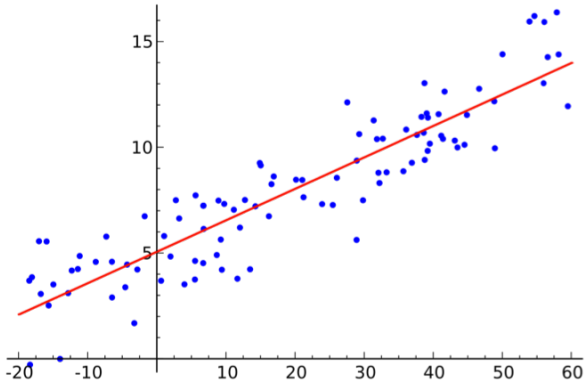
Tufts COMP 135: Introduction to Machine Learning
https://www.cs.tufts.edu/comp/135/2019s/

Linear Regression & Gradient Descent



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

Prof. Mike Hughes

LR & GD Unit Objectives

- Exact solutions of least squares
  - 1D case without bias
  - 1D case with bias
  - General case

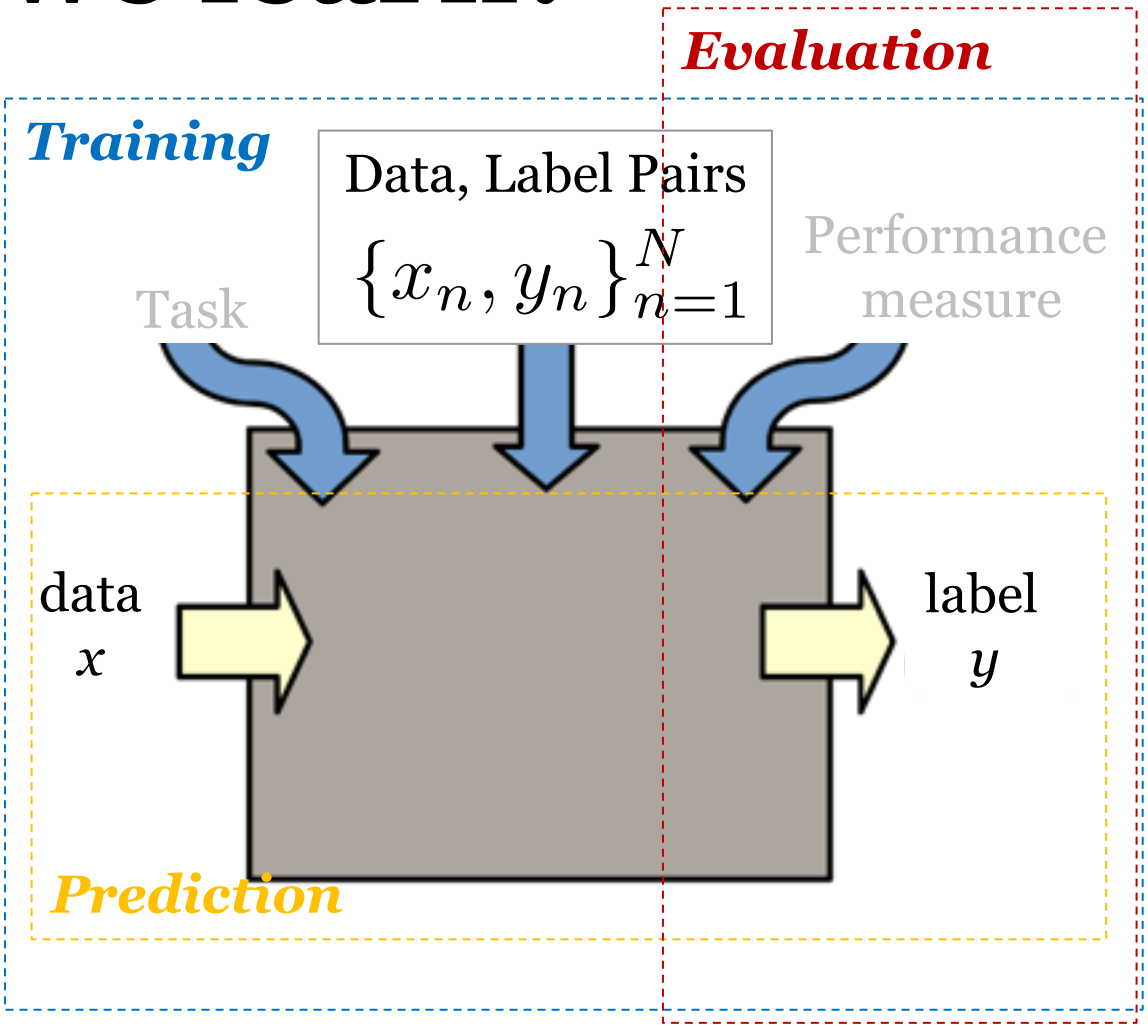
- Gradient descent for least squares

