DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

September 18 2018

Logistics

- HW 1 is on Piazza and Gradescope
- Deadline: Friday, Sept. 28, 2018
- Office hours
  - Alina: Thu 4:30-6:00pm (ISEC 625)
  - Anand: Tue 2-3pm (ISEC 605)
- How to submit HW
  - Create a PDF and submit on Gradescope before 11:59pm the day assignment is due
  - Include link to code and ReadMe file
  - Use Jupyter notebook in R or Python

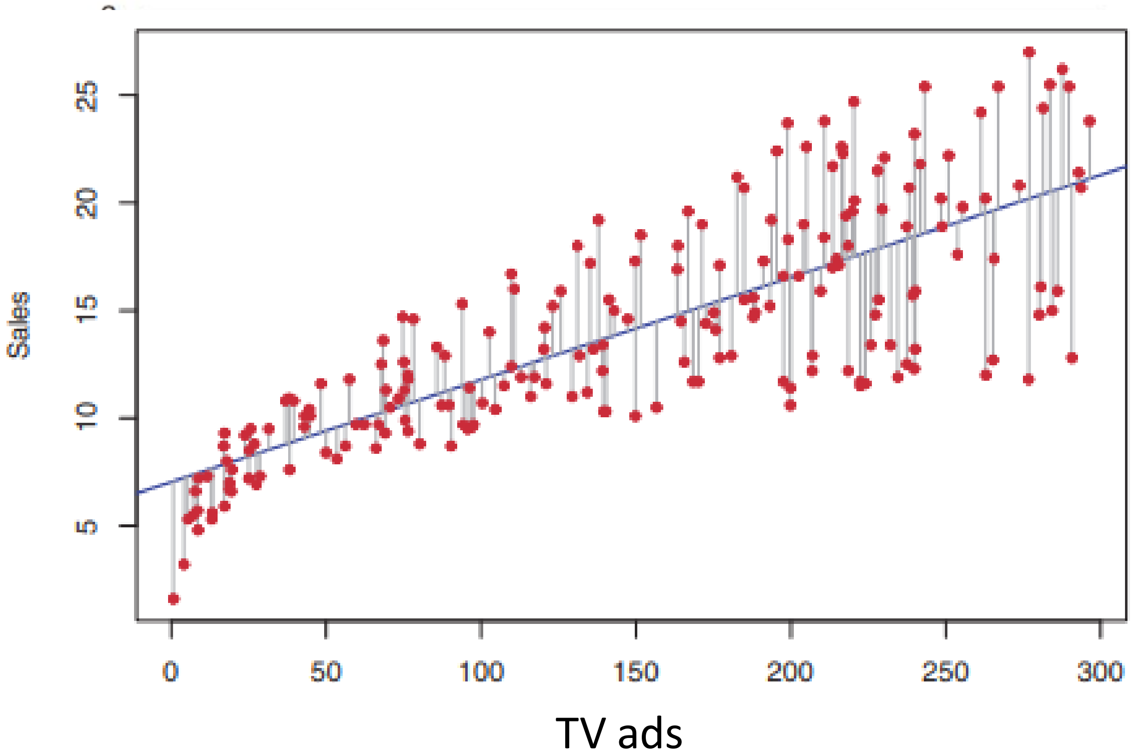
Collaboration policy

- What is allowed
  - You can discuss the homework with your colleagues
  - You can post questions on Piazza and come to office hours
  - You can search for online resources to better understand class concepts
- What is not allowed
  - Sharing your written answers with colleagues
  - Sharing your code or receiving code from colleague
  - Do not use code from the Internet!

Linear regression

Given:

- Data $\boldsymbol{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\right\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ Features
- Corresponding labels $\boldsymbol{y} = \left\{y^{(1)}, \ldots, y^{(n)}\right\}$ where $y^{(i)} \in \mathbb{R}$ Response variables



Simple linear regression

- Dataset $x^{(i)} \in R, y^{(i)} \in R, h_\theta(x) = \theta_0 + \theta_1 x$
- $J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)^2$ loss

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) = 0$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{2}{n}\sum_{i=1}^{n} x^{(i)}\left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right) = 0$$

