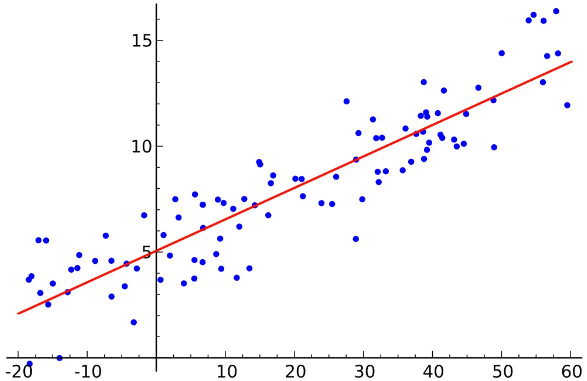
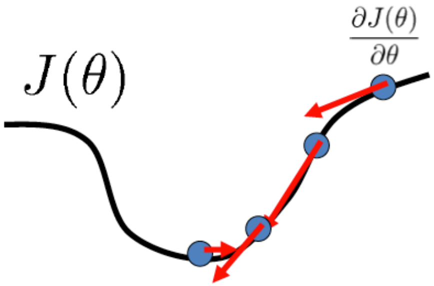
Tufts COMP 135: Introduction to Machine Learning
https://www.cs.tufts.edu/comp/135/2020f/

Gradient Descent





Many slides attributable to:
Erik Sudderth (UCI)
Finale Doshi-Velez (Harvard)
James, Witten, Hastie, Tibshirani (ISL/ESL books)

Prof. Mike Hughes

Today’s Objectives (day 06)

Gradient descent

- Key idea: Find optimal (or close to optimal) parameters for an objective by repeatedly stepping downhill
- Benefits:
  - Widely useful for many ML tasks. Likely useful for any objective that is differentiable
  - In contrast, exact solutions (like our formulas for optimal weights for least squares) are hard to derive for most objectives
- Practical problems:
  - Local vs. global minima
  - How to pick step size
  - How to assess convergence
  - Initialization

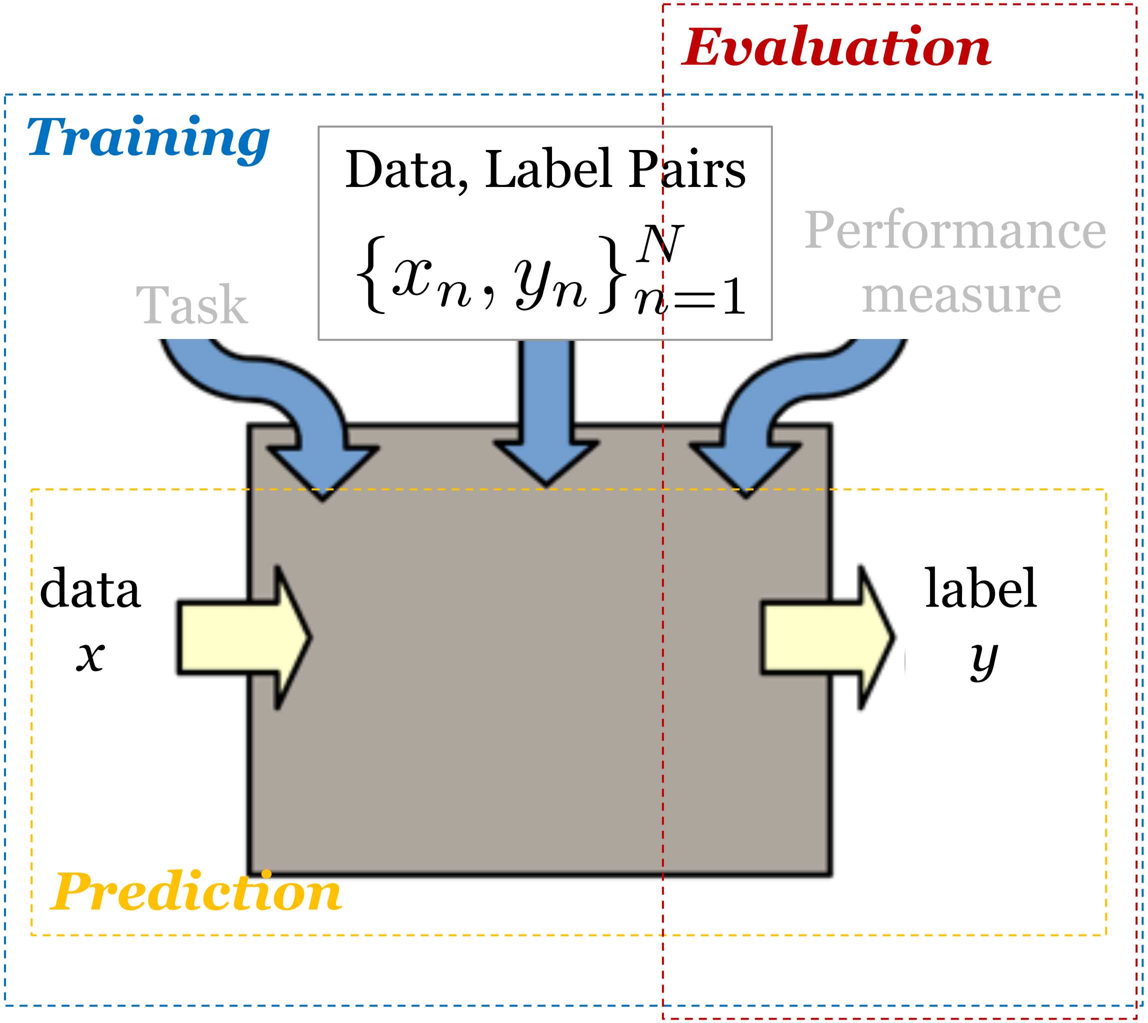
- Lab: Gradient descent for linear regression

What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



Task: Regression

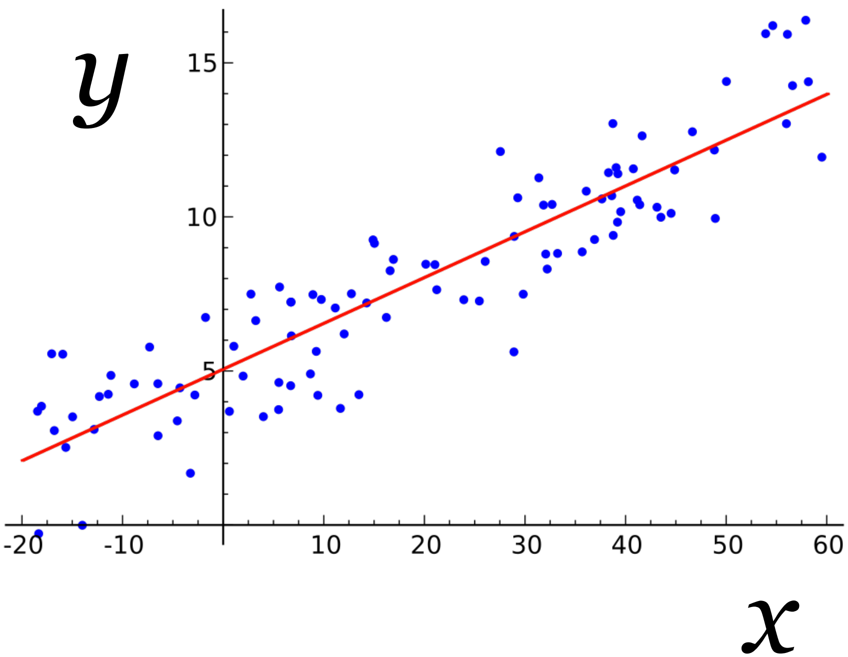
Supervised Learning

regression

Unsupervised Learning

Reinforcement Learning

y is a numeric variable e.g. sales in $$



Linear Regression

Optimization problem: “Least Squares”

