Linear Regression

Shusen Wang

Warm-up: Vector and Matrix

Vector and Matrix

Vector (n-dim)

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

Matrix ($n \times d$)

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1d} \\ a_{21} & a_{22} & \cdots & a_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nd} \end{bmatrix}$$

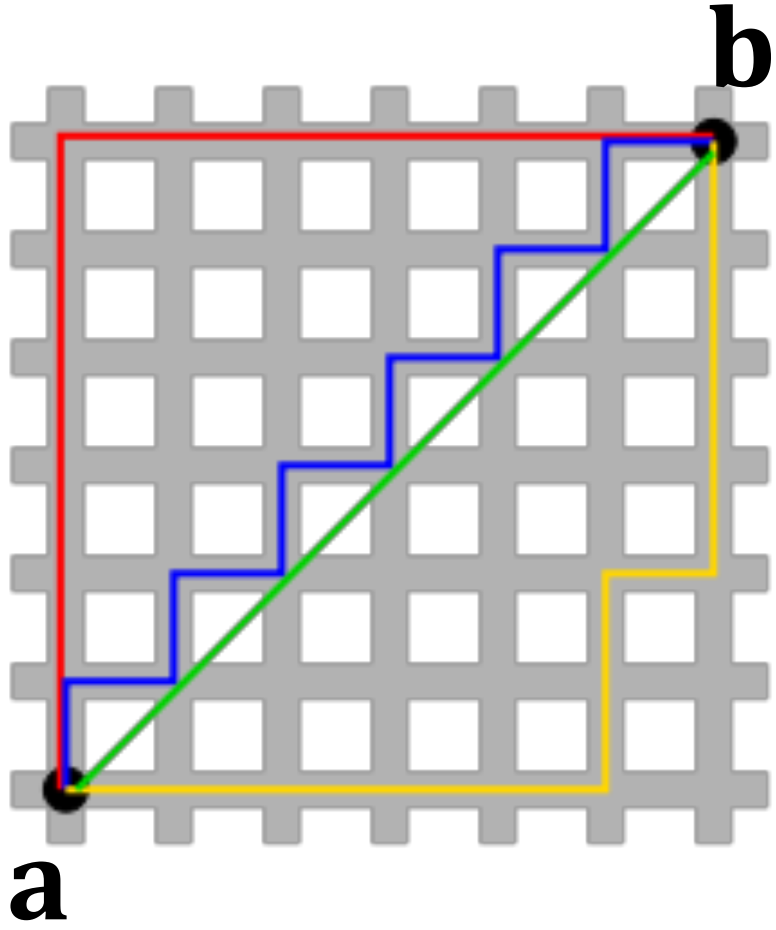
Row and columns

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_{:1} & \mathbf{a}_{:2} & \cdots & \mathbf{a}_{:d} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{1:} \\ \mathbf{a}_{2:} \\ \vdots \\ \mathbf{a}_{n:} \end{bmatrix}$$

Vector Norms

- The ℓ_p norm: $\|\mathbf{x}\|_p := \left(\sum_i |x_i|^p\right)^{1/p}$.
- The ℓ_2 norm: $\|\mathbf{x}\|_2 = \left(\sum_i x_i^2\right)^{1/2}$ (the Euclidean norm).
- The ℓ_1 norm $\|\mathbf{x}\|_1 = \sum_i |x_i|$.
- The ℓ_∞ norm is defined by $\|\mathbf{x}\|_\infty = \max_i |x_i|$.

Vector Norms



- The ℓ_2-distance (Euclidean distance):
  $||\mathbf{a} - \mathbf{b}||_2$ (green line)

- The ℓ_1-distance (Manhattan distance):
  $||\mathbf{a} - \mathbf{b}||_1$ (red, blue, yellow lines)

Transpose and Rank

Transpose:

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^T = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

Square matrix: a matrix with the same number of rows and columns.

Symmetric: a square matrix $\mathbf{A}$ is symmetric if $\mathbf{A}^T = \mathbf{A}$.

Rank: the number of linearly independent rows (or columns).

Full rank: a square matrix is full rank if the rank equals to #columns.

Eigenvalue Decomposition

- Let $\mathbf{A}$ be any $n \times n$ symmetric matrix.
- Eigenvalue decomposition: $\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{v}_i \mathbf{v}_i^T$.
- Eigenvalues satisfy $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.
- Eigenvectors satisfy $\mathbf{v}_i^T \mathbf{v}_j = 0$ for all $i \neq j$.