What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



Task: Regression

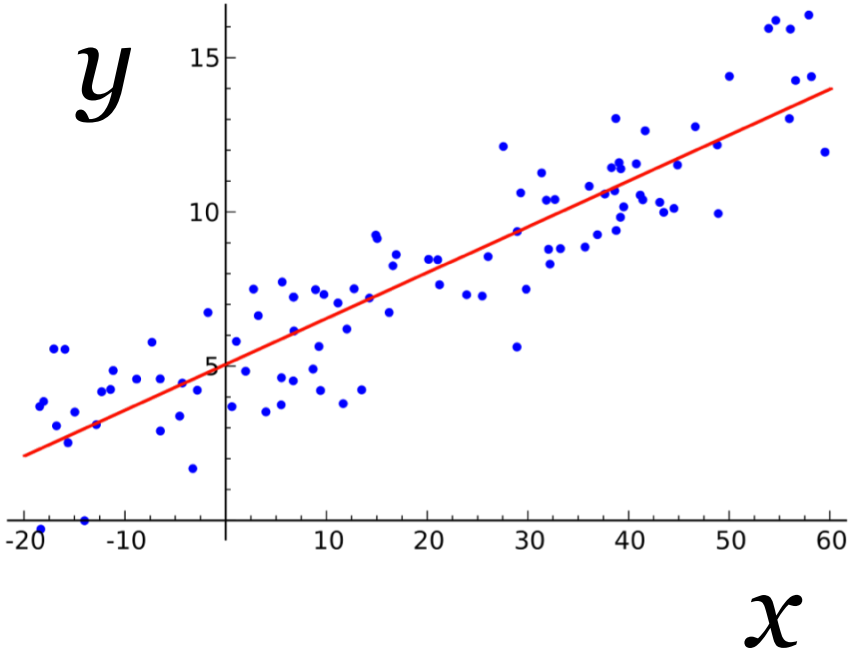
Supervised Learning

regression

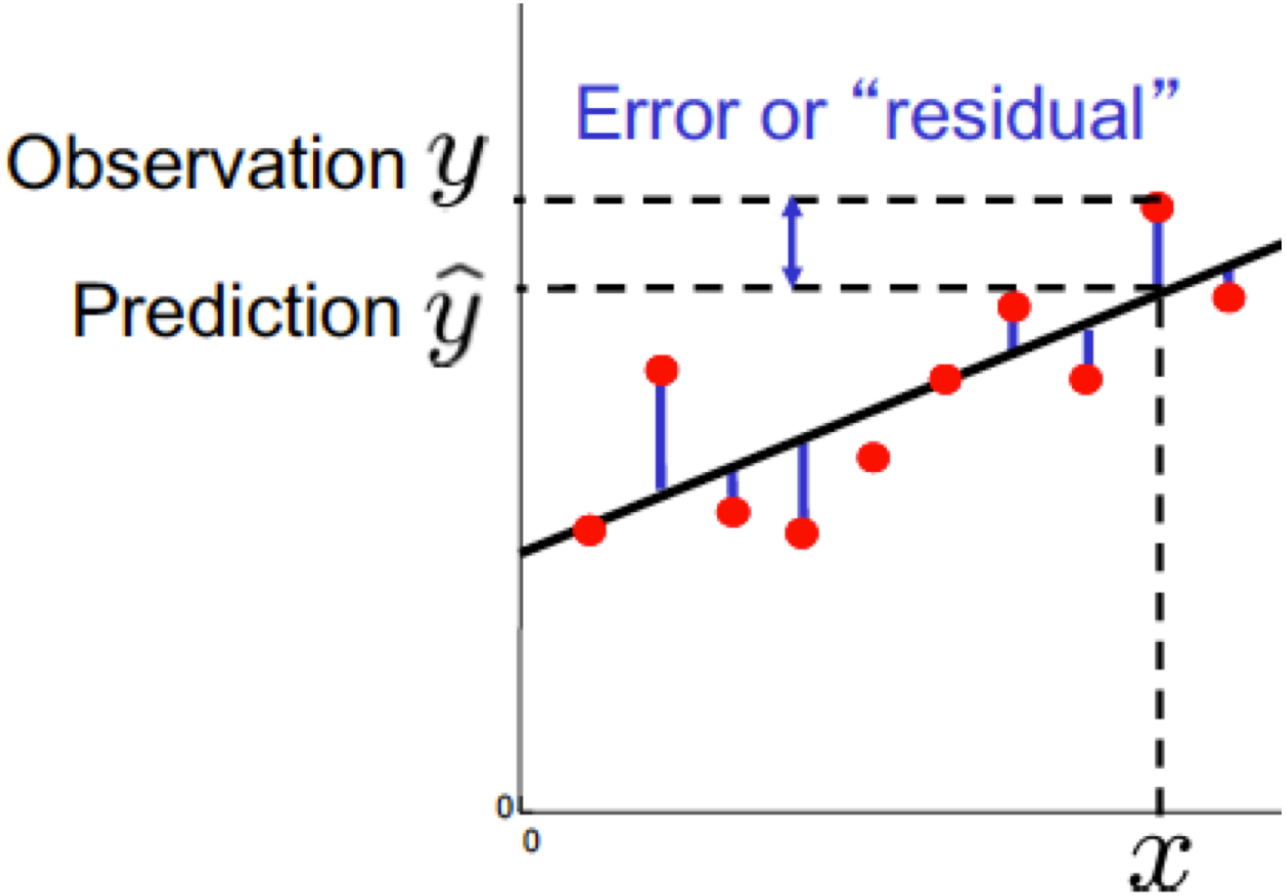
Unsupervised Learning

Reinforcement Learning

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



Visualizing errors



Regression: Evaluation Metrics

- mean squared error $$\frac{1}{N}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2$$