$$\min_{\theta \in \mathbb{R}^{F+1}} \quad \sum_{n=1}^{N} (y_n - \hat{y}(x_n, \theta))^2$$

$$\tilde{X} = \begin{bmatrix} x_{11} & \dots & x_{1F} & 1 \\ x_{21} & \dots & x_{2F} & 1 \\ & & \dots & \\ x_{N1} & \dots & x_{NF} & 1 \end{bmatrix}$$

$$\min_{\theta \in \mathbb{R}^{F+1}} \quad \underbrace{(y - \tilde{X}\theta)^T (y - \tilde{X}\theta)}_{J(\theta)}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

We can solve **this particular** optimization problem exactly.

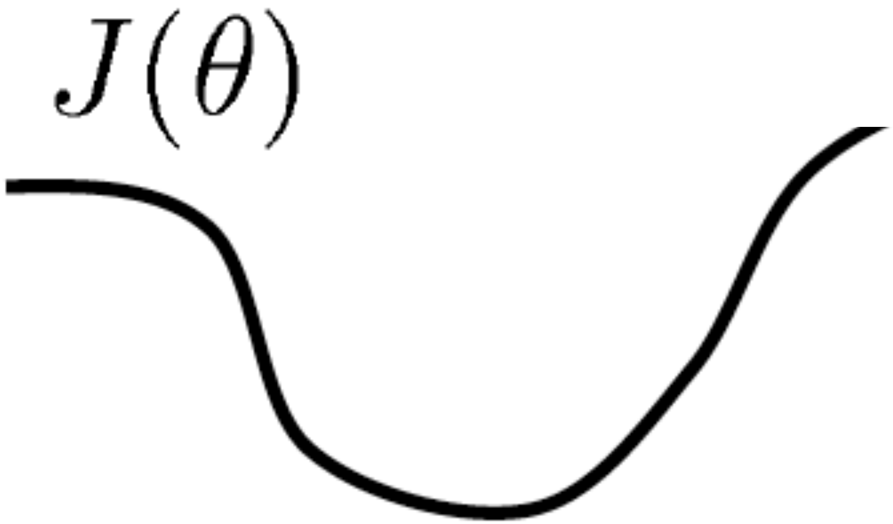
However, what happens when:

- We want to use another error than squared error?
- We want to add an L1 penalty?
- We have so many dimensions (F > 1 million) that solving the linear system exactly is prohibitive
- We want to change the objective in some other way

Is there a general purpose way to solve optimization problems like

$$\min_{\theta} \quad J(\theta)$$

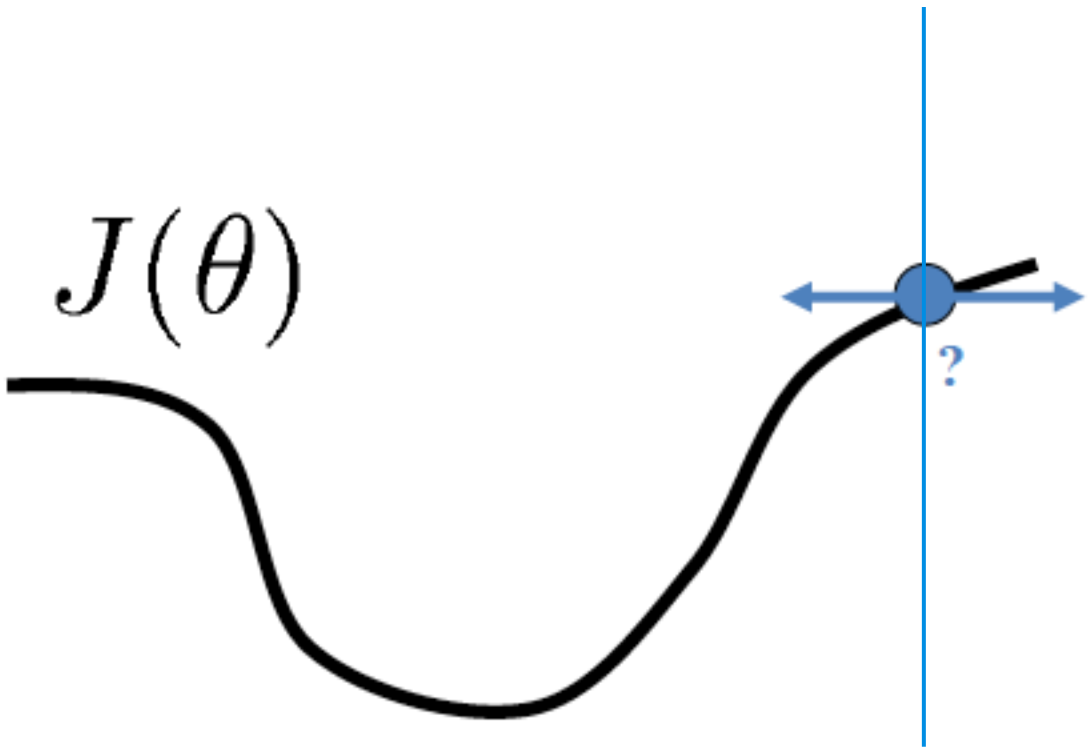
Assumptions



- Loss function J is smooth and deterministic
- We can evaluate our loss function at any value of parameter theta
- We can evaluate the first derivative ("gradient") of our loss at any value of parameter theta