- Solution of min loss

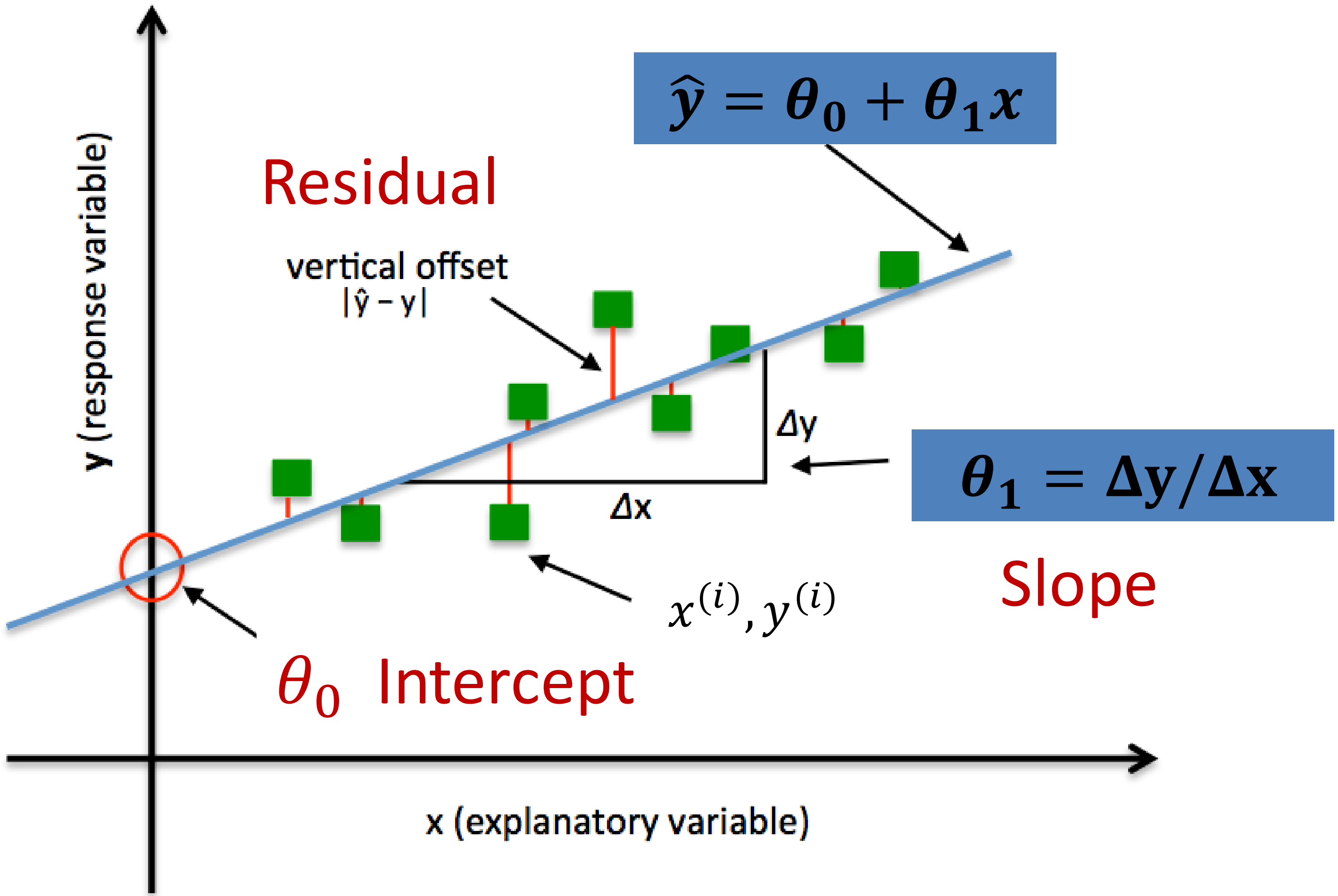
$$-\ \theta_0 = \bar{y} - \theta_1 \bar{x}$$

$$-\ \theta_1 = \frac{\sum (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum\left(x^{(i)} - \bar{x}\right)^2}$$

$$\bar{x} = \frac{\sum_{i=1}^{n} x^{(i)}}{n}$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y^{(i)}}{n}$$

