# Optimization for Data Science
# Lecture 04: Gradient Descent

Kimon Fountoulakis

School of Computer Science
University of Waterloo

19/09/2019

# Outline

- Gradient descent
- Line-search techniques for gradient descent

# Some comments

- For the sake of generalization let's consider the following problem

$$\text{minimize } f(x)$$

- Our analysis will be done for this general problem, and it will hold for our denoising problem by simply setting

$$f(x) = \frac{\lambda}{2}\|Dx\|_2^2 + \frac{1}{2}\|x - z_{noisy}\|_2^2$$

# Lipschitz continuous gradient

- We will say that the gradient of $f$ is **Lipschitz continuous** with positive **Lipschitz constant** $L$ if

$$\|\nabla f(z) - \nabla f(s)\|_2 \leq L\|z - s\|_2 \ \forall z, s$$

- The distance of any two gradients cannot be much larger than the distance of the points.

- Fairly common assumption that is true for the data science problems that we will study. It is used in analyzing optimization algorithms.

# Lipschitz continuous gradient: example

$$f(x) = x^T A x - b^T x$$

$$\nabla f(x) = Ax - b$$

**Assumption: matrix A is symmetric**

$$\begin{aligned} \|\nabla f(z) - \nabla f(s)\|_2 &= \|A(z - s)\|_2 \\ &\leq \|A\|_2 \|z - s\|_2 \end{aligned}$$

- Thus gradient of $f$ is Lipschitz continuous with Lipschitz constant

$$L = \|A\|_2$$

# Fundamental theorem of calculus for multivariate functions

- For any “x” and “y” we have the following equality

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \int_0^1 \left(\nabla f(x + t(y - x)) - \nabla f(x)\right)^T (y - x) dt$$

# A corollary of the fundamental theorem of calculus

- From the fundamental theorem of calculus, the Cauchy-Schwartz and Lipschitz continuity of the gradient we get that

$$f(y) \leq f(x) + \nabla f(x)^T (y-x) + \frac{L}{2}\|y-x\|_2^2 \quad \forall x, y \in \mathbb{R}^n$$

**Proof**

$$f(y) = f(x) + \nabla f(x)^T (y-x) + \int_0^1 \left(\nabla f(x + t(y-x)) - \nabla f(x)\right)^T (y-x) dt \longleftarrow$$ From the Fundamental Theorem of Calculus

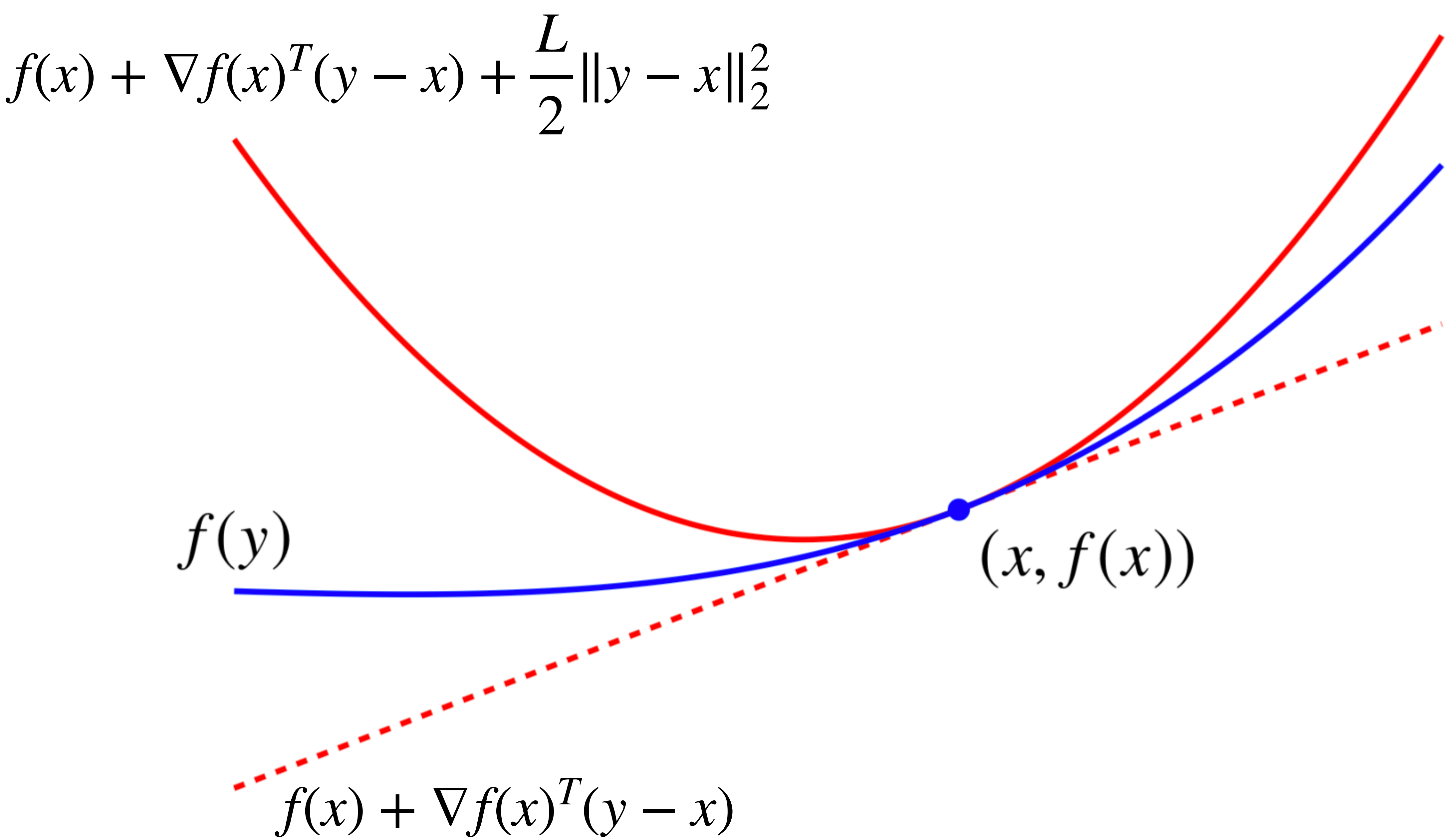
$$\leq f(x) + \nabla f(x)^T (y-x) + \int_0^1 \|\nabla f(x + t(y-x)) - \nabla f(x)\|_2 \|y-x\|_2 dt \longleftarrow$$ We used Cauchy-Schwartz