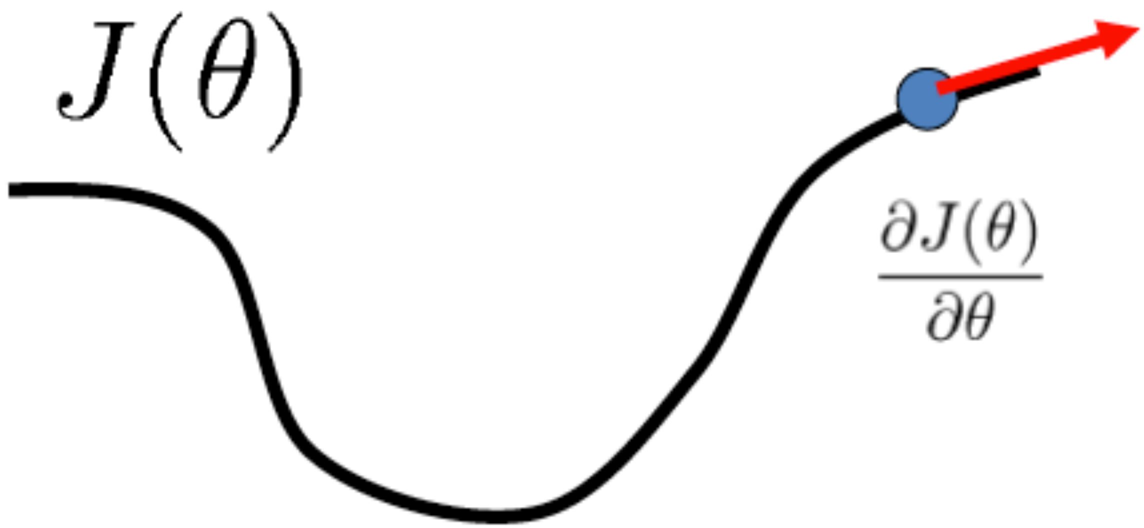
$$\frac{\partial J(\theta)}{\partial \theta}$$

Idea: Optimize via small steps

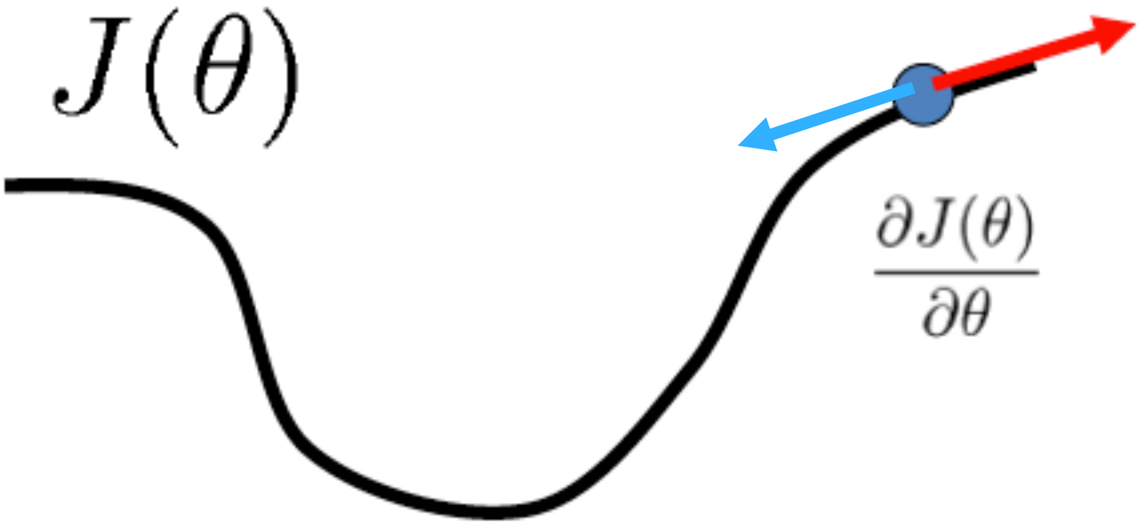


Current "guess" for
parameter theta

Derivatives point **uphill**



To minimize, go **downhill**



Step in the opposite **direction** of the derivative

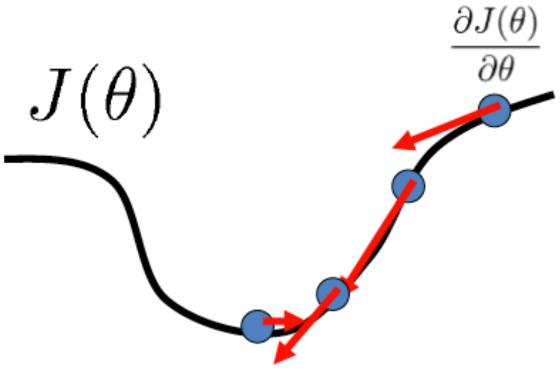
Steepest descent algorithm

input: initial $\theta \in \mathbb{R}$

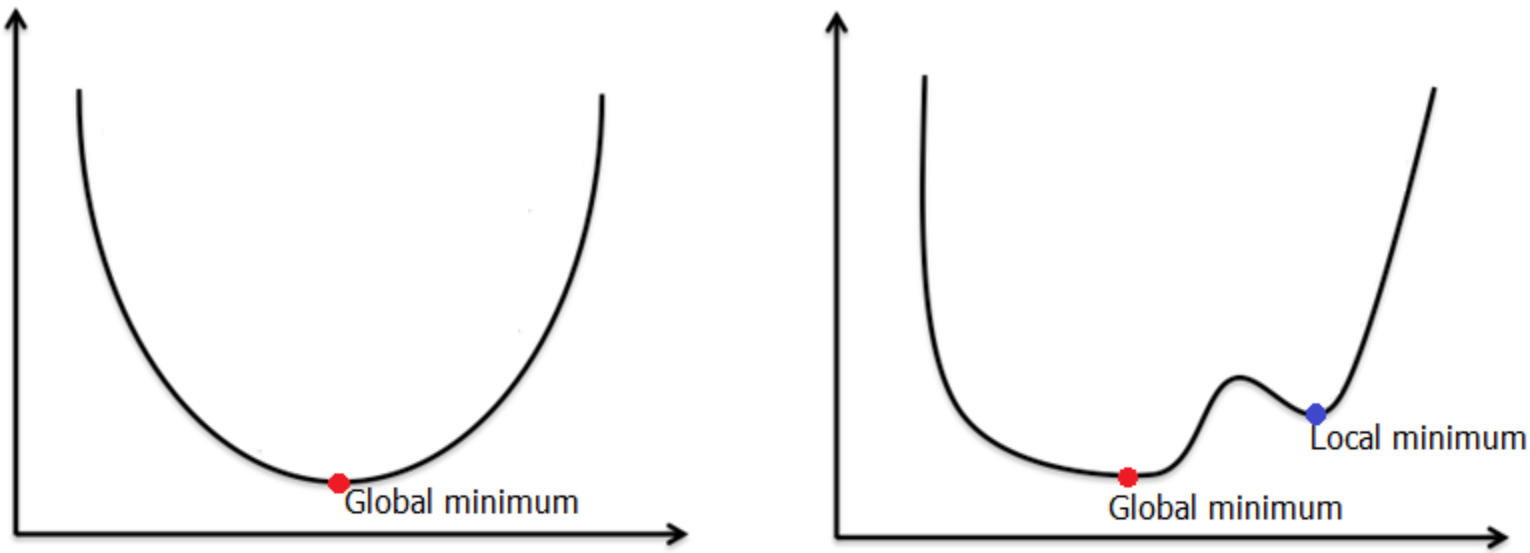
input: step size $\alpha \in \mathbb{R}_+$

while not converged:

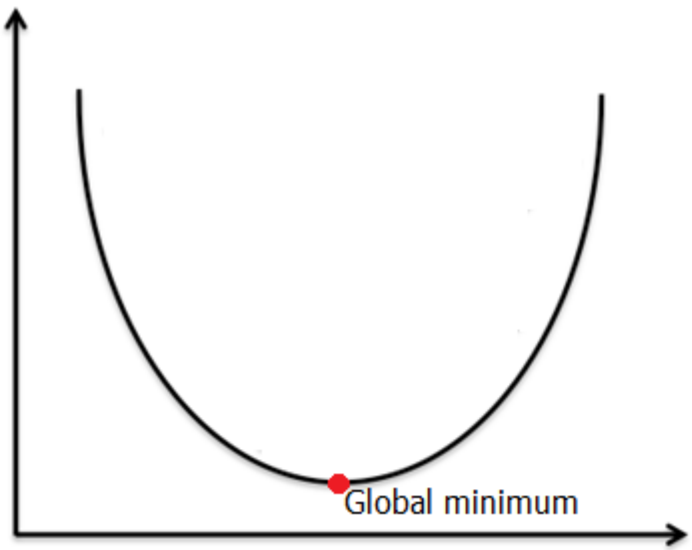
$$\theta \leftarrow \theta - \alpha \frac{d}{d\theta} J(\theta)$$



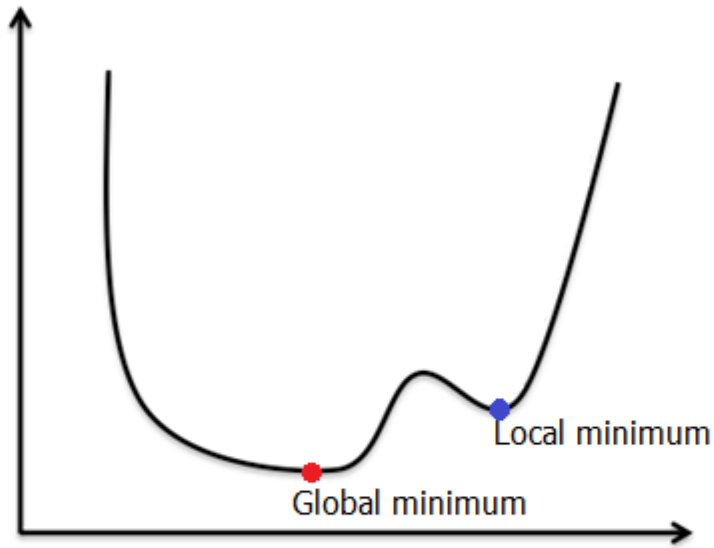
Will gradient descent always find same solution?



Will gradient descent always find same solution?

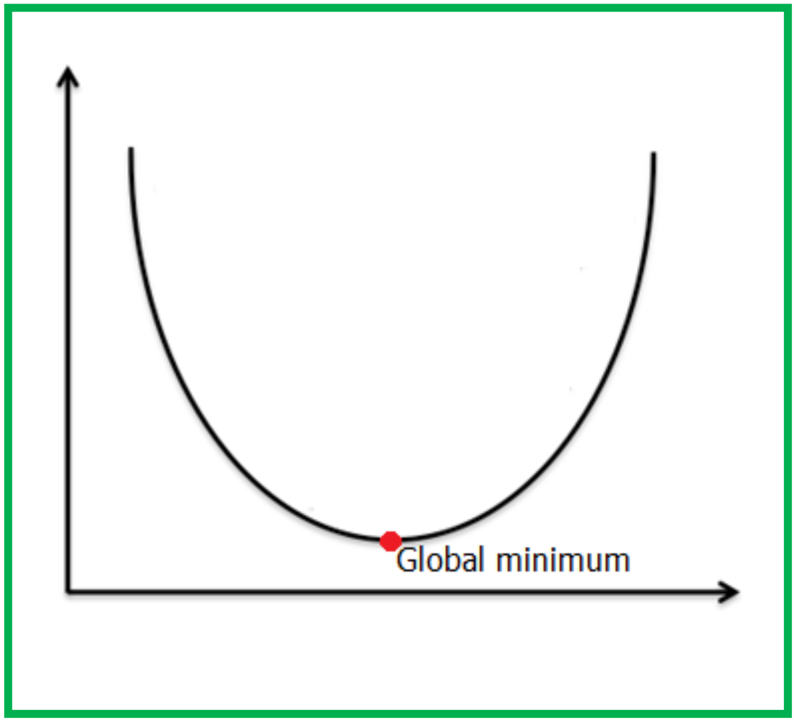


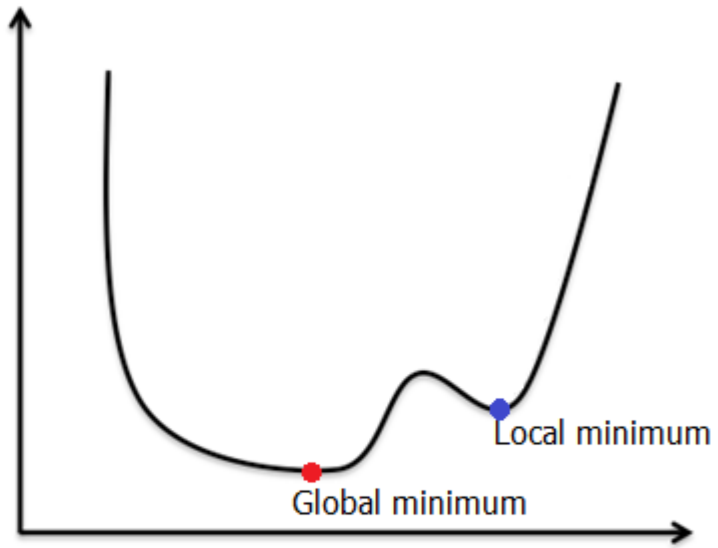
Yes, if loss looks like this



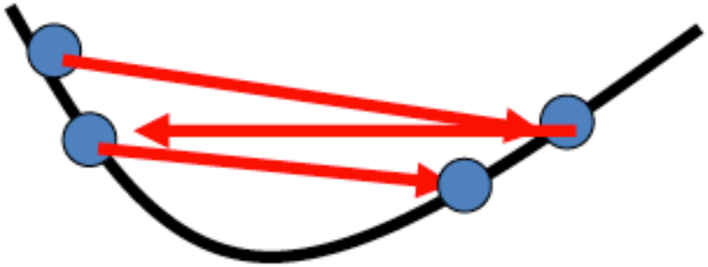
Not if multiple local minima exist

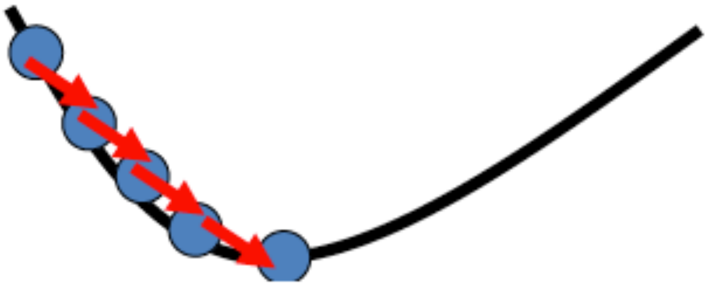
Linear Regression is convex!