Interpretation

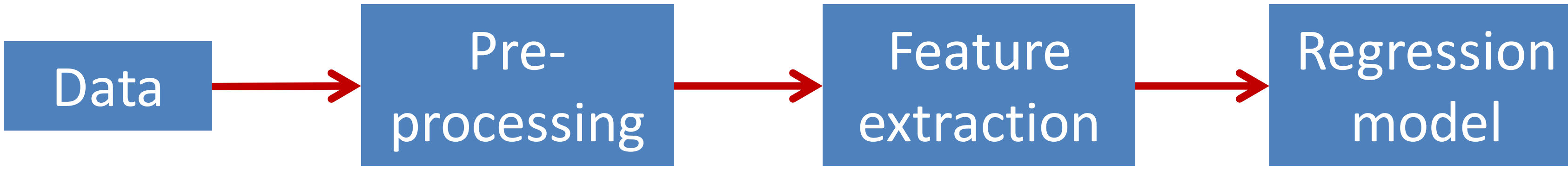


$$h_\theta(x) = \theta_0 + \theta_1 x$$

MSE$= \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

Regression Learning

Training



Labeled
$x^{(i)}, y^{(i)}$

Normalization
Standardization

Feature
Selection

Train MSE

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Testing

Price
Risk score
Test MSE

Unlabeled
x'

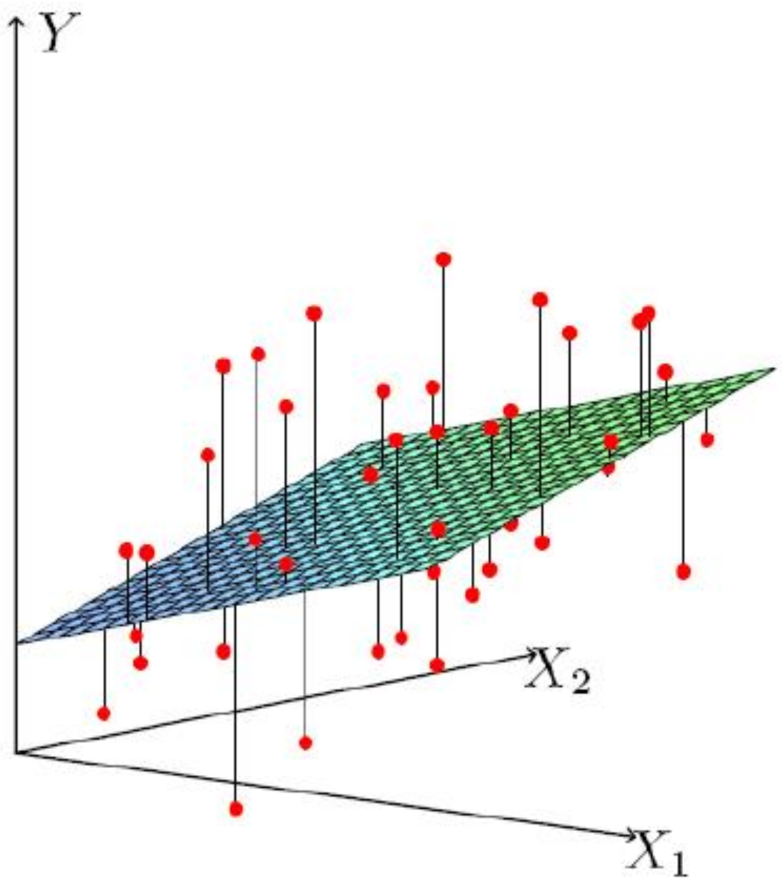
$$h_\theta(x') = \theta_0 + \theta_1 x'$$

Outline

- Multiple linear regression
  - Derivation in matrix form
- Practical issues
  - Feature scaling and normalization
  - Outliers
  - Categorical variables
- Gradient descent
  - Efficient algorithm for optimizing loss function
  - Training LR with Gradient Descent

Multiple Linear Regression

- Dataset: $x^{(i)} \in R^d, y^{(i)} \in R$



Use Vectorization

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model:

$$h(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\mathsf{T} = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as $h(\boldsymbol{x}) = \boldsymbol{\theta}^\mathsf{T} \boldsymbol{x}$

Use Vectorization

- Consider our model for n instances:

$$h\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)} = \theta^T x^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

$\mathbb{R}^{(d+1)\times 1}$ $\qquad$ $\mathbb{R}^{n\times(d+1)}$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

Loss function

- For the linear regression cost function:

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$= \frac{1}{n}\left|\left|X\theta - y\right|\right|^2$$

Euclidian Norm $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$

Matrix and vector gradients

If $y = f(x), y \in R$ scalar , $x \in R^n$ vector

$$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x_1} \quad \frac{\partial y}{\partial x_2} \quad \cdots \quad \frac{\partial y}{\partial x_n} \right]$$

If $y = f(x), y \in R^m, x \in R^n$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Jacobian matrix

Properties

- If w, x are $(d \times 1)$ vectors, $\frac{\partial w^T x}{\partial x} = w$
- If A: $(n \times d)$ x: $(d \times 1)$, $\frac{\partial Ax}{\partial x} = A$