- $\mathbf{A}$ is full rank $\Longleftrightarrow$ all the eigenvalues are nonzero.

Warm-up: Optimization

Optimization: Basics

Optimization problem: $\min_{\mathbf{w}} f(\mathbf{w})\,; \quad \text{s.t. } \mathbf{w} \in \mathcal{C}\,.$

- $\mathbf{w} = [w_1, \cdots, w_d]$: optimization variables
- $f : \mathbb{R}^d \mapsto \mathbb{R}$: objective function
- $\mathcal{C}$ (a subset of $\mathbb{R}^d$) : feasible set

Optimization: Basics

Optimization problem: $\min_{\mathbf{w}} f(\mathbf{w})$; $\boxed{\text{s.t. } \mathbf{w} \in \mathcal{C} \text{ .}}$

Constraint

- $\mathbf{w} = [w_1, \cdots, w_d]$: optimization variables
- $f : \mathbb{R}^d \mapsto \mathbb{R}$: objective function
- $\mathcal{C}$ (a subset of $\mathbb{R}^d$) : feasible set

Optimization: Basics

Optimization problem: $\min_{\mathbf{w}} f(\mathbf{w})$; s. t. $\mathbf{w} \in \mathcal{C}$.

- $\mathbf{w} = [w_1, \cdots, w_d]$: optimization variables
- $f : \mathbb{R}^d \mapsto \mathbb{R}$: objective function
- $\mathcal{C}$ (a subset of $\mathbb{R}^d$) : feasible set

- $\mathbf{w}^\star = \operatorname{argmin}_{\mathbf{w} \in \mathcal{C}} f(\mathbf{w})$ is the optimal solution to the problem
  - $f(\mathbf{w}^\star) \leq f(\mathbf{w})$ for all the vectors $\mathbf{w}$ in the set $\mathcal{C}$.
  - $\mathbf{w}^\star$ may not exist; if it exists, it may not be unique.

Least Squares Regression

The Linear Regression Task

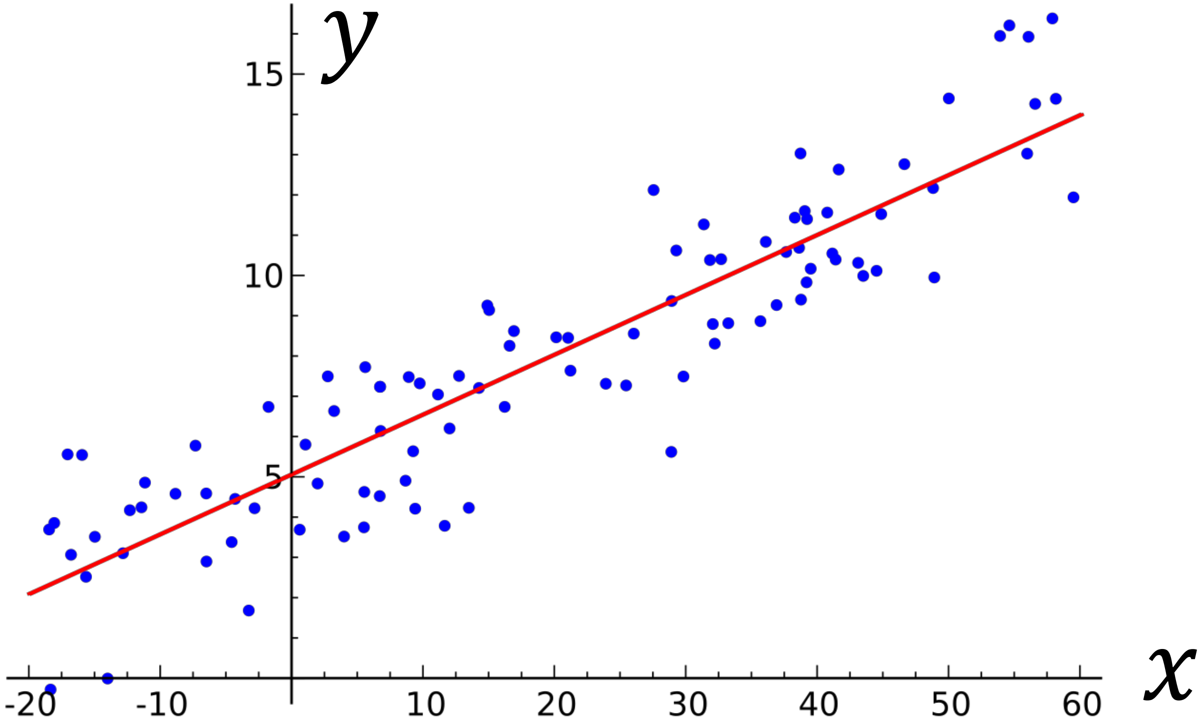
Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