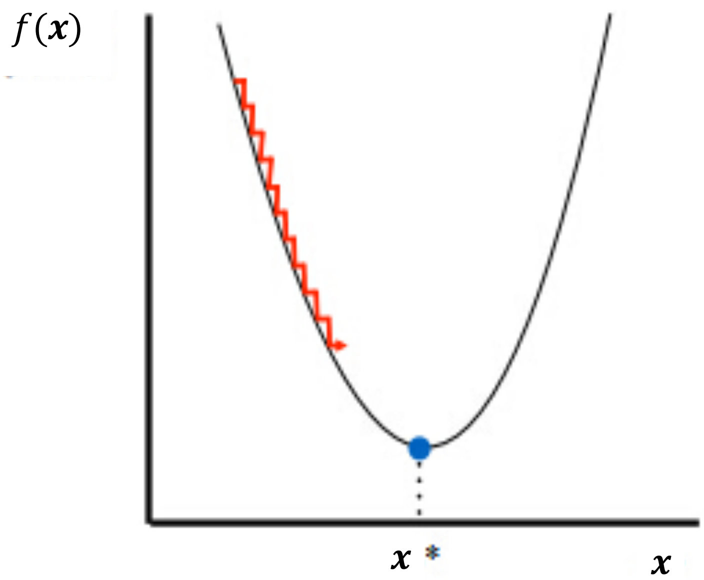
How to set step size?



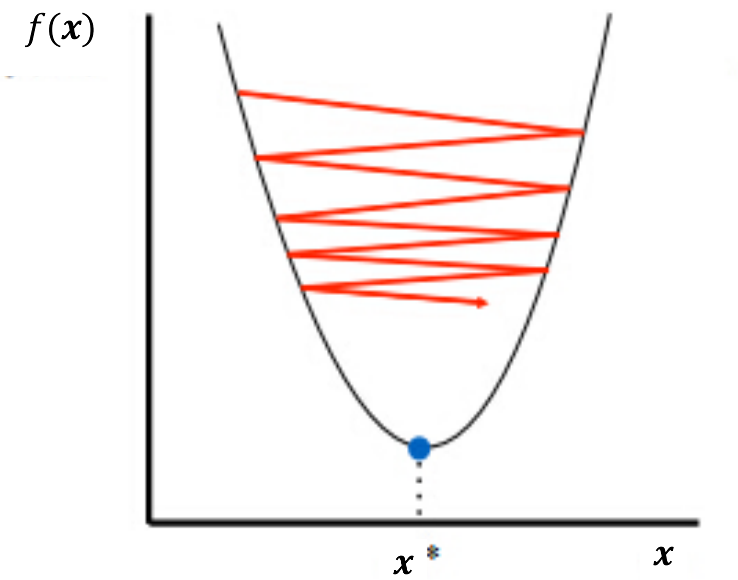


Intuition: 1D gradient descent

Choosing good step size matters!



Too small: converge very slowly



Too big: overshoot and even diverge

Debugging: loss vs iterations

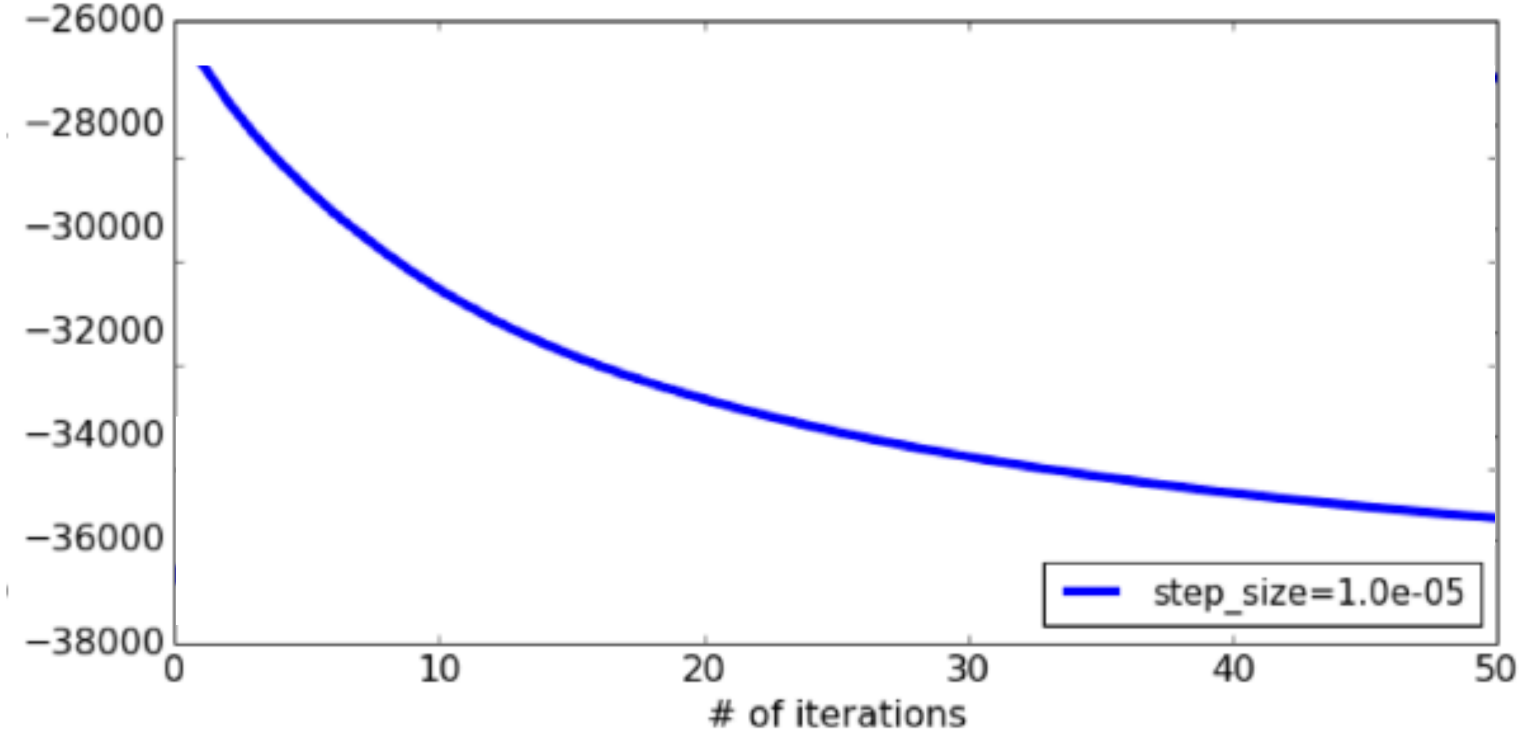


Figure Credit: Emily Fox (UW)