- If A: $(d \times d)$ x: $(d \times 1)$, $\frac{\partial x^T Ax}{\partial x} = (A + A^T)x$
- If A symmetric: $\frac{\partial x^T Ax}{\partial x} = 2Ax$
- If x: $(d \times 1)$, $\frac{\partial ||x||^2}{\partial x} = 2x^T$

Min loss function

– Notice that the solution is when $\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$

$$J(\theta) = \frac{1}{n} \left\| X\theta - y \right\|^2$$

Using chain rule

$$f(\theta) = h\big(g(\theta)\big), \frac{\partial f(\theta)}{\partial \theta} = \frac{\partial h(g(\theta))}{\partial \theta} \frac{\partial g(\theta)}{\partial \theta}$$

$$h(x) = \left\| x \right\|^2, g(\theta) = X\theta - y$$

$$h'(x) = 2x^T, g'(\theta) = X$$

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{2}{n}[(X\,\theta - y)^T X] = 0 \;\Rightarrow X^T\;(X\theta - y) = 0$$

Closed Form Solution: $\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$

Closed-form solution

- Can obtain $\boldsymbol{\theta}$ by simply plugging $\boldsymbol{X}$ and $\boldsymbol{y}$ into

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

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

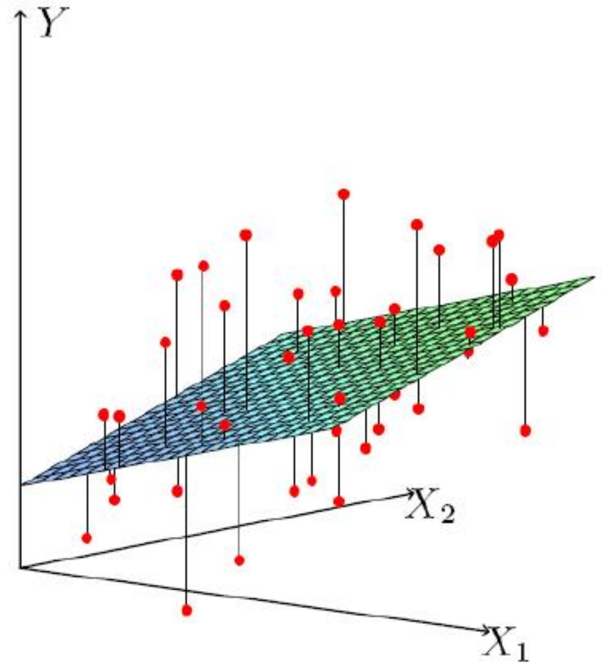
- If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $\mathrm{d} \leq \mathrm{n}$