1-dim ($d = 1$) example:

Solution:

$$y_i \approx 0.15\, x_i + 5.0$$

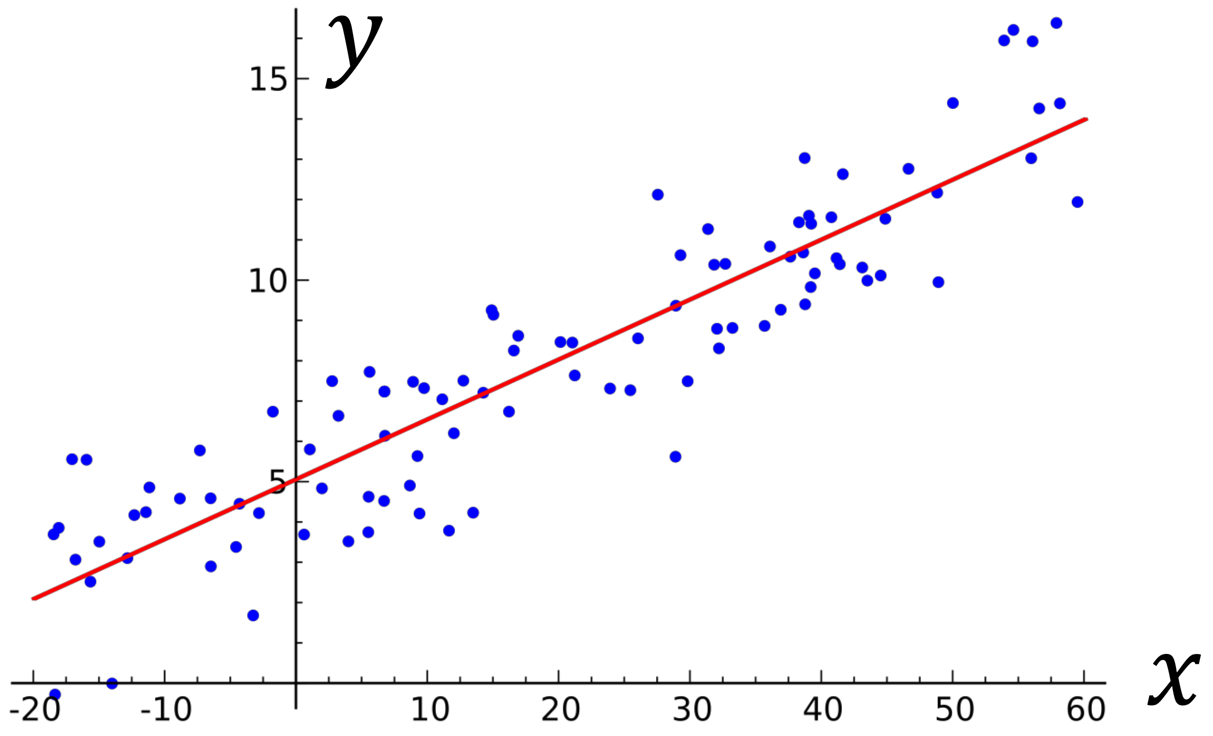


The Linear Regression Task

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

Question (regard training):
how to compute $\mathbf{w}$ and b?



The Linear Regression Task

Input: vectors $\mathbf{x}_1, \cdots, \mathbf{x}_n \in \mathbb{R}^d$ and labels $y_1, \cdots, y_n \in \mathbb{R}$

Output: a vector $\mathbf{w} \in \mathbb{R}^d$ and scalar $b \in \mathbb{R}$ such that $\mathbf{x}_i^T \mathbf{w} + b \approx y_i$.

Method: least squares regression.