$$\leq f(x) + \nabla f(x)^T (y-x) + \int_0^1 Lt\|y-x\|_2^2 dt \longleftarrow$$ We used L-smoothness

$$\leq f(x) + \nabla f(x)^T (y-x) + L\|y-x\|_2^2 \int_0^1 t dt$$

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

# A corollary of the fundamental theorem of calculus



# Assumptions

- We will assume that function $f$ is differentiable.

- We will also assume that the gradient of $f$ is Lipschitz continuous

$$\|\nabla f(z) - \nabla f(s)\|_2 \leq L\|z - s\|_2 \ \forall z, s$$

# Making gradient descent

- We start with an initial guess $x_0$
- This guess can simply be an all-zeros vector or a random vector.
- We consider the following update

$$x_1 = x_0 + h$$

- How do we pick $h$?
- One trivial idea is to $\quad \text{minimize } f(x_0 + h)$
- But this is not any different than the original problem.

# Making gradient descent

- A better idea is to approximate the given objective function $f$ with a simpler function. The new function is simpler in the sense that it is easier to minimize.

- For example, let's use the corollary of the Fundamental Theorem Calculus of $f$ at $x_0$. We have that

$$f(x_0 + h) \leq f(x_0) + \nabla f(x_0)^T h + \frac{L}{2} \|h\|_2^2$$

- The RHS is a new function that upper bounds the original. We could maybe use this.

# Making gradient descent

- Let's re-arrange a few terms in the Corollary of the Fundamental Theorem of Calculus.

$$f(x_0 + h) - f(x_0) \leq \nabla f(x_0)^T h + \frac{L}{2} \|h\|_2^2$$

- Assuming that $x_0$ is not a local minimizer, then, ideally, we want to pick $h$ such that

$$f(x_0 + h) - f(x_0) < 0$$

- That's because such an $h$ guarantees that the objective function at the new point is less than the objective function at the previous point.

# Making gradient descent

- One way to guarantee that

$$f(x_0 + h) - f(x_0) < 0$$

- is by picking an $h$ such that

$$\nabla f(x_0)^T h + \frac{L}{2}\|h\|_2^2 < 0$$

- That is because we have the following from the Corollary of the Fundamental Theorem of Calculus

$$f(x_0 + h) - f(x_0) \leq \nabla f(x_0)^T h + \frac{L}{2}\|h\|_2^2$$

# Making gradient descent

- Let's pick an $h$ that minimizes

$$\text{minimize}_h \ \nabla f(x_0)^T h + \frac{L}{2} \|h\|_2^2$$

- Function $\nabla f(x_0)^T h + \frac{L}{2} \|h\|_2^2$ is convex w.r.t $h$. Therefore, the minimizer is given by setting its gradient to zero, which gives

$$h = -\frac{1}{L} \nabla f(x_0)$$

# Making gradient descent

- Using

$$h = -\frac{1}{L}\nabla f(x_0)$$

- And the Corollary of the Fundamental Theorem of Calculus we get

$$f(x_0 - \frac{1}{L}\nabla f(x_0)) - f(x_0) \leq -\frac{1}{2L}\|\nabla f(x_0)\|_2^2 < 0$$

- If $x_0$ is not a stationary point ($\nabla f(x_0) \neq 0$) then we get

$$f(x_0 - \frac{1}{L}\nabla f(x_0)) < f(x_0)$$

# Making gradient descent

- The update formula for the first step is

$$x_1 = x_0 - \frac{1}{L}\nabla f(x_0)$$

- More generally, for iteration $k$ we get

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$$

# How do we terminate?

- We terminate the algorithm when

$$\|\nabla f(x_k)\|_2 \leq \epsilon$$

- For some user defined parameter $\epsilon > 0$.

