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

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

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

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

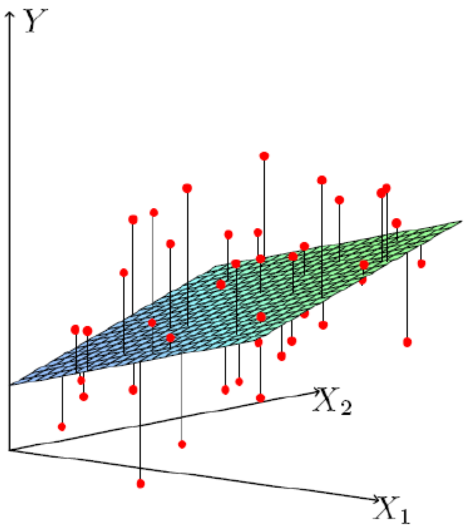
$$\bar{y} = \frac{\sum_{i=1}^{N} y_i}{N}$$

Multiple Linear Regression

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

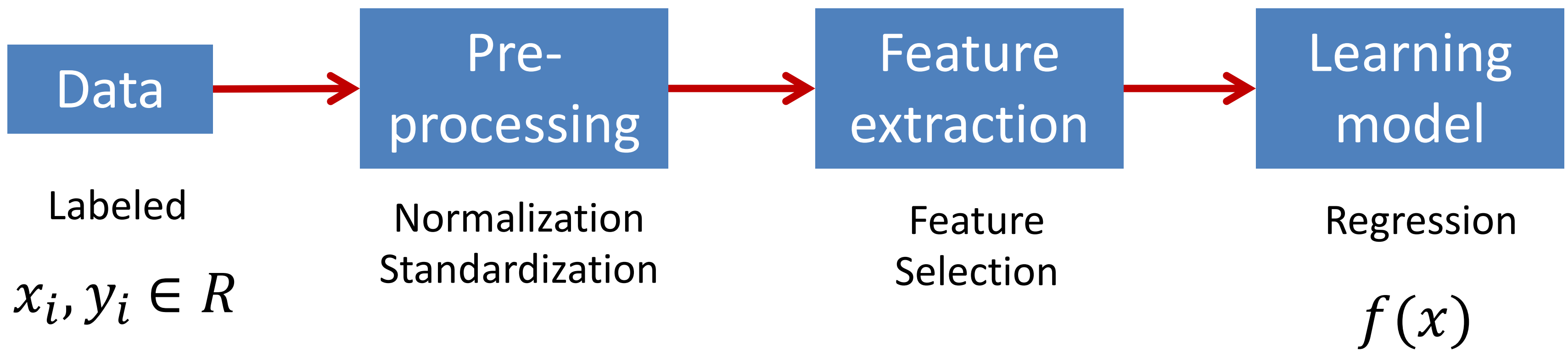
$$\theta = (X^\intercal X)^{-1} X^\intercal y$$

Closed-form optimum
solution for linear regression

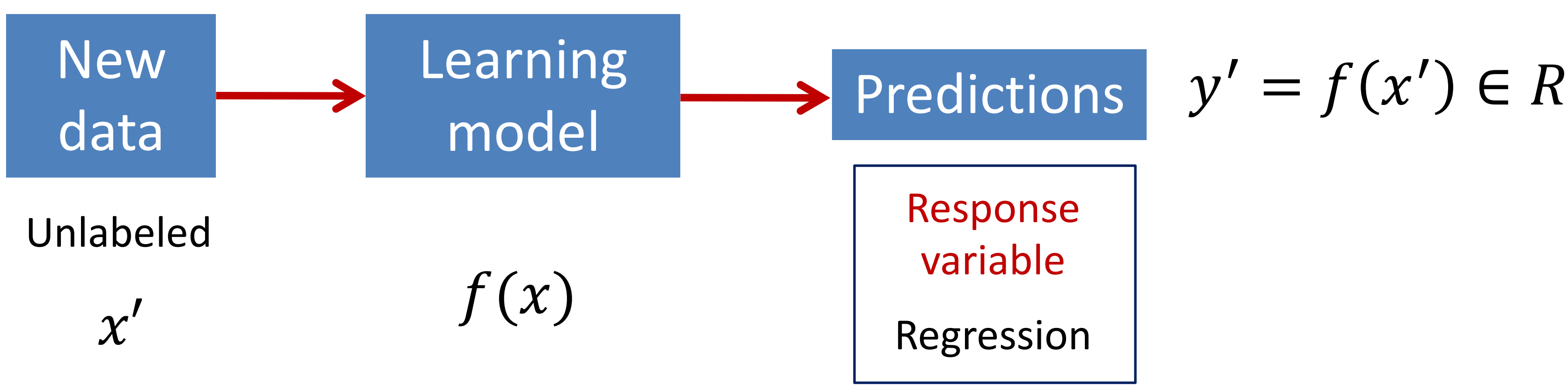


Supervised Learning: Regression

Training



Testing



Practical issues: Feature Standardization

- 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: $$x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{s_j}$$ for j = 1...d (not x_0!)
    - s_j is the standard deviation of feature j
- Must apply the same transformation to instances for both training and prediction
- Outliers can cause problems

