DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

February 4 2021

Outline

- Linear regression
  - Feature standardization
  - Outliers
- Gradient descent optimization
  - General algorithm
  - Instantiation for linear regression

Solution for simple linear regression

- Dataset $x_i \in R, y_i \in R, h_\theta(x) = \theta_0 + \theta_1 x$
- $J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\theta_0 + \theta_1 x_i - y_i)^2$ MSE / Loss

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{2}{N}\sum_{i=1N}(\theta_0 + \theta_1 x_i - y_i) = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{2}{N}\sum_{i=1}^{N} x_i(\theta_0 + \theta_1 x_i - y_i) = 0$$

- Solution of min loss

$$-\ \theta_0 = \bar{y} - \theta_1 \bar{x}$$

$$-\ \theta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{cov(x, y)}{Var(x)}$$

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$\bar{y} = \frac{\sum_{i=1}^{N} y_i}{N}$$

Multiple Linear Regression

- Dataset: $x_i \in R^{d}, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{N} \sum (\theta^T x_i - y_i)^2$

$$X = \begin{bmatrix} 1 & \cdots \\ 1 & \cdots \\ \vdots & \\ 1 & \cdots \end{bmatrix}$$

TRAINING EXAMPLE i

FEATURE j

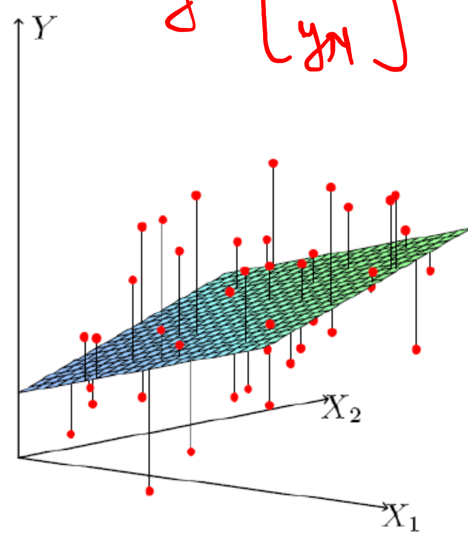
size: $N \times (d+1)$

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

VECTOR OF RESPONSES

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$



Closed-form optimum
solution for linear regression

Supervised Learning: Regression

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled

$x_i, y_i \in R$

Normalization
Standardization

Feature
Selection

Regression

$f(x)$

Testing

New data → Learning model → Predictions

$y' = f(x') \in R$

Unlabeled

x'

$f(x)$

Response variable

Regression

Feature Standardization

Input $X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1j} & \cdots & x_{1d} \\ 1 & & & & & \\ \vdots & x_{i1} & \cdots & x_{ij} & \cdots & x_{id} \\ 1 & x_{N1} & \cdots & x_{Nj} & \cdots & x_{Nd} \end{bmatrix}$

$$\mu_j = \frac{1}{N} \sum_{i=1}^{N} x_{ij}$$ SAMPLE MEAN

$$s_j = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \mu_j)^2}$$

SAMPLE STD DEV.

1) $x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{s_j}$

MEAN 0

STD. DEV: 1

Practical issues: Feature Standardization

AT TRAINING:

- Rescales features to have zero mean and unit variance
  - Let μ_j be the mean of feature j:

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

  - Replace each value with:

$$\boxed{x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{s_j}}$$

for j = 1...d
(not x_0!)

    - s_j is the standard deviation of feature j

AT TESTING:

- SAME TRANSFORMATION USING μ_j & s_j FROM TRAINING

Other feature normalization

2) • Min-Max rescaling

– $x_{ij} \leftarrow \frac{x_{ij} - min_j}{max_j - min_j}$ $\in [0,1]$

– min_j and max_j: min and max value of feature j

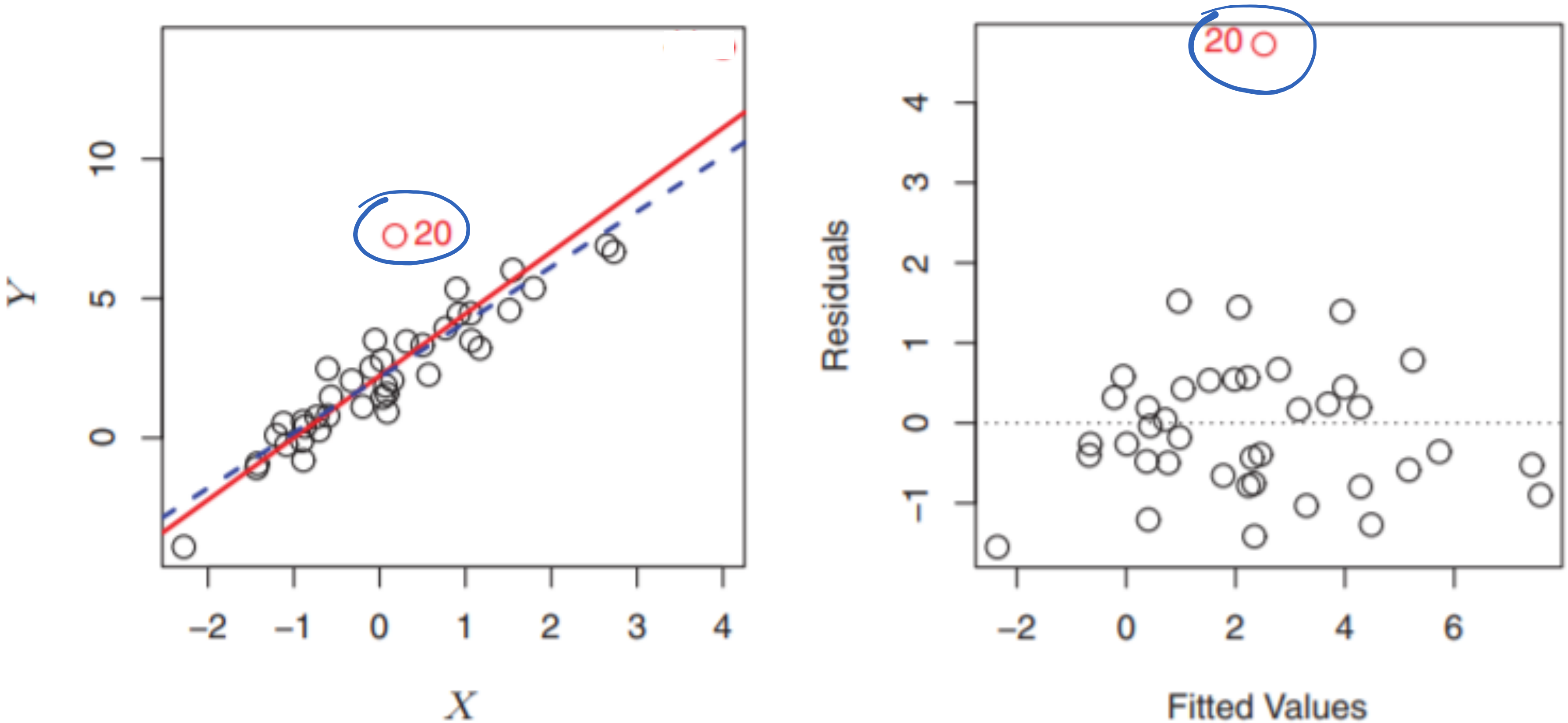
3) • Mean normalization

– $x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{max_j - min_j}$ MEAN 0

Feature standardization/normalization

- Goal is to have individual features on the same scale
- Is a pre-processing step in most learning algorithms
- Necessary for linear models and Gradient Descent
- Different options:
  - Feature standardization
  - Feature min-max rescaling
  - Mean normalization

Practical issues: Outliers



- Dashed model is without outlier point
- Linear regression is not resilient to outliers!
- Outliers can be eliminated based on residual value
- Other methods to eliminate outliers (anomaly detection)

Categorical variables

- Predict credit card balance
  - Age
  - Income
  - Number of cards
  - Credit limit
  - Credit rating
- Categorical variables
  - Student (Yes/No)
  - State (50 different levels)

How to generate numerical representations of these?

Indicator Variables

- One-hot encoding
- Binary (two-level) variable
  - Add new feature $x_j = 1$ if student and 0 otherwise
- Multi-level variable
  - State: 50 values
  - $x_{MA} = 1$ if State $=$ MA and $0,$ otherwise
  - $x_{NY} = 1$ if State $=$ NY and $0,$ otherwise
  - …
  - How many indicator variables are needed?
- Disadvantages: data becomes too sparse for large number of levels
  - Will discuss feature selection later in class

How to optimize loss functions?

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$

MSE

- $J(\theta) = \frac{1}{N} \sum (\theta^T x_i - y_i)^2$ Loss / cost
  - Strictly convex function (unique minimum)
- General method to optimize a multi-variate function
  - Practical (low asymptotic complexity)
  - Convergence guarantees to global minimum

GRADIENT DESCENT

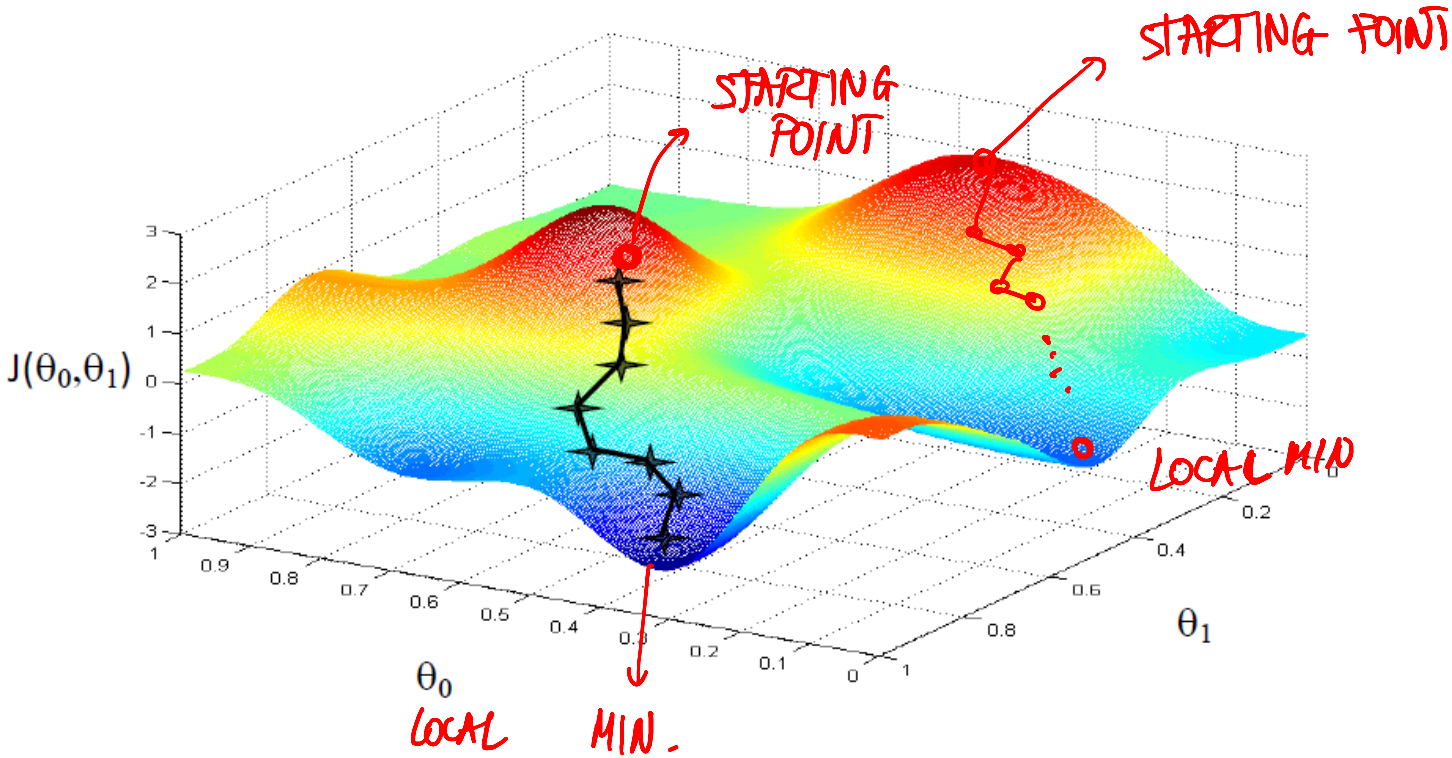
What Strategy to Use?