# Gradient descent: pseudo-code

- While $\|\nabla f(x_k)\|_2 > \epsilon$

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$$

# How to choose L?

- We assumed that the gradient of $f$ is Lipschitz continuous

$$\|\nabla f(z) - \nabla f(s)\|_2 \leq L\|z - s\|_2 \; \forall z, s$$

- But the constant $L$ is used in the gradient descent algorithm. This means that up to this point, we have to know $L$ to use gradient descent.

- However, in practice knowing $L$ is not necessary.

# How to choose L?

- When we constructed gradient descent, we made the decision to pick an $h$ such that it minimizes

$$\text{minimize}_h \ \nabla f(x_0)^T h + \frac{L}{2}\|h\|_2^2$$

- to guarantee that

$$f(x_0 + h) - f(x_0) < 0$$

- which gave us the following step

$$h = -\frac{1}{L}\nabla f(x_0)$$

# How to choose L?

- But maybe solving the following problem is not necessary

$$\text{minimize}_h \; \nabla f(x_0)^T h + \frac{L}{2}\|h\|_2^2$$

- to guarantee that

$$f(x_0 + h) - f(x_0) < 0$$

- We simply need to pick an $h$ such that

$$\nabla f(x_0)^T h + \frac{L}{2}\|h\|_2^2 < 0$$

# How to choose L?

- We can choose $h$ as a multiple of the gradient.

$$h = -\alpha \nabla f(x_0)$$

- Replacing this into the inequality $\nabla f(x_0)^T h + \frac{L}{2}\|h\|_2^2 < 0$ we get

$$-\alpha \|\nabla f(x_0)\|_2^2 + \alpha^2 \frac{L}{2} \|\nabla f(x_0)\|_2^2 < 0$$

- Solving w.r.t to $\alpha$ we get

$$\alpha < \frac{2}{L}$$

# How to choose L?

- This means that for any $\alpha$ that satisfies

$$\alpha < \frac{2}{L}$$

- and $h$ defined as

$$h = -\alpha \nabla f(x_0)$$

- We guarantee that the objective function is decreased

$$f(x_0 + h) - f(x_0) < 0$$

# How to choose $\alpha$?: line-search

- Remember that our goal is to guarantee that the objective function is decreased

$$f(x_{k+1}) < f(x_k)$$

- and we decided to set

$$h = -\alpha \nabla f(x_0)$$

- But we do not know $\alpha$ yet.

- We can find an $\alpha$ that guarantees that the objective function is decreased by using a technique called **line-search.**

# Line-search

- Start with a guess $\alpha = 1$
- Check if the progress inequality is satisfied

$$f(x_k - \alpha \nabla f(x_k)) < f(x_k)$$

- If not, then half $\alpha$ i.e., $\alpha \leftarrow \alpha/2$
- Worst case, if $\alpha < 1/L$ then the above condition will be satisfied and the line-search procedure will terminate.

# Line-search: how much does it cost?

- In worst-case the condition will be satisfied after

$$\frac{\log L}{\log 2}$$

- Each iteration requires recomputing the objective function. Say that computing the objective function costs $T_f$ time. Thus, the total cost in worst-case is

$$T_f \frac{\log L}{\log 2}$$

# Is it better than simply computing the Lipschitz-constant?

- For denoising we have to compute the Lipschitz constant once, before the beginning of the algorithm. But computing the Lipschitz constant can be expensive.

- On the other hand, for line-search we have to perform it at each iteration of the algorithm, but each call to a line-search procedure is inexpensive.

# Is it better than simply computing the Lipschitz-constant?

**Example**

- Time for Gradient Descent using the Lipschitz constant for our denoising problem is **152 seconds**.

    - Out of the **152 seconds, 147 seconds** were spent on computing the Lipschitz constant.

- Time for Gradient Descent using line-search was **3 seconds!!**

# A better way to choose L: Armijo line-search

- Start with a guess $\alpha = 1$
- Check if the objective is **sufficiently** decreased

$$f(x_k - \alpha \nabla f(x_k)) \leq f(x_k) - \alpha\gamma \|\nabla f(x_k)\|_2^2 \quad \text{for } \gamma \in \left(0, \frac{1}{2}\right]$$

- If not, then half $\alpha$ i.e., $\alpha \leftarrow \alpha/2$

# Implementation details for denoising

- For denoising the dominant costs for computing the gradient are

**Bad implementation**

$$\nabla f(x) = \overbrace{\left(\lambda \mathrm{real}(\underbrace{D^* D}_{\textbf{One matrix-matrix product}}) + I\right) x}^{\textbf{One matrix-vector product}} - z_{noisy}$$

# Implementation details for denoising

- We have to perform the matrix-vector products first!!

$$\nabla f(x) = \lambda \text{real}(D^*(Dx)) + x - z_{noisy}$$

**First matrix-vector product**

**Second matrix-vector product**