- mean absolute error $$\frac{1}{N}\sum_{n=1}^{N}|y_n - \hat{y}_n|$$

Linear Regression

Parameters:

weight vector $w = [w_1, w_2, \ldots w_f \ldots w_F]$

bias scalar b

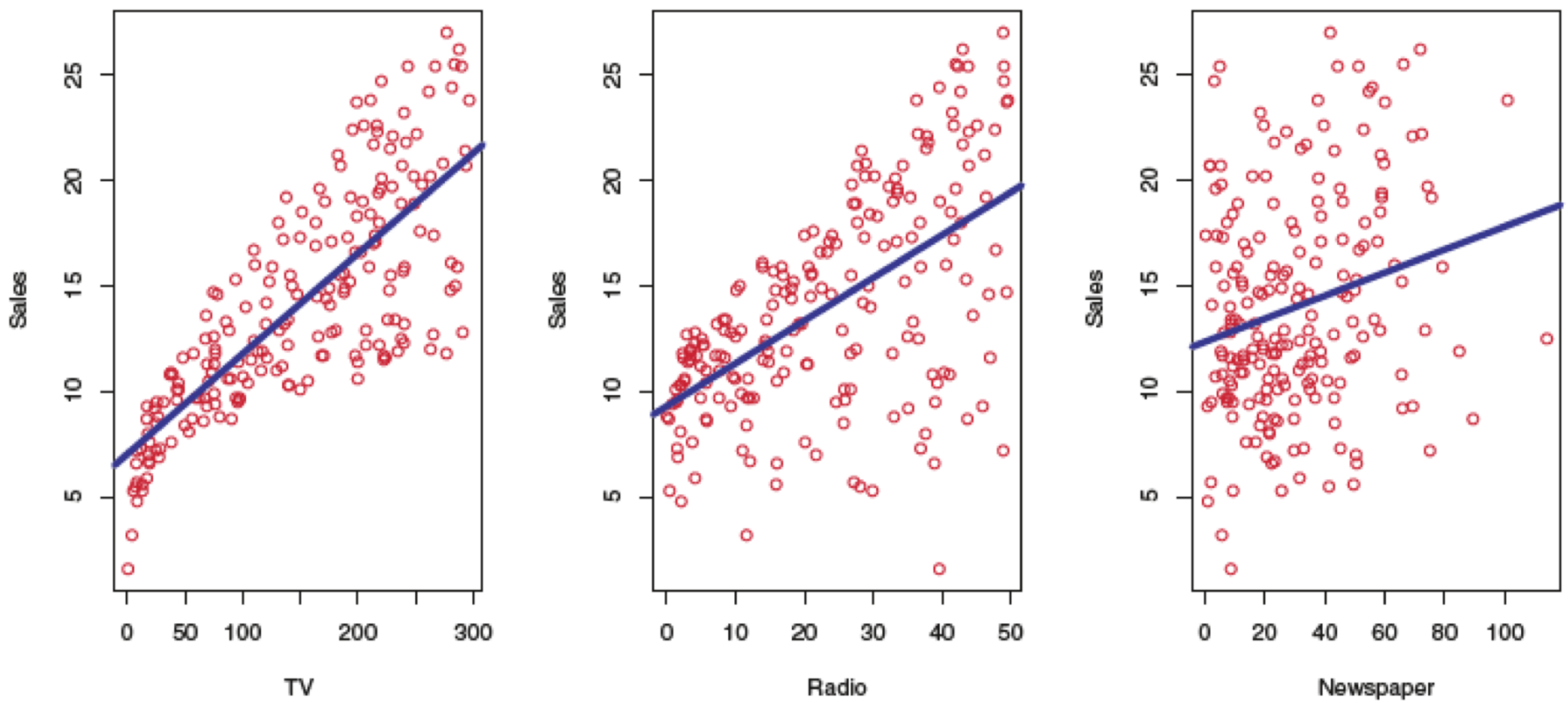
Prediction:

$$\hat{y}(x_i) \triangleq \sum_{f=1}^{F} w_f x_{if} + b$$

Training:

find weights and bias that minimize error

Sales vs. Ad Budgets



Linear Regression: Training

Optimization problem: “Least Squares”

$$\min_{w,b} \sum_{n=1}^{N} \Big(y_n - \hat{y}(x_n, w, b) \Big)^2$$

Linear Regression: Training

Optimization problem: "Least Squares"

$$\min_{w,b} \sum_{n=1}^{N} \Big(y_n - \hat{y}(x_n, w, b) \Big)^2$$

Exact formula for optimal values of w, b exist!