Follow the Slope

Follow the direction of steepest descent!

How to optimize $J(\theta)$?

- Choose initial value for θ RANDOM
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Gradient Descent

GOAL: FIND θ TO MIN $J(\theta)$

1. INITIALIZE θ AT RANDOM
2. REPEAT UNTIL STOPPING CONDITIONS
3. $\theta_j \leftarrow \theta_j - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_j}$; $j = 0, 1, \dots, d$

$$\theta = \begin{bmatrix} \theta_0 \\ \vdots \\ \theta_d \end{bmatrix}$$

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

VECTOR UPDATE RULE

Gradient Descent

ASSIGNMENT

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

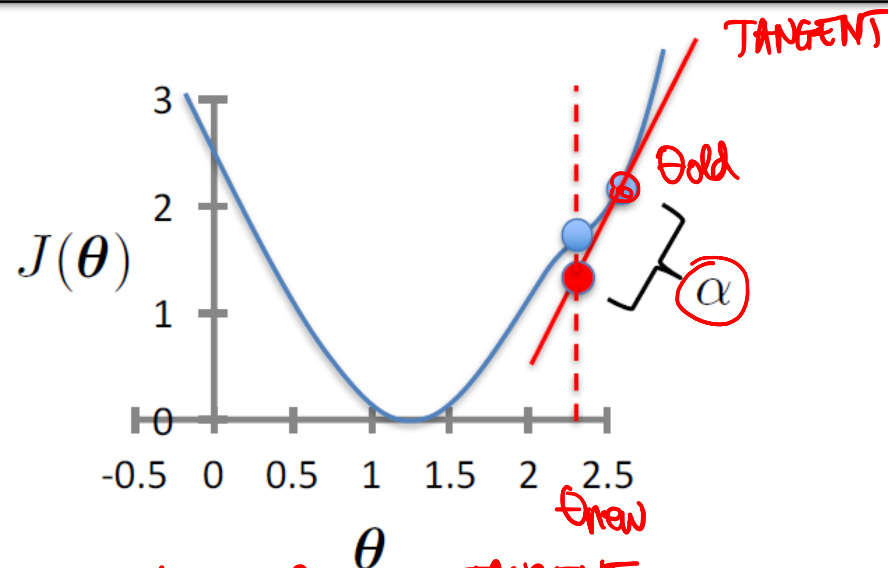
new old old old

$$\theta_j \leftarrow \theta_j - \alpha \boxed{\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})}$$

simultaneous update
for j = 0 ... d

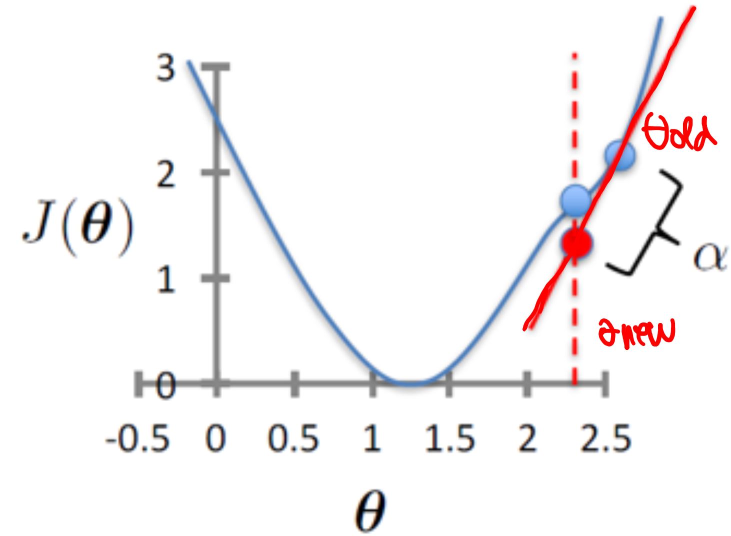
learning rate (small)
e.g., α = 0.05

α = STEP SIZE



DIRECTION: SLOPE OF TANGENT
NEGATIVE DIRECTION OF GRADIENT

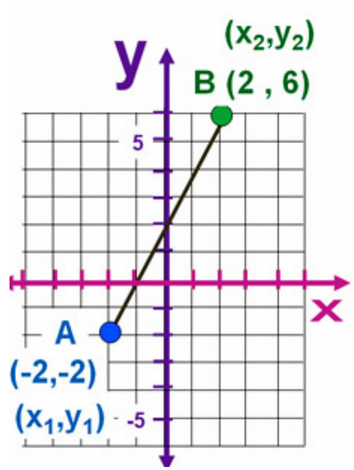
Gradient Descent



$\theta \leftarrow \theta - \alpha \cdot m$

$\theta - 2\alpha$.