$$AGA = A$$

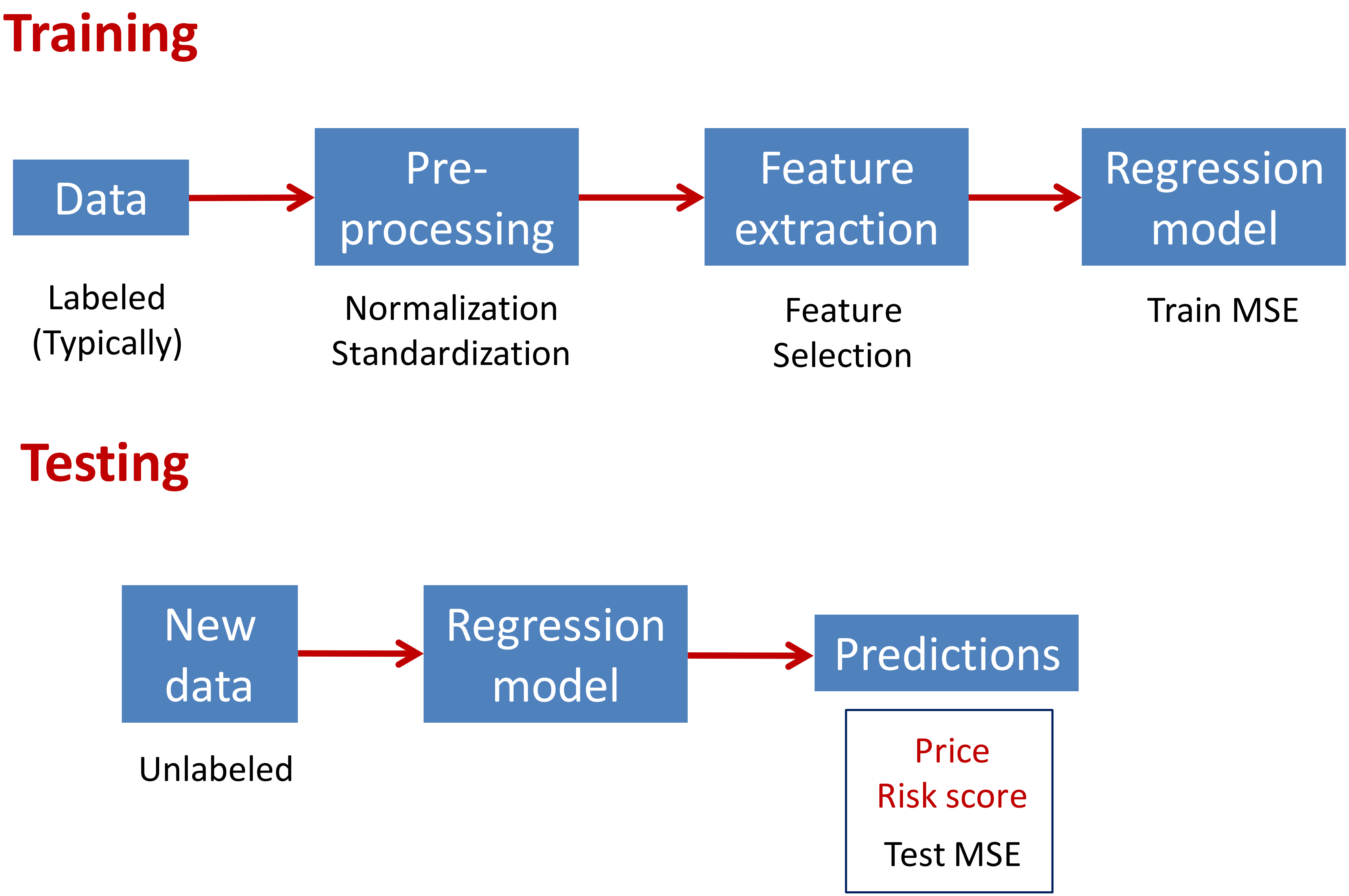
Multiple Linear Regression

- Dataset: $x^{(i)} \in R^d, y^{(i)} \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{n}\sum_{i=1}^{n}\left(\theta^T x^{(i)} - y^{(i)}\right)^2$ loss / cost

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



Regression Learning



Feature Standardization

- Rescales features to have zero mean and unit variance
  - Let μ_j be the mean of feature j: $\mu_j = \frac{1}{n} \sum_{i=1}^{n} x_j^{(i)}$
  - Replace each value with:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

