DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

January 17 2019

Logistics

- HW 1 is on Piazza and Gradescope
- Deadline: Friday, Jan. 25, 2019
- Office hours
  - Alina: Thu 4:30-6:00pm (ISEC 625)
  - Ewen: Mon 5:30-6:30pm (ISEC 605)
- How to submit HW
  - Create a PDF and submit on Gradescope before 11:59pm the day assignment is due
  - Submit zip of code in Google form
  - Should include ReadMe file on how to run code
  - Preferred: 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 directly code from the Internet!

Outline

- Terminology for supervised learning
- Multiple linear regression
  – Derivation of optimal model in matrix form
- Practical issues
  – Feature scaling and normalization
  – Outliers
  – Categorical variables
- Lab

Terminology

- Hypothesis space $H = \{f: X \rightarrow Y\}$
- Training data $\mathrm{D} = (x_{\mathrm{i}}, y_{\mathrm{i}}) \in X \times Y$
- Features: $x_i \in X$
- Labels $y_{\mathrm{i}} \in Y$
  - Classification: discrete $y_{\mathrm{i}} \in \{0,1\}$
  - Regression: $y_{\mathrm{i}} \in \mathrm{R}$
- Loss function: $L(f, D)$
  - Measures how well f fits training data
- Training algorithm: Find hypothesis $\hat{f}: X \rightarrow Y$
  - $\hat{f} = \underset{f \in H}{\operatorname{argmin}} \; L(f, D)$

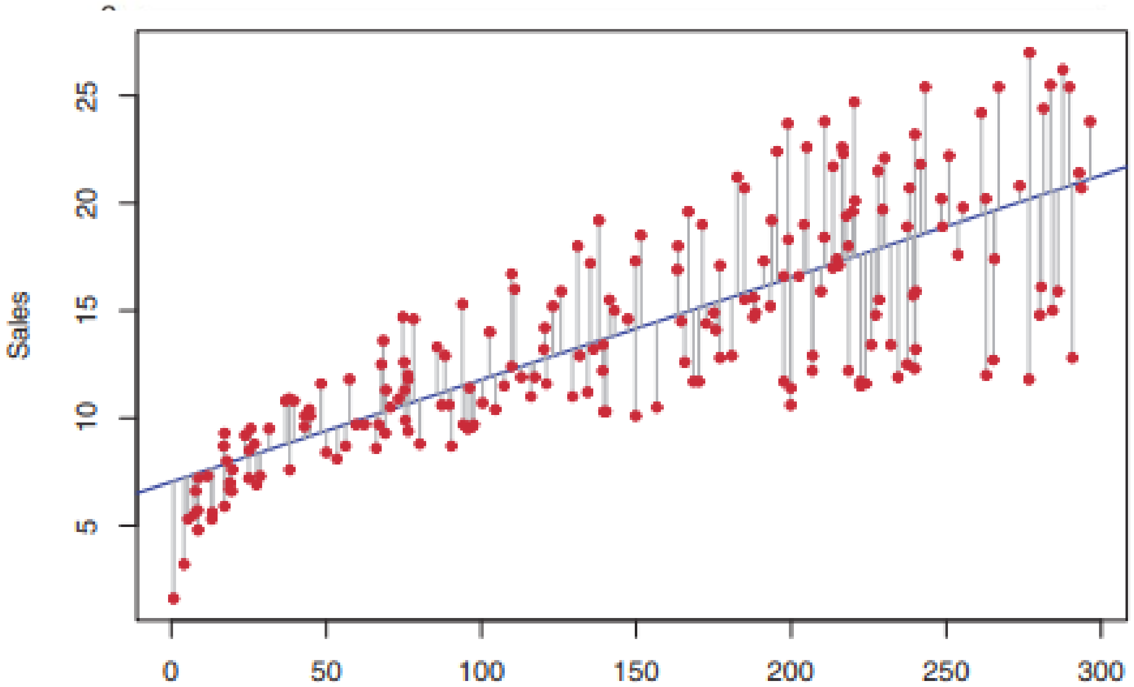
Linear regression

Given:

– Data $\boldsymbol{X} = \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)}\right\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$ Features