If step size is **too small**

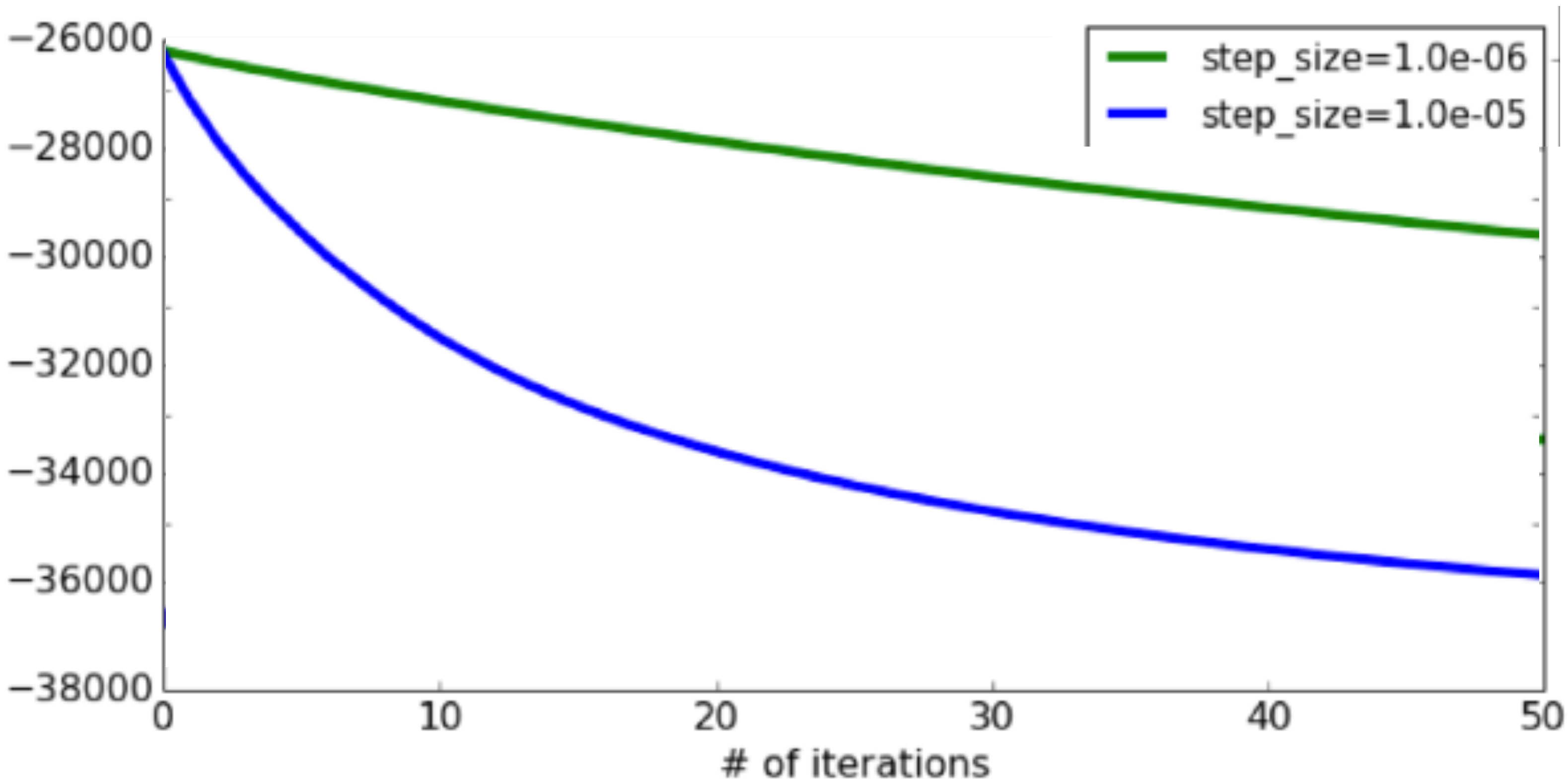


Figure Credit: Emily Fox (UW)