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

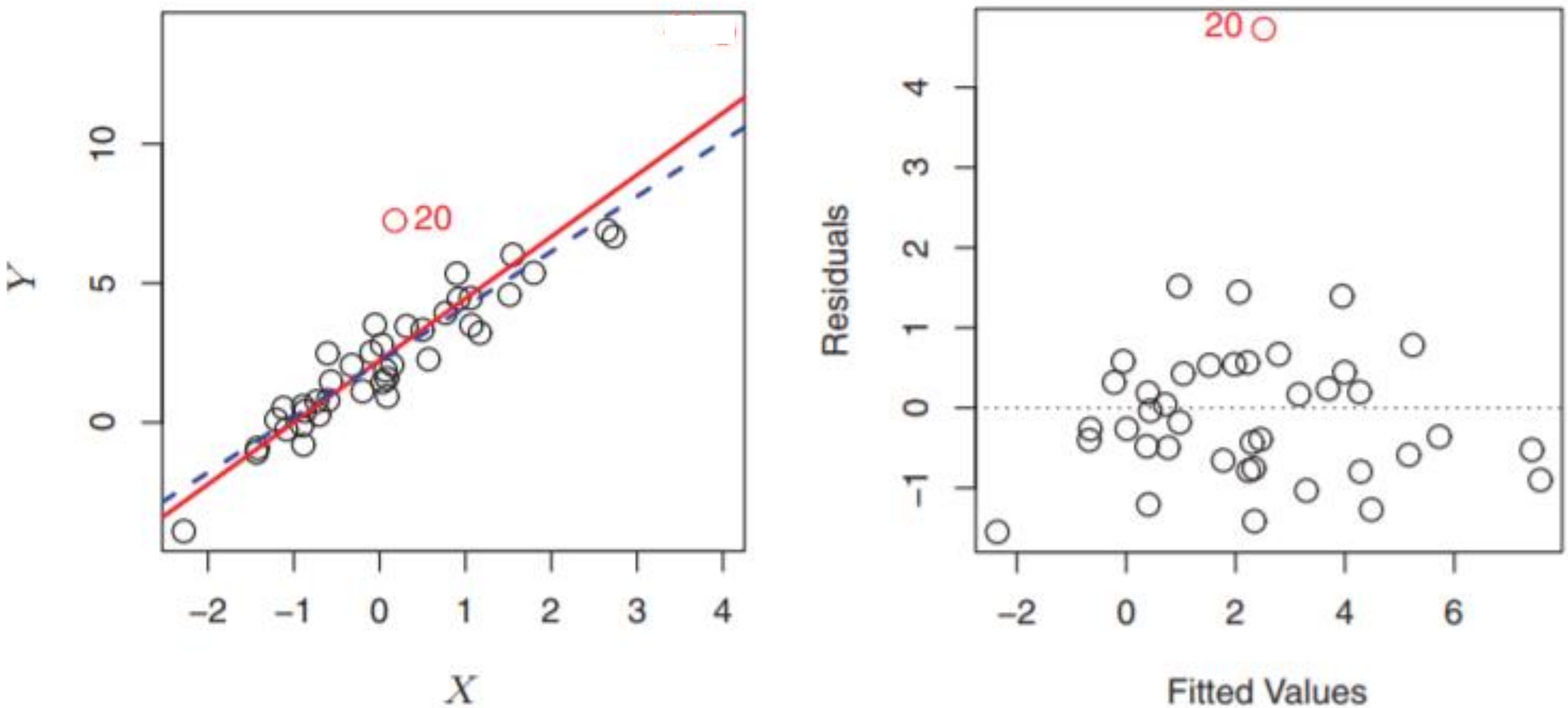
    - s_j is the standard deviation of feature j

- Must apply the same transformation to instances for both training and prediction
- Outliers can cause problems

Other feature normalization

- Re-scaling
  - $x_j^{(i)} \leftarrow \frac{x_j^{(i)} - min_j}{max_j - min_j} \in [0,1]$
  - min_j and max_j: min and max value of feature j
- Mean normalization
  - $x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{max_j - min_j}$
  - Mean 0

Outliers



- Dashed model is without outlier point
- Linear regression is not resilient to outliers!
- Outliers can be eliminated based on residual value
    - Other techniques for outlier 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)

Indicator Variables

- 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

Comparison with ANOVA

- ANOVA
  - General statistical method for comparing populations
  - Example 1: Is the income of MA and NY residents similar?
  - Example 2: Is there any difference between patients with certain treatment or no treatment?
- Linear regression
  - Learning algorithm used for predicting responses on new data
  - Example 1: Predict the income of US residents
  - Example 2: Predict survival of patients
  - Hypothesis testing for coefficient equal to zero is similar to ANOVA

Outline

- Multiple linear regression
  - Derivation in matrix form
- Practical issues
  - Feature scaling and normalization
  - Outliers
  - Categorical variables
- Gradient descent
  - Efficient algorithm for optimizing loss function
  - Training LR with Gradient Descent