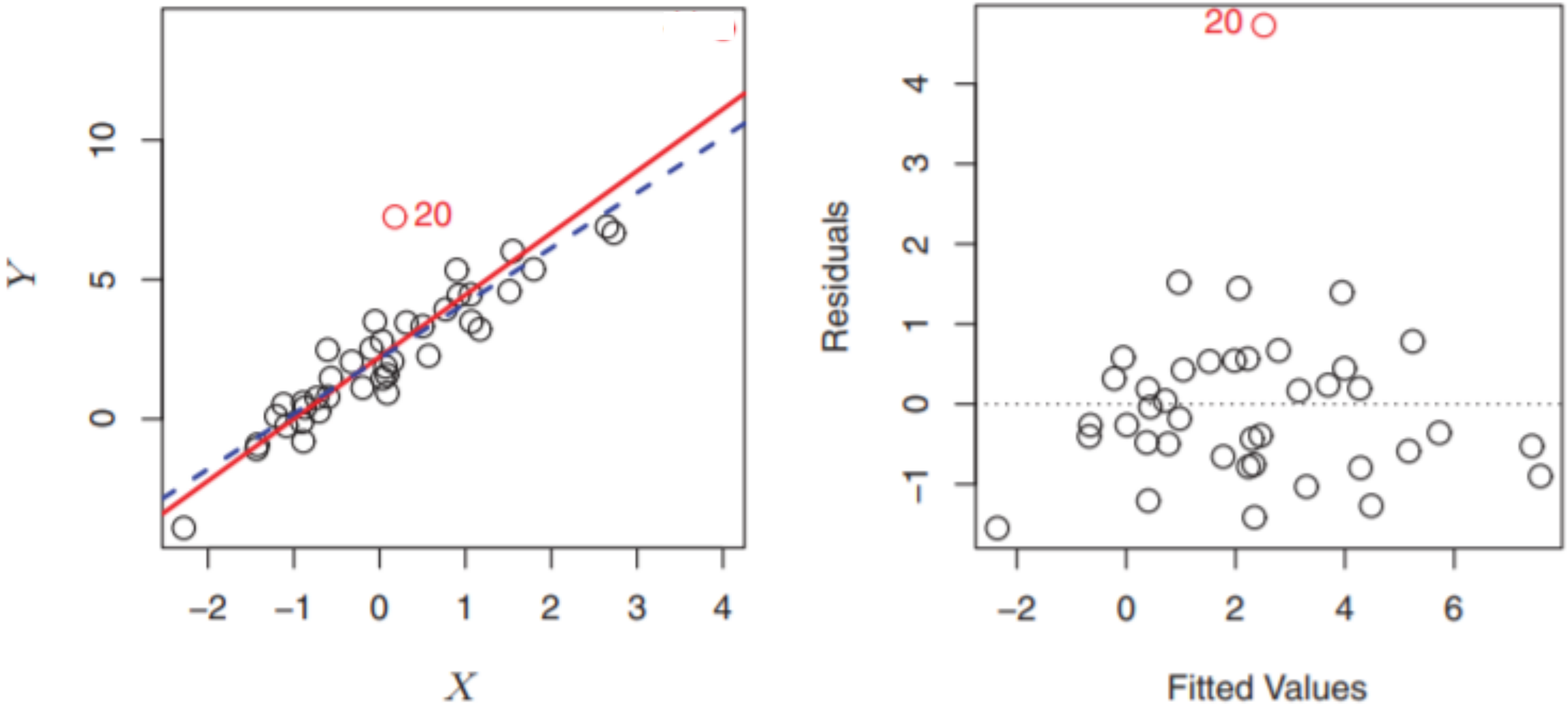
Other feature normalization

- 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
- 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?