With only one feature (F=1):

$$w = \frac{\sum_{n=1}^{N} (x_n - \bar{x})(y_n - \bar{y})}{\sum_{n=1}^{N} (x_n - \bar{x})^2}$$

$$\bar{x} = \text{mean}(x_1, \ldots x_N)$$

$$\bar{y} = \text{mean}(y_1, \ldots y_N)$$

$$b = \bar{y} - w\bar{x}$$

Where does this come from?

Linear Regression: Training

Optimization problem: "Least Squares"

$$\min_{w,b} \sum_{n=1}^{N} \Big(y_n - \hat{y}(x_n, w, b) \Big)^2$$

Exact formula for optimal values of w, b exist!

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

With many features (F >= **1**):

$$[w_1 \dots w_F \;\; b]^T = (\tilde{X}^T \tilde{X})^{-1} \tilde{X}^T y$$

Where does this come from?

Derivation Notes

http://www.cs.tufts.edu/comp/135/2019s/notes/day03_linear_regression.pdf

When does the Least Squares estimator exist?

- Fewer examples than features (N < F)

  Infinitely many solutions!

- Same number of examples and features (N=F)

  Optimum exists if X is full rank

- More examples than features (N > F)

  Optimum exists if X is full rank

More compact notation

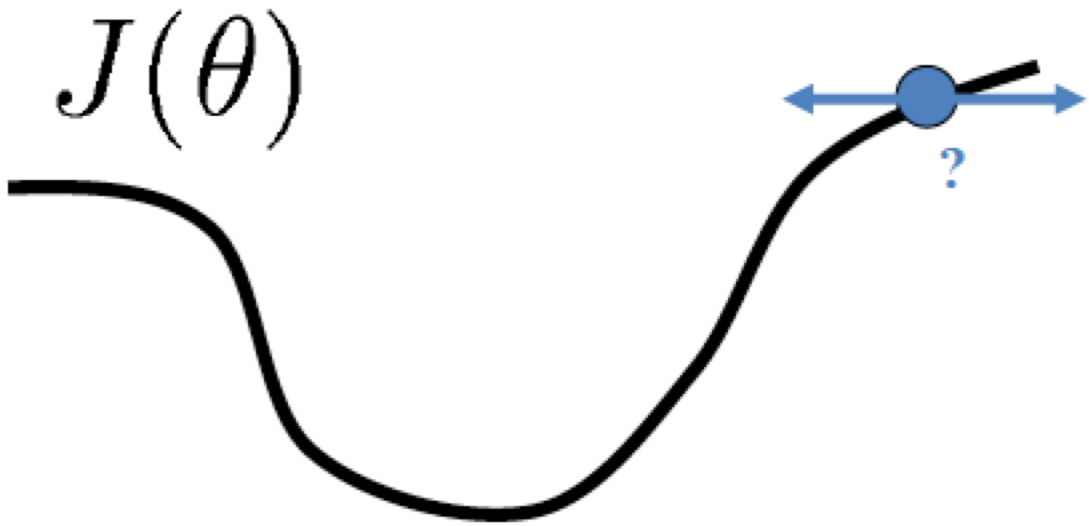
$$\theta = [b \; w_1 \; w_2 \ldots w_F]$$

$$\tilde{x}_n = [1 \; x_{n1} \; x_{n2} \ldots x_{nF}]$$

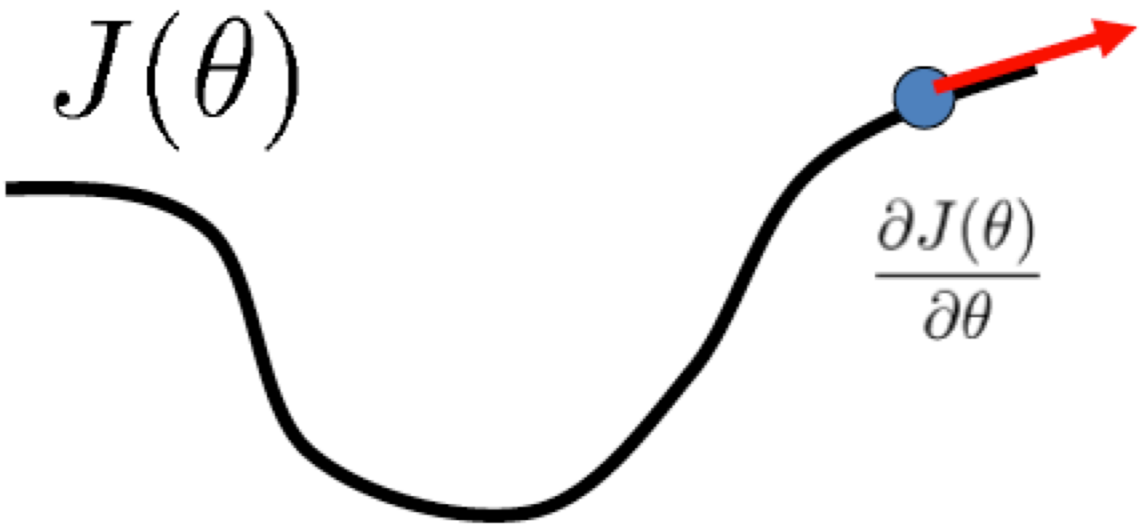
$$\hat{y}(x_n, \theta) = \theta^T \tilde{x}_n$$