- The optimization model:

$$\min_{\mathbf{w}, b} L(\mathbf{w}, b), \quad \text{where } L(\mathbf{w}, b) = \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^{\mathbf{2}}$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} L(\mathbf{w}, b), \qquad \text{where } L(\mathbf{w}, b) = \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} L(\mathbf{w}, b), \quad \text{where } L(\mathbf{w}, b) = \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + \boxed{b} - y_i\right)^2$$

Intercept

Least Squares Regression

- The optimization model:

$$\min_{\mathbf{w},b} L(\mathbf{w}, b), \qquad \text{where } L(\mathbf{w}, b) = \sum_{i=1}^{n} \left(\mathbf{x}_i^T \mathbf{w} + b - y_i\right)^2$$

$$\Updownarrow$$

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i\right)^2$$

- Define $\bar{\mathbf{x}}_i = [\mathbf{x}_i; 1] \in \mathbb{R}^{d+1}$
- Define $\bar{\mathbf{w}} = [\mathbf{w}, b] \in \mathbb{R}^{d+1}$
- $\Rightarrow \quad \mathbf{x}_i^T \mathbf{w} + b = \bar{\mathbf{x}}_i^T \bar{\mathbf{w}}$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \sum_{i=1}^{n} \left(\bar{\mathbf{x}}_i^T \bar{\mathbf{w}} - y_i \right)^2$$

$$\Updownarrow$$

Matrix form: $$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad ||\bar{\mathbf{X}}\, \bar{\mathbf{w}} - \mathbf{y}||_2^2$$

$$\bar{\mathbf{X}} = \begin{bmatrix} \mathbf{x}_1^T & 1 \\ \mathbf{x}_2^T & 1 \\ \vdots & \vdots \\ \mathbf{x}_n^T & 1 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad \bar{\mathbf{X}}\bar{\mathbf{w}} - \mathbf{y} = \begin{bmatrix} \bar{\mathbf{x}}_1^T \bar{\mathbf{w}} - y_1 \\ \bar{\mathbf{x}}_2^T \bar{\mathbf{w}} - y_2 \\ \vdots \\ \bar{\mathbf{x}}_n^T \bar{\mathbf{w}} - y_n \end{bmatrix}$$

$n \times (d+1)$ $\qquad$ $n \times 1$ $\qquad$ $n \times 1$

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad ||\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}||_2^2$$

Tasks

Methods

Algorithms

Linear Regression

Least Squares Regression

LASSO

Least Absolute Deviations

?

Least Squares Regression

- The optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad ||\bar{\mathbf{X}}\, \bar{\mathbf{w}} - \mathbf{y}||_2^2$$

Tasks	Methods	Algorithms
	Least Squares Regression	Analytical Solution
Linear Regression	LASSO	Gradient Descent (GD)
	Least Absolute Deviations	Conjugate Gradient (CG)

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Gradient: $$\frac{\partial \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2}{\partial \bar{\mathbf{w}}} = 2(\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \bar{\mathbf{X}}^T\mathbf{y})$$

Algorithms

Analytical Solution

Gradient Descent (GD)

Conjugate Gradient

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Gradient: $\dfrac{\partial\ \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2}{\partial\ \bar{\mathbf{w}}} = 2(\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \bar{\mathbf{X}}^T\mathbf{y}) \boxed{= \mathbf{0}}$

1st-order optimal condition

Algorithms

Analytical Solution

Gradient Descent (GD)

Conjugate Gradient

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad ||\bar{\mathbf{X}}\, \bar{\mathbf{w}} - \mathbf{y}||_2^2$$

Gradient: $\dfrac{\partial\ ||\bar{\mathbf{X}}\, \bar{\mathbf{w}} - \mathbf{y}||_2^2}{\partial\ \bar{\mathbf{w}}} = 2(\bar{\mathbf{X}}^T \bar{\mathbf{X}}\, \bar{\mathbf{w}} - \bar{\mathbf{X}}^T \mathbf{y}) = \mathbf{0}$

Normal equation: $\bar{\mathbf{X}}^T \bar{\mathbf{X}}\, \bar{\mathbf{w}} = \bar{\mathbf{X}}^T \mathbf{y}$

Assume $\bar{\mathbf{X}}^T \bar{\mathbf{X}}$ is full rank.

