DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

February 2 2021

Today’s Outline

- Announcements
  - First tutorial (recording and notebook) available on Canvas
  - Panda tutorial by Omkar on Wed at 4pm
- Multiple Linear Regression
  - Vector and matrix gradients
  - Closed-form solution derivation
- Lab in Python
  - Simple LR
  - Multiple LR
- Practical issues when training LR models

Supervised Learning: Regression

Training

Data → Pre-processing → Feature extraction → Learning model

Labeled

$x_i, y_i \in R$

Normalization

Feature Selection

Regression

$f(x)$

Testing

New data → Learning model → Predictions

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

Unlabeled

x'

$f(x)$

Response variable

Regression

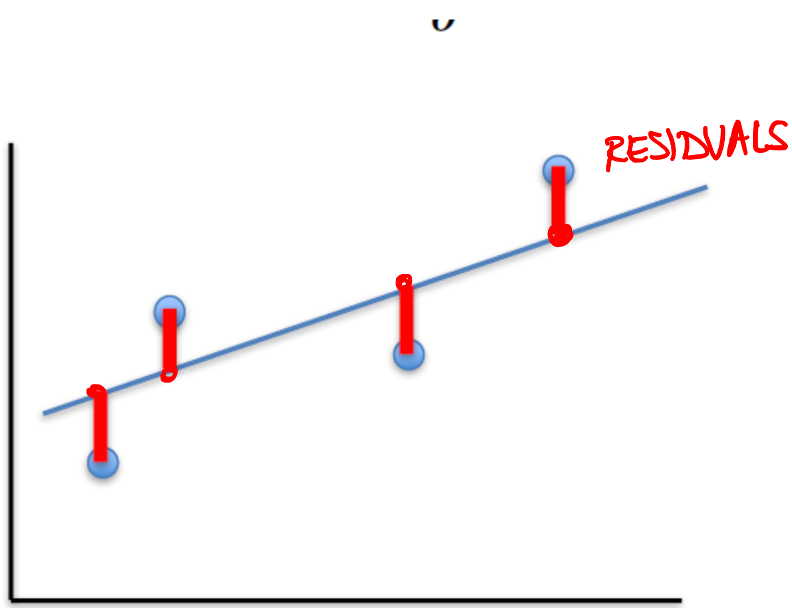
Least-Squares Linear Regression

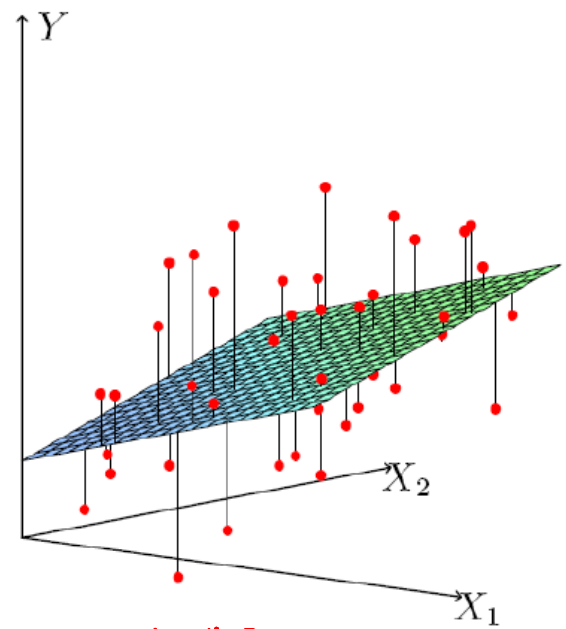
$$\text{MSE}: \quad J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \big[\underbrace{h_\theta(x_i) - y_i}_{\text{RESIDUAL}}\big]^2$$