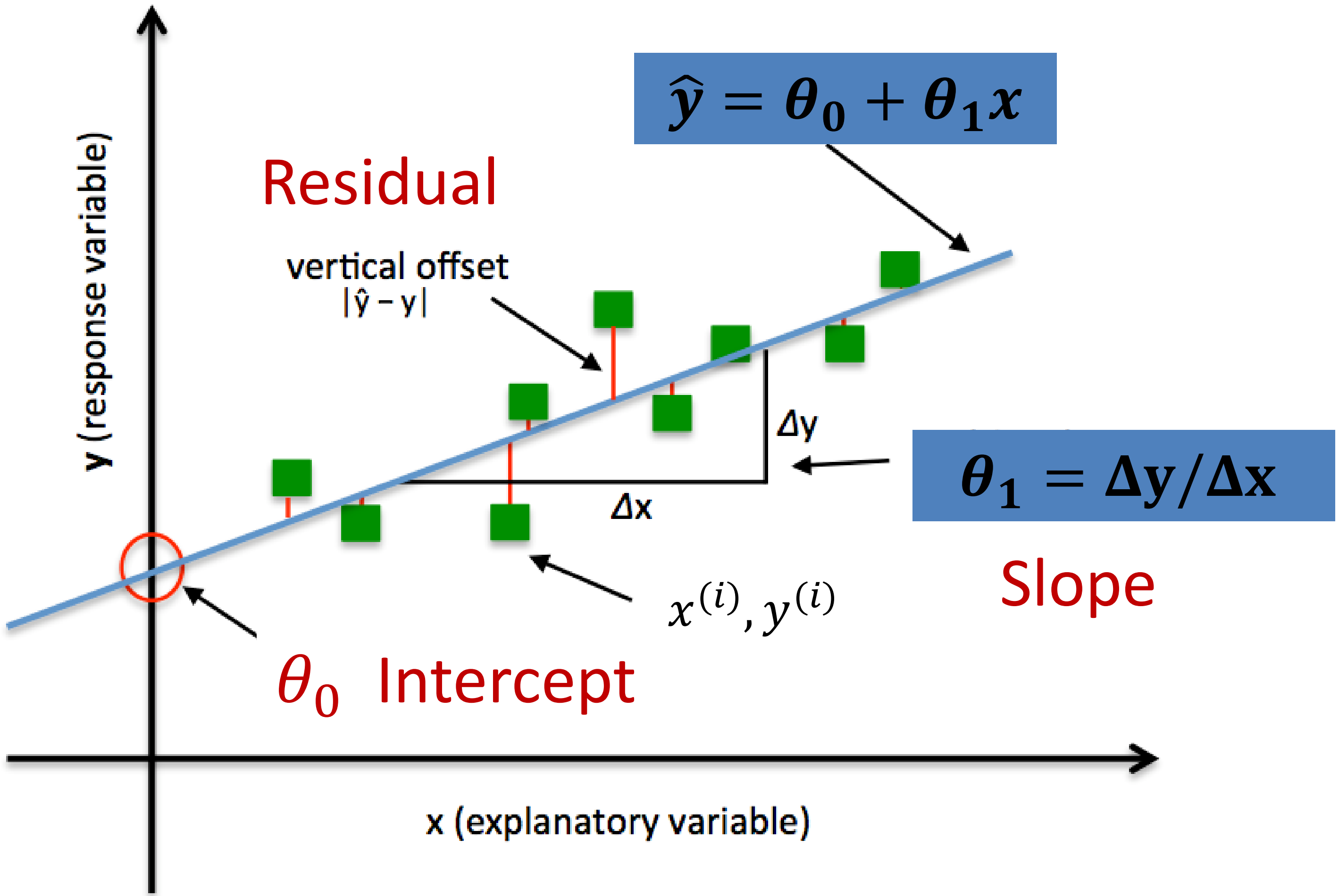
– Corresponding labels $\boldsymbol{y} = \left\{y^{(1)}, \ldots, y^{(n)}\right\}$ where $y^{(i)} \in \mathbb{R}$

Response variables



Simple Linear Regression: 1 predictor

Interpretation



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

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

Regression Learning

Training

Data → Pre-processing → Feature extraction → Regression model

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

Normalization
Standardization

Feature
Selection

Min MSE

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

Testing

New data → Regression model → Predictions

Unlabeled
x'

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

Price
Risk score
Test MSE

Simple Linear Regression

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

  MSE / Loss

- Solution of min loss

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

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

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

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

Variance of x (denominator)

Co-variance of x and y (numerator)

How Well Does the Model Fit?