Analytical solution: $\bar{\mathbf{w}}^{\star} = (\bar{\mathbf{X}}^T \bar{\mathbf{X}})^{-1}\, \bar{\mathbf{X}}^T \mathbf{y}$

Algorithms

- Analytical Solution
- Gradient Descent (GD)
- Conjugate Gradient

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Gradient: $\frac{\partial \; \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2}{\partial \;\; \bar{\mathbf{w}}} = 2\left(\bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}} \; - \bar{\mathbf{X}}^T\mathbf{y}\right) = \mathbf{0}$

Gradient descent repeats:

1. Compute gradient: $\mathbf{g}_t = \bar{\mathbf{X}}^T\bar{\mathbf{X}}\,\bar{\mathbf{w}}_t \; - \; \bar{\mathbf{X}}^T\mathbf{y}$
2. Update: $\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - \alpha_t\,\mathbf{g}_t$

Algorithms

Analytical Solution

Gradient Descent (GD)

Conjugate Gradient

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Convergence: after $O\left(\kappa \log \frac{1}{\epsilon}\right)$ iterations,

$$\left|\left|\bar{\mathbf{X}}\,(\bar{\mathbf{w}}_t - \bar{\mathbf{w}}^\star)\right|\right|_2 \leq \epsilon \left|\left|\bar{\mathbf{X}}\,(\bar{\mathbf{w}}_0 - \bar{\mathbf{w}}^\star)\right|\right|_2.$$

$\kappa = \frac{\lambda_{\max}(\bar{\mathbf{X}}^T\bar{\mathbf{X}})}{\lambda_{\min}(\bar{\mathbf{X}}^T\bar{\mathbf{X}})}$ is the condition number.

Algorithms

- Analytical Solution
- Gradient Descent (GD)
- Conjugate Gradient

Least Squares Regression

- Solve the optimization model:

$$\min_{\overline{\mathbf{w}} \in \mathbb{R}^{d+1}} \quad \left|\left|\overline{\mathbf{X}}\, \overline{\mathbf{w}} - \mathbf{y}\right|\right|_2^2$$

Convergence: after $O\left(\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ iterations,

$$\left|\left|\overline{\mathbf{X}}\,(\overline{\mathbf{w}}_t - \overline{\mathbf{w}}^\star)\right|\right|_2 \leq \epsilon \left|\left|\overline{\mathbf{X}}\,(\overline{\mathbf{w}}_0 - \overline{\mathbf{w}}^\star)\right|\right|_2.$$

$\kappa = \frac{\lambda_{\max}(\overline{\mathbf{X}}^T\overline{\mathbf{X}})}{\lambda_{\min}(\overline{\mathbf{X}}^T\overline{\mathbf{X}})}$ is the condition number.

The pseudo-code of CG is available at the Wikipedia.

Algorithms

Analytical Solution

Gradient Descent (GD)

Conjugate Gradient

Least Squares Regression

- Solve the optimization model:

$$\min_{\bar{\mathbf{w}} \in \mathbb{R}^{d+1}} \ \|\bar{\mathbf{X}}\,\bar{\mathbf{w}} - \mathbf{y}\|_2^2$$

Tasks	Methods	Algorithms
	Least Squares Regression	Analytical Solution
Linear Regression	LASSO	Gradient Descent (GD)
	Least Absolute Deviations	Conjugate Gradient

Solve Least Squares in Python

1. Load Data

```
from keras.datasets import boston_housing

(x_train, y_train), (x_test, y_test) = boston_housing.load_data()

print('shape of x_train: ' + str(x_train.shape))
print('shape of x_test: ' + str(x_test.shape))
print('shape of y_train: ' + str(y_train.shape))
print('shape of y_test: ' + str(y_test.shape))
```

```
shape of x_train: (404, 13)
shape of x_test: (102, 13)
shape of y_train: (404,)
shape of y_test: (102,)
```

2. Add A Feature

```
import numpy

n, d = x_train.shape
xbar_train = numpy.concatenate((x_train, numpy.ones((n, 1))),
                               axis=1)

print('shape of x_train: ' + str(x_train.shape))
print('shape of xbar_train: ' + str(xbar_train.shape))
```

```
shape of x_train: (404, 13)
shape of xbar_train: (404, 14)
```

3. Solve the Least Squares

Analytical solution: $\bar{\mathbf{w}} = \left(\boxed{\bar{\mathbf{X}}^T \bar{\mathbf{X}}}\right)^{-1} \bar{\mathbf{X}}^T \mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Squares

Analytical solution: $\bar{\mathbf{w}} = \boxed{(\bar{\mathbf{X}}^T \bar{\mathbf{X}})^{-1}} \bar{\mathbf{X}}^T \mathbf{y}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Squares