If step size is **large**

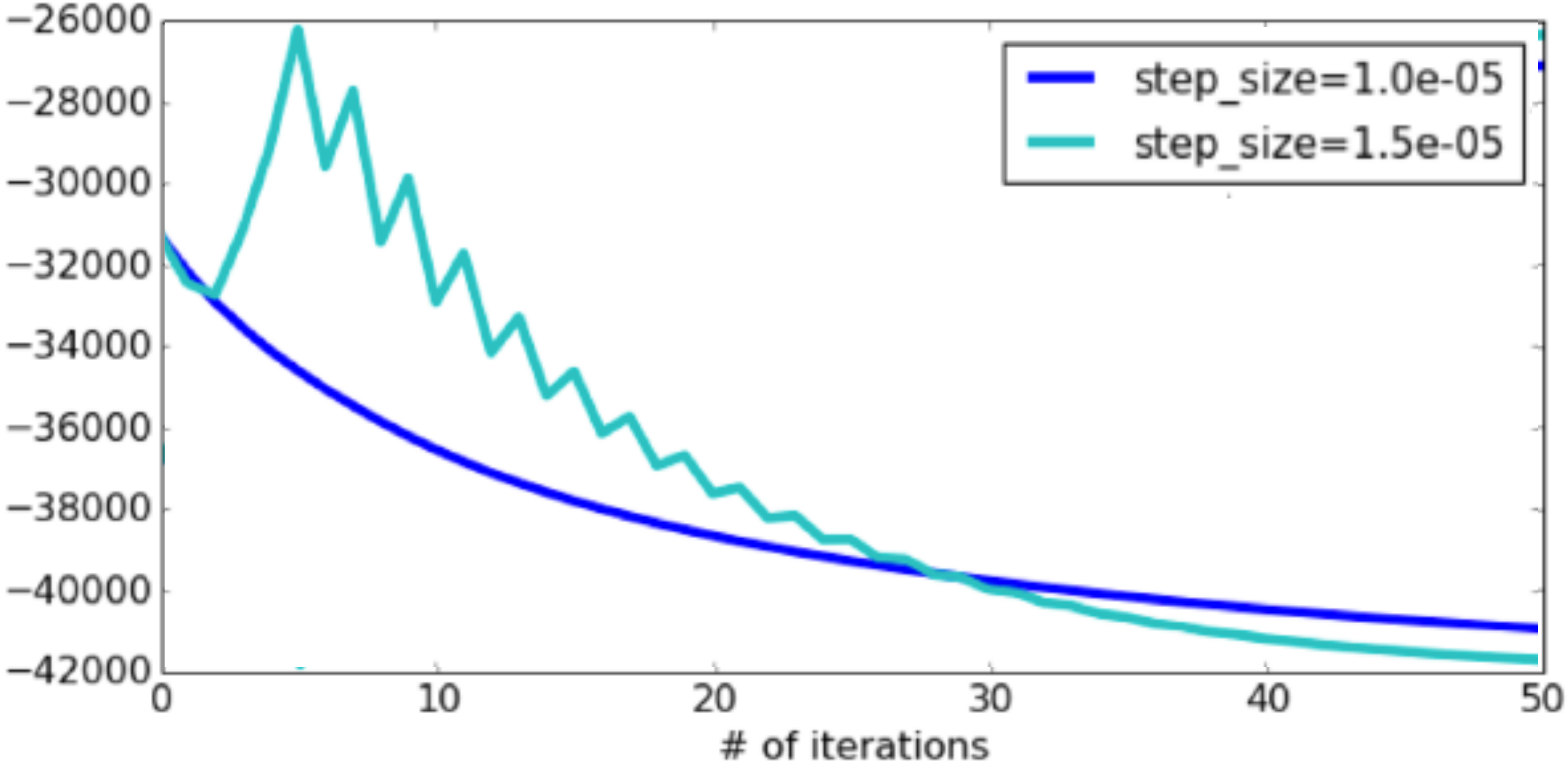


Figure Credit: Emily Fox (UW)

If step size is **way too large**

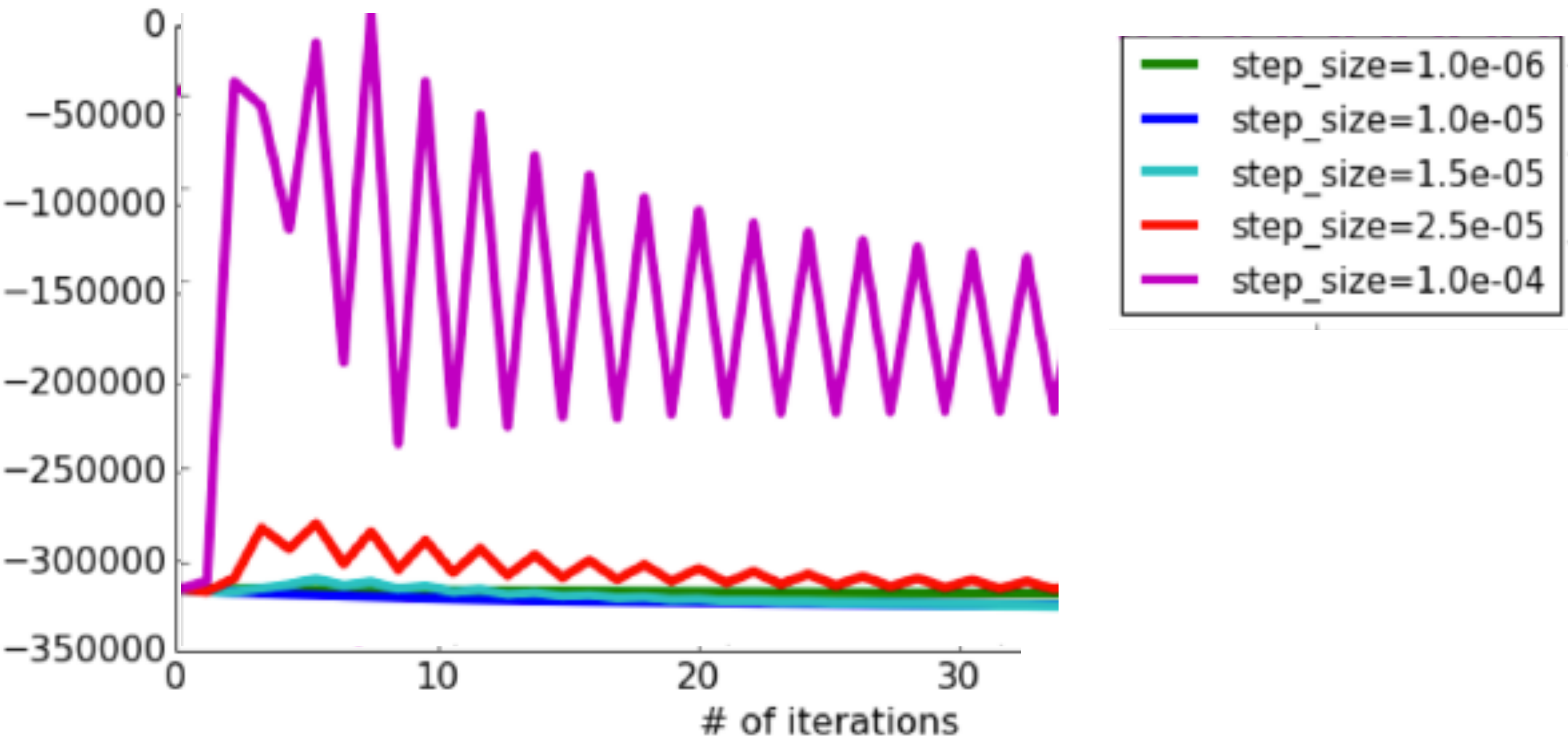
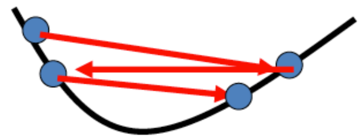
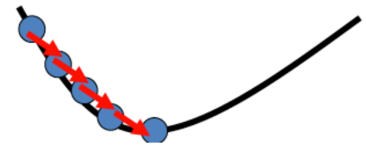


Figure Credit: Emily Fox (UW)

Hints for picking step sizes





- Don’t blindly trust the default
- Don’t try just one!

- Try several values (spaced like 10^{-4}, 10^{-2}, 1, 100) until
  - Find one clearly too small
  - Find one clearly too large (oscillation / divergence)
  - Then focus on finding “just right” value in between

- Always make trace plots!
  - loss vs iteration
  - gradient vs iteration
  - parameter vs iteration

- Smarter choices for step size (advanced material, beyond today’s class)
  - Decaying methods
  - Line search methods
  - Second-order methods
  - Adaptive methods

How to assess convergence?