- Correlation between feature and response
  - Pearson's correlation coefficient

Co-variance of x and y

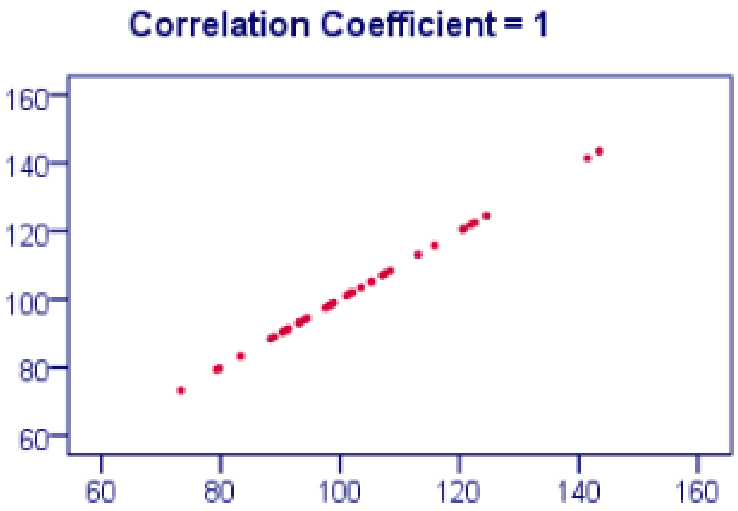
$$\mathrm{Cor}(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}},$$

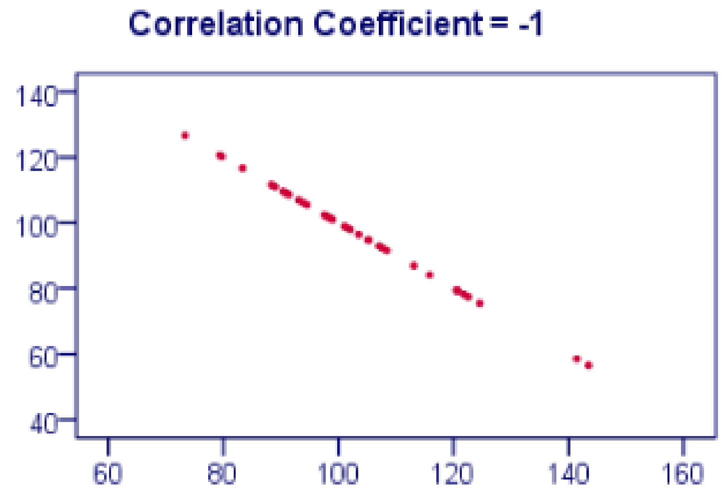
Standard deviation x

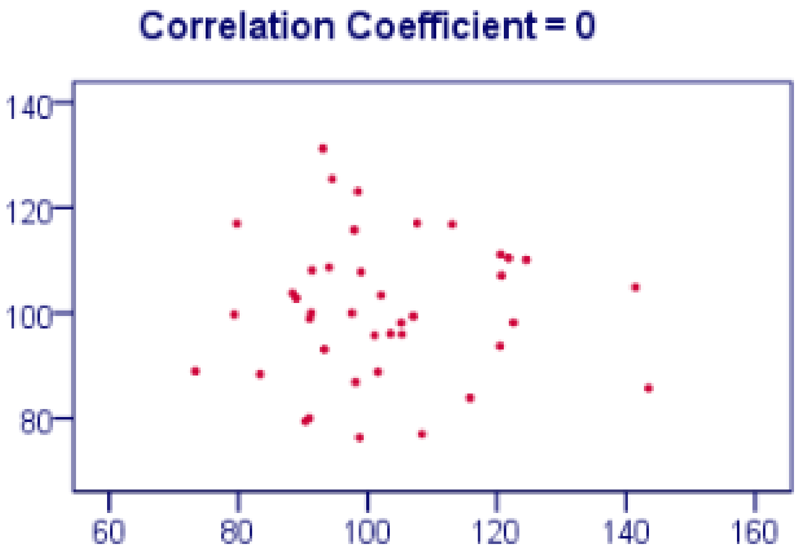
Standard deviation y

- Measures linear dependence between x and y
- Positive coefficient implies positive correlation
  - The closer to 1 the coefficient is, the stronger the correlation
- Negative coefficient implies negative correlation
  - The closer to -1 the coefficient is, the stronger the correlation