x_i, y_i TRAINING DATA

x_i → FEATURES

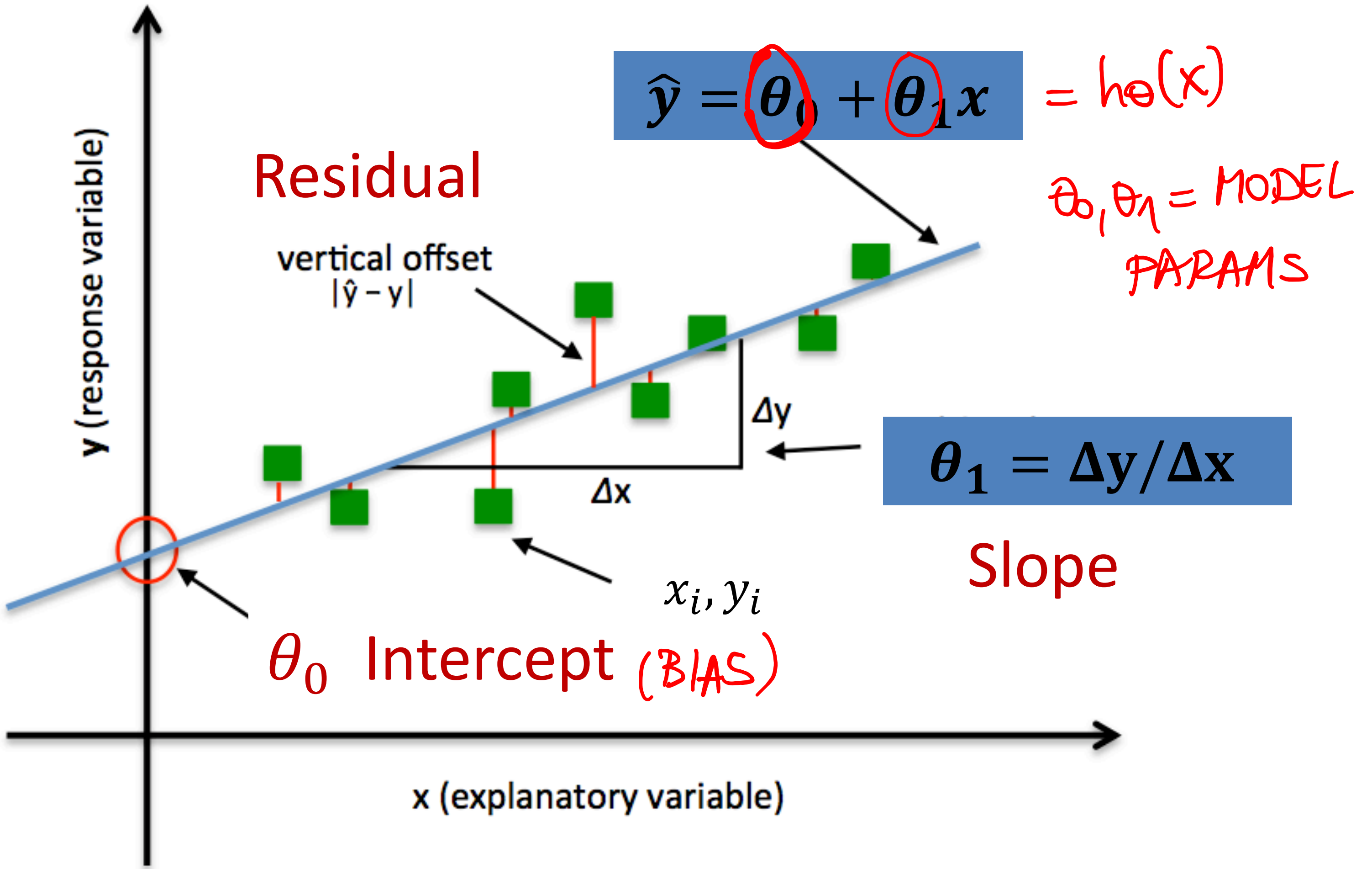
y_i → RESPONSE VARIABLE





OLS : ORDINARY LEAST SQUARES

Interpretation



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

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

Solution for simple linear regression

TRAINING $i=\overline{1,N}$

- Dataset $x_i \in R, y_i \in R, h_\theta(x) = \theta_0 + \theta_1 x$

Min
- $J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(\theta_0 + \theta_1 x_i - y_i)^2$ MSE / Loss