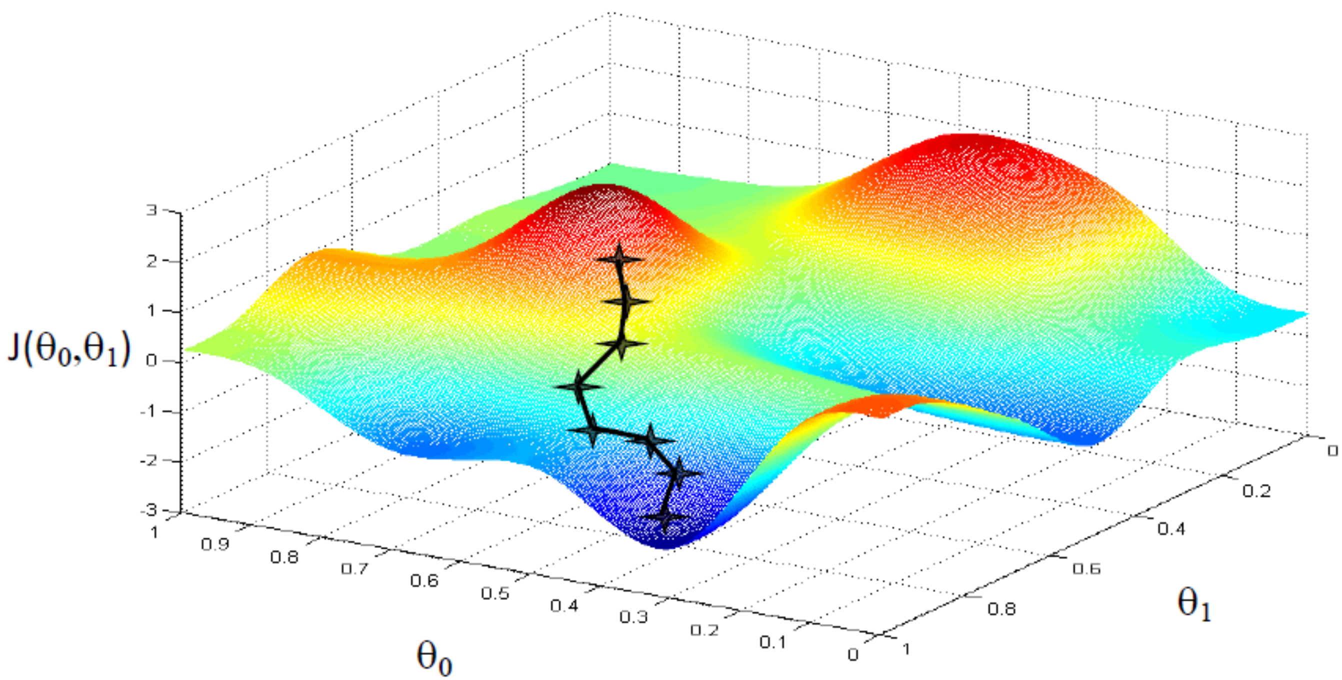
- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- $J(\theta) = \frac{1}{N} \sum (\theta^T x_i - y_i)^2$ Loss / cost
  - Strictly convex function (unique minimum)
- General method to optimize a multi-variate function
  - Practical (low asymptotic complexity)
  - Convergence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

Follow the Slope

Follow the direction of steepest descent!

How to optimize $J(\theta)$?

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



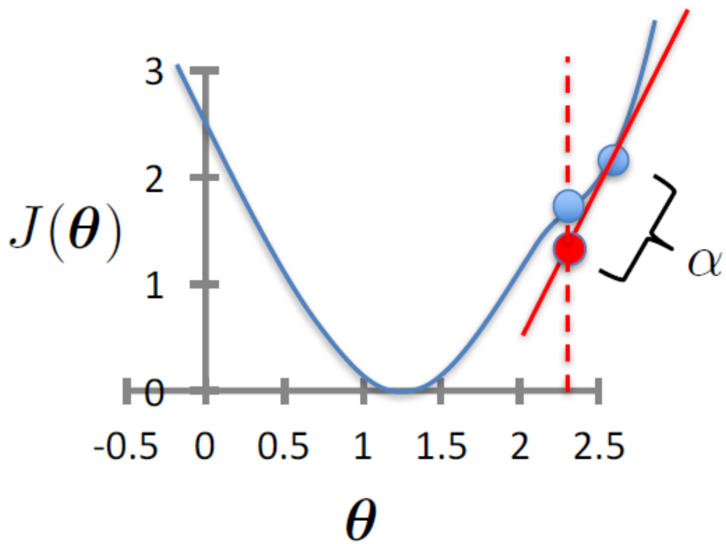
Gradient Descent

- 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



- Gradient = slope of line tangent to curve
- Function decreases faster in negative direction of gradient
- Larger learning rate => larger step