Analytical solution: $\mathbf{\bar{w}} = \boxed{(\mathbf{\bar{X}}^T \mathbf{\bar{X}})^{-1}} \boxed{\mathbf{\bar{X}}^T \mathbf{y}}$

```
xx = numpy.dot(xbar_train.T, xbar_train)
xx_inv = numpy.linalg.pinv(xx)
xy = numpy.dot(xbar_train.T, y_train)
w = numpy.dot(xx_inv, xy)
```

3. Solve the Least Squares

Training Mean Squared Error (MSE): $\frac{1}{n} \left|\left| \mathbf{y} - \bar{\mathbf{X}} \bar{\mathbf{w}} \right|\right|_2^2$

```
y_lsr = numpy.dot(xbar_train, w)
diff = y_lsr - y_train
mse = numpy.mean(diff * diff)
print('Train MSE: ' + str(mse))
```

```
Train MSE: 22.004800838348142
```

Linear Regression for Housing Price

Linear Regressor

$\bar{\mathbf{w}}$

Train

label, y_i	**Price**	Selling price of the property ($1,000)
features, $\mathbf{x}_i$	**Beds**	Number of bedrooms in the house
	Baths	Number of bathrooms in the house
	Square Feet	Size of the house in square feet
	Miles to Resort	Miles from the property to the downtown resort area
	Miles to Base	Miles from the property to the base of the ski resort's mountain
	Acres	Lot size in number of acres
	Cars	Number of cars that will fit into the garage
	Years Old	Age of the house, in years, at the time it was listed
	DoM	Number of days the house was on the market before it sold

Linear Regression for Housing Price

Features of a House, $\mathbf{x}'$

➔ Extend it to $\bar{\mathbf{x}}'$

Linear Regressor $\bar{\mathbf{w}}$

Predict

Price: $\bar{\mathbf{w}}^T \bar{\mathbf{x}}' = \$500\text{K}$

Train

label, y_i

features, $\mathbf{x}_i$

Price	Selling price of the property ($1,000)
Beds	Number of bedrooms in the house
Baths	Number of bathrooms in the house
Square Feet	Size of the house in square feet
Miles to Resort	Miles from the property to the downtown resort area
Miles to Base	Miles from the property to the base of the ski resort's mountain
Acres	Lot size in number of acres
Cars	Number of cars that will fit into the garage
Years Old	Age of the house, in years, at the time it was listed
DoM	Number of days the house was on the market before it sold

4. Make Prediction for Test Samples

- Add a feature to the test feature matrix: $\mathbf{X}_{\text{test}} \rightarrow \overline{\mathbf{X}}_{\text{test}}$.
- Make prediction by: $\mathbf{y}_{\text{pred}} = \overline{\mathbf{X}}_{\text{test}} \overline{\mathbf{w}}$.

```
n_test, _ = x_test.shape
xbar_test = numpy.concatenate((x_test, numpy.ones((n_test, 1))), axis=1)
y_pred = numpy.dot(xbar_test, w)
```

4. Make Prediction for Test Samples

- Add a feature to the test feature matrix: $\mathbf{X}_{\text{test}} \rightarrow \overline{\mathbf{X}}_{\text{test}}$.
- Make prediction by: $\mathbf{y}_{\text{pred}} = \overline{\mathbf{X}}_{\text{test}} \overline{\mathbf{w}}$.
- MSE (testing): $\frac{1}{n_{\text{test}}} \left\| \mathbf{y}_{\text{pred}} - \mathbf{y}_{\text{test}} \right\|_2^2$

```
# mean squared error (testing)

diff = y_pred - y_test
mse = numpy.mean(diff * diff)
print('Test MSE: ' + str(mse))
```

```
Test MSE: 23.19559925640985
```

Training MSE is **22.0**

4. Make Prediction for Testing Samples

Test MSE is **23.2**

Also known as out-of-sample error.

The gap is an indicator of generalization.

Training MSE is **22.0**

Also known as in-sample error.

5. Compare with Baseline

Baseline:

- whatever the features are, the prediction is $\text{mean}(\mathbf{y})$.

```
y_mean = numpy.mean(y_train)

diff = y_pred - y_mean
mse = numpy.mean(diff * diff)
print('Test MSE: ' + str(mse))
```

```
Test MSE: 57.38297638530044
```

Test MSE of least squares is **23.19**