$$J(\theta) \triangleq \sum_{n=1}^{N} (y_n - \hat{y}(x_n, \theta))^2$$

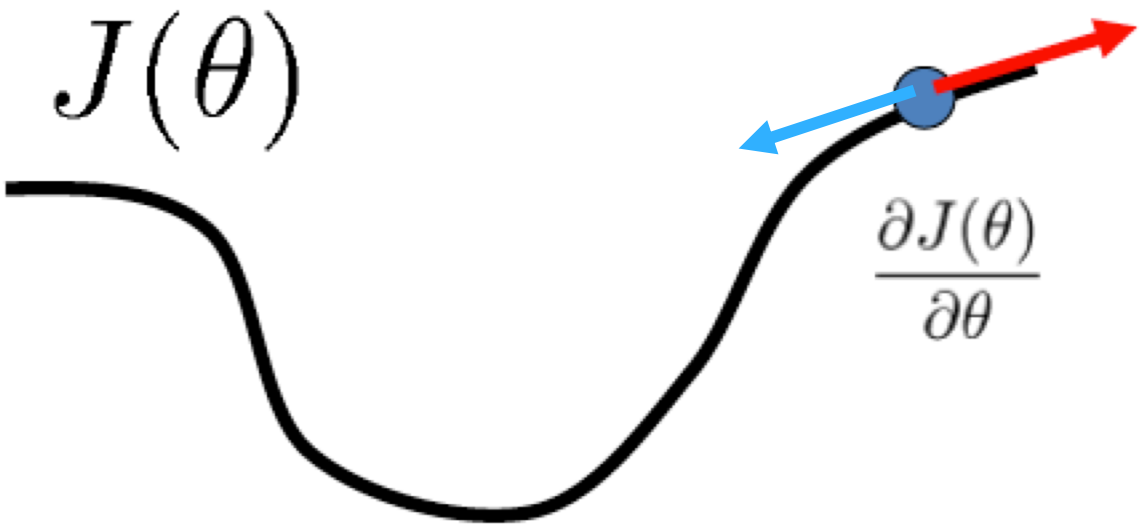
Idea: Optimize via small steps



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:

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

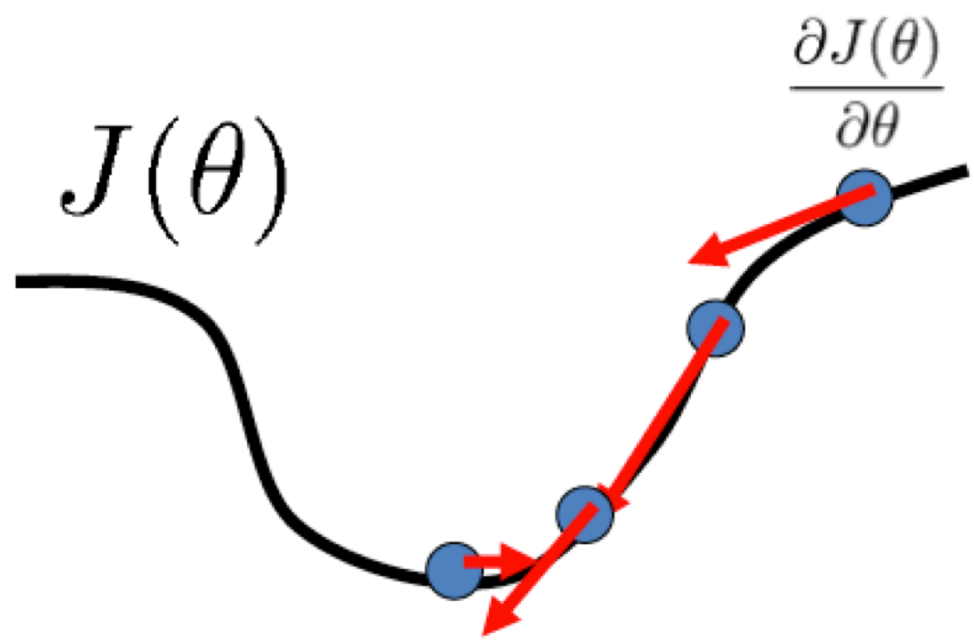
Steepest descent algorithm

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

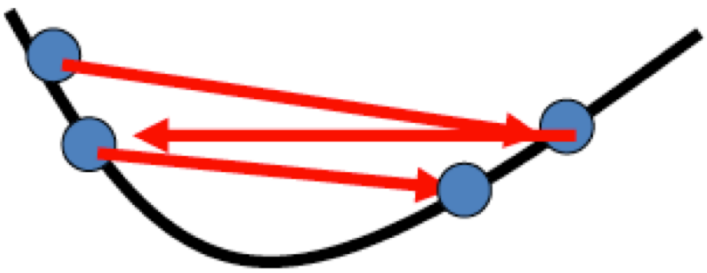
input: step size $\alpha \in \mathbb{R}_+$