DECREASE



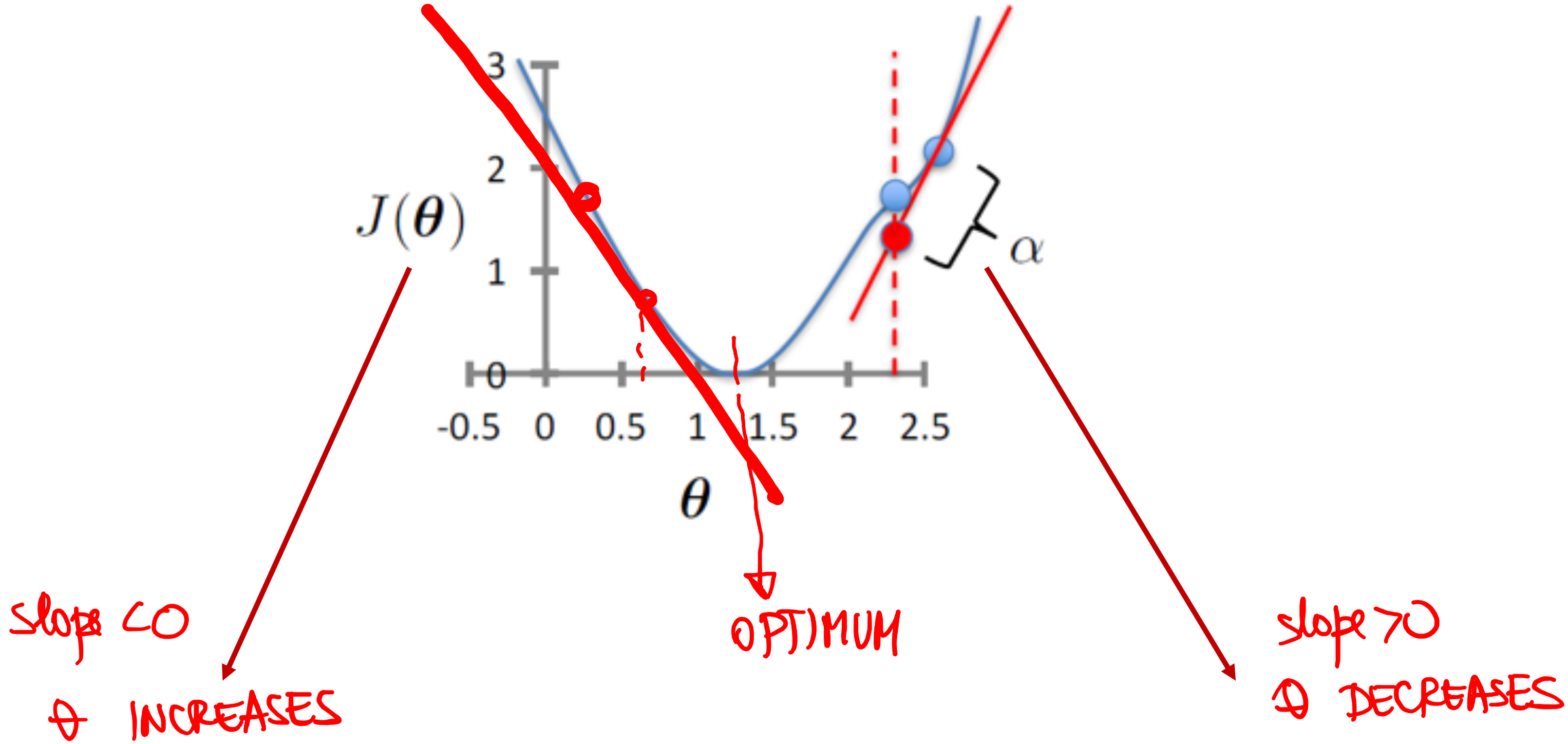
The Gradient "m" is:

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta Y}{\Delta X}$$

$$m = \frac{6 - {}^-2}{2 - {}^-2}$$

$$m = 8 / 4 = 2 \checkmark$$

Gradient Descent



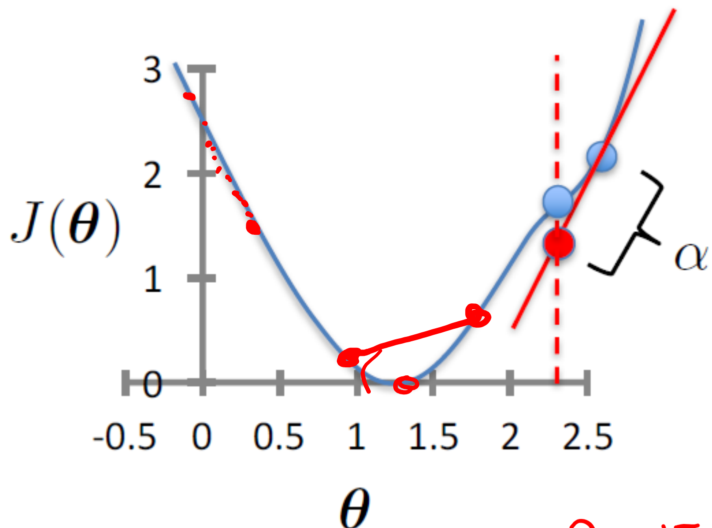
Stopping Condition

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



- When should the algorithm stop?

1) IF $\theta_{new} = \theta_{old}$ STOP
2) IF $\|\theta_{new} - \theta_{old}\| < \varepsilon$, STOP
3) BOUND NUMBER ITERATIONS

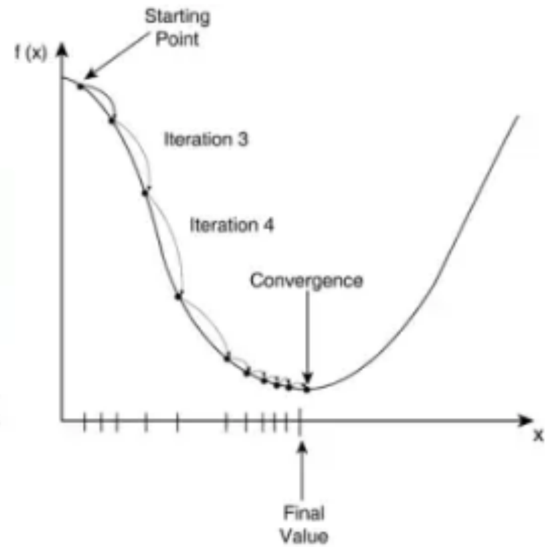
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