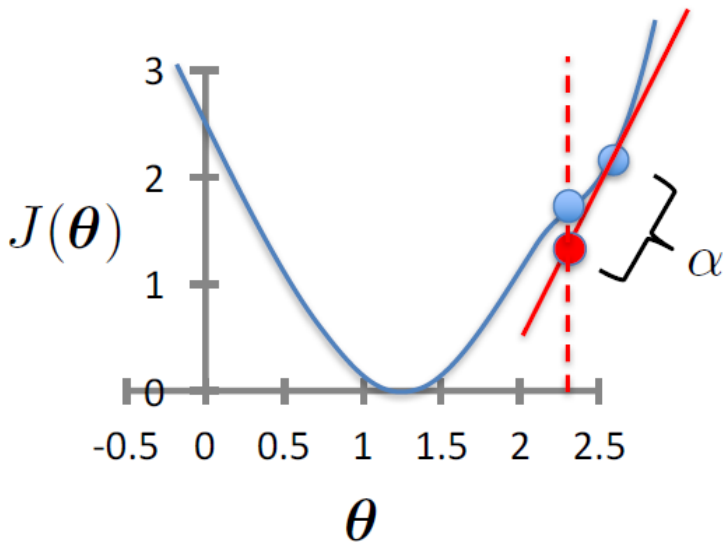
Gradient Descent

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

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

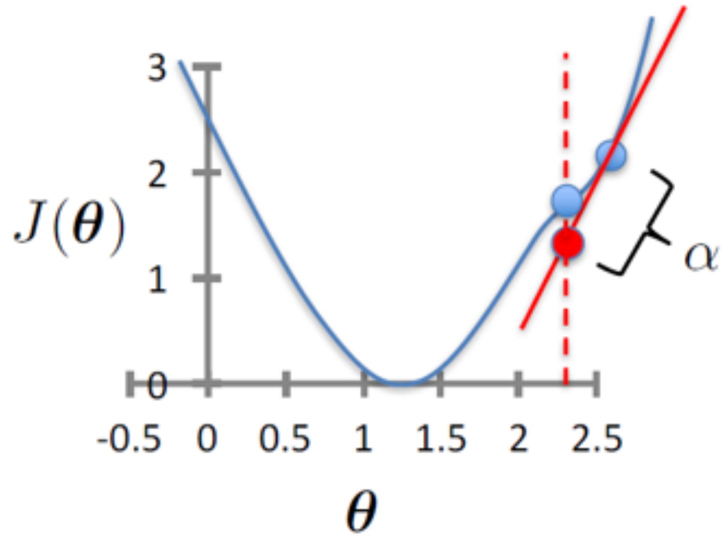
simultaneous update
for j = 0 ... d

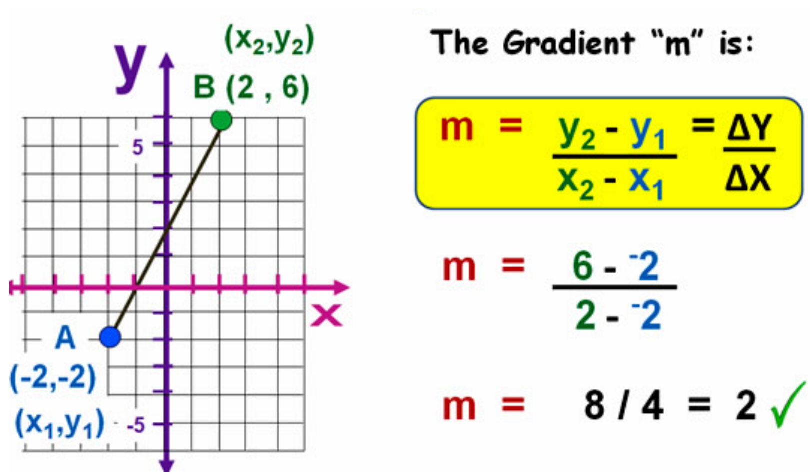
learning rate (small)
e.g., α = 0.05



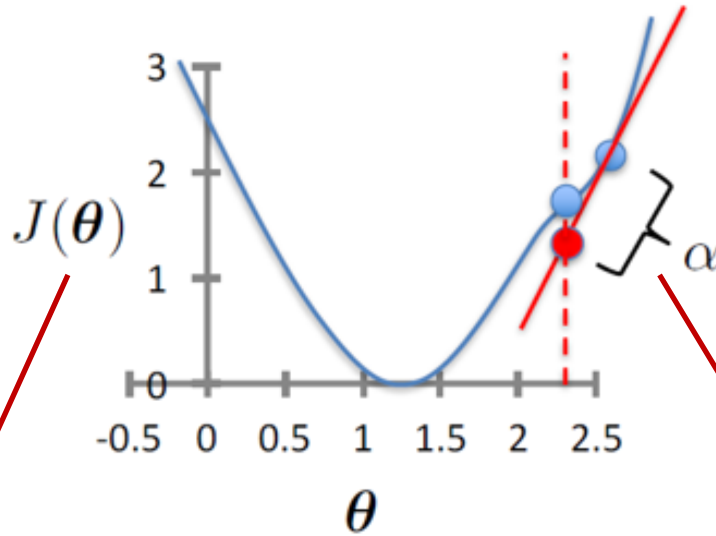
Vector update rule: $\theta \leftarrow \theta - \frac{\partial J(\theta)}{\partial \theta}$

Gradient Descent





Gradient Descent



- If θ is on the left of minimum, slope is negative
- Increase value of θ

- If θ is on the right of minimum, slope is positive
- Decrease value of θ

In both cases θ gets closer to minimum

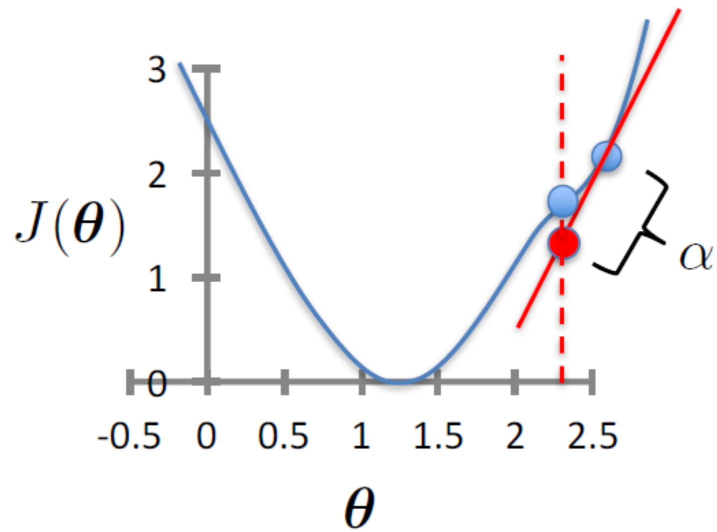
Stopping Condition

- 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?
- When the update in θ is below some threshold
- Or maximum number of iterations is reached