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

- Solution of min loss

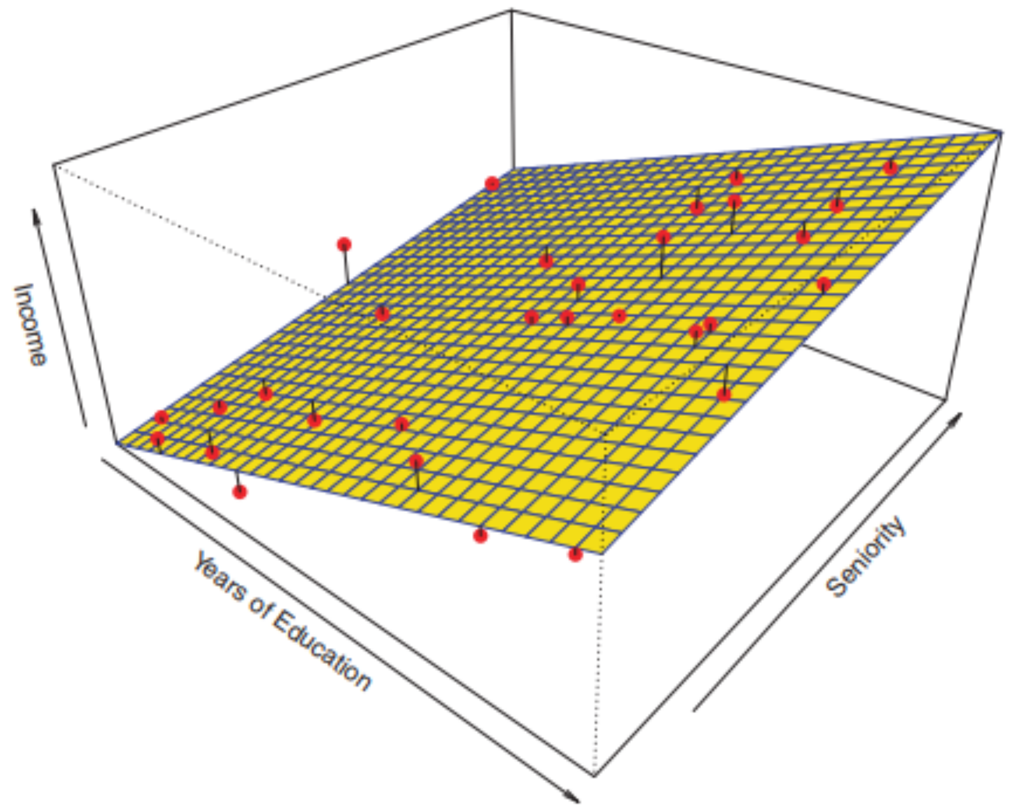
$$- \theta_0 = \bar{y} - \theta_1 \bar{x}$$

$$- \theta_1 = \frac{\frac{1}{N-1}\sum (x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{N-1}\underbrace{\sum (x_i - \bar{x})^2}_{Var(x)}} = \frac{Cov(x,y)}{Var(x)}$$

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

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

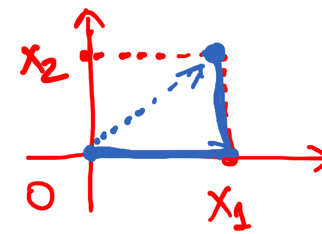
Multiple Linear Regression



- Linear Regression with at least 2 predictors $d \geqslant 2$
- Dataset: $x_i \in R^d, y_i \in R$

Vector Norms

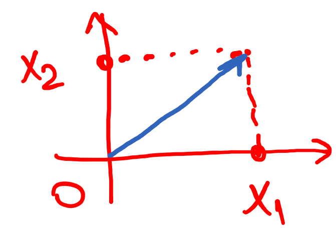
Vector norms: A norm of a vector ||x|| is informally a measure of the "length" of the vector.



