Machine Learning

Linear Regression

Yuan-Kai Wang (王 元 凱)

ykwang@fju.edu.tw
http://www.ykwang.tw

Department of Electrical Engineering
Fu Jen Catholic University
輔仁大學電機工程系

2016~2025

Goal of This Unit

- This unit explains linear regression
  - It is one of supervised learning algorithms
- **Gradient descent**: the algorithm for linear regression
  - It is also used for logistic regression, neural network, deep neural network,...
- **Overfitting** and **model selection**

References

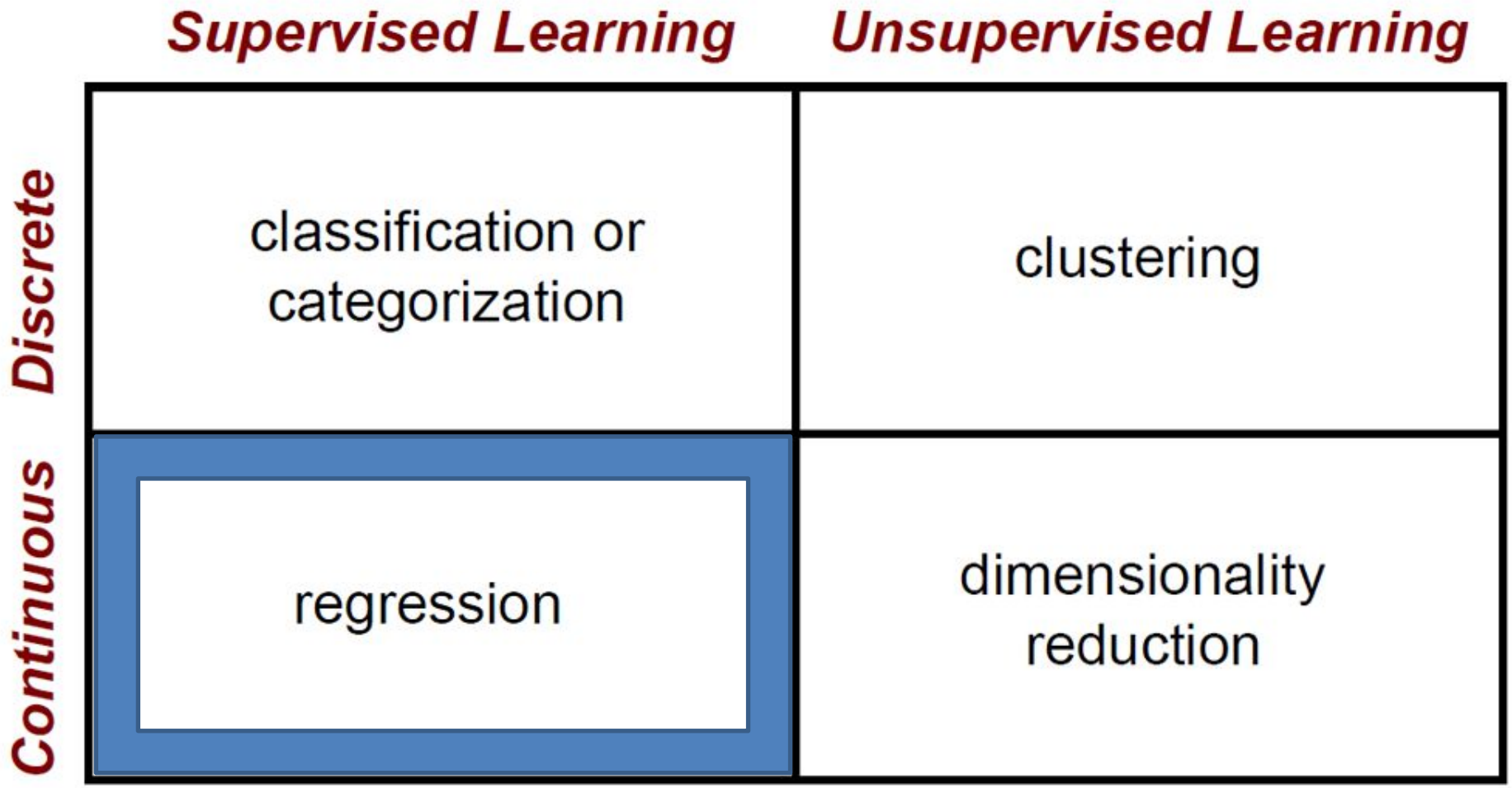
- Machine Learning in Action, by Peter Harrington, Manning Publications Co., 2012.
  - Chapter 8 Predicting numeric values: regression
- Applied Linear Statistical Models 5th, M. Kutner, C. Nachtsheim, J. Neter, W. Li, McGraw-Hill, 2005.
  - Chapter 1 Linear Regression with One Predictor Variable
  - Chapter 2 Inferences in Regression and Correlation Analysis
  - Chapter 6 Multiple Regression I

Contents

- What is linear regression
- Sum of square error
- Least square
- Gradient descent
- Overfitting and model selection
- Conclusion

WHAT IS LINEAR REGRESSION

Regression

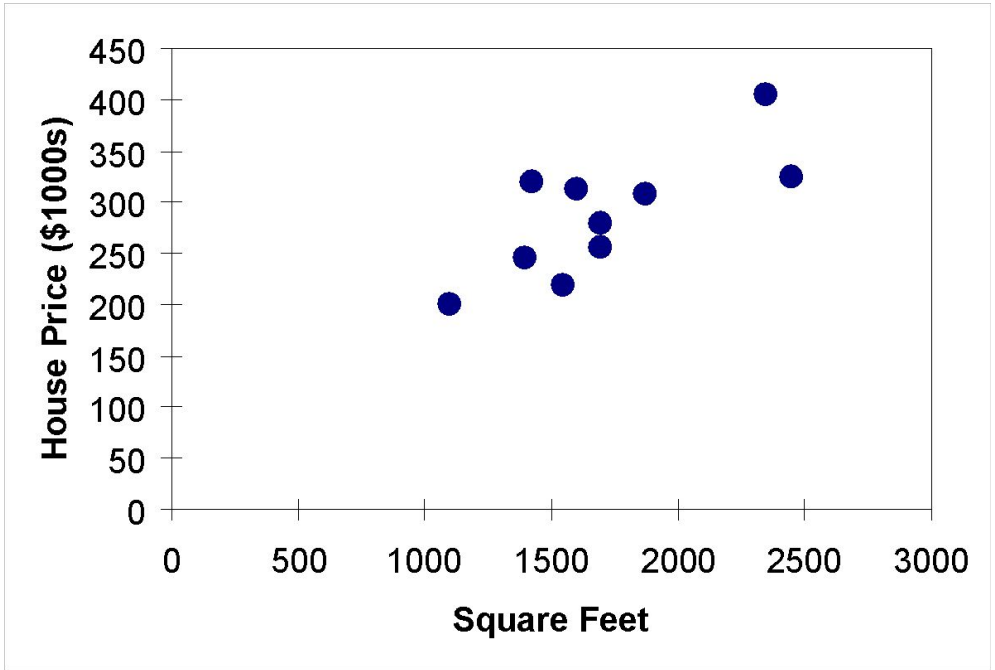

Supervised Learning
Unsupervised Learning
Discrete
Continuous
classification or categorization
clustering
regression
dimensionality reduction

Example: House Price

House Price in $1000s (Y)	Square Feet (X)
245	1400
312	1600
279	1700
308	1875
199	1100
219	1550
405	2350
324	2450
319	1425
255	1700