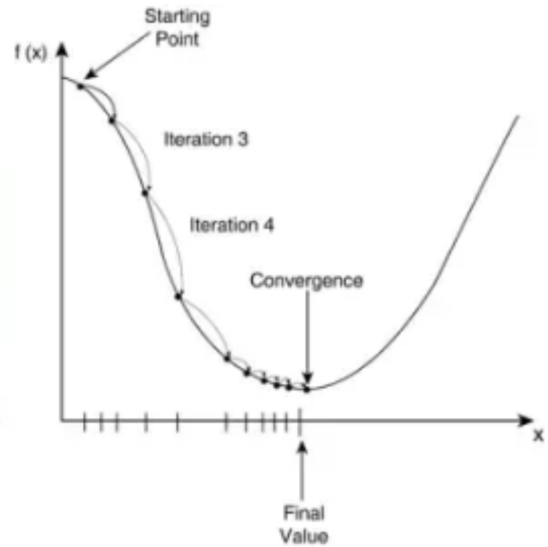
Gradient Descent

- 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

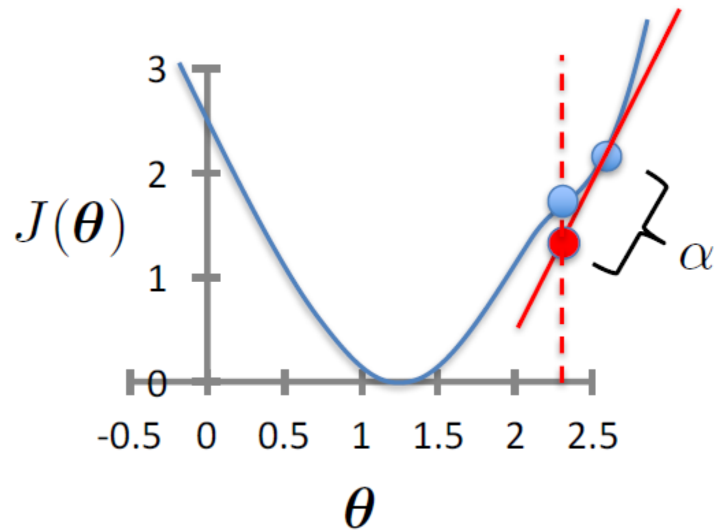
Gradient Descent

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

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

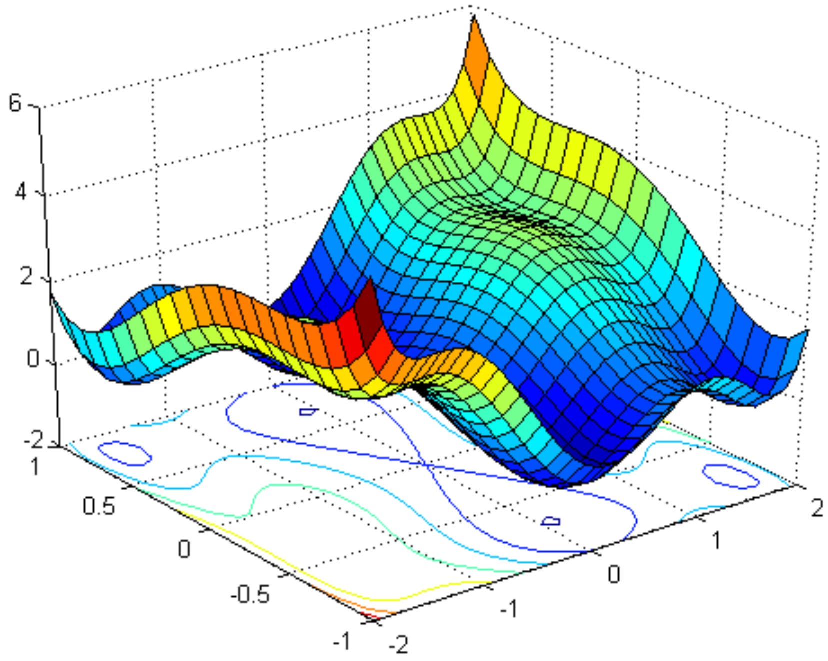
simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