L1

L_p

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



L2

– Common norms: L_1, L_2 (Euclidean)

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

EUCLIDEAN

$\|x\|_2 = \sqrt{x_1^2 + x_2^2}$

– L_∞

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

Vector products

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

- Vector dot (inner) product:

$$x^T y \in \mathbb{R} = \begin{array}{c} \text{ROW} \\ \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \\ \text{SIZE: } 1 \times n \end{array} \begin{array}{c} \text{COLUMN} \\ \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \\ \text{SIZE: } n \times 1 \end{array} = \sum_{i=1}^{n} x_i y_i . \in \mathbb{R}$$

- Vector outer product:

$$xy^T \in \mathbb{R}^{m \times n} = \begin{array}{c} \text{COLUMN} \\ \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \\ \text{SIZE: } m \times 1 \end{array} \begin{array}{c} \text{ROW} \\ \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} \\ \text{SIZE: } 1 \times n \end{array} = \begin{array}{c} \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix} \\ \text{SIZE } m \times n \end{array}$$

Hypothesis Multiple LR

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

MULTIPLE LR: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d$] (*)

$$x = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix}$$

$1 \rightarrow$ BIAS

MODEL PARAMETER

$d = \#$ features

$d+1$

(*) $h_\theta(x) = \theta^T x$

DOT PRODUCT OF θ and x

Training data

N training examples

$x_i = (x_{i1}, x_{i2}, \dots, x_{id})$ d features

$y_1, \dots, y_N$ RESPONSE VAR

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1j} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2j} & \dots & x_{2d} \\ \vdots & \vdots & & & & \\ 1 & x_{i1} & \dots & x_{ij} & \dots & x_{id} \\ \vdots & \vdots & & & & \\ 1 & x_{N1} & \dots & x_{Nj} & \dots & x_{Nd} \end{bmatrix}$$

TRAINING EXAMPLE i

FEATURE j

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

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

size $N \times 1$

Use Vectorization

$\hat{y}_i = \mathrm{PREDICTED\ RESPONSE}$

- $\hat{y}_i = h_\theta(x_i) = \sum_{j=0}^{d} \theta_j x_{ij} = \theta^T x_i$

MATRIX: $X = \begin{pmatrix} \boxed{1 \quad x_{11} \quad \cdots \quad x_{1d}} \\ \vdots \\ 1 \quad x_{N1} \quad \cdots \quad x_{Nd} \end{pmatrix}$ $\quad \theta = \begin{bmatrix} \boxed{\begin{matrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{matrix}} \end{bmatrix}$

$X\theta = \begin{bmatrix} \theta_0 + \theta_1 x_{11} + \cdots + \theta_d x_{1d} \\ \vdots \\ \theta_0 + \theta_1 x_{N1} + \cdots + \theta_d x_{Nd} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_N \end{bmatrix} = \hat{y}$

size $N \times 1$

- $h_\theta(x) = X\theta$ VECTORIZED FORM

$\hat{y} = X\theta$

Loss function MSE

- For the linear regression cost function:

MSE: $$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}[h_\theta(x_i) - y_i]^2$$

$$= \frac{1}{N}\underbrace{\sum_{i=1}^{N}[\hat{y}_i - y_i]^2}$$

$$= \frac{1}{N}\|\hat{y} - y\|_2^2$$

$$= \frac{1}{N}\|X\theta - y\|_2^2$$

$$y - \hat{y} = \begin{bmatrix} y_1 - \hat{y}_1 \\ \vdots \\ y_N - \hat{y}_N \end{bmatrix}$$

$$\|y - \hat{y}\|_2 = \sqrt{(y_1 - \hat{y}_1)^2 + \dots + (y_N - \hat{y}_N)^2}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_N \end{bmatrix}$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ . \\ . \\ . \\ \hat{y}_N \end{bmatrix}$$

Matrix and vector gradients

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

$$\frac{\partial f(x)}{\partial x} = \left[\frac{\partial f(x)}{\partial x_1} \quad \cdots \cdots \quad \frac{\partial f(x)}{\partial x_n} \right]$$

VECTOR GRADIENT

ROW VECTOR.