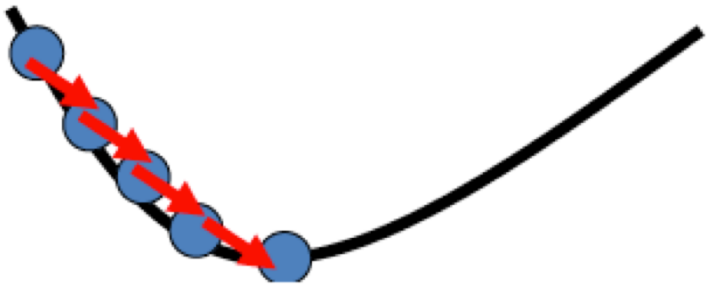
while not converged:

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

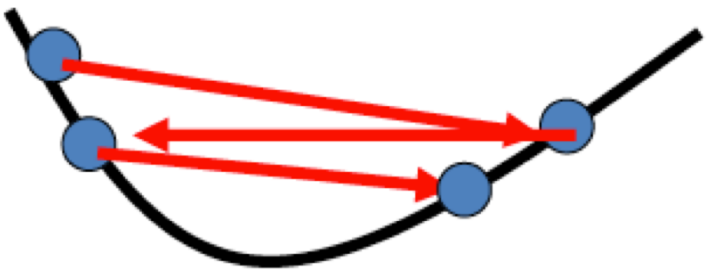


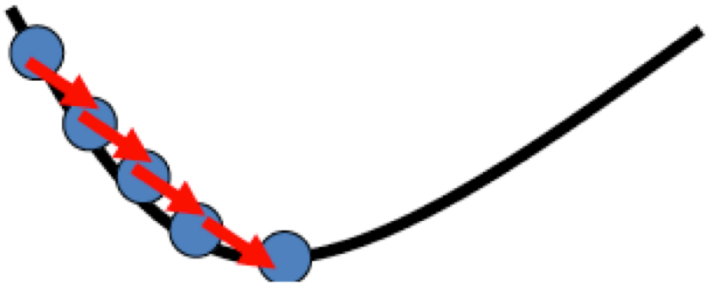
How to set step size?





How to set step size?





- Simple and usually effective: pick small constant

$$\alpha = 0.01$$

- Improve: **decay** over iterations

$$\alpha_t = \frac{C}{t} \qquad \alpha_t = (C + t)^{-0.9}$$

- Improve: Line search for best value at each step

How to assess convergence?

- Ideal: stop when derivative equals zero

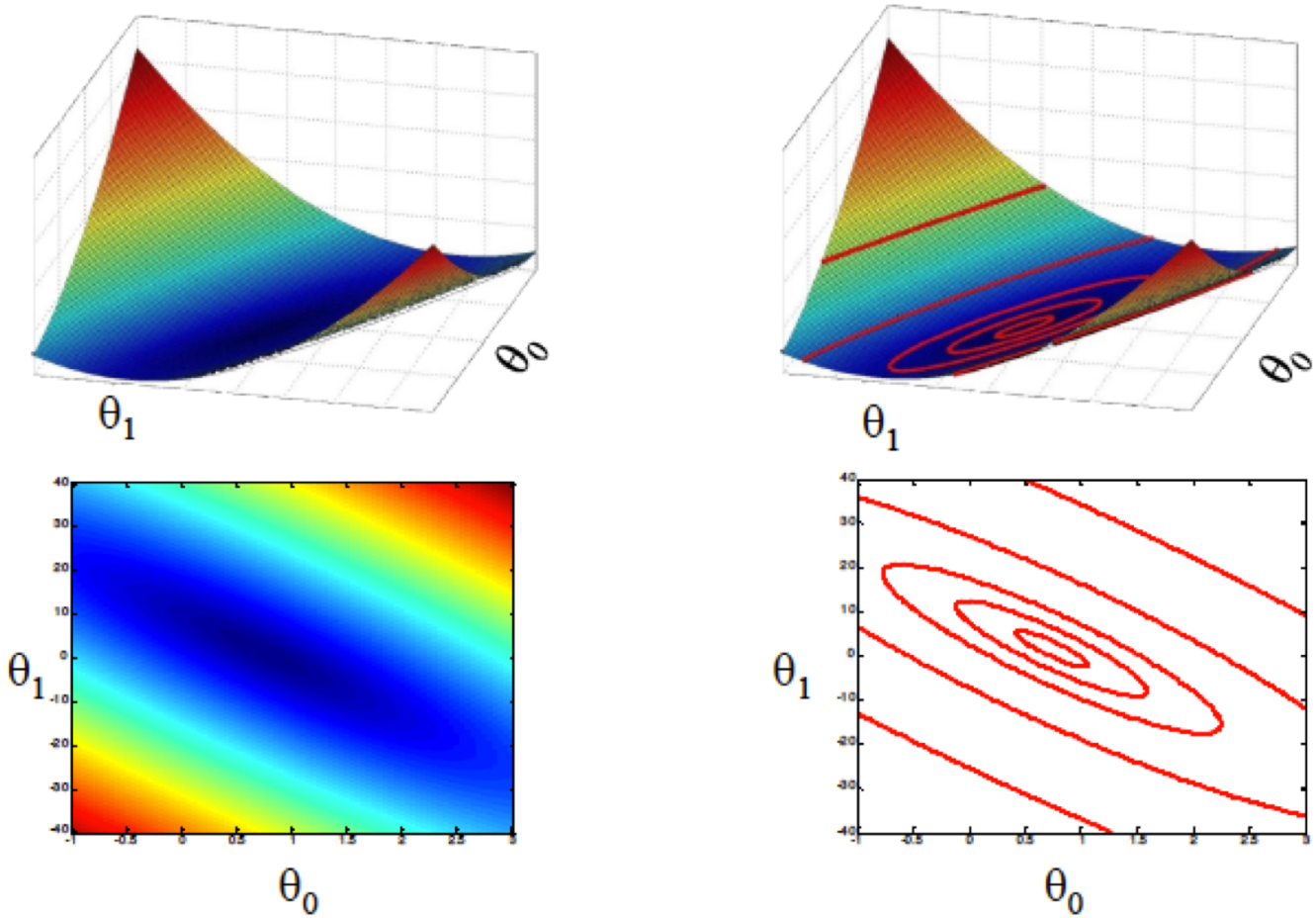
- Practical heuristics: stop when ...
  - when change in loss becomes small