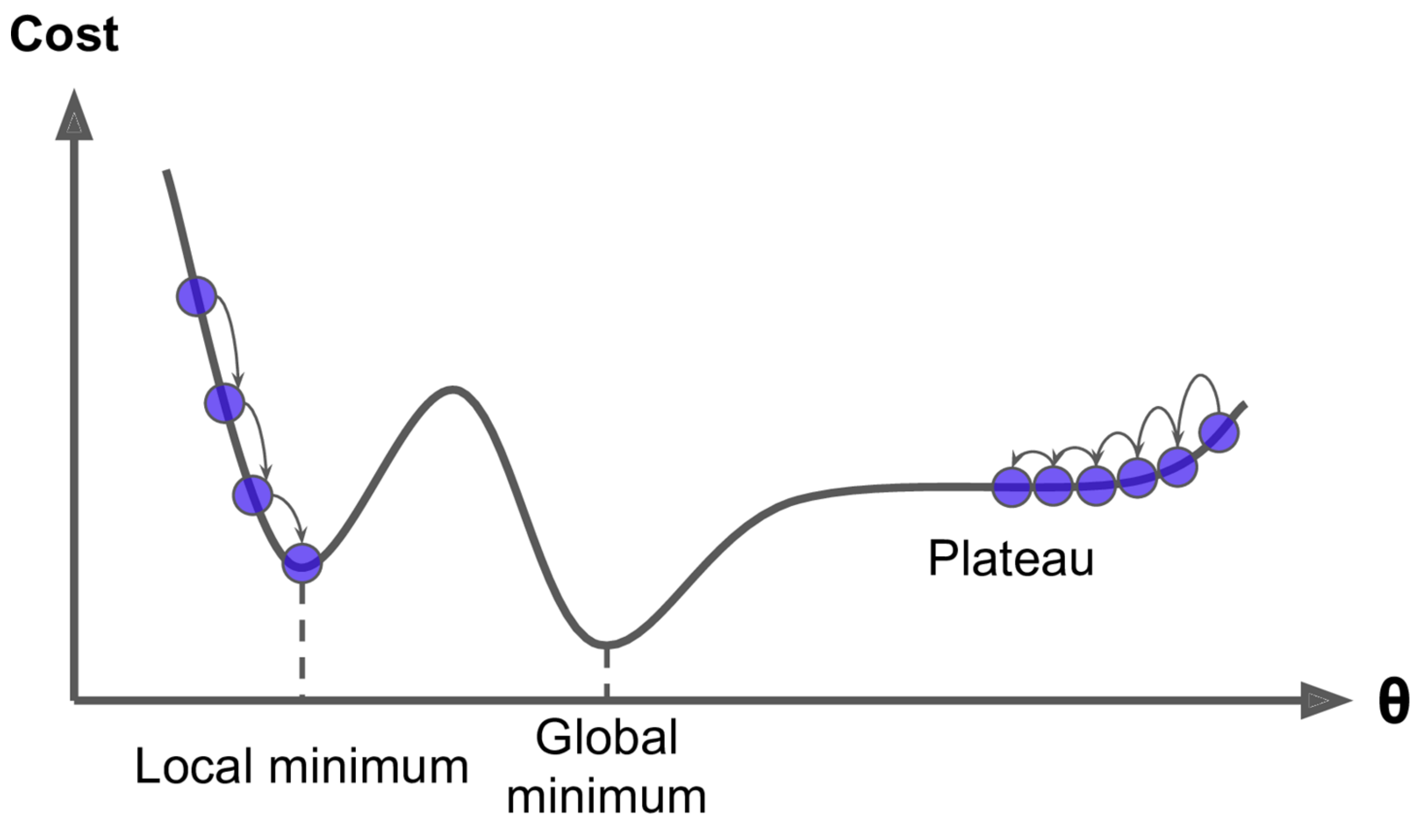
- What happens when θ reaches a local minimum?
- The slope is 0, and gradient descent converges!
- Strictly convex functions only have global minimum

Complex loss function



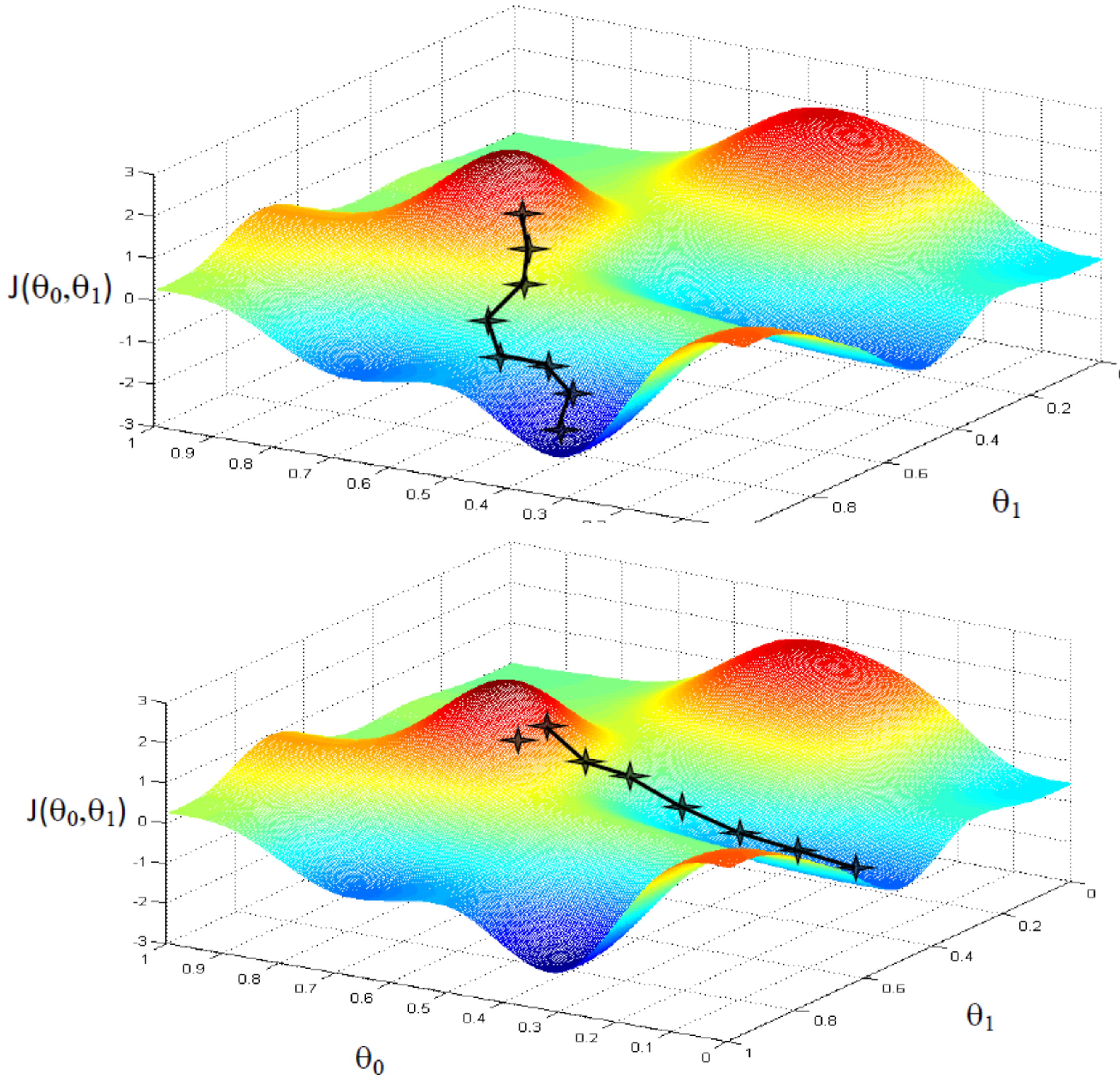
- Complex loss functions are more difficult to optimize