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

$$\frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

MATRIX GRADIENT

JACOBIAN MATRIX

SIZE $m \times n$

Properties

1) - If w, x are$(d\times 1)$ vectors, $\frac{\partial w^T x}{\partial x} = w^T$
2) - 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: $\frac{\partial x^T Ax}{\partial x} = 2Ax$

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

④ $w = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}$; $x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$

- $w^T x = w_1 x_1 + \dots + w_d x_d \in \mathbb{R}$

$$\frac{\partial w^T x}{\partial x} = \left[\frac{\partial w^T x}{\partial x_1} \; \dots\dots \; \frac{\partial w^T x}{\partial x_d} \right] = [w_1 \;\; w_2 \dots\dots w_d] = w^T$$

⑤ $x = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$

$\|x\|^2 = x_1^2 + \dots + x_d^2 \in \mathbb{R}$

$$\frac{\partial \|x\|^2}{\partial x} = \left[\frac{\partial \|x\|^2}{\partial x_1} \; \dots\dots \; \frac{\partial \|x\|^2}{\partial x_d} \right] = [2x_1 \;\; 2x_2 \dots \;\; 2x_d] = 2x^T$$

Min loss function

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

→ X: SIZE $\boxed{N \times (d+1)}$

θ: SIZE $(d+1) \times 1$

y: SIZE $N \times 1$

$$J(\theta) = \frac{1}{N} \left|\underbrace{|X\theta - y|}\right|^2 \in \mathbb{R}$$

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

CHAIN RULE

$$= \frac{2}{N} (X\theta - y)^T \cdot X = 0$$

TRANSPOSE $\left[\begin{array}{l} (A^T)^T = A \\ (A \cdot B)^T = B^T A^T \end{array} \right.$

$$(X\theta - y)^T \cdot X = 0$$

$$X^T (X\theta - y) = 0 \Rightarrow (X^T X)\theta = X^T y$$

CLOSED FORM OPTIMAL SOLUTION TO MIN MSE

$$\boxed{\theta = (X^T X)^{-1} \cdot X^T y}$$

$(d+1) \times (d+1) \cdot (d+1) \times N, (N \times 1) \to$ SIZE $(d+1) \times 1$

INPUT: X MATRIX TRAINING DATA

y VECTOR RESPONSES

OUTPUT: $\theta = (X^T X)^{-1} X^T y$ MIN MSE

ASSUMED $X^T X$ INVERTIBLE

FOR. $d=1$, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$

X: size $N \times 2$

How Well Does the Model Fit?

- Residual Sum of Squares
  - $RSS = \sum[R_i\ ]^2 = \sum[y_i - (\theta_0 + \theta_1 x_i)]^2$
- Total Sum of Squares
  - $TSS = \sum[y_i - \bar{y}]^2$ Var (y)
  - Total variance of the response
- Proportion of variability in Y that can be explained using X
  - $R^2 = 1 - \frac{RSS}{TSS} \in [0,1]$

1 FEATURE

- Correlation between feature and response

$$\rho = \ Corr(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}$$

- For simple regression R^2 is equal to ρ^2
- For multiple LR, R^2 can be used

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

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

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{id} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_N \end{bmatrix}$$

- 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 $\mathrm{d} \leq \mathrm{n}$

A MATRIX, G PSEUDO-INV

$$AGA = A$$

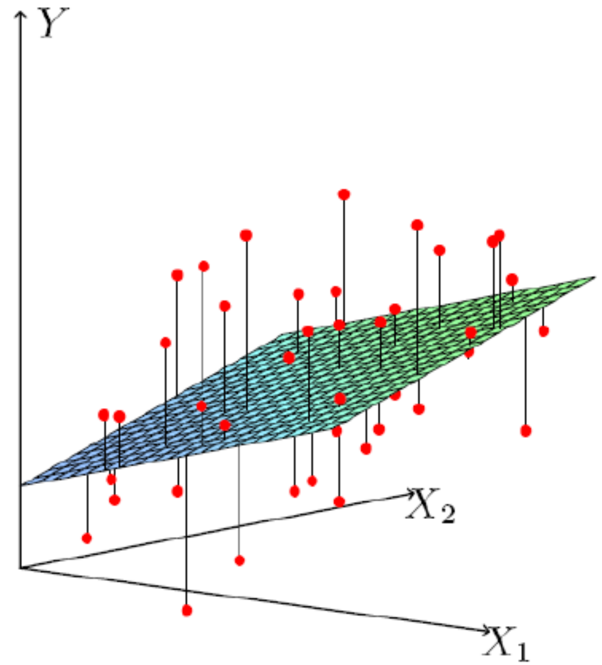
If A is invertible $\Rightarrow G = A^{-1}$

