ECE 595: Machine Learning I

Lecture 5.1: Gradient Descent

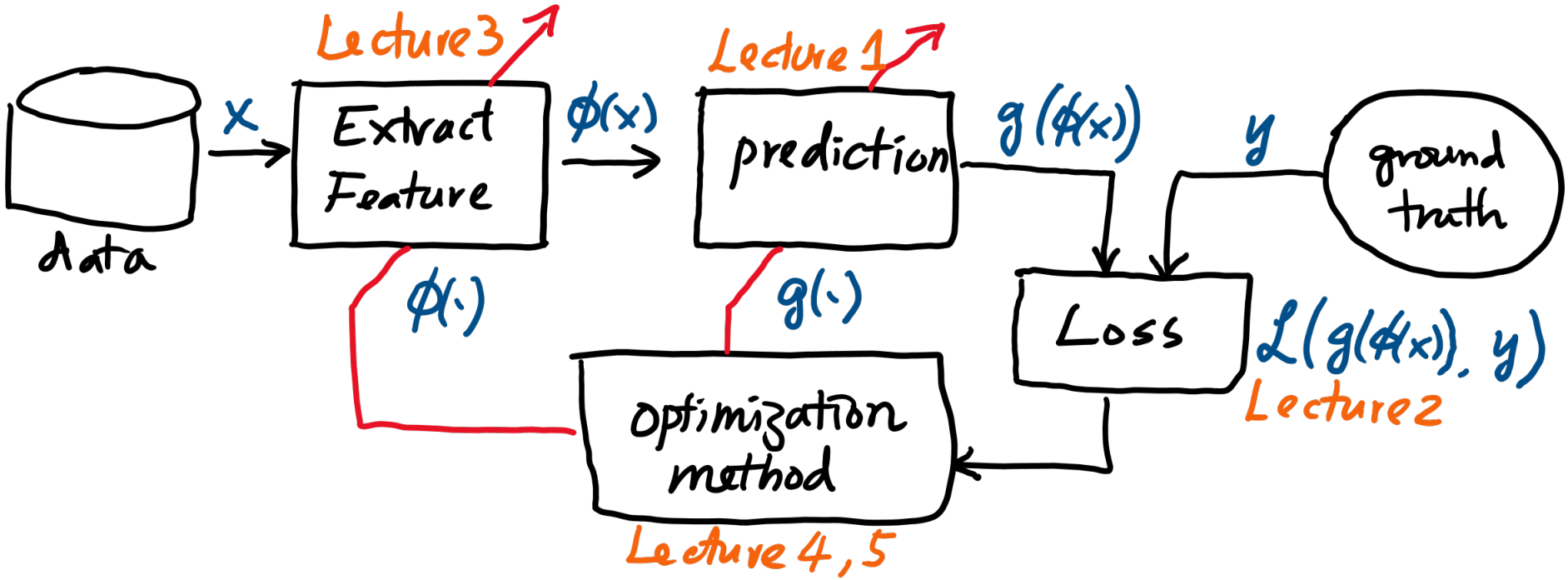
Spring 2020

Stanley Chan

School of Electrical and Computer Engineering
Purdue University

PURDUE
UNIVERSITY

Outline



Outline

Mathematical Background

- Lecture 4: Intro to Optimization
- Lecture 5: Gradient Descent

Lecture 5: Gradient Descent

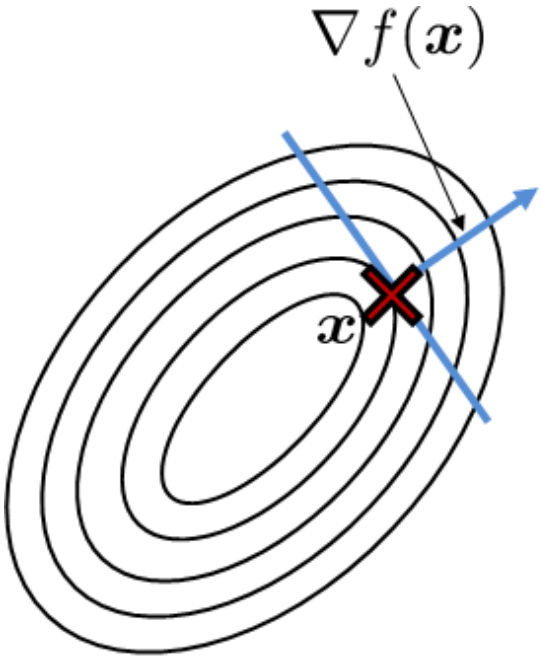
- Gradient Descent
  - Descent Direction
  - Step Size
  - Convergence
- Stochastic Gradient Descent
  - Difference between GD and SGD
  - Why does SGD work?

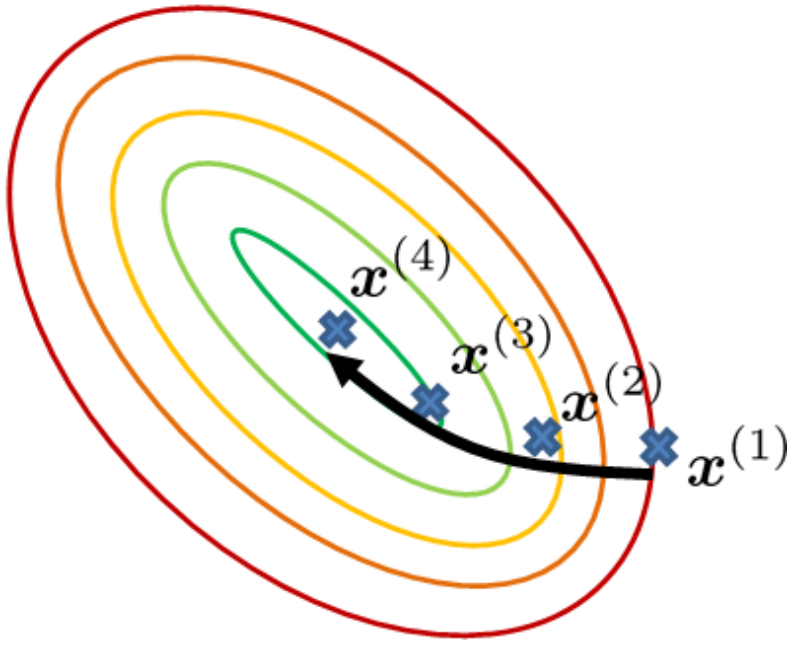
Gradient Descent

The algorithm:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha^{(t)} \nabla f(\boldsymbol{x}^{(t)}), \qquad t = 0, 1, 2, \ldots,$$

where $\alpha^{(t)}$ is called the **step size**.





Why is the direction $-\nabla f(\boldsymbol{x})$?

- Recall (Lecture 4): If $\boldsymbol{x}^*$ is optimal, then

$$\underbrace{\lim_{\epsilon \to 0} \frac{1}{\epsilon} [f(\boldsymbol{x}^* + \epsilon \boldsymbol{d}) - f(\boldsymbol{x}^*)]}_{\geq 0, \ \forall \boldsymbol{d}} = \nabla f(\boldsymbol{x}^*)^T \boldsymbol{d}$$

$$\Longrightarrow \qquad \nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} \geq 0, \quad \forall \boldsymbol{d}$$

- But if $\boldsymbol{x}^{(t)}$ is not optimal, then we want

$$f(\boldsymbol{x}^{(t)} + \epsilon \boldsymbol{d}) \leq f(\boldsymbol{x}^{(t)})$$