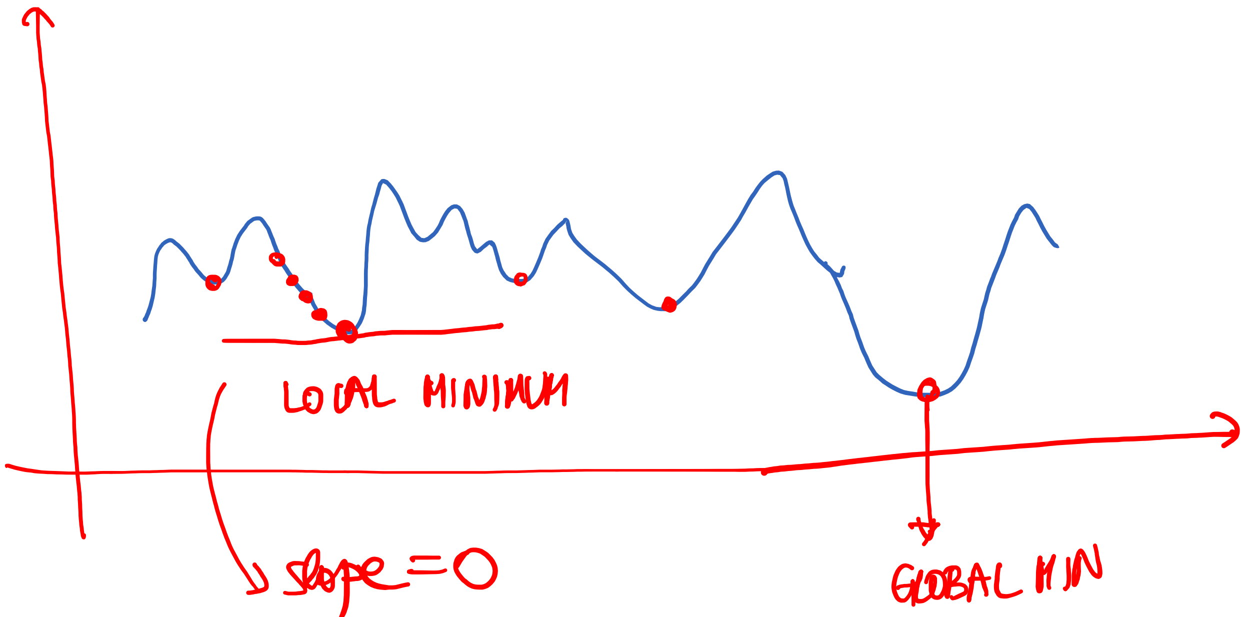
simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



- As you approach the minimum, the slope gets smaller, and GD will take smaller steps

LOCAL MIN VS. GLOBAL MIN



$$\theta \leftarrow \theta - \alpha \underbrace{\boxed{\frac{\partial J(\theta)}{\partial \theta}}}_{0}$$

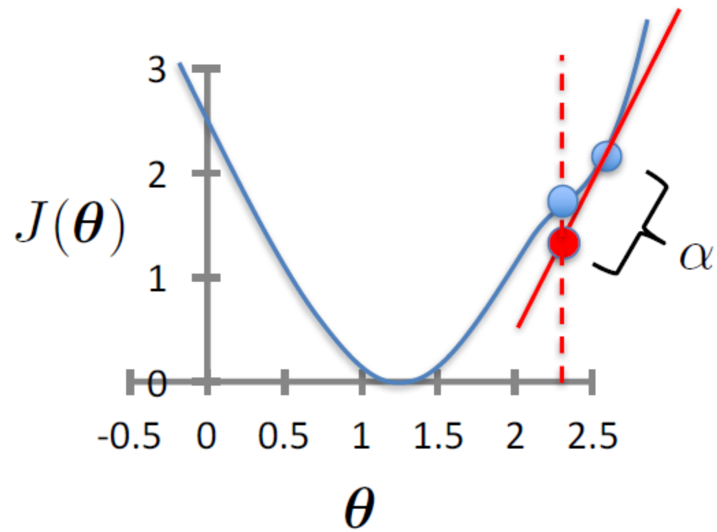
Gradient Descent

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

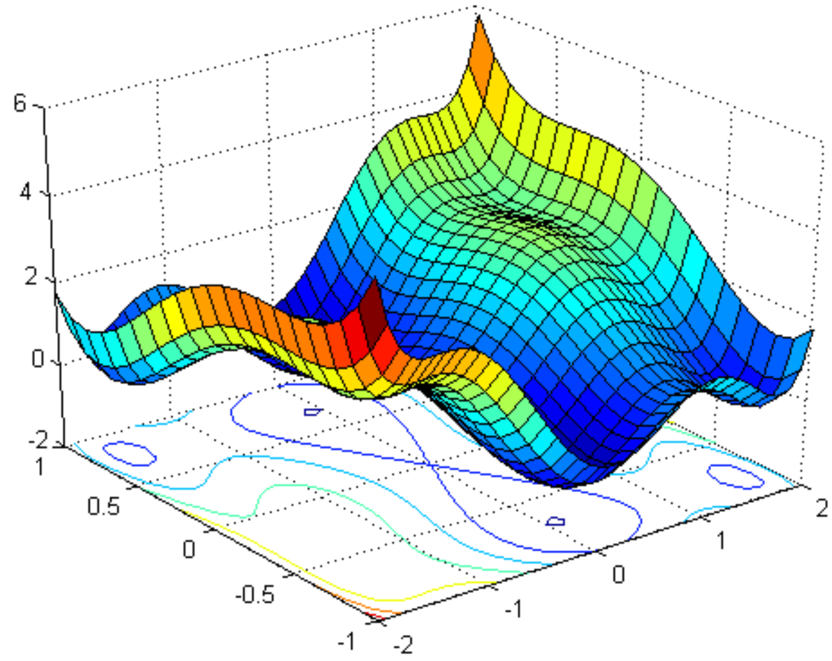
simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