USE RIDGE REG. $\Rightarrow$ ALWAYS INVERTIBLE

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

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



Lab Simple Linear Regression

```
#!/usr/bin/env python

import numpy as np
import matplotlib.pyplot as plt

import pandas as pd
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV

from sklearn.datasets import load_boston
boston_dataset = load_boston()
```

Boston house prediction dataset

Lab

```
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
boston['MEDV'] = boston_dataset.target
boston.head(5)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
print(len(boston))
boston.shape
```

```
506
```

```
(506, 14)
```

```python
correlation_matrix = boston.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<AxesSubplot:>
```

Lab

```python
# Simple LR
X = pd.DataFrame(np.c_[boston['RM']], columns = ['RM'])

Y = boston['MEDV']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=5)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 1)
(102, 1)
(404,)
(102,)
```

```python
slr = LinearRegression()
slr.fit(X_train, Y_train)
```

Lab

```
print(slr.intercept_)
print(slr.coef_)
```

```
-32.839129906011266
[8.82345634]
```

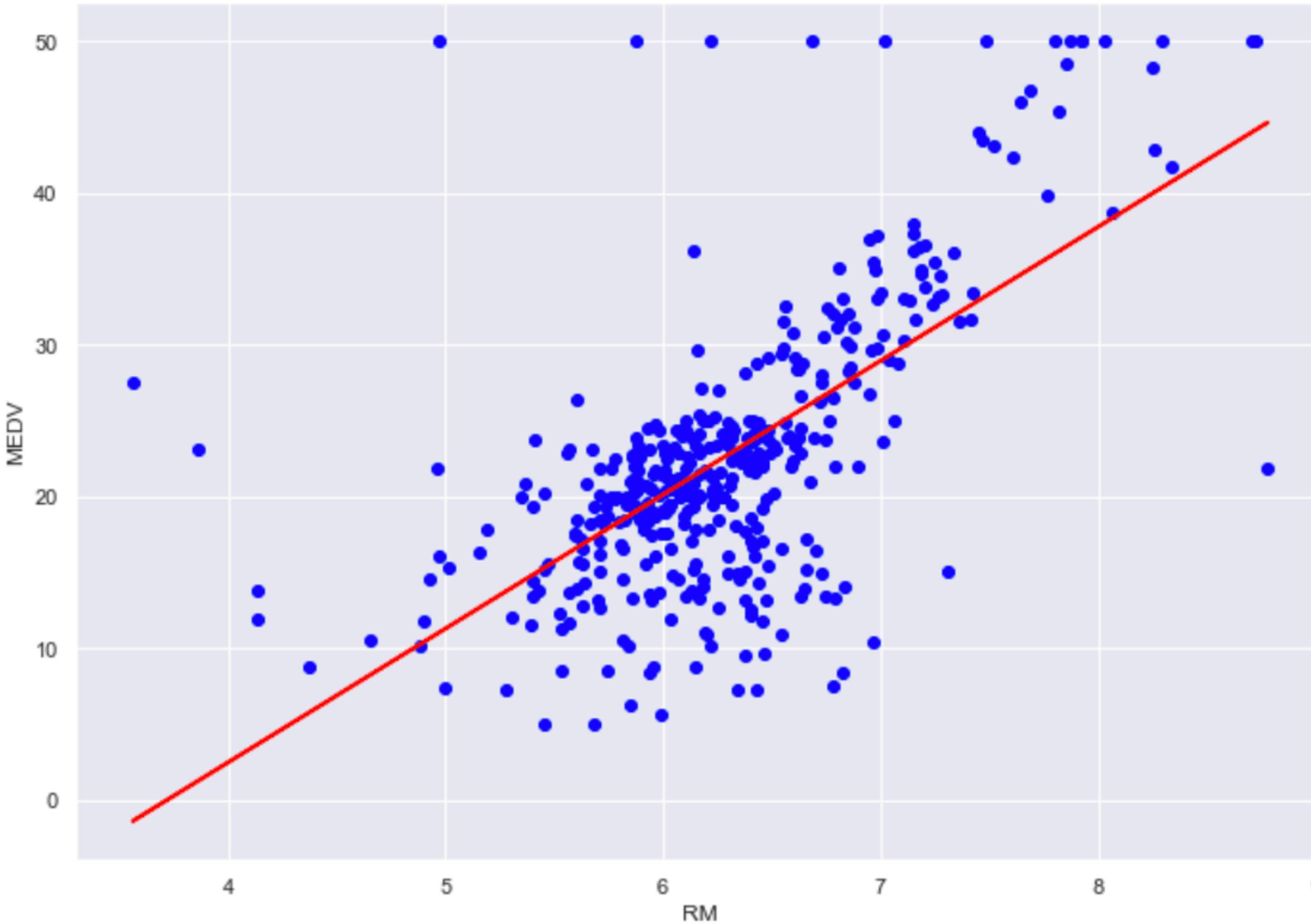
```
Y_train_predict = slr.predict(X_train)
mse = mean_squared_error(Y_train, Y_train_predict)

print("The model performance for training set")
print('MSE is {}'.format(mse))
```

```
The model performance for training set
MSE is 48.612648648611334
```

```
df = pd.DataFrame({'Actual': Y_train, 'Predicted': Y_train_predict})
df.head()
```