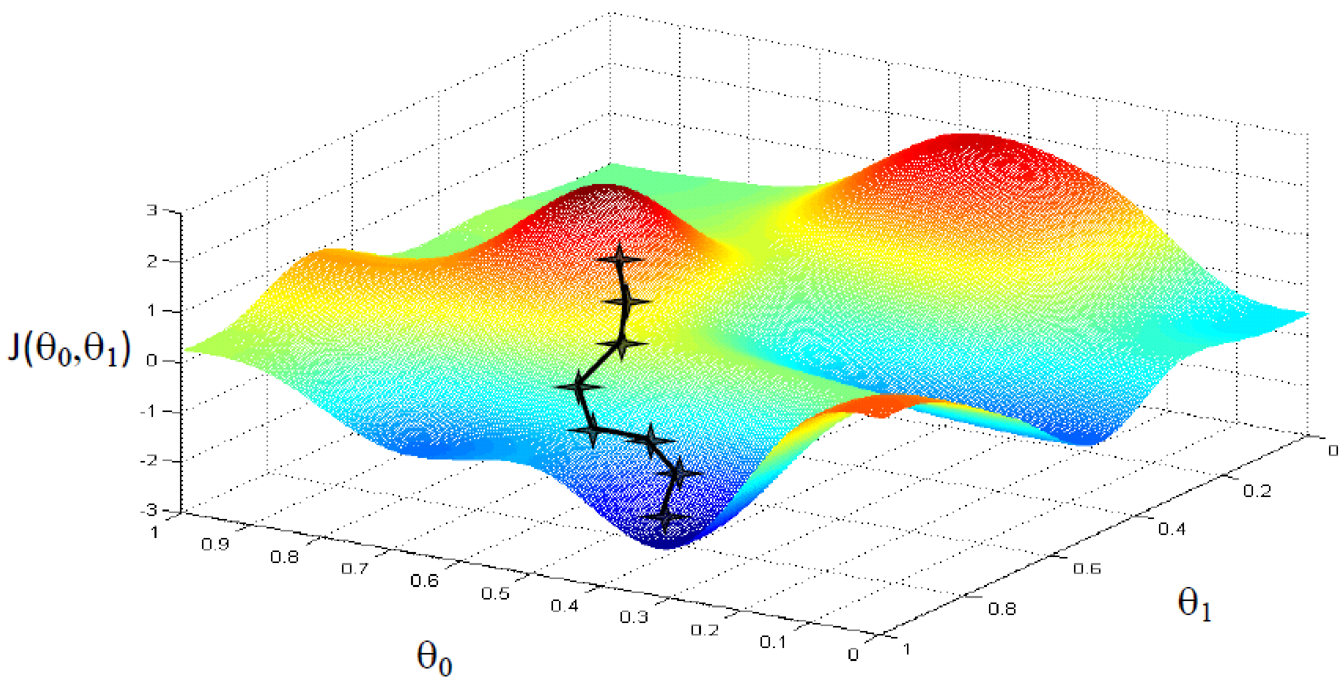
What Strategy to Use?

Follow the Slope

Follow the direction of steepest descent!

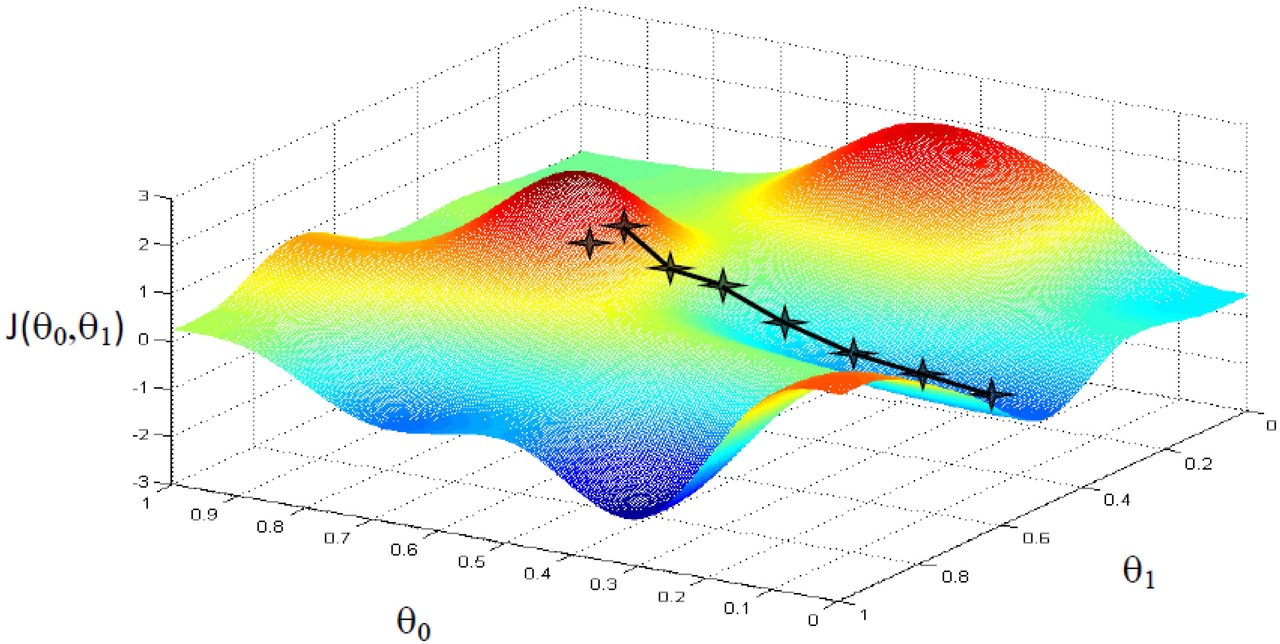
How to optimize $J(\theta)$?