GD Convergence Issues



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

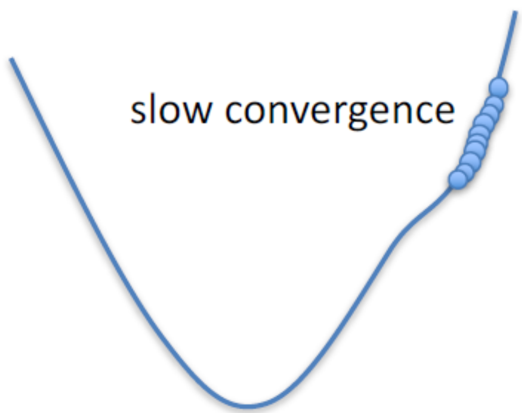
Solution: start from multiple random locations



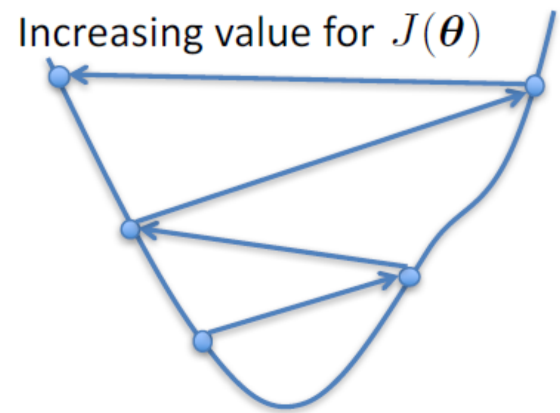
$J(\theta_0,\theta_1)$
θ_1
$J(\theta_0,\theta_1)$
θ_0
θ_1