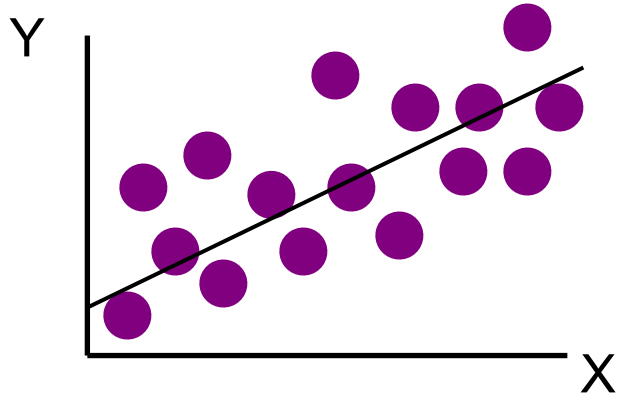
Data

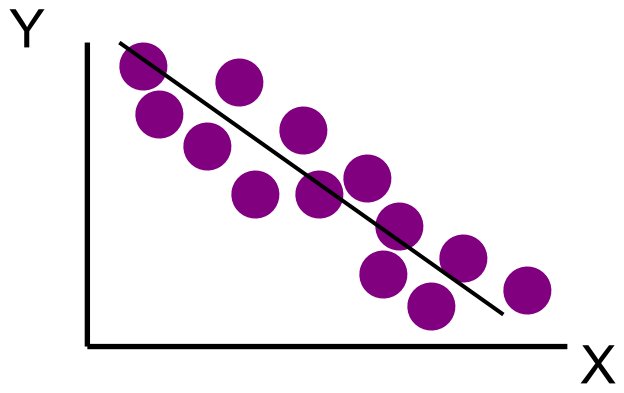
Scatter plot



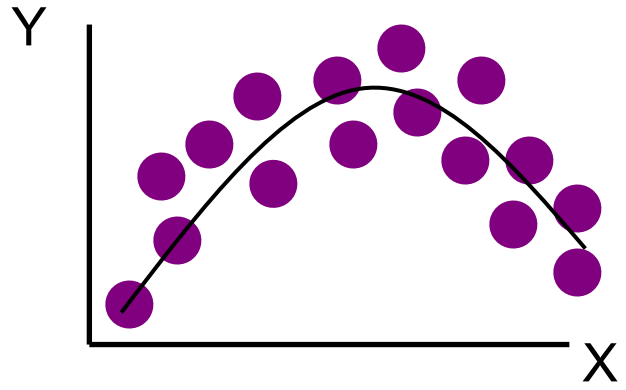
Correlation: Linear vs. Nonlinear

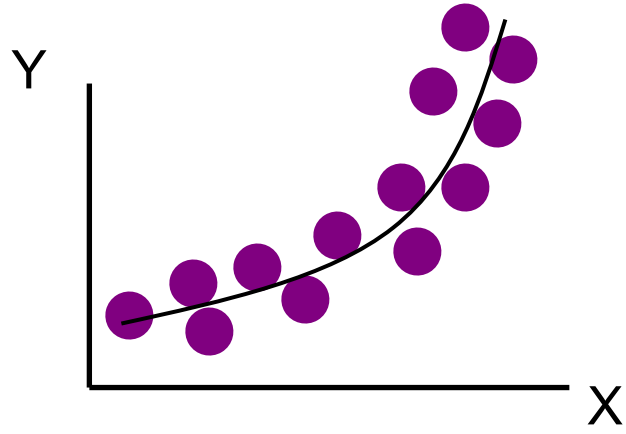
Linear relationships





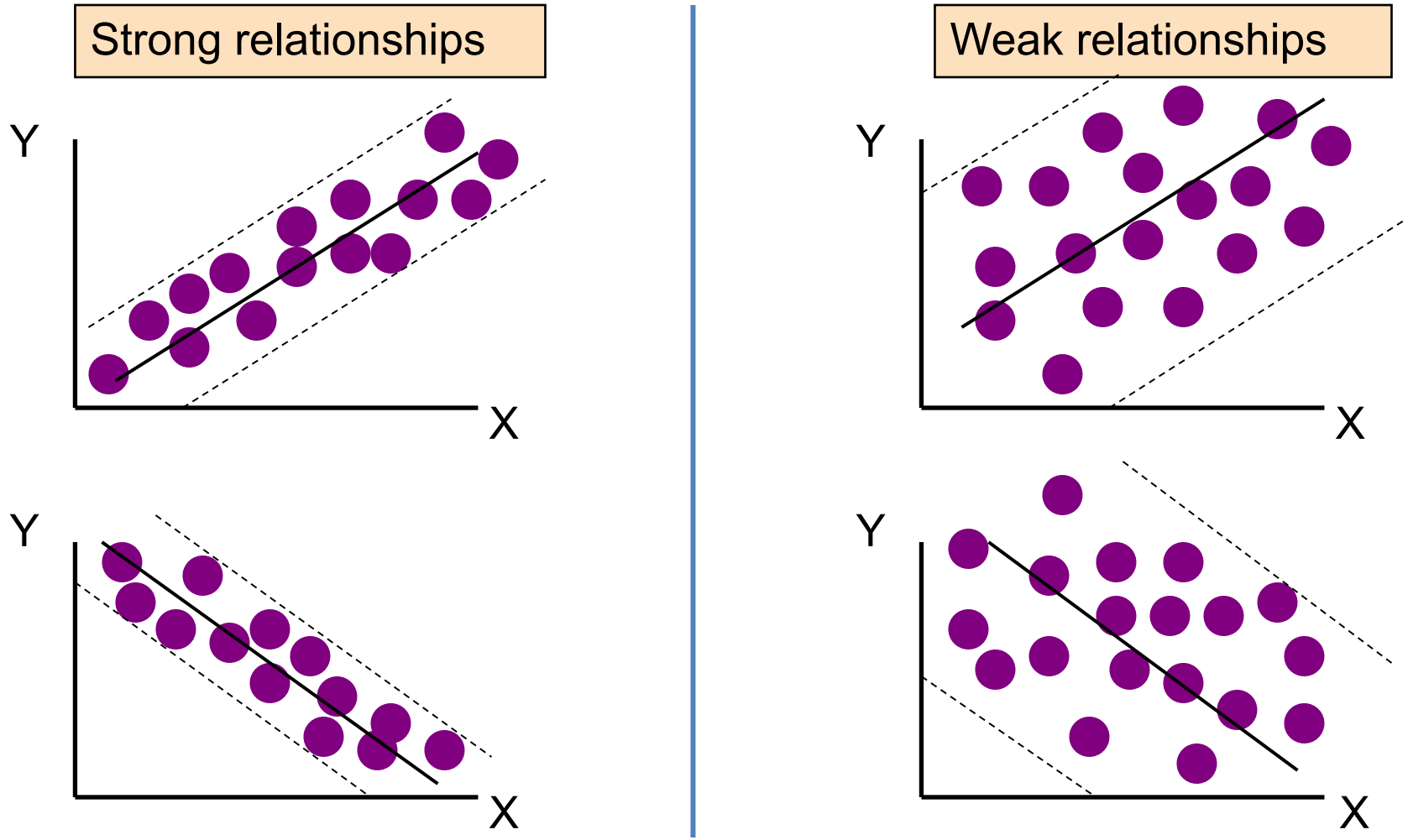
Curvilinear relationships





Scatter Plot of Data

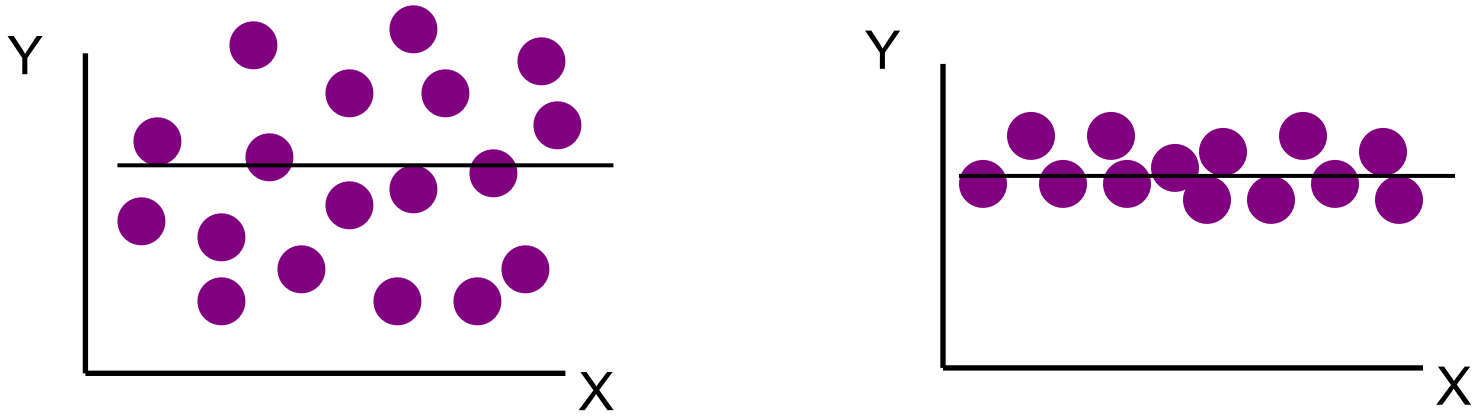
Correlation: Strong vs. Weak



Scatter Plot of Data

No Correlation

No relationship

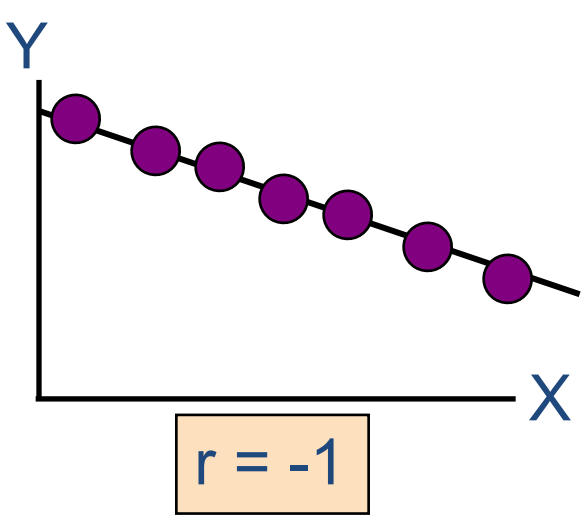


Scatter Plot of Data

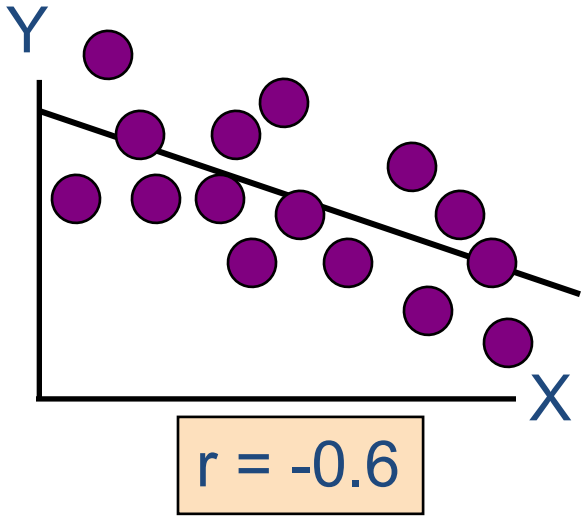
Correlation of Data

- Visualization of correlation
  - Ascending, descending
  - Strong, weak
- Quantification and mathematics for correlation
  - Correlation coefficient, covariance matrix
  - linear/nonlinear function