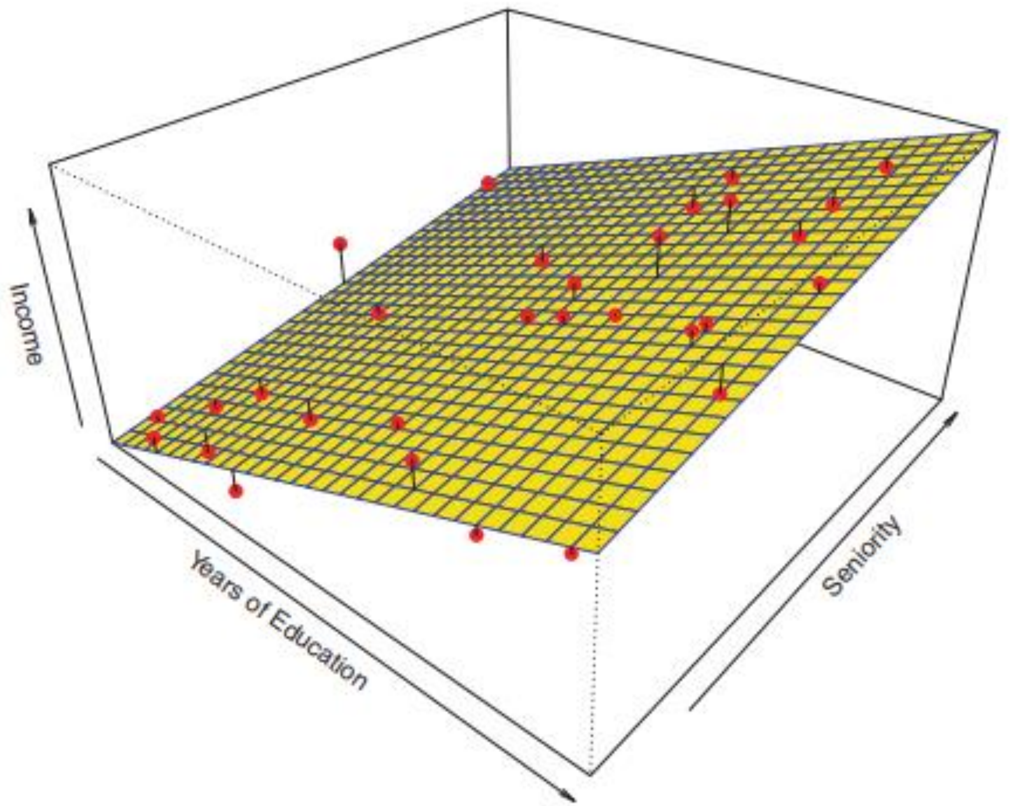
Correlation Coefficient





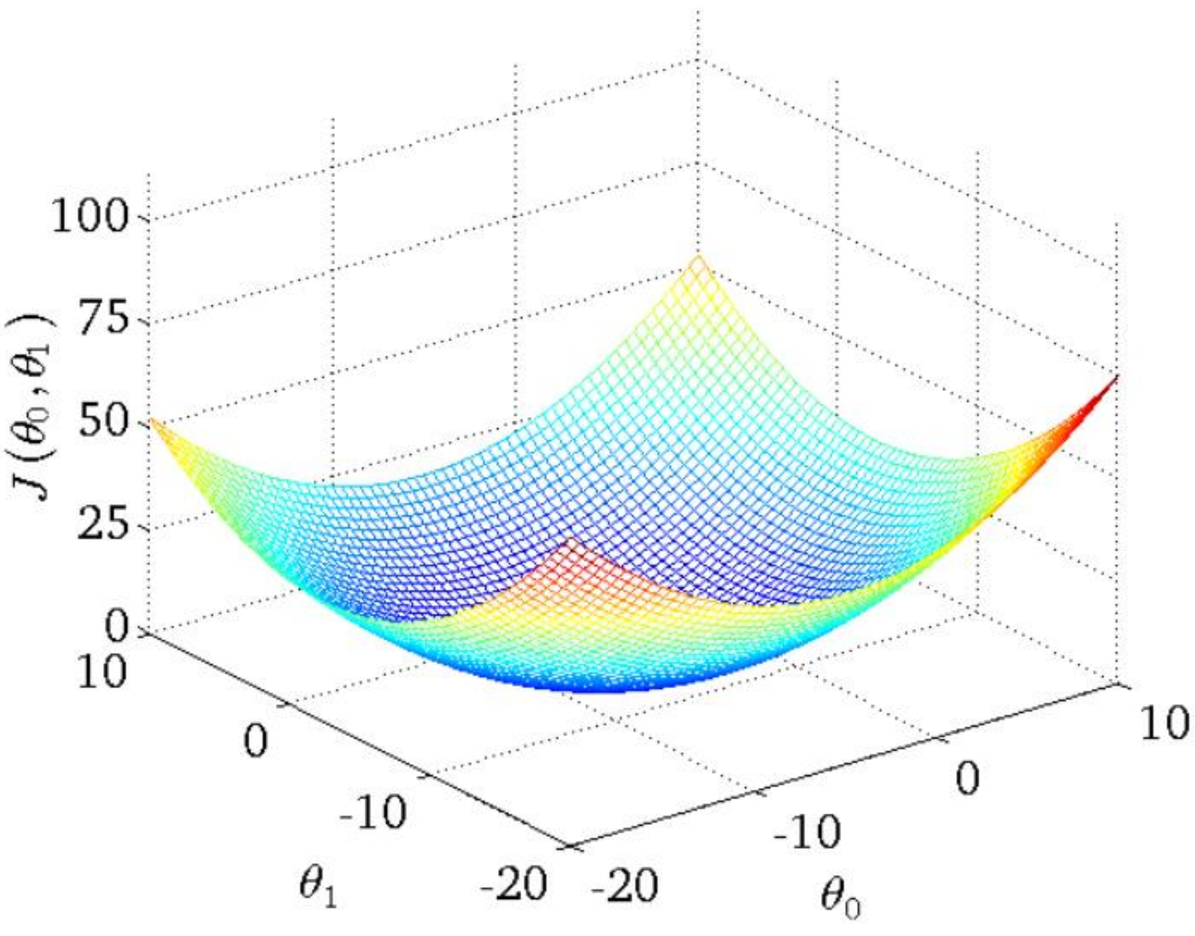


Multiple Linear Regression



- Linear Regression with 2 predictors
- Dataset: $x^{(i)} \in R^d, y^{(i)} \in R$

MSE function



Convex function implies unique minimum

Vector Norms

Vector norms: A norm of a vector ||x|| is informally a measure of the "length" of the vector.

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

- Common norms: L_1, L_2 (Euclidean)

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \qquad \|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- $L_{infinity}$

$$\|x\|_\infty = \max_i |x_i|$$

Vector products

We will use lower case letters for vectors
The elements are referred by x_i.

- Vector dot (inner) product:

$$x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ x_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i.$$

- Vector outer product:

$$xy^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$$

Hypothesis Multiple LR

- Linear Model

- Consider our model:

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

- Let

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

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

Vector inner product

Training data

Feature 1 Feature d

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

Training example i

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

- Total number of training example: *n*
- Dimension of training data point (number of features): *d*

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

Model parameter

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

Training data

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

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

Model prediction vector $\hat{y}$

Loss function MSE

- For the linear regression cost function:

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

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{n}||\hat{y} - y||^2$$

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

Let:

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