- Ideal: stop when derivative equals zero

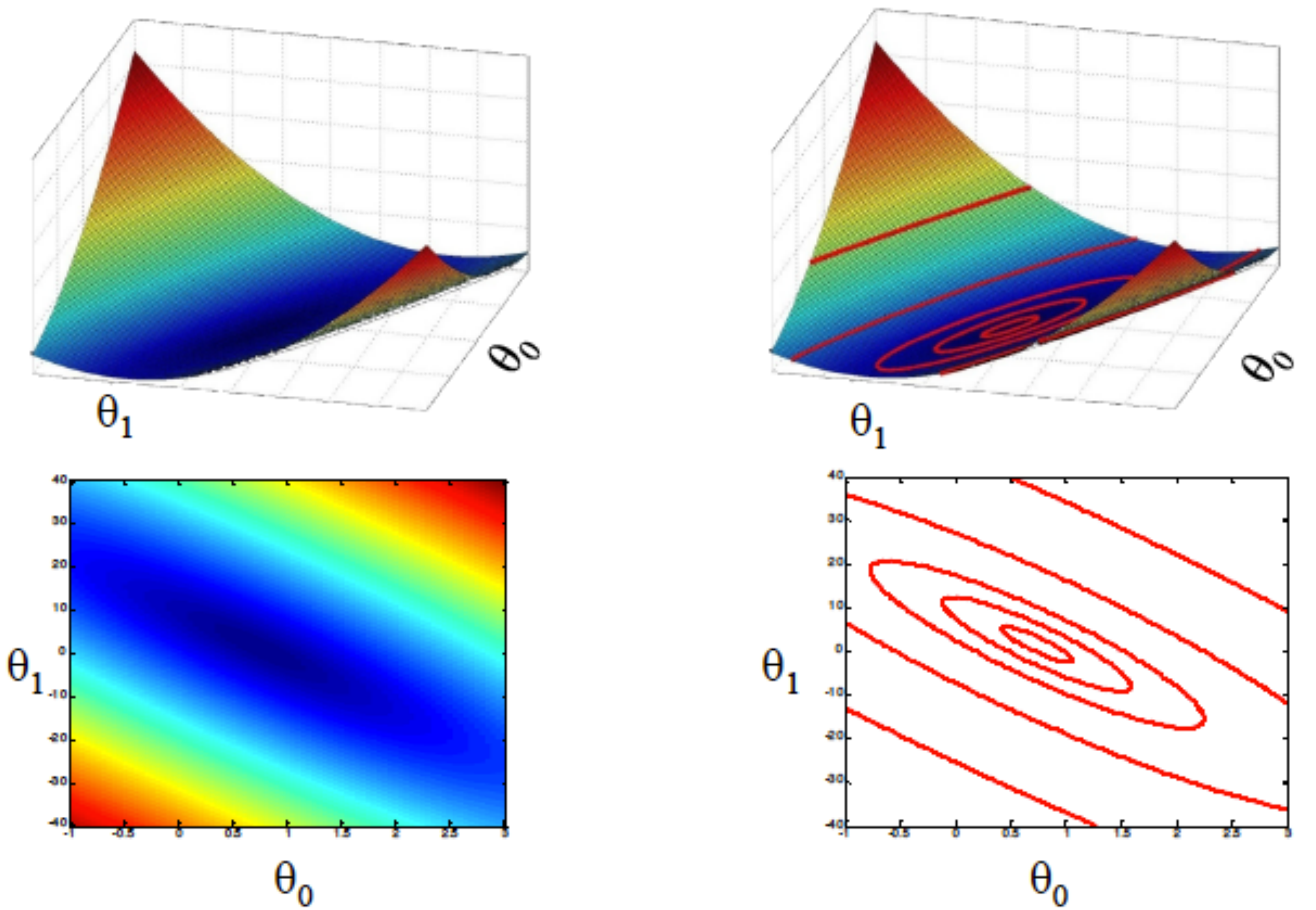
- Practical heuristics: stop when …
  - change in loss becomes "small enough"

$$|J(\theta_t) - J(\theta_{t-1})| < \epsilon$$

  - net step size is indistinguishable from zero

$$\alpha \left|\frac{d}{d\theta} J(\theta)\right| < \epsilon$$

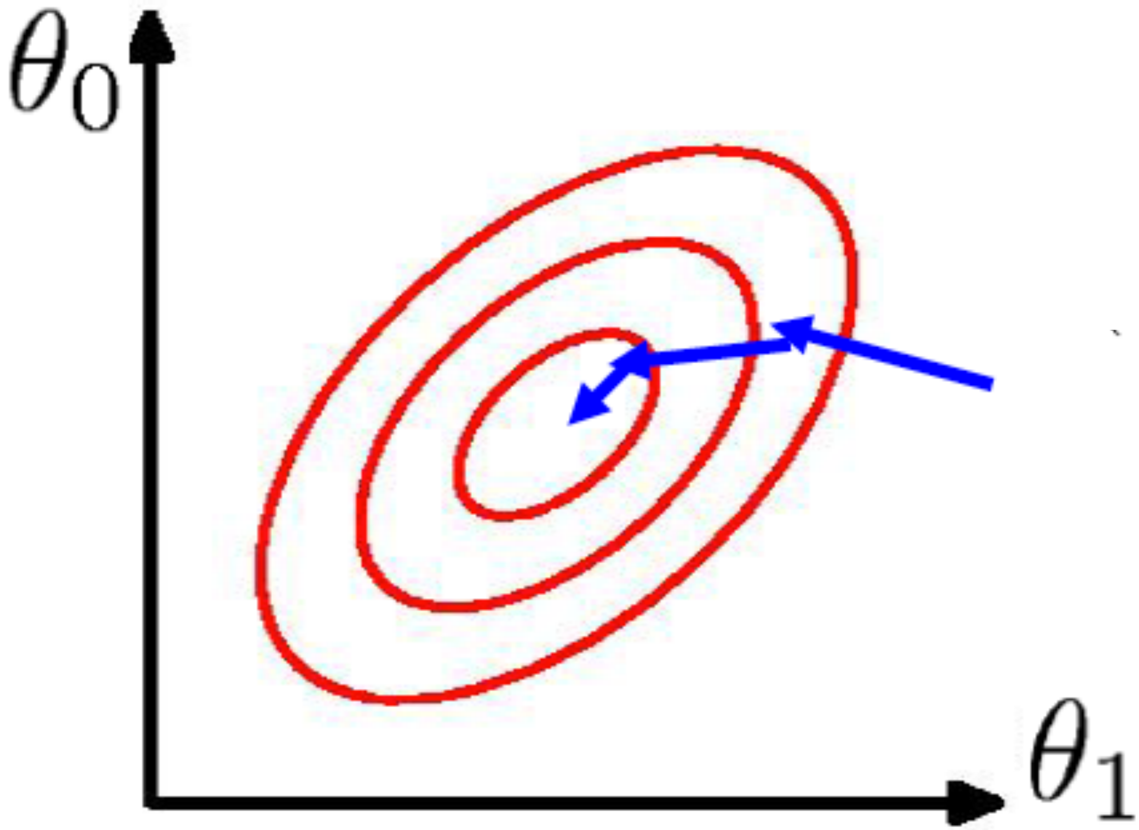
Visualizing loss function when parameter has 2 dimensions



"Level set" contours : all points with same function value

Gradient descent when parameter has 2 dimensions

gradient = vector of partial derivatives



Today's Objectives (day 06)

- Gradient descent
  - Key idea: Find optimal (or close to optimal) parameters for an objective by repeatedly stepping downhill
  - Benefits:
    - Widely useful for many ML tasks. Can be used for any objective that is differentiable
    - In contrast, exact solutions (like our formulas for optimal weights for least squares) are hard to do for most objectives
  - Practical problems:
    - Local vs. global minima
    - How to pick step size
    - How to assess convergence
    - Initialization

- Lab: Gradient descent for linear regression with one slope parameter