- So,

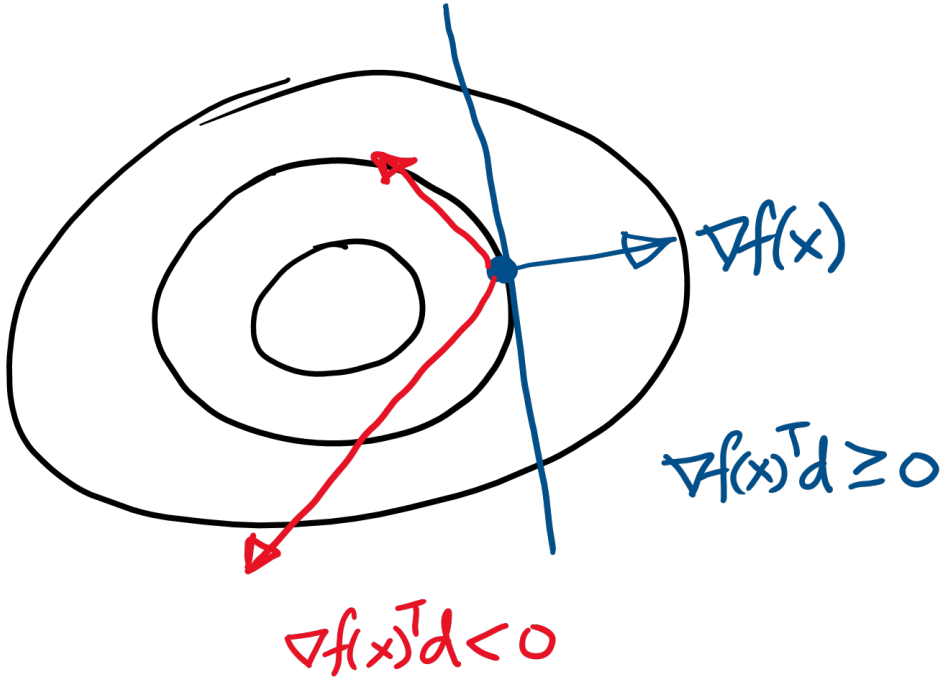
$$\underbrace{\lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big[f(\boldsymbol{x}^{(t)} + \epsilon \boldsymbol{d}) - f(\boldsymbol{x}^{(t)}) \Big]}_{\leq 0, \ \text{for some } \boldsymbol{d}} = \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}$$

$$\Longrightarrow \qquad \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d} \leq 0$$

Descent Direction

Pictorial illustration:

- $\nabla f(\boldsymbol{x})$ is **perpendicular** to the contour.
- A search direction $\boldsymbol{d}$ can either be on the positive side $\nabla f(\boldsymbol{x})^T \boldsymbol{d} \geq 0$ or negative side $\nabla f(\boldsymbol{x})^T \boldsymbol{d} < 0$.
- Only those on the negative side can reduce the cost.
- All such $\boldsymbol{d}$'s are called the **descent directions**.



The Steepest $\boldsymbol{d}$

Previous slide: If $\boldsymbol{x}^{(t)}$ is not optimal yet, then some $\boldsymbol{d}$ will give

$$\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d} \leq 0.$$

- So, let us make $\nabla f(\boldsymbol{x}^{(t)})^T$ as negative as possible.

$$\boldsymbol{d}^{(t)} = \underset{\|\boldsymbol{d}\|_2 = \delta}{\text{argmin}} \ \nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d},$$

- We need δ to control the magnitude; Otherwise $\boldsymbol{d}$ is unbounded.
- The solution is