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



How to optimize $J(\theta)$?

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



Different starting point

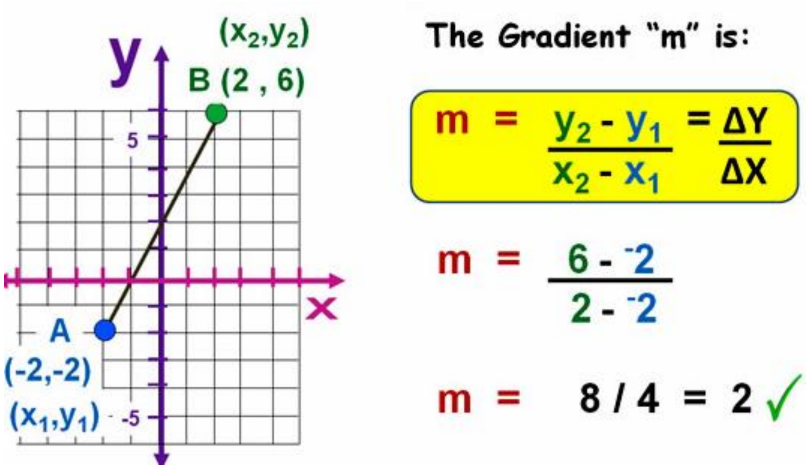
Gradient Descent

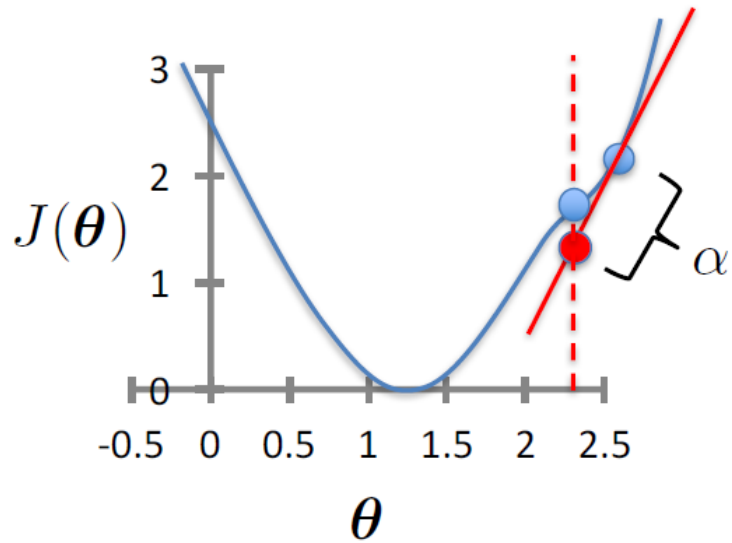
- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

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

simultaneous update for j = 0 ... d

learning rate (small)
e.g., α = 0.05





Gradient = slope of line tangent to curve at the same point

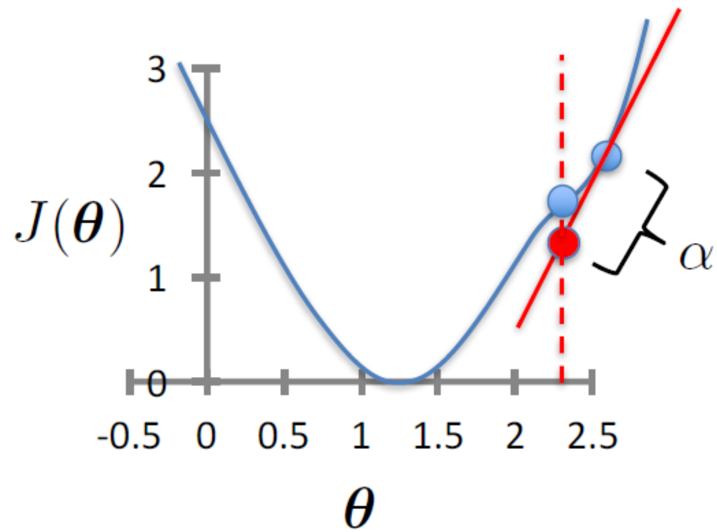
Gradient Descent

- 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!

Review

- Solution for multiple linear regression can be computed in closed form
  - Matrix inversion is computationally intense
- Gradient descent is an efficient algorithm for optimization and training LR
  - The most widely used algorithm in ML!
  - Many variants (SGD, Coordinate descent, etc.)
  - Converges if objective is convex
- In practice several techniques can help generate more robust models
  - Outlier removal
  - Feature scaling

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!