-x))^T (y-x) - \nabla f(x)^T (y-x)$$

$$= \left[\nabla f(x+t(y-x)) - \nabla f(x)\right]^T (y-x)$$

Use Cauchy-Schwarz Inequality ($a^T b \le \|a\|_2 \|b\|_2$)

$$\le \underbrace{\|\nabla f(x+t(y-x)) - \nabla f(x)\|_2}_{\text{Lipschitz continuous gradients}} \|y-x\|_2$$

$$\le L \|x+t(y-x) - x\|_2 \|y-x\|_2$$

$$g'(t) - g'(0) \le tL \|y-x\|_2^2$$

$$\int_0^1 g'(t)\,dt = g(1) - g(0)$$

$$\Rightarrow g(1) = g(0) + \int_0^1 g'(t)\,dt$$

$$\le g(0) + \int_0^1 \left(tL\|y-x\|_2^2 + g'(0)\right)dt$$

$$f(y) \le f(x) + L\|y-x\|_2^2 \left.\frac{t^2}{2}\right|_0^1 + g'(0)$$

$$\le f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|_2^2$$

Descent Lemma (when $\Delta x = \nabla f(x)$)

Descent Lemma (when $\Delta x = \nabla f(x)$)

Let $f \in C^1_L(\mathbb{R}^n)$ with $\operatorname{dom} f$ being convex. Let us consider the iterative method:

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

and focus on the case $\Delta x^{(k)} = -\nabla f(x^{(k)})$. Then, as long as $0 < t^{(k)} < \frac{2}{L}$, $\Delta x^{(k)}$ is a desent direction, i.e. $f(x^{(k+1)}) < f(x^{(k)})$ as long as $x^{(k)}$ is not x^*.

Remarks: A range of step sizes guarantee descent. The larger L is, the smaller is the range.

Proof: Let $y = x^{(k)} + t^{(k)} \Delta x^{(k)}$

and $x = x^{(k)}$ in the Quadratic upper bound.

$$f(x^{(k)} + t^{(k)} \Delta x^{(k)}) \leq f(x^{(k)}) + \nabla f(x^{(k)})^T (\cancel{x^{(k)}} + t^{(k)} \Delta x^{(k)} - \cancel{x^{(k)}})$$
$$+ \frac{L}{2} \| \cancel{x^{(k)}} + t^{(k)} \Delta x^{(k)} - \cancel{x^{(k)}} \|_2^2$$

$$f(x^{(k+1)}) \leq f(x^{(k)}) + t^{(k)} \nabla f(x^{(k)})^T \Delta x^{(k)} + t^{(k)^2} \frac{L}{2} \| \Delta x^{(k)} \|_2^2$$

Now put $\Delta x^{(k)} = -\nabla f(x^{(k)})$

$$f(x^{(k+1)}) \leq f(x^{(k)}) - t^{(k)} \| \nabla f(x^{(k)}) \|_2^2 + t^{(k)^2} \frac{L}{2} \| \nabla f(x^{(k)}) \|_2^2$$

need < 0

$$\leq f(x^{(k)}) - \left(t^{(k)} - t^{(k)^2}\frac{L}{2}\right)\|\nabla f(x)\|_2^2$$

Clearly, $f(x^{(k+1)}) < f(x^{(k)})$

most reduction when this is the largest.

if and only if

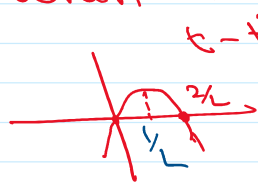
$$t^{(k)} - t^{(k)^2}\frac{L}{2} > 0$$

$$\iff t^{(k)^2}\frac{L}{2} < t^{(k)}$$

$$\iff t^{(k)} < \frac{2}{L}$$

$\blacksquare$

What is the $t^{(k)}$ when 'L' is known?



The best $t^{(k)}$, when it comes to most reduction for gradient descent is $t^{(k)} = \frac{1}{L}$.