Choosing Learning Rate

α too small



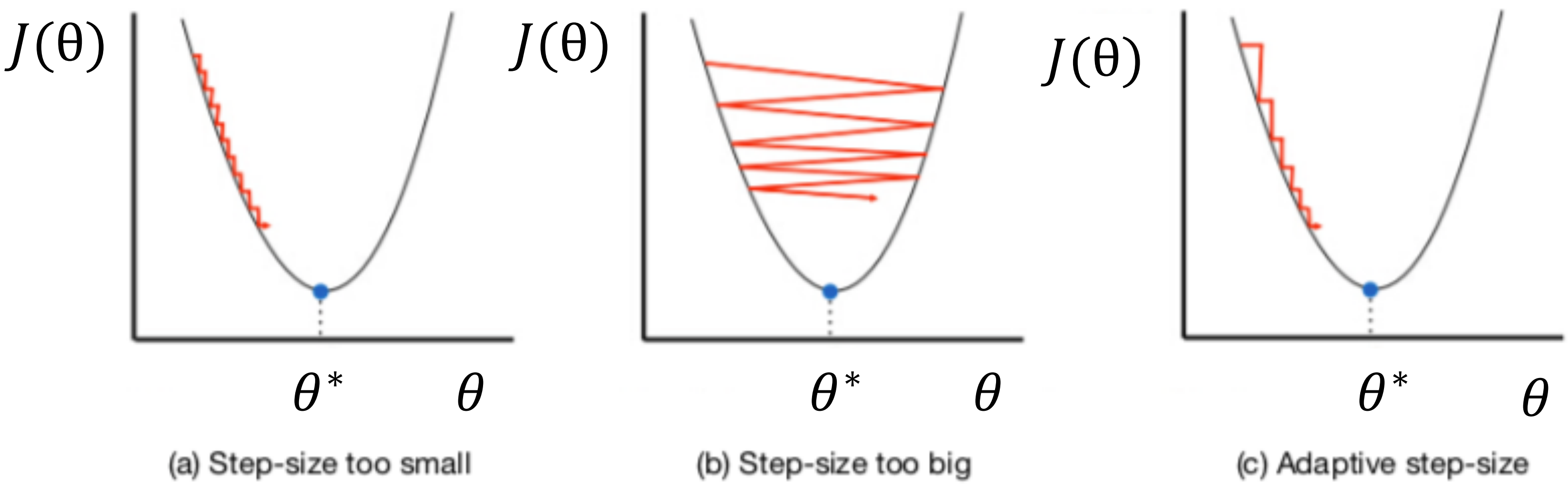
α too large



- 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

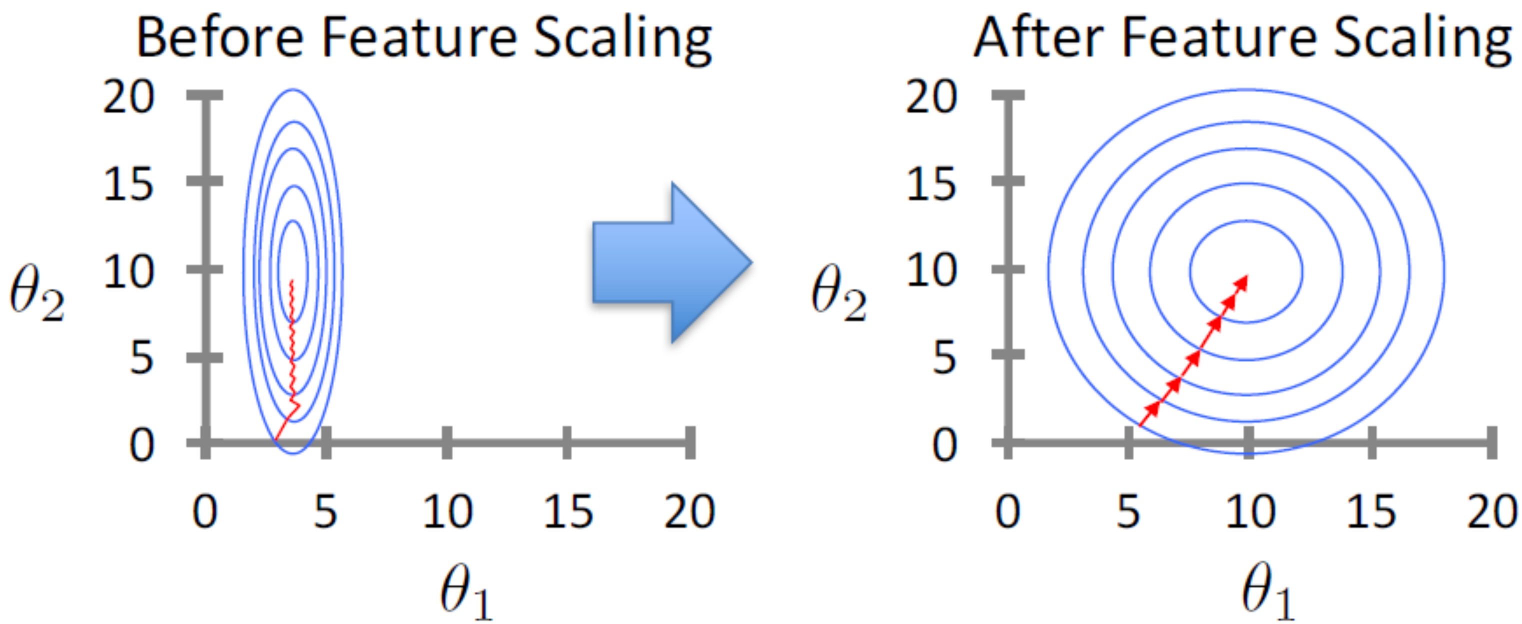
Adaptive step size



- 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



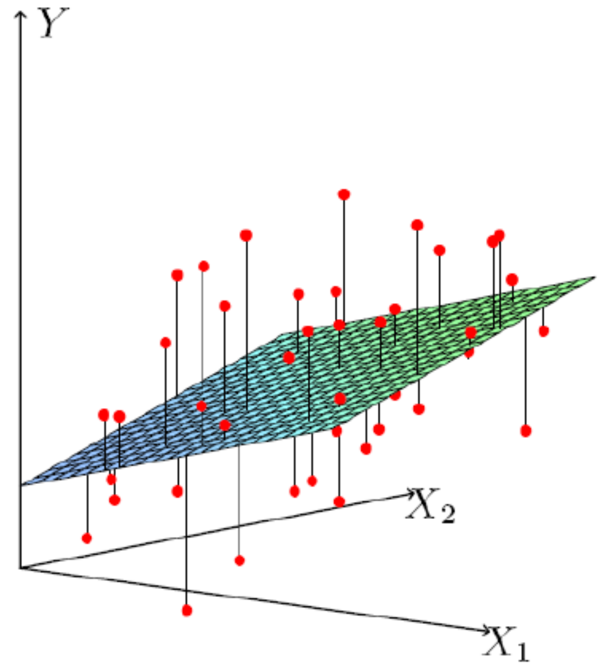
- Makes gradient descent converge *much* faster

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

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

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

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

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

GD for Linear Regression

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

$||\theta_{new} - \theta_{old}|| < \epsilon$ or
iterations == MAX_ITER

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

simultaneous update for j = 0 ... d

- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_\theta(x_i)$
  - Use this stored value in the update step loop
- 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