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

  - when step size is indistinguishable from zero

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

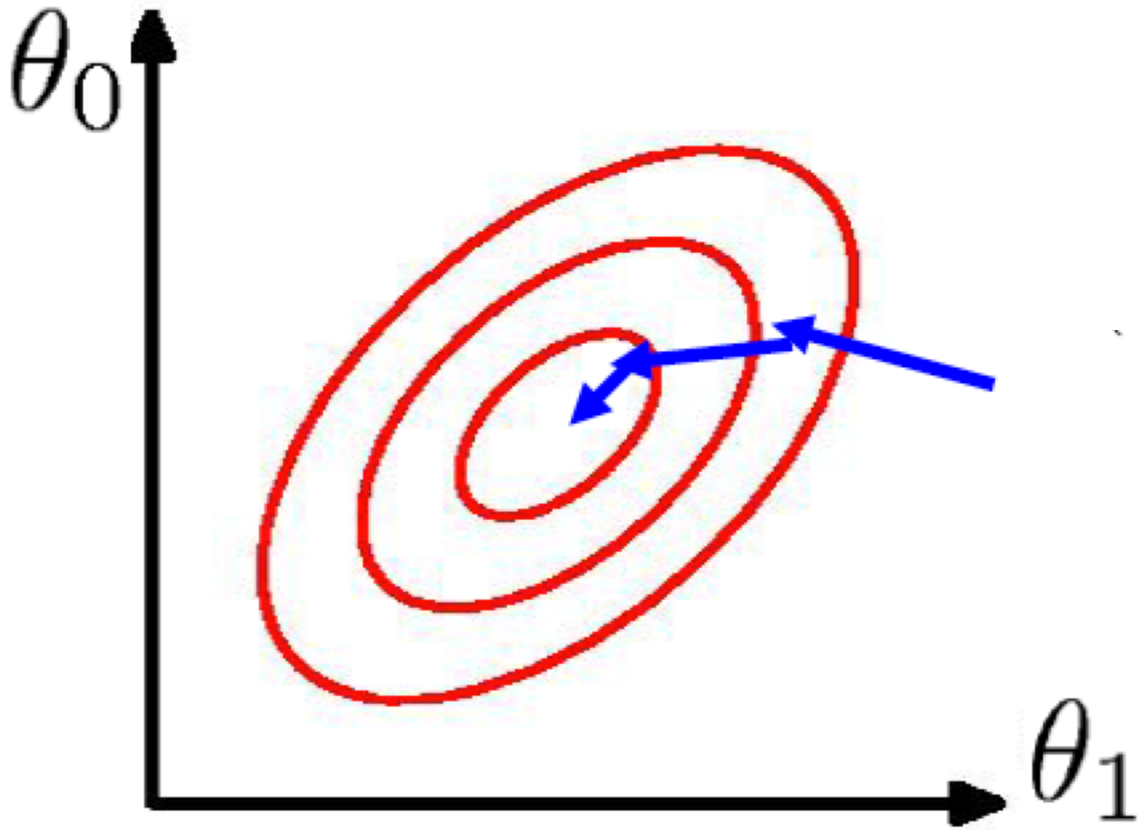
Visualizing the cost function



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

In 2D parameter space

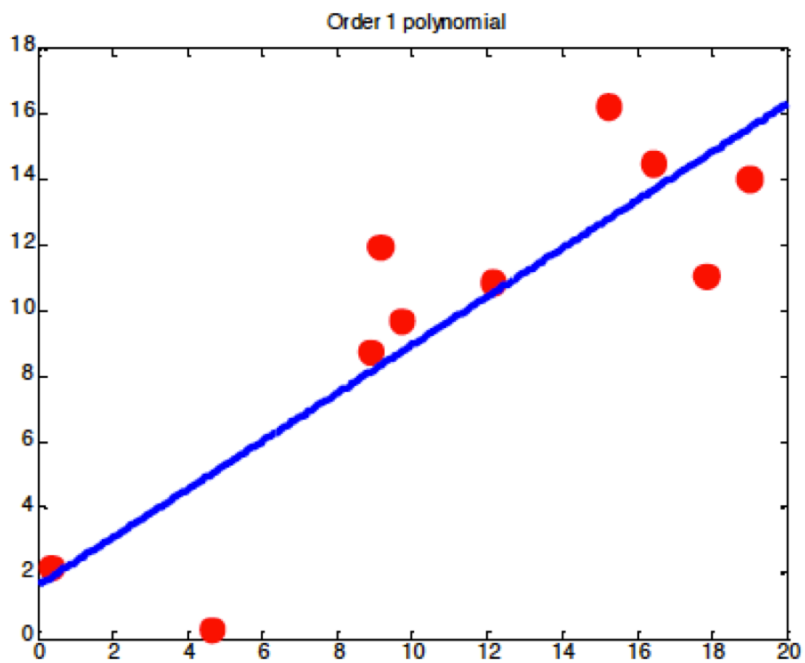
gradient = vector of partial derivatives

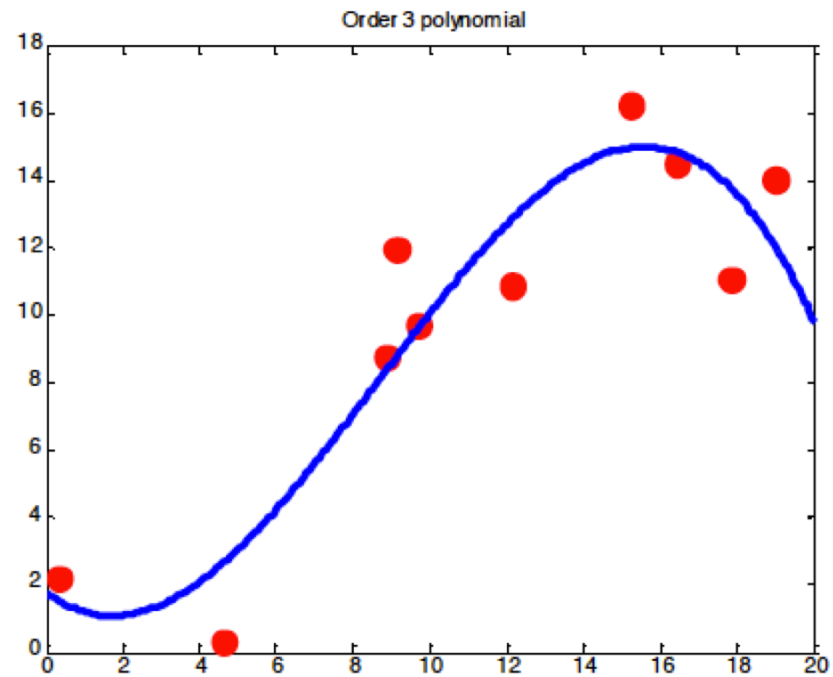


Gradient Descent DEMO

https://github.com/tufts-ml-courses/comp135-19s-assignments/blob/master/labs/GradientDescentDemo.ipynb

Fitting a line isn't always ideal





(c) Alexander Ihler

Can fit **linear** functions to **nonlinear** features

A nonlinear function of x:

$$\hat{y}(x_i) = \theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \theta_3 x_i^3$$

Can be written as a linear function of $\phi(x_i) = [x_i \quad x_i^2 \quad x_i^3]$

$$\hat{y}(\phi(x_i)) = \theta_0 + \theta_1 \phi(x_i)_1 + \theta_2 \phi(x_i)_2 + \theta_3 \phi(x_i)_3$$

"Linear regression" means linear in the parameters (weights, biases)

Features can be arbitrary transforms of raw data

What feature transform to use?

- Anything that works for your data!
  - sin / cos for periodic data
  - polynomials for high-order dependencies

$$\phi(x_i) = [x_i \quad x_i^2 \quad x_i^3]$$

  - interactions between feature dimensions

$$\phi(x_i) = [x_{i1}x_{i2} \quad x_{i3}x_{i4}]$$

  - Many other choices possible