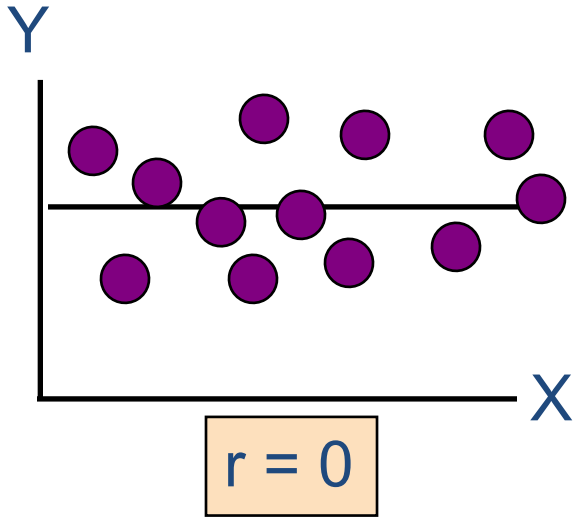
Correlation Coefficients



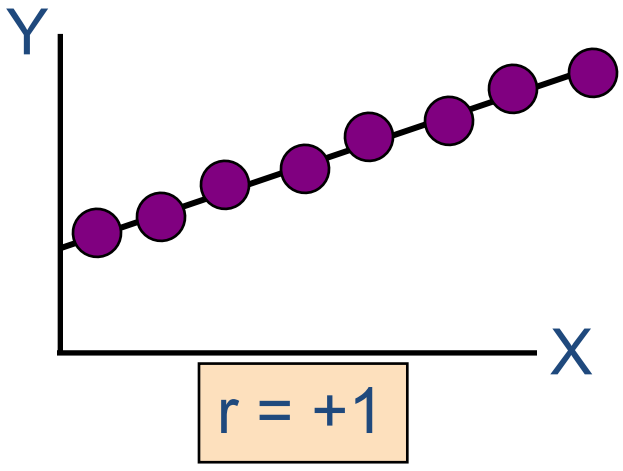
r = -1



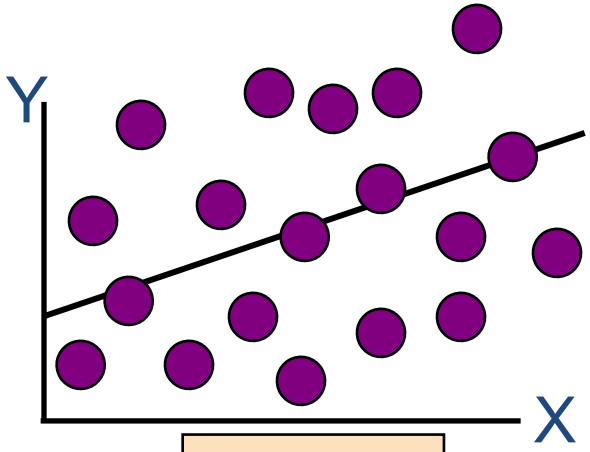
r = -0.6



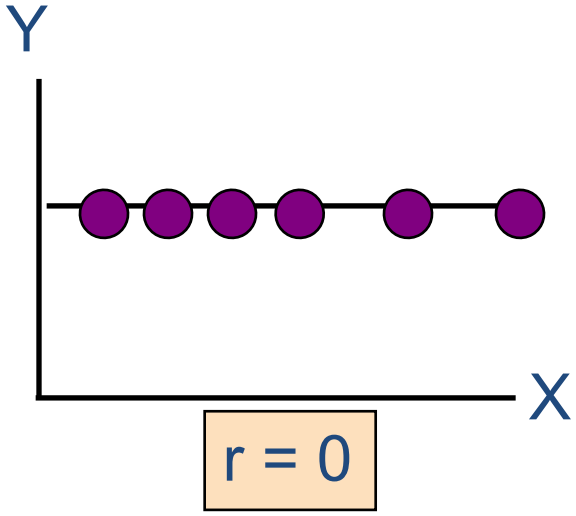
r = 0



r = +1



r = +0.3

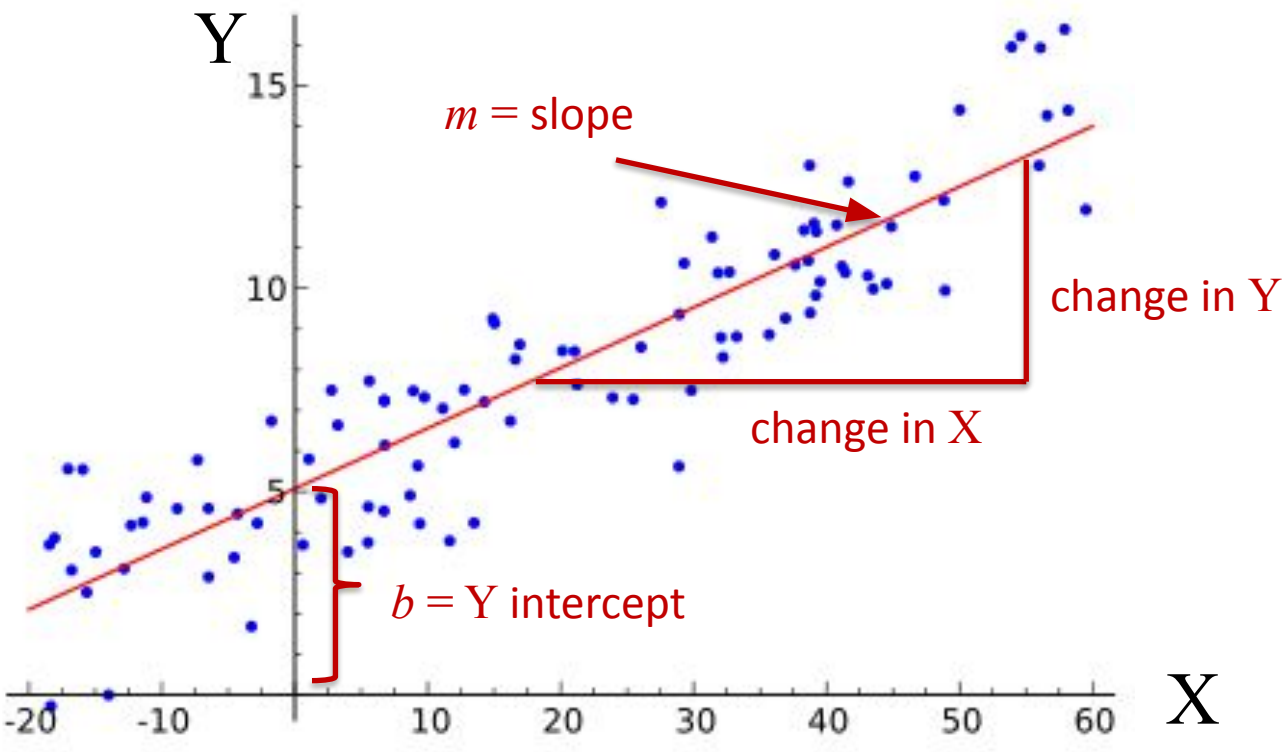


r = 0

Linear Function

Find the best line to explain the data

linear function: $Y = f(X) = mX+b$

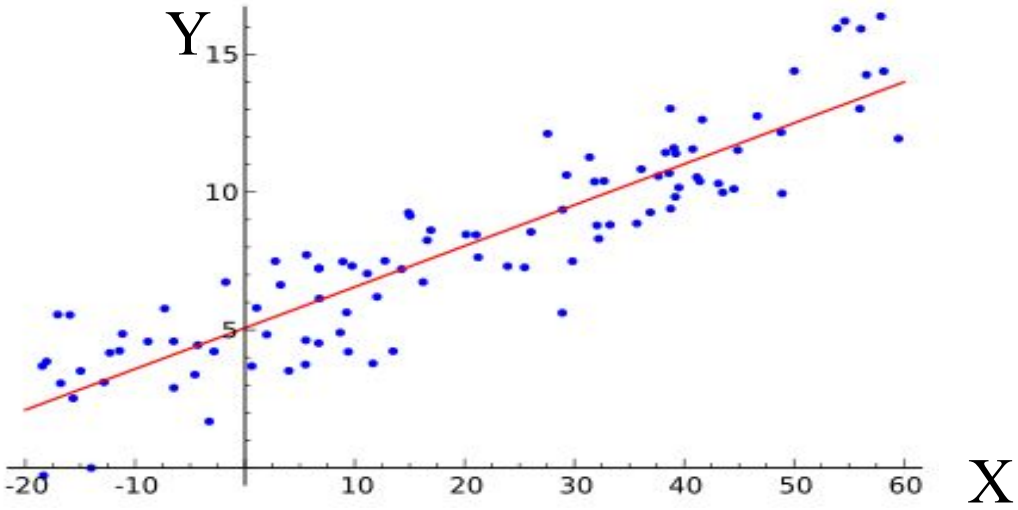


Linear Regression

- It is an algorithm that
  - Given a set of *linear* data (x, y)
  - Find the linear function $y = mx + b$

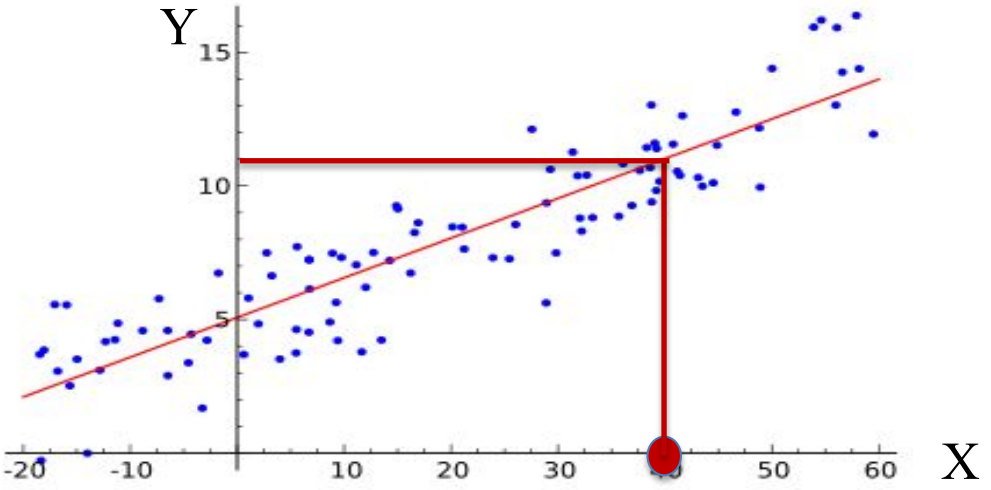
(x, y) are noisy data
$y=mx+b$ is a best-fit line

Why find the line function?
→ Predict future



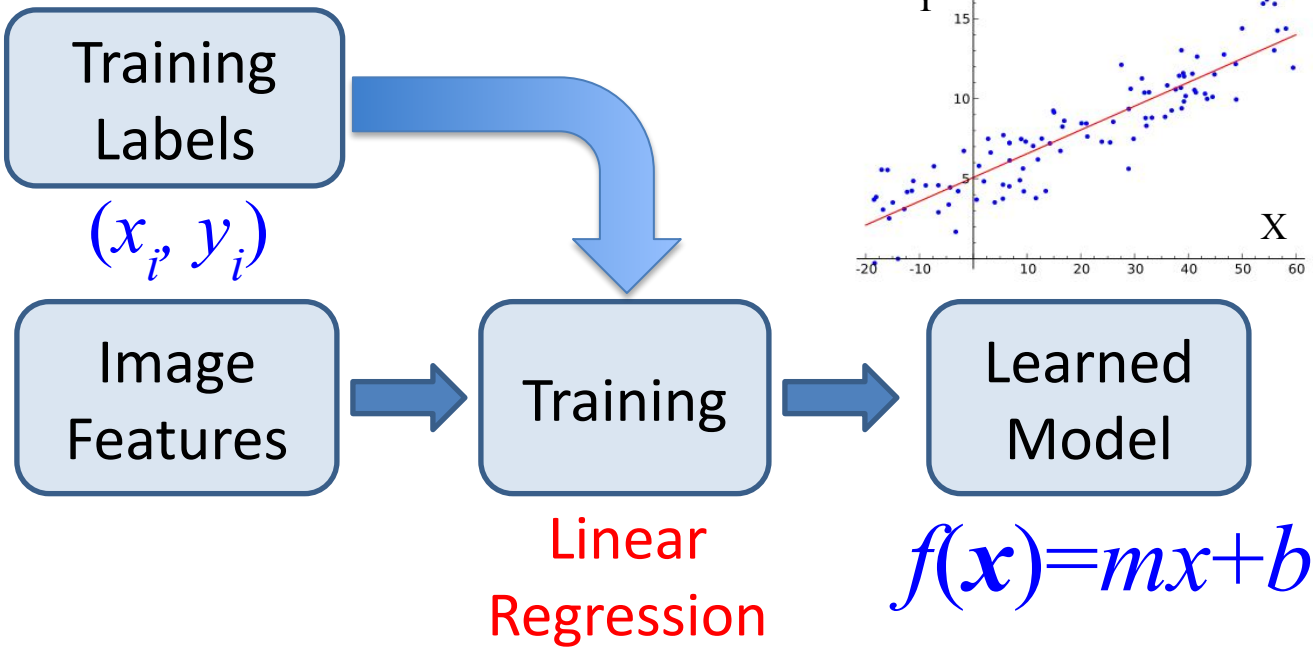
Prediction by Linear Regression

- Find the best-fit function $y = mx + b$ for the data (x, y)
  => training
- When we later get a $x' = 40$ we can predict $y' = 11$
  => testing
  prediction

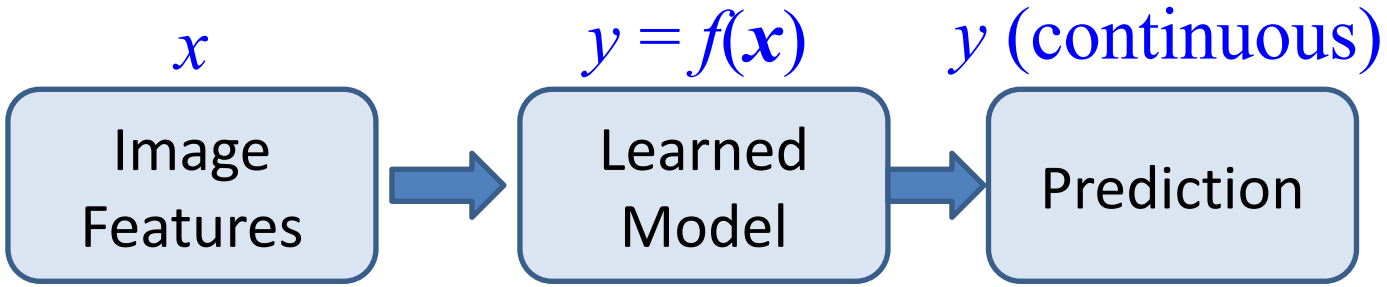


Linear Regression

Training

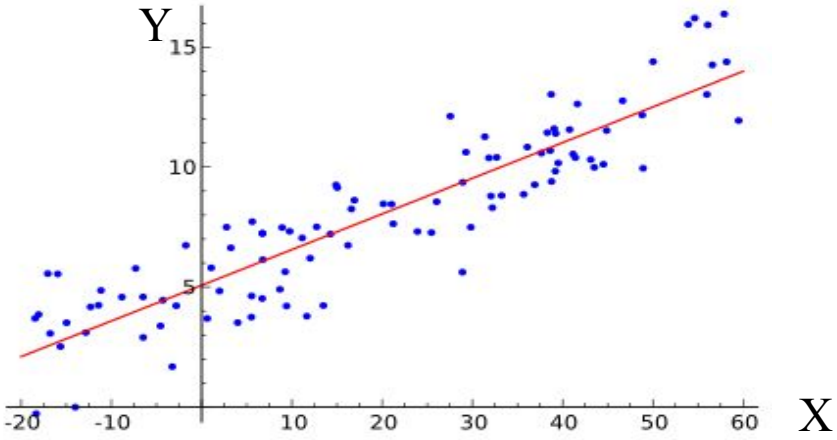


Testing



How to Find the Linear Function

- Concept
  - Best fit
    ⇒ minimum SSE (sum of square error)
- Methods
  - Least square
  - **Gradient descent**

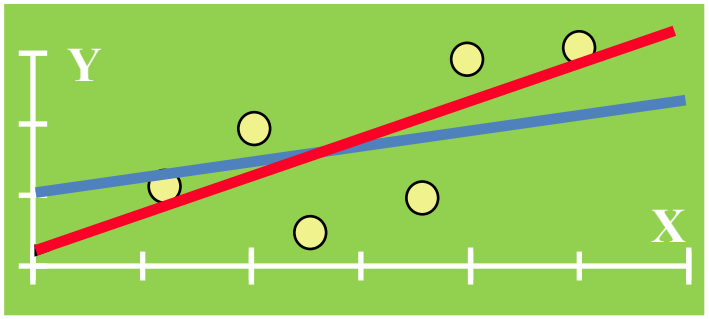


Best fit = minimize SSE

SUM OF SQUARE ERROR (SSE)

Square Errors

- Suppose we have the data set (x_i, y_i)
- "Best Fit" means the difference between actual y_i and predicted $\hat{y}_i$ are a minimum



$$Y = m_1 X + b_1$$
$$Y = m_2 X + b_2$$

$$\hat{y}_i = \hat{m} x_i + \hat{b}$$

Minimum Error

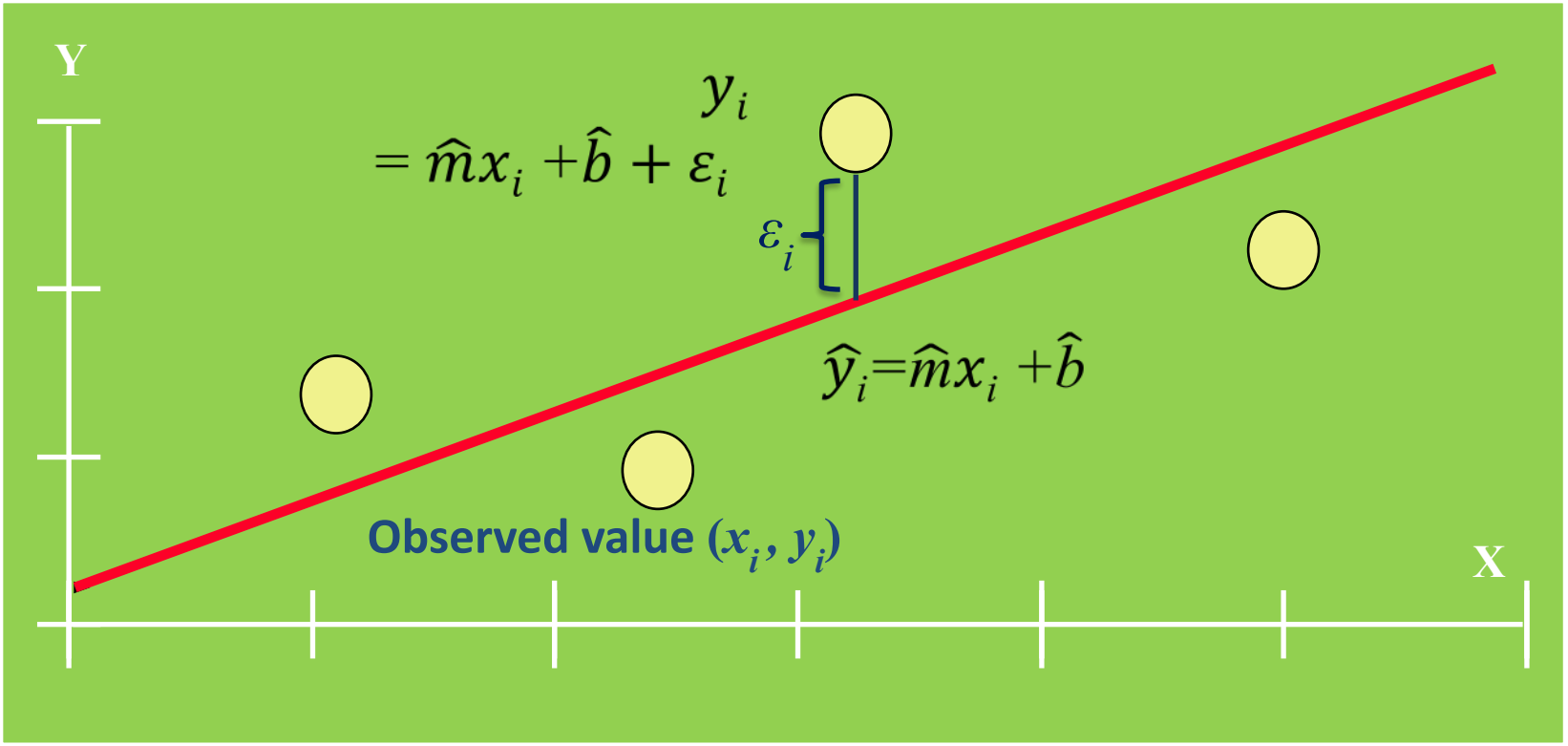
$$\sum_{i=1}^{n} (y_i - \hat{y}_i)$$

- But negative differences decrease the error
- So we need square error

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \varepsilon_i^2$$

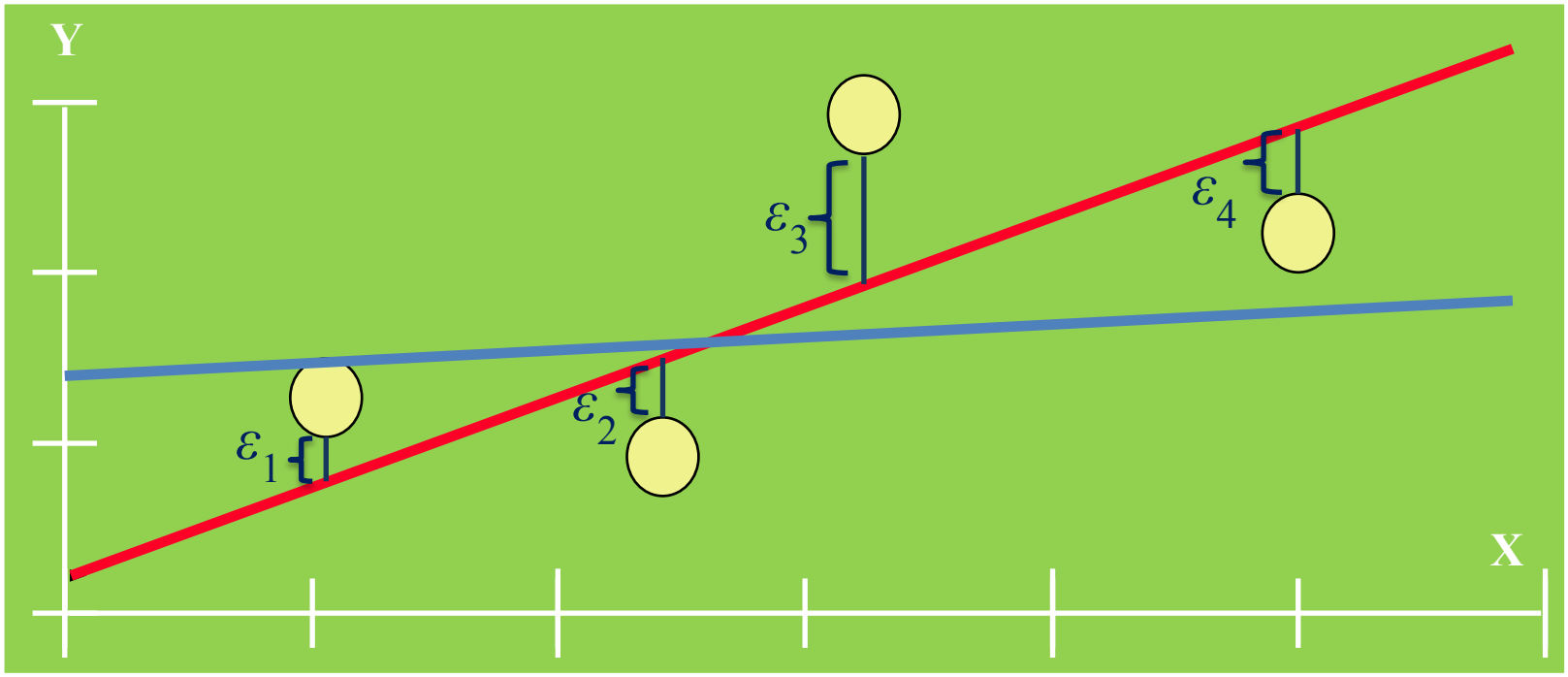
Square Errors Are Random

$Y = mX + b + \varepsilon$ ε is random error



Sum of Square Errors (SSE)

Total square errors: $\sum_{i=1}^{n} \varepsilon_i^2 = \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2$



Minimum SSE (sum of square error) ⇒ Best fit

First method to minimize SSE

LEAST SQUARE

Least Square (LS) Method

- Minimize the SSE (Sum of Squared Errors) by Least Square
- Least square (LS)

$$Q = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{m}x_i - \hat{b})^2$$

$$\frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2x_i(y_i - mx_i - b)$$

$$\frac{\partial Q}{\partial b} = \sum_{i=1}^{n} -2(y_i - mx_i - b)$$

Let $\frac{\partial Q}{\partial m} = 0$ and $\frac{\partial Q}{\partial b} = 0$

Least Square Solution

- Prediction equation $\hat{y} = \hat{m}x + \hat{b}$
- Slope $\hat{m}$ $\hat{m} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$
- Intercept $\hat{b}$ $\hat{b} = \bar{y} - \hat{m}\bar{x}$

Optimum solution

It looks great to find solutions of linear function
But it is good only for one X

How about high dimensions: $(X_1, X_2, \ldots, X_N)$

Chapter 8 in Machine Learning in Action

Chapter 1 in Applied Linear Statistical Models

Multiple Linear Regression

$$Y = w_0 + w_1 X_1 + w_2 X_2 + \cdots + w_N X_N = W^T X$$

$$W = (w_0, w_1, \cdots, w_N)^T, \qquad X = (1, X_1, \cdots, X_N)^T$$

Its LS solution is computationally terrible

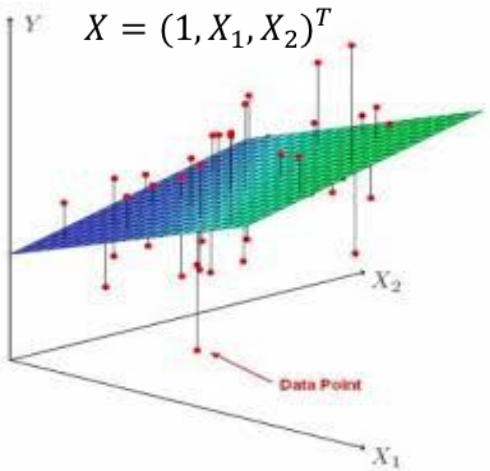
Suppose we have n data points

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \underbrace{X}_{n \times (N+1)} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1N} \\ 1 & x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nN} \end{bmatrix}$$

$$\widehat{W} = (X^T X)^{-1} (X^T X) Y$$

Assume n=1000, N=100, $(X^T X)$ is a 101x101 matrix

Example: N=2



So we need gradient descent method: computationally efficient

Chapter 6 in Applied Linear Statistical Models

Second method to minimize SSE; It is very popular and important

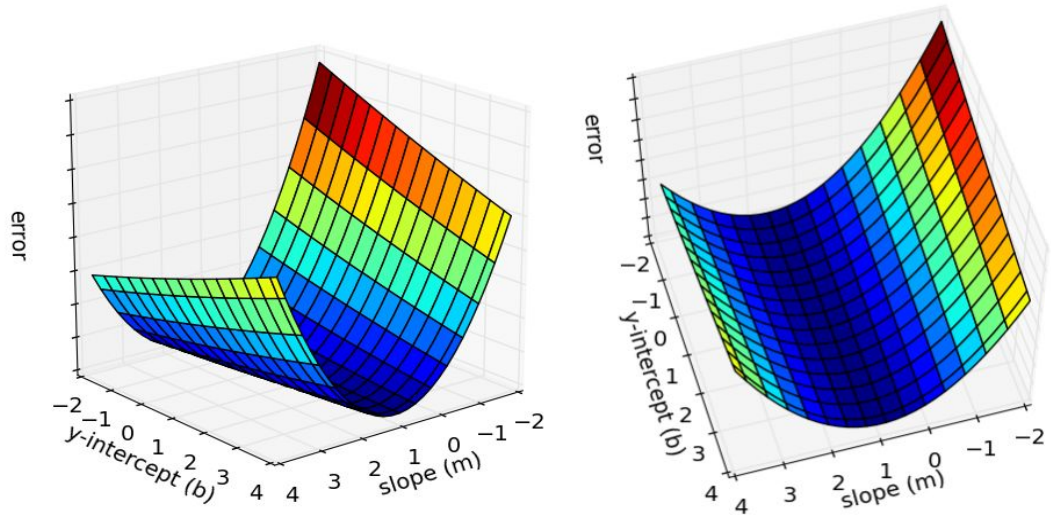
GRADIENT DESCENT

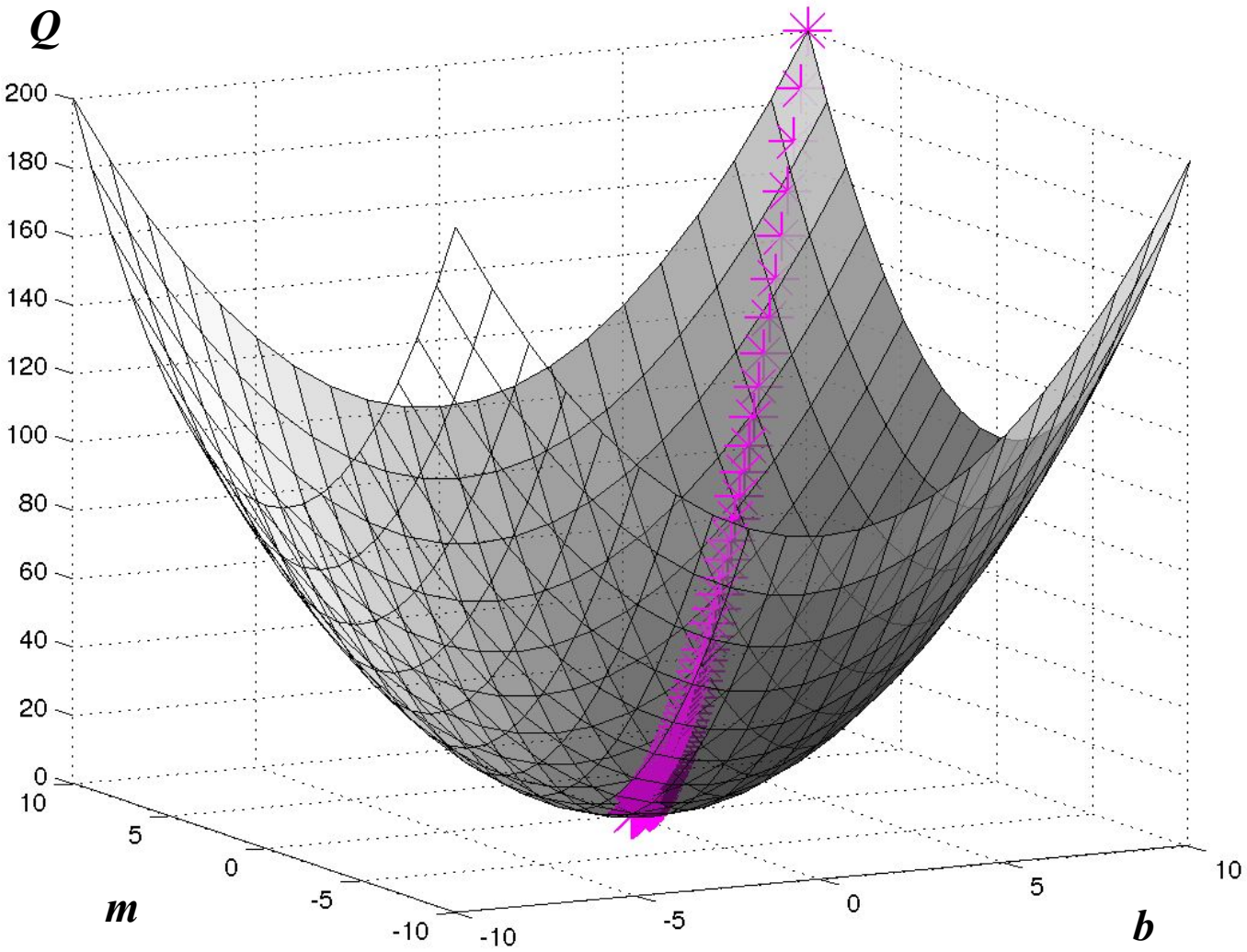
Gradient Descent

- It is a very important set of iterative algorithms
- It uses **gradient** updates to find solution

$$Q = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{m}x_i - \hat{b})^2$$

$$\frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2x_i(y_i - mx_i - b) \qquad \frac{\partial Q}{\partial b} = \sum_{i=1}^{n} -2(y_i - mx_i - b)$$





$$Q = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{m}x_i - \hat{b})^2$$

Iterated Update of Parameters by Gradients

Compute gradients of the model parameters m, b: $\nabla m, \nabla b$

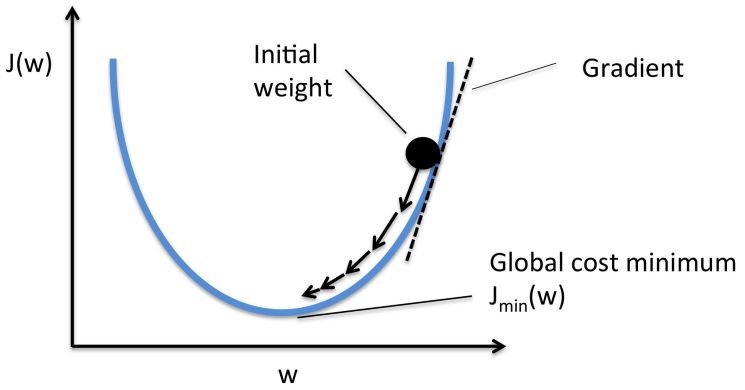
Add the gradients to the model parameters m, b

$$m' = m + \alpha \nabla m$$

$$b' = b + \alpha \nabla b$$

$$\nabla m = \frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2x_i(y_i - mx_i - b)$$

$$\nabla b = \frac{\partial Q}{\partial b} = \sum_{i=1}^{n} -2\,(y_i - mx_i - b)$$



Where α is called the **learning rate**.

These updates happen **many times** in one pass over the dataset.

Source: "Chapter 6 Multiple Regression I", Applied Linear Statistical Models 5th, M. Kutner, C. Nachtsheim, J. Neter, W. Li, McGraw-Hill, 2005

Pseudo Code of Gradient Descent

```
def  linear_regression(X, y, m_current=0, b_current=0,
                       epochs=1000, learning_rate=0.0001):
    n = float(len(y))
    for i in range(epochs):
        y_current = (m_current * X) + b_current
        cost = sum([data**2 for data in (y-y_current)]) / n
        m_gradient = -(2/N) * sum(X * (y - y_current))
        b_gradient = -(2/N) * sum(y - y_current)
        m_current = m_current + (learning_rate * m_gradient)
        b_current = b_current + (learning_rate * b_gradient)
    return  m_current, b_current, cost
```

[Linear Regression Using Gradient Descent in 10 Lines of Code]

$\hat{y}_i = \hat{m}x_i + \hat{b}$

$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

$\nabla m = \frac{\partial Q}{\partial m} = \sum_{i=1}^{n} -2x_i(y_i - mx_i - b)$

$\nabla b = \frac{\partial Q}{\partial b} = \sum_{i=1}^{n} -2\,(y_i - mx_i - b)$

$m' = m + \alpha\,\nabla m$

$b' = b + \alpha\,\nabla b$

OVERFITTING & MODEL SELECTION

An Example

1. $y = b + w \cdot x$

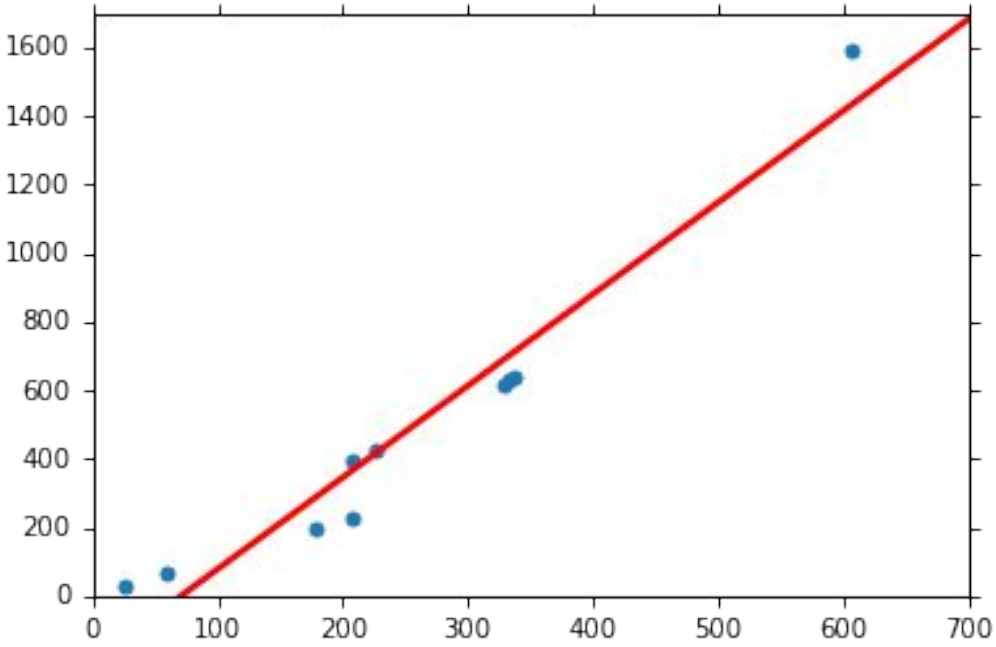
2. $y = b + w_1 \cdot x + w_2 \cdot x^2$

3. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$

4. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$

5. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$

Training Data



5 models for the training data
Which model is the best one?

Source: Machine Learning,李宏毅,台大電機,2016.

An Example

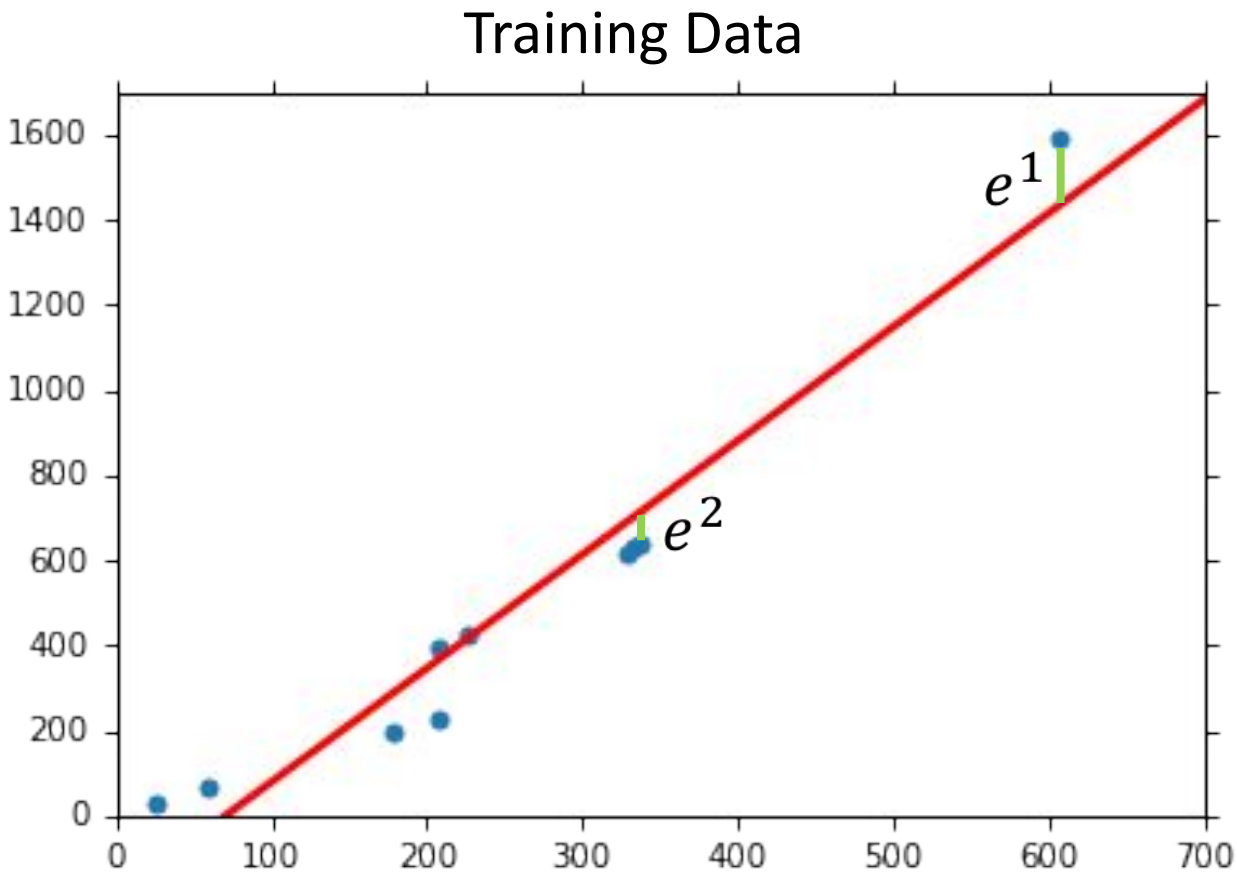
$y = b + w \cdot x$

b = -188.4

w = 2.7

Average Error on Training Data

$$= \sum_{n=1}^{10} e^n = 31.9$$



Source: Machine Learning,李宏毅,台大電機,2016.

How's the Result? - Generalization

$y = b + w \cdot x$

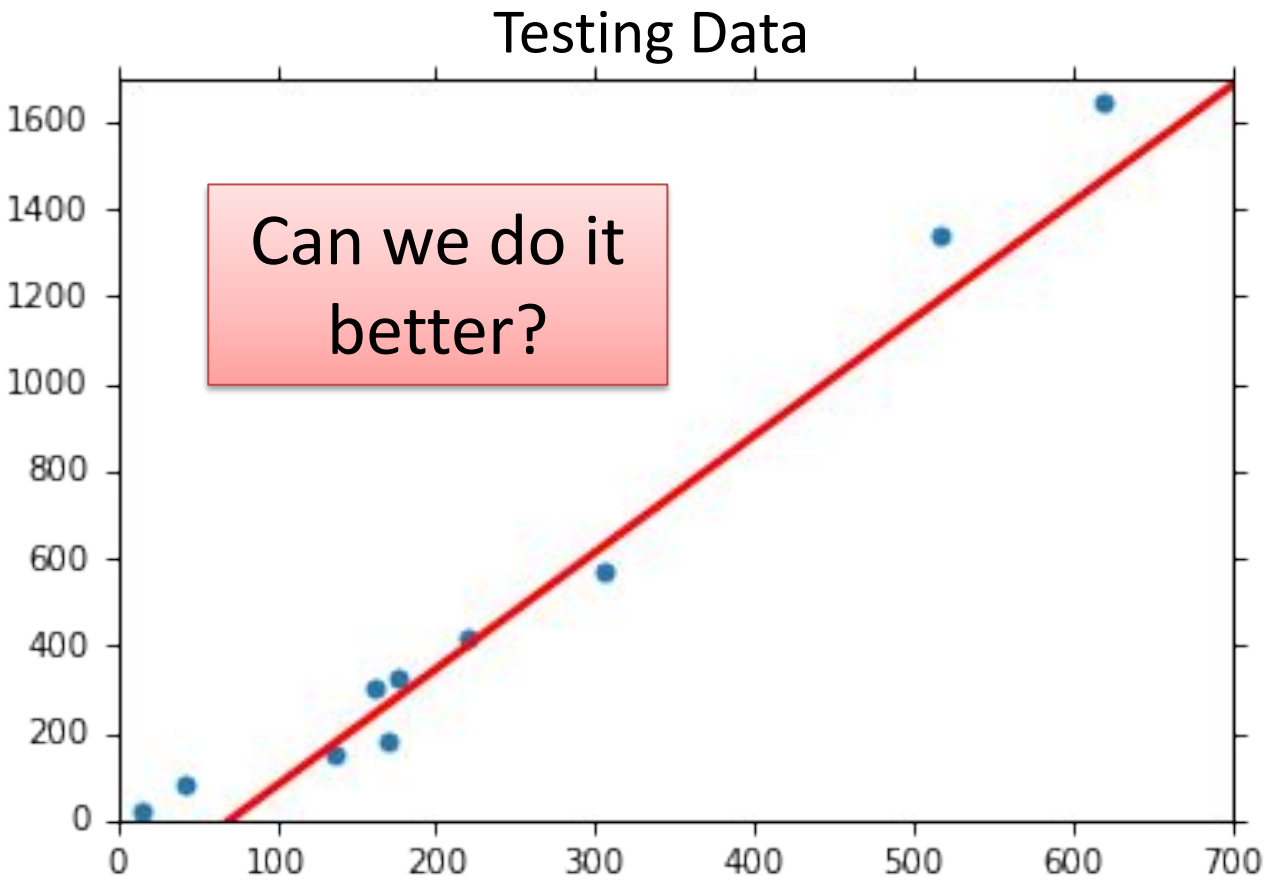
b = -188.4

w = 2.7

Average Error on Testing Data

$$= \sum_{n=1}^{10} e^n = 35.0$$

> Average Error on Training Data



Source: Machine Learning,李宏毅,台大電機,2016.

Selecting another Model

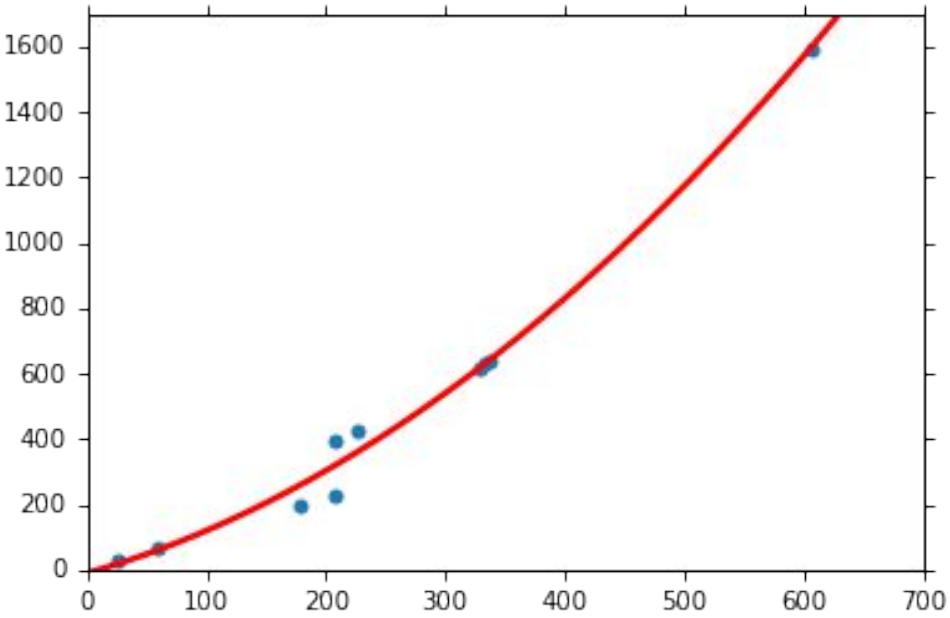
$$y = b + w_1 \cdot x + w_2 \cdot x^2$$

Best Function

b = -10.3

w_1 = 1.0, w_2 = 2.7 x 10^{-3}

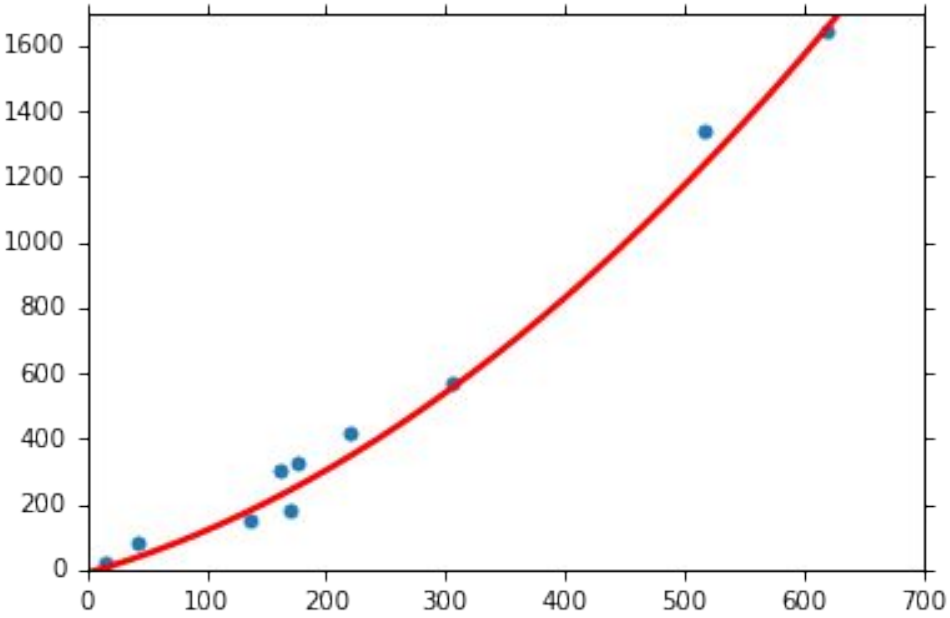
Average Training Error = 15.4



Testing:

Average Test Error = 18.4

Better! Could it be even better?



Source: Machine Learning,李宏毅,台大電機,2016.

Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot x^2 + w_3 \cdot x^3$$

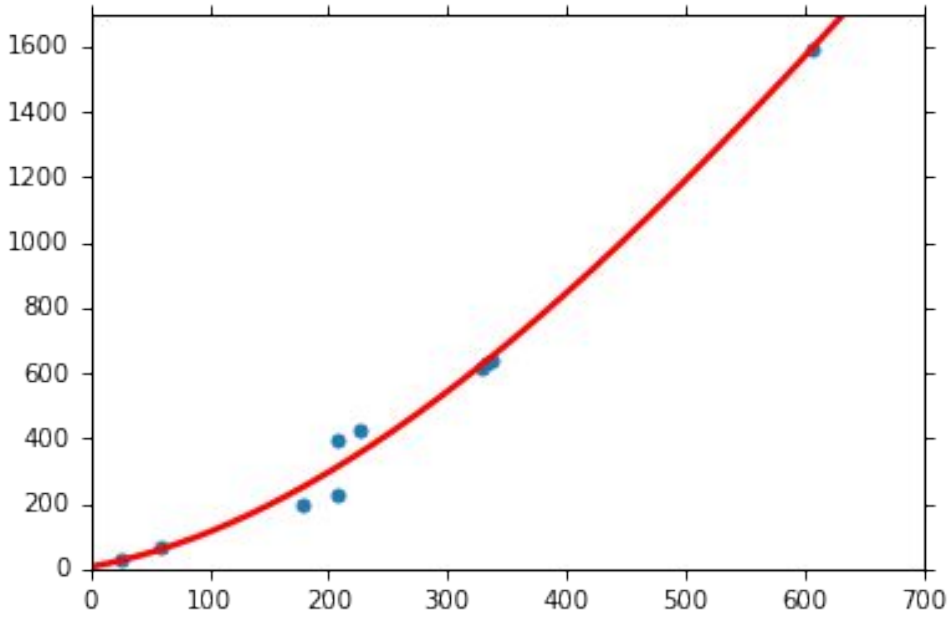
Best Function

$b = 6.4$, $w_1 = 0.66$

$w_2 = 4.3 \times 10^{-3}$

$w_3 = -1.8 \times 10^{-6}$

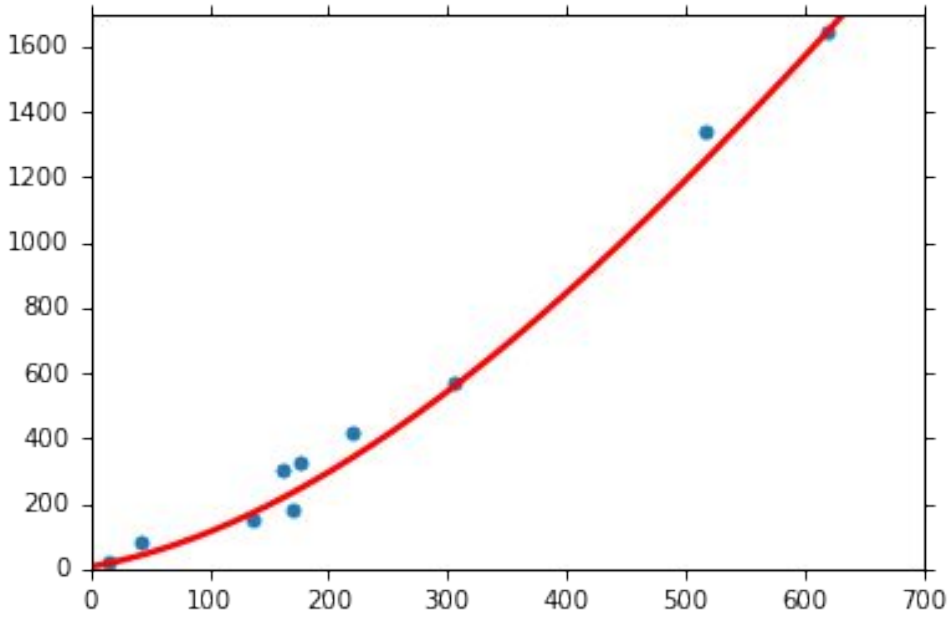
Average Error = 15.3



Testing:

Average Error = 18.1

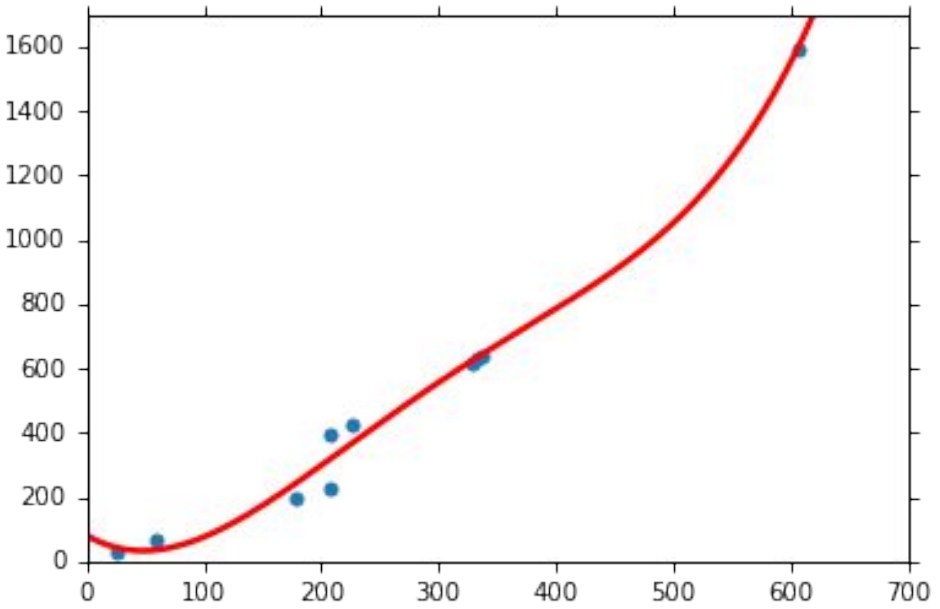
Slightly better.
How about more complex model?



Source: Machine Learning,李宏毅,台大電機,2016.

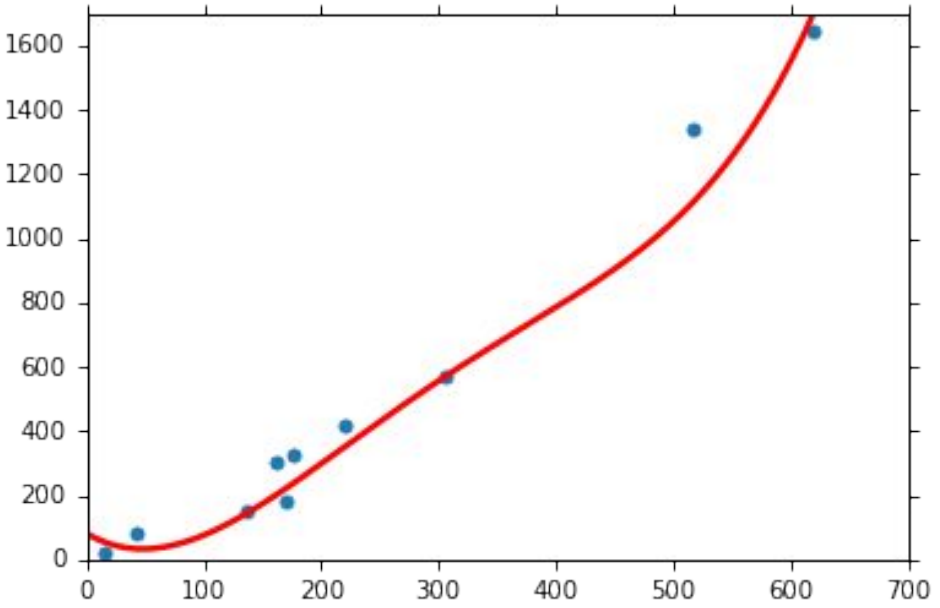
Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$$



Best Function

Average Error = 14.9



Testing:

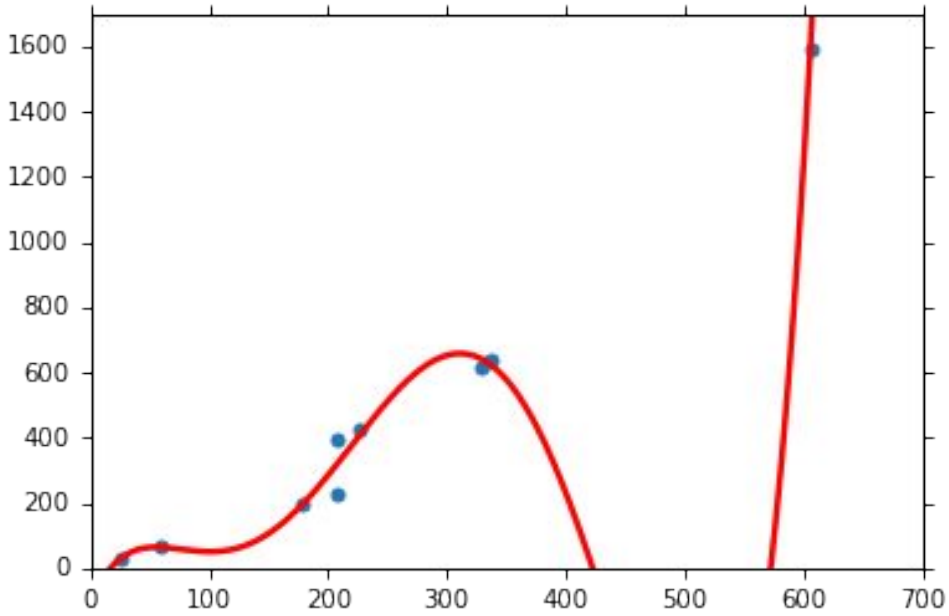
Average Error = 28.8

The results become worse ...

Source: Machine Learning,李宏毅,台大電機,2016.

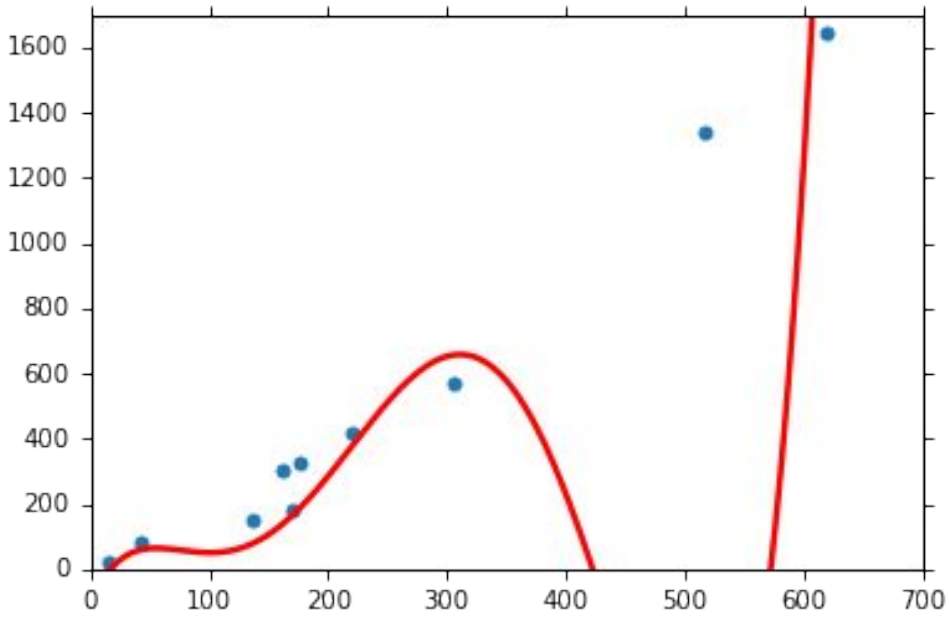
Selecting another Model

$$y = b + w_1 \cdot x_{cp} + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$$



Best Function

Average Error = 12.8



Testing:

Average Error = 232.1

The results are so bad.

Source: Machine Learning,李宏毅,台大電機,2016.

Model Selection

1. $y = b + w \cdot x$

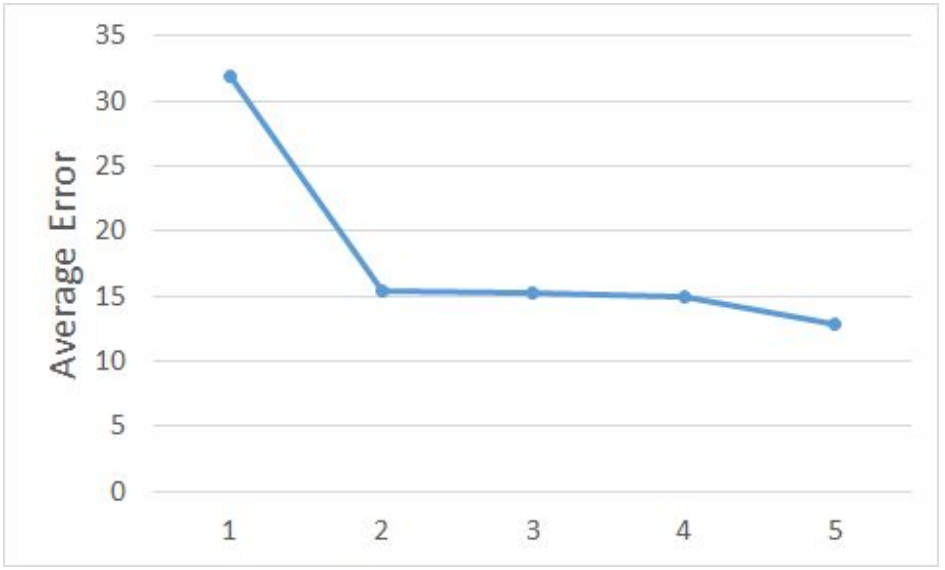
2. $y = b + w_1 \cdot x + w_2 \cdot x^2$

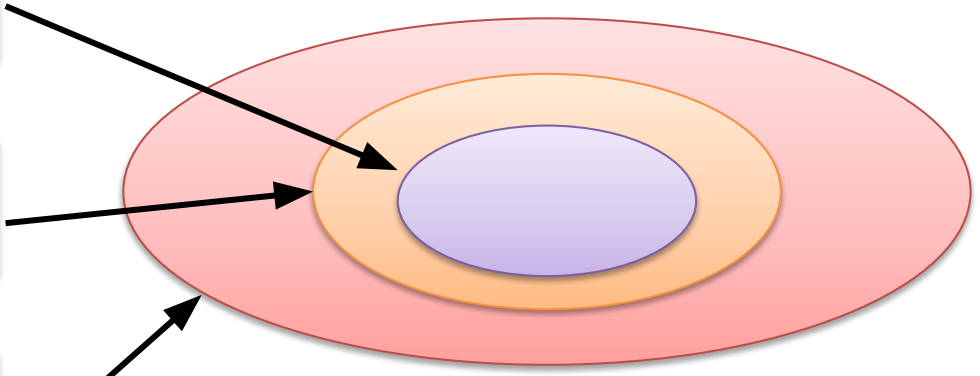
3. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3$

4. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4$

5. $y = b + w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot x^3 + w_4 \cdot x^4 + w_5 \cdot x^5$

Training Data