Ex: $f(x)$

General Descent Method

__Initialize:__ $x^{(0)} \in \operatorname{dom} f$

$k \leftarrow 0$

__Repeat__

__1.__ Determine a descent direction $\Delta x^{(k)}$
(i.e., $-\nabla f(x^{(k)})^T \Delta x^{(k)} > 0$)

__2.__ __Line Search:__ Choose a step size $t^{(k)} > 0$

__3.__ __Update the iterate:__ $x^{(k+1)} \leftarrow x^{(k)} + t^{(k)} \Delta x^{(k)}$

__Until__ stopping criterion is satisfied.

__Special Cases:__

① Gradient descent: $\Delta x^{(k)} = -\nabla f(x^{(k)})$

② Newton's method: $\Delta x^{(k)} = -\nabla^2 f(x^{(k)}) \nabla f(x^{(k)})$

__Line Search:__ How to pick the step size in a descent method?

① Choose a fixed step size $t^{(k)} = \eta \quad \forall\ k \geq 0$.

__e.g.,__ If $f \in C^1_L(\mathbb{R}^n)$ and L is known or can be computed efficiently then pick

$$\eta = \frac{1}{L} \text{ for gradient descent.}$$

In other cases, trial and error helps pick a step size.

Issues with this approach:

- Sometimes it is too costly to compute L.
- Sometimes functions are not C^1_L.
- Even when one has descent in some iterations, does not mean the step size would work in all points in domf.

Unless L is known, the longer the step size, the better 'perhaps'.

(2) Variable step size $t^{(k)}$

↳ The most common approach in the literature.

(a) Decaying step size policy

$$t^{(k)} \to 0 \quad \text{as} \quad k \to \infty$$

Typical policy

(i) $t^{(k)} \to 0$ as $k \to \infty$

(ii) $\sum_{k=0}^{\infty} t^{(k)} = \infty$

(iii) $\sum_{k=0}^{\infty} \left[t^{(k)}\right]^2 < \infty$

e.g.; $t^{(k)} = \dfrac{\text{const}}{k}$

↳ Often used in machine learning / stochastic

↳ often used in machine learning | Stochastic optimization.

(b) Search for a nice step size that reduces the objective function using a subroutine in each iteration k.

↳ often used in practical deterministic optimization problems.

↳ we will study this in detail under 'Exact line search' and 'Inexact line search'.

<u>Exact line Search</u>

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

When we have fixed $\Delta x^{(k)}$ and are looking for $t^{(k)}$, we are effectively looking at a one-dimensional function:

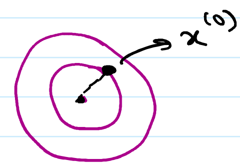
$$\tilde{f}(t) = f(x^{(k)} + t^{(k)} \Delta x^{(k)})$$

Pick $t^{(k)} = \underset{t \geq 0}{\arg\min}\ \tilde{f}(t)$

<u>Example:</u> Say $f(x) = x_1^2 + x_2^2$

$$\nabla f(x) = 2x$$

Let $x^{(0)} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \mathbb{1}$



$$\Rightarrow x^{(1)} = x^{(0)} - 2t^{(1)} x^{(0)}$$

$$= (1-2t^{(1)})\,\mathbb{1} = (1-\tfrac{1}{2}\times 2)\,\mathbb{1} = 0\cdot\mathbb{1} = \begin{bmatrix}0\\0\end{bmatrix}$$

$$\tilde{f}(t) = f(x^{(1)}) = (1-2t)^2\cdot 1 + (1-2t)^2\cdot 1$$

$$= 2(1-2t)^2$$

$$\boxed{t^{(1)}} = \underset{t\geq 0}{\arg\min}\ \tilde{f}(t) = \underset{t\geq 0}{\arg\min}\ 2(1-2t)^2 = \tfrac{1}{2}$$

$$\tilde{f}'(t) = 4(1-2t)\cdot -2 = -8(1-2t)$$

$$\tilde{f}'(t) = 0 \Rightarrow -8(1-2t) = 0$$

$$t = \tfrac{1}{2}$$

Exact line search, in which one solves a one-dim optimization problem in each iteration, works in cases where:

1. The solution has an analytical form.
2. It might be computationally feasible to numerically solve the 1-D problem.

But if the cost of exact line search is too much, we resort to 'inexact line search'.

↳ Backtracking (Armijo-Goldstein line search)