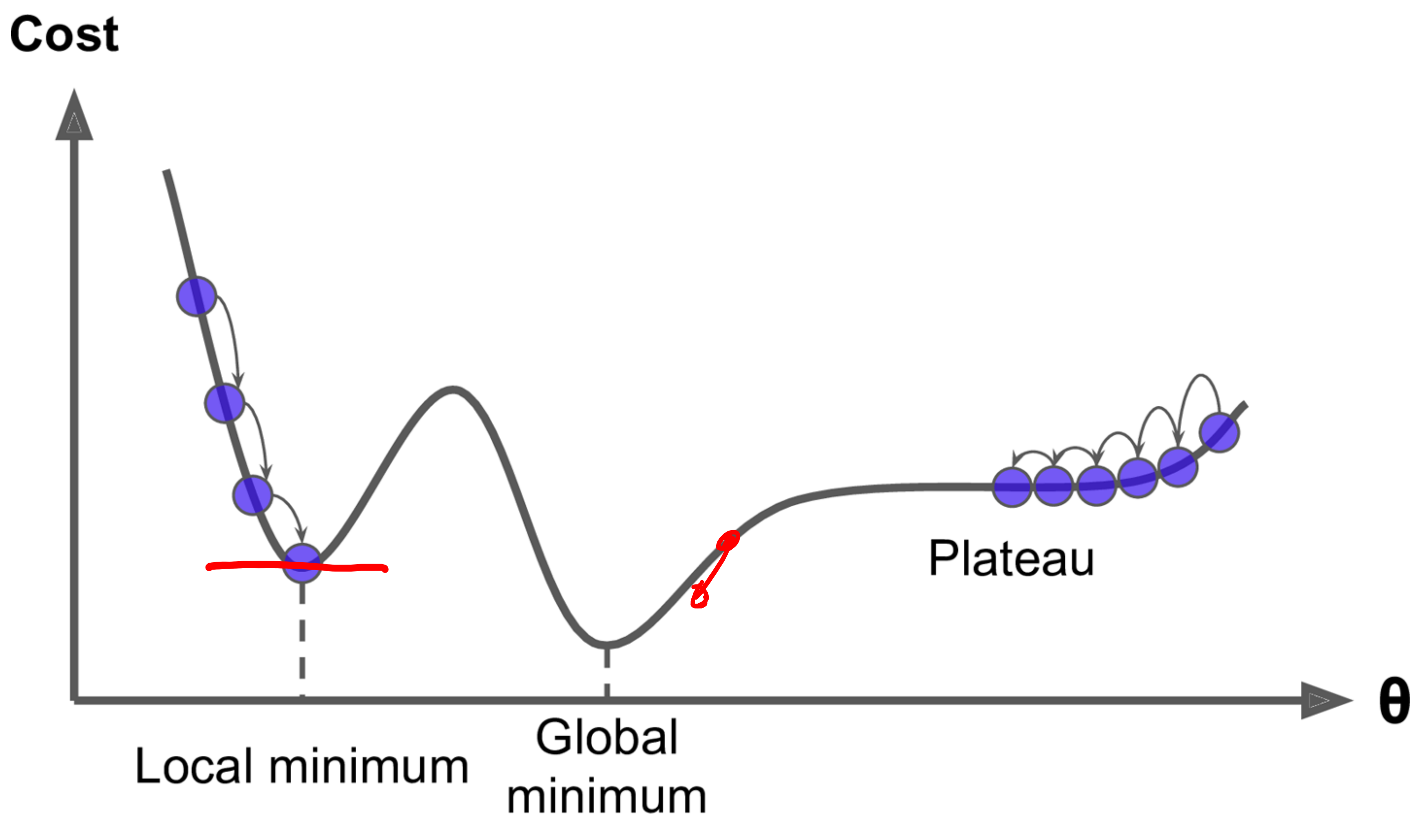
- What happens when θ reaches a local minimum?