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

Matrix and vector gradients

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

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

Vector gradient (row vector)

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 (Matrix gradient)

Properties

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

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

Min loss function

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

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

Using chain rule

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

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

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

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

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

Vectorization

- Two options for operations on training data
  - Matrix operations
  - For loops to update individual entries
- Most software packages are highly optimized for matrix operations
  - Python numpy
  - Preferred method!

- Matrix operations are much faster than loops!

Closed-form solution

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

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

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

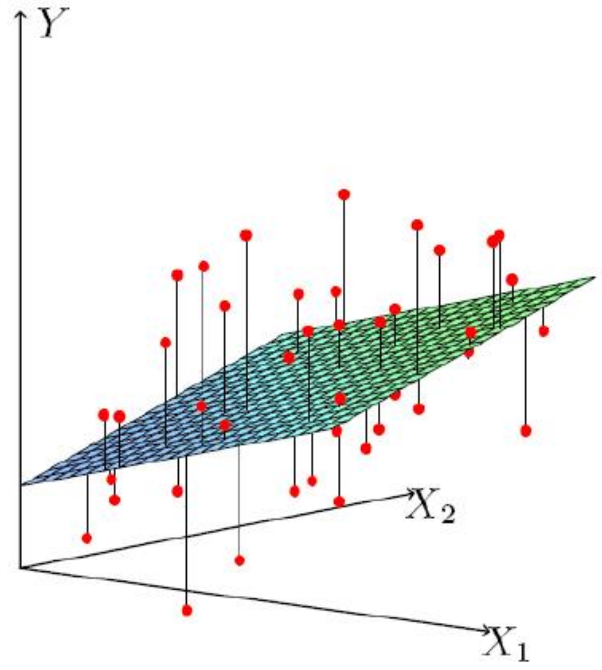
- If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $\mathrm{d} \le \mathrm{n}$

$$AGA = A$$

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

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



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
- Mean 0 and Standard Deviation 1

Other feature normalization

- Min-Max rescaling
  - $x_j^{(i)} \leftarrow \frac{x_j^{(i)} - min_j}{max_j - min_j} \in [0,1]$
  - 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

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

Review

- Solution for multiple linear regression can be computed in closed form
  - Matrix inversion is computationally intense
  - We will discuss an efficient training algorithms (gradient descent)
- 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!