	Actual	Predicted
33	13.1	17.463395
283	50.0	37.069115
418	8.8	19.722200
502	20.6	21.160423
402	12.1	23.666285

```python
plt.scatter(X_train, Y_train,  color='blue')
plt.plot(X_train, Y_train_predict, color='red', linewidth=2)
plt.xlabel('RM')
plt.ylabel('MEDV')
plt.show()
```



Multiple LR Lab

```
# Multiple LR

#X_multi = pd.DataFrame(np.c_[boston['LSTAT'], boston['RM']], columns = ['LSTAT','RM'])
X_multi = boston.loc[:, boston.columns != 'MEDV']
Y = boston['MEDV']

X_m_train, X_m_test, Y_m_train, Y_m_test = train_test_split(X_multi, Y, test_size = 0.2, random_state=5)
print(X_m_train.shape)
print(X_m_test.shape)
print(Y_m_train.shape)
print(Y_m_test.shape)
```

```
(404, 13)
(102, 13)
(404,)
(102,)
```

```
mlr = LinearRegression()
mlr.fit(X_m_train, Y_m_train)
```

```
LinearRegression()
```

Multiple LR Lab

```
coeff_df = pd.DataFrame(mlr.coef_, X_m_train.columns, columns=['Coefficient'])
coeff_df
```

	Coefficient
CRIM	-0.130800
ZN	0.049403
INDUS	0.001095
CHAS	2.705366
NOX	-15.957050
RM	3.413973
AGE	0.001119
DIS	-1.493081
RAD	0.364422
TAX	-0.013172
PTRATIO	-0.952370
B	0.011749
LSTAT	-0.594076

Simple vs Multiple LR

```
print(slr.intercept_)
print(slr.coef_)
```

```
-32.839129906011266
[8.82345634]
```

```
Y_train_predict = slr.predict(X_train)
mse = mean_squared_error(Y_train, Y_train_predict)

print("The model performance for training set")
print('MSE is {}'.format(mse))
```

```
The model performance for training set
MSE is 48.612648648611334
```

```
Y_m_train_predict = mlr.predict(X_m_train)
mse = mean_squared_error(Y_m_train, Y_m_train_predict)

print("The model performance for training set")
print("--------------------------------------")
print('MSE is {}'.format(mse))
print("\n")
```

```
The model performance for training set
--------------------------------------
MSE is 22.477090408387635
```

Simple vs Multiple LR

```
df_m = pd.DataFrame({'Actual': Y_train, 'Predicted simple': Y_train_predict, 'Predicted multi': Y_m_train_predict})
df_m.head(10)
```

	Actual	Predicted simple	Predicted multi
33	13.1	17.463395	13.828770
283	50.0	37.069115	44.528528
418	8.8	19.722200	3.915991
502	20.6	21.160423	22.377959
402	12.1	23.666285	18.235923
368	50.0	11.013448	25.523748
201	24.1	21.531008	29.439747
310	16.1	11.039918	18.694533
343	23.9	26.242734	27.856463
230	24.3	19.933962	24.644734