Complex loss function

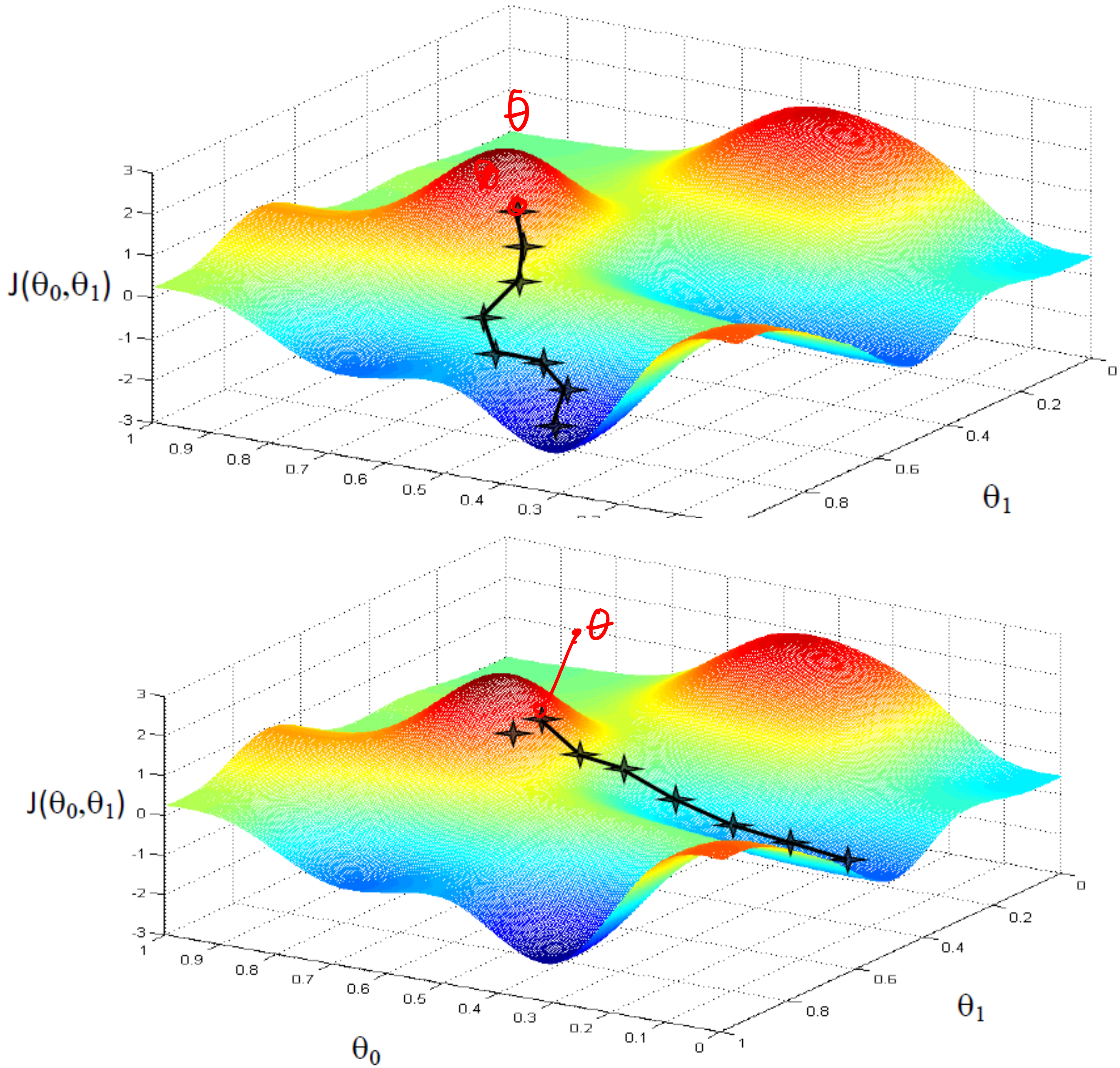


- Complex loss functions are more difficult to optimize

GD Convergence Issues



- Local minimum: Gradient descent stops
- Plateau: Almost flat region where slope is small



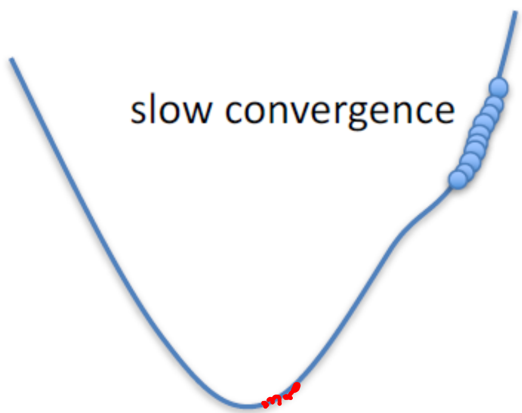
$J(\theta_0,\theta_1)$
θ_1
$J(\theta_0,\theta_1)$
θ_0
θ_1

Adjusting Learning Rate

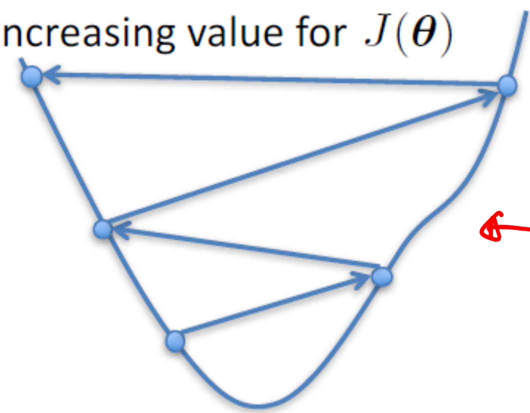
α too small	α too large

Choosing Learning Rate

α too small



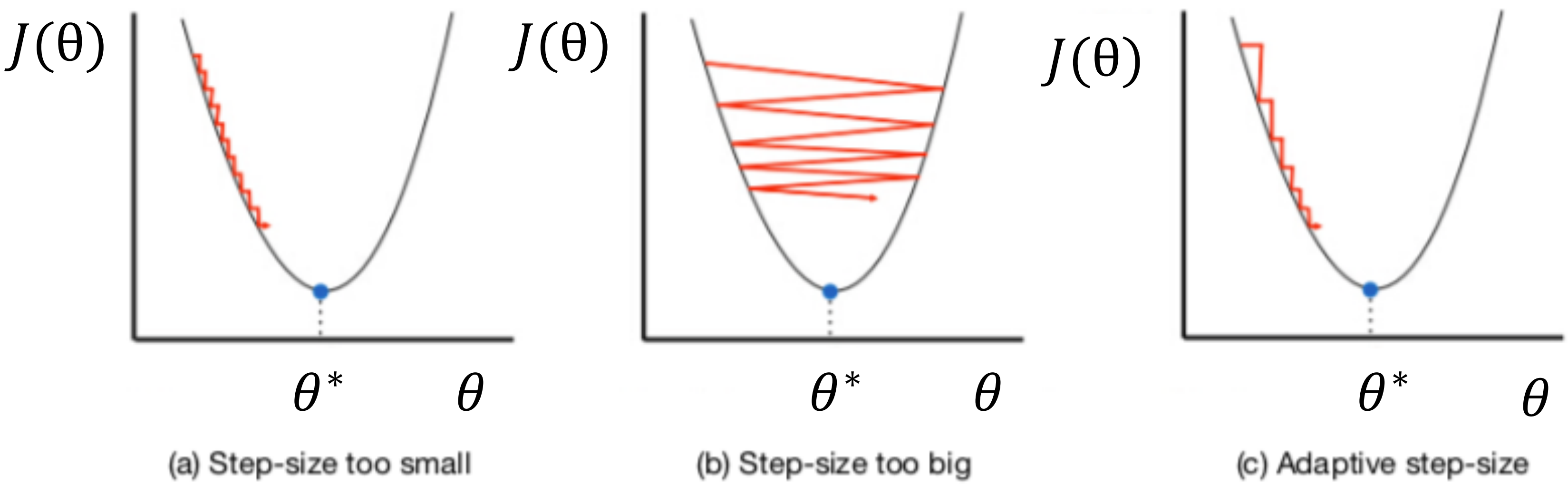
α too large



- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\boldsymbol{\theta})$ each iteration
- The value should decrease at each iteration
- If it doesn't, adjust α