$$\boldsymbol{d}^{(t)} = -\nabla f(\boldsymbol{x}^{(t)})$$

- Why? By Cauchy Schwarz,

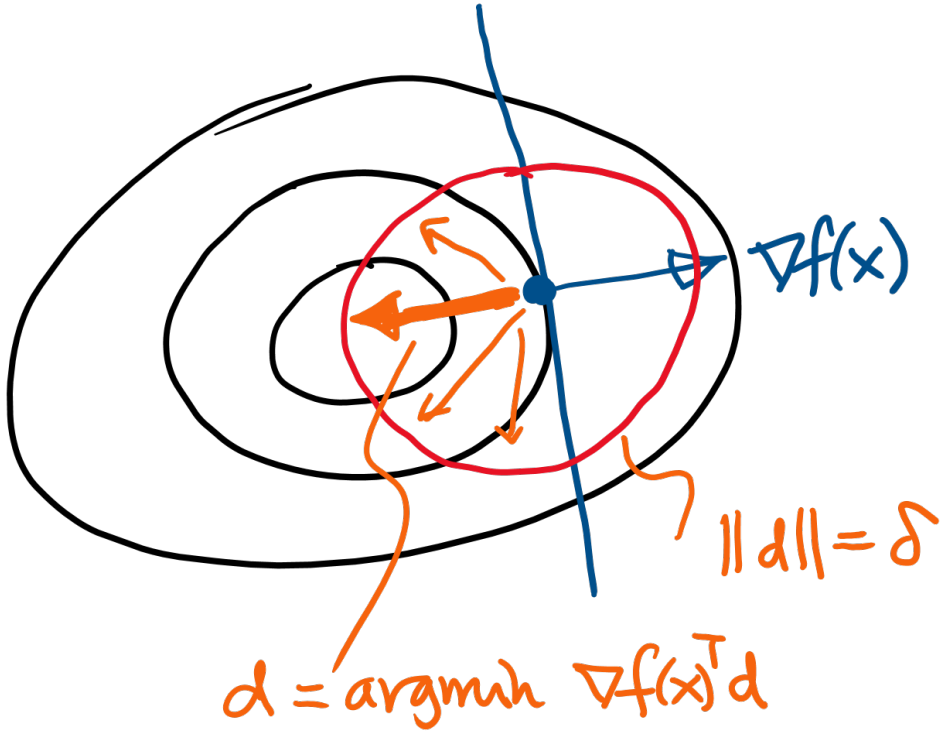
$$\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d} \geq -\|\nabla f(\boldsymbol{x}^{(t)})\|_2 \|\boldsymbol{d}\|_2.$$

- Minimum attained when $\boldsymbol{d} = -\nabla f(\boldsymbol{x}^{(t)})$.
- Set $\delta = \|\nabla f(\boldsymbol{x}^{(t)})\|_2$.

Steepest Descent Direction

Pictorial illustration:

- Put a ball surrounding the current point.
- All $\boldsymbol{d}$'s inside the ball are feasible.
- Pick the one that minimizes $\nabla f(\boldsymbol{x})^T \boldsymbol{d}$.
- This direction must be parallel (but opposite sign) to $\nabla f(\boldsymbol{x})$.



Step Size

The algorithm:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha^{(t)} \nabla f(\boldsymbol{x}^{(t)}), \qquad t = 0, 1, 2, \ldots,$$

where $\alpha^{(t)}$ is called the step size.

- **1. Fixed step size**

$$\alpha^{(t)} = \alpha.$$

- **2. Exact line search**

$$\alpha^{(t)} = \underset{\alpha}{\text{argmin}} \quad f\left(\boldsymbol{x}^{(t)} + \alpha \boldsymbol{d}^{(t)}\right),$$

  - E.g., if $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}$, then

$$\alpha^{(t)} = -\frac{\nabla f(\boldsymbol{x}^{(t)})^T \boldsymbol{d}^{(t)}}{\boldsymbol{d}^{(t)T} \boldsymbol{H} \boldsymbol{d}^{(t)}}.$$

- **3. Inexact line search**:
  Amijo / Wolfe conditions. See Nocedal-Wright Chapter 3.1.

Convergence

Let $\boldsymbol{x}^*$ be the global minimizer. Assume the followings:

- Assume f is twice differentiable so that $\nabla^2 f$ exist.
- Assume $0 \preceq \lambda_{\min} \boldsymbol{I} \preceq \nabla^2 f(\boldsymbol{x}) \preceq \lambda_{\max} \boldsymbol{I}$ for all $\boldsymbol{x} \in \mathbb{R}^n$
- Run gradient descent with exact line search.

Then, (Nocedal-Wright Chapter 3, Theorem 3.3)

$$\begin{aligned} f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^*) &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^2 \left(f(\boldsymbol{x}^{(t)}) - f(\boldsymbol{x}^*)\right) \\ &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^4 \left(f(\boldsymbol{x}^{(t-1)}) - f(\boldsymbol{x}^*)\right) \\ &\leq \quad \vdots \\ &\leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}}\right)^{2t} \left(f(\boldsymbol{x}^{(1)}) - f(\boldsymbol{x}^*)\right). \end{aligned}$$

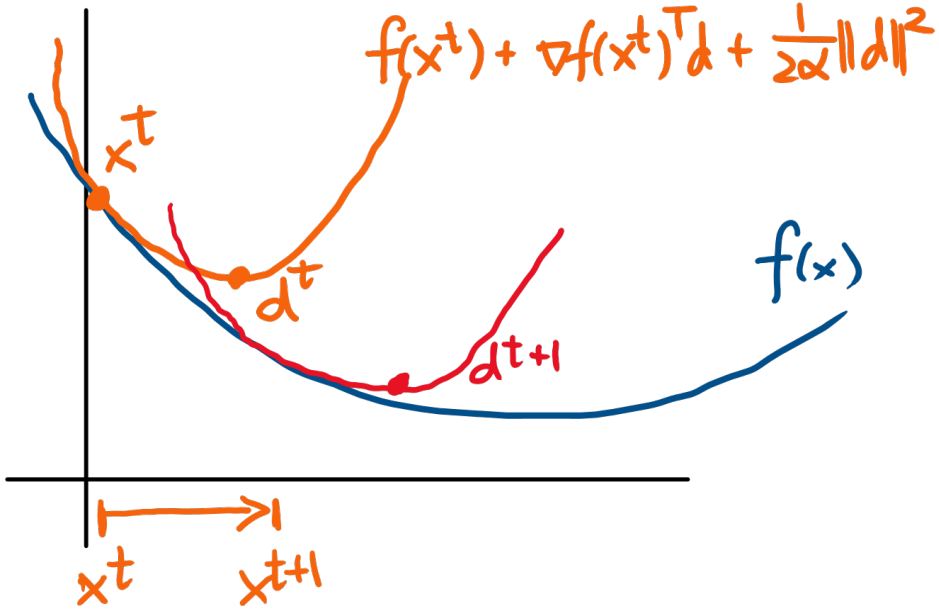
Thus, $f(\boldsymbol{x}^{(t)}) \to f(\boldsymbol{x}^*)$ as $t \to \infty$.

Understanding Convergence

- Gradient descent can be viewed as **successive approximation**.
- Approximate the function as

$$f(\boldsymbol{x}^t + \boldsymbol{d}) \approx f(\boldsymbol{x}^t) + \nabla f(\boldsymbol{x}^t)^T \boldsymbol{d} + \frac{1}{2\alpha}\|\boldsymbol{d}\|^2.$$

- We can show that the $\boldsymbol{d}$ that minimizes $f(\boldsymbol{x}^t + \boldsymbol{d})$ is $\boldsymbol{d} = -\alpha \nabla f(\boldsymbol{x}^t)$.
- This suggests: Use a **quadratic function** to locally approximate f.
- Converge when curvature α of the approximation is not too big.



Advice on Gradient Descent

- Gradient descent is useful because
  - Simple to implement (compared to ADMM, FISTA, etc)
  - Low computational cost per iteration (no matrix inversion)
  - Requires only first order derivative (no Hessian)
  - Gradient is available in deep networks (via back propagation)
- Most machine learning has built-in (stochastic) gradient descents
- Welcome to implement your own, but you need to be careful
  - Convex non-differentiable problems, e.g., ℓ_1-norm
  - Non-convex problem, e.g., ReLU in deep network
  - Trap by local minima
  - Inappropriate step size, a.k.a. learning rate
- Consider more "transparent" algorithms such as `CVX` when
  - Formulating problems. No need to worry about algorithm.
  - Trying to obtain insights.