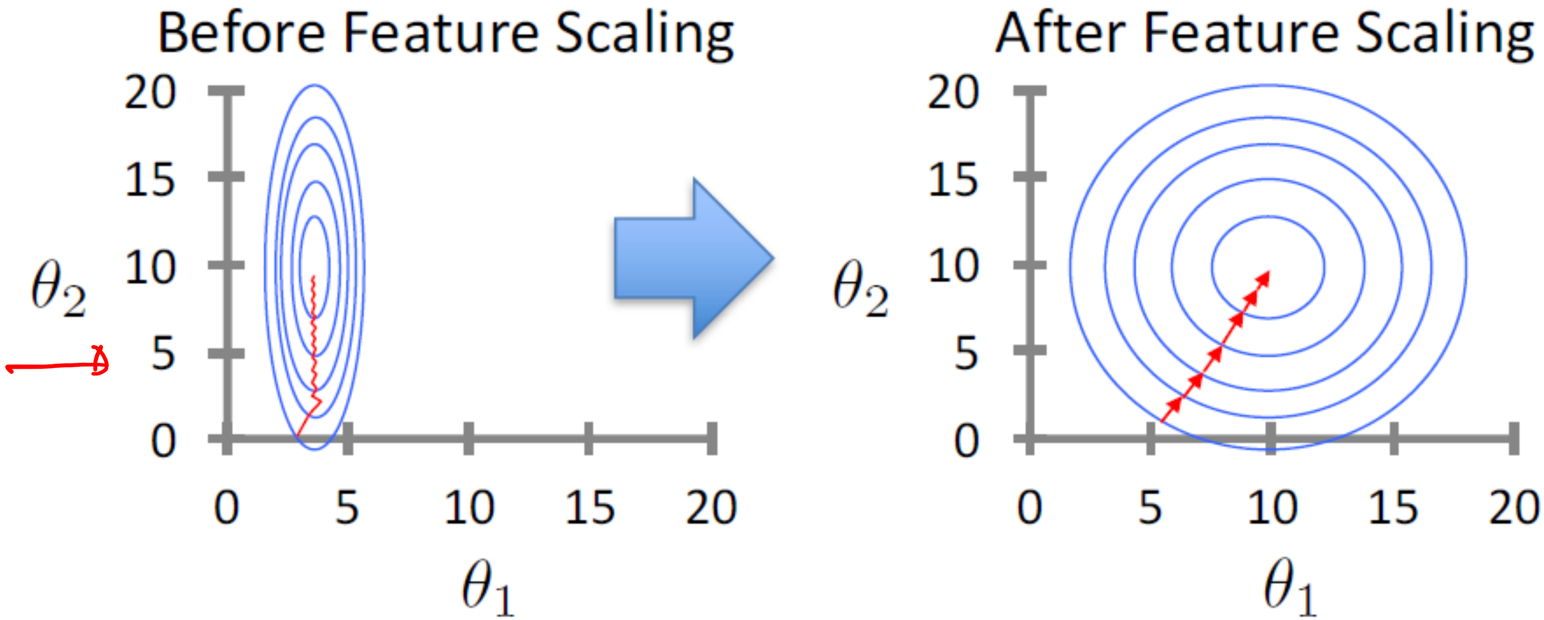
Adaptive step size



- Start with large step size and reduce over time, adaptively
- Line search method
- Measure how objective decreases

Feature Scaling

- **Idea:** Ensure that feature have similar scales



- Makes gradient descent converge *much* faster

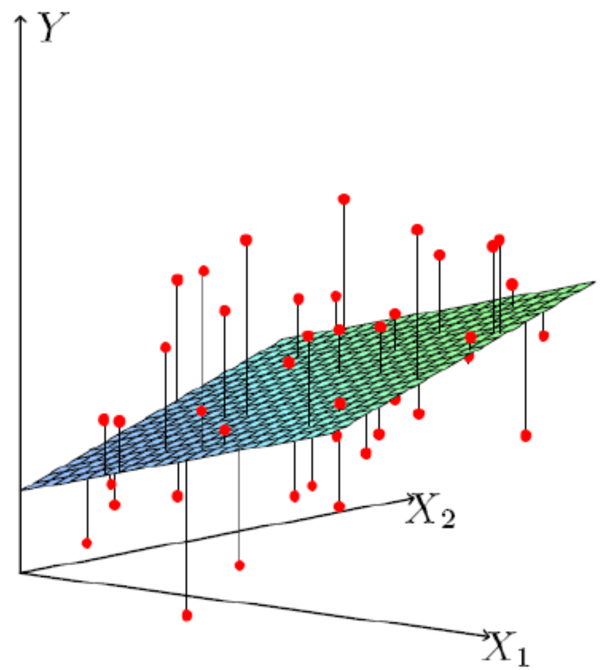
Multiple Linear Regression

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{N}\sum(\theta^T x_i - y_i)^2$ Loss / cost

$h_\theta(x_i)$

$$\theta = (X^\top X)^{-1} X^\top y$$

MSE is a strictly convex function and has unique minimum



GD for Multiple Linear Regression

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

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

$$h_\theta(x) = X\theta$$

$$J(\theta) = \frac{1}{N} \| h_\theta(x) - y \|^2 = \frac{1}{N} \| X\theta - y \|^2$$

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{N} 2 (X\theta - y)^T \cdot X$$

$$\boxed{\theta \leftarrow \theta - \alpha \frac{2}{N} (X\theta - y)^T \cdot X}$$

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[h_\theta(x_i) - y_i\right] = \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=0}^{d}\theta_k x_{ik} - y_i\right]^2$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{2}{N}\sum_{i=1}^{N}\left[\sum \theta_k x_{ik} - y_i\right]\cdot x_{ij}$$

$$= \frac{2}{N}\sum_{i=1}^{N}\left[h_\theta(x_i) - y_i\right] x_{ij}$$

$$\boxed{\theta_j \leftarrow \theta_j - \alpha \frac{2}{N}\sum_{i=1}^{N}\left[h_\theta(x_i) - y_i\right] x_{ij}}$$

GD for Linear Regression

- Initialize θ
- Repeat until convergence $||\theta_{new} - \theta_{old}|| < \epsilon$ or iterations == MAX_ITER

$$\theta_j \leftarrow \theta_j - \alpha \frac{2}{N} \sum_{i=1}^{N} (\boxed{h_\theta(x_i)} - y_i) x_{ij}$$

θ_{old}

simultaneous update for j = 0 ... d

- Assume convergence when $\|\boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old}\|_2 < \epsilon$

L_2 norm: $\|\boldsymbol{v}\|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$

Gradient Descent in Practice

- Asymptotic complexity
  - N is size of training data, d is feature dimension, and T is number of iterations
- Most popular optimization algorithm in use today
- At the basis of training
  - Linear Regression
  - Logistic regression
  - SVM
  - Neural networks and Deep learning
  - Stochastic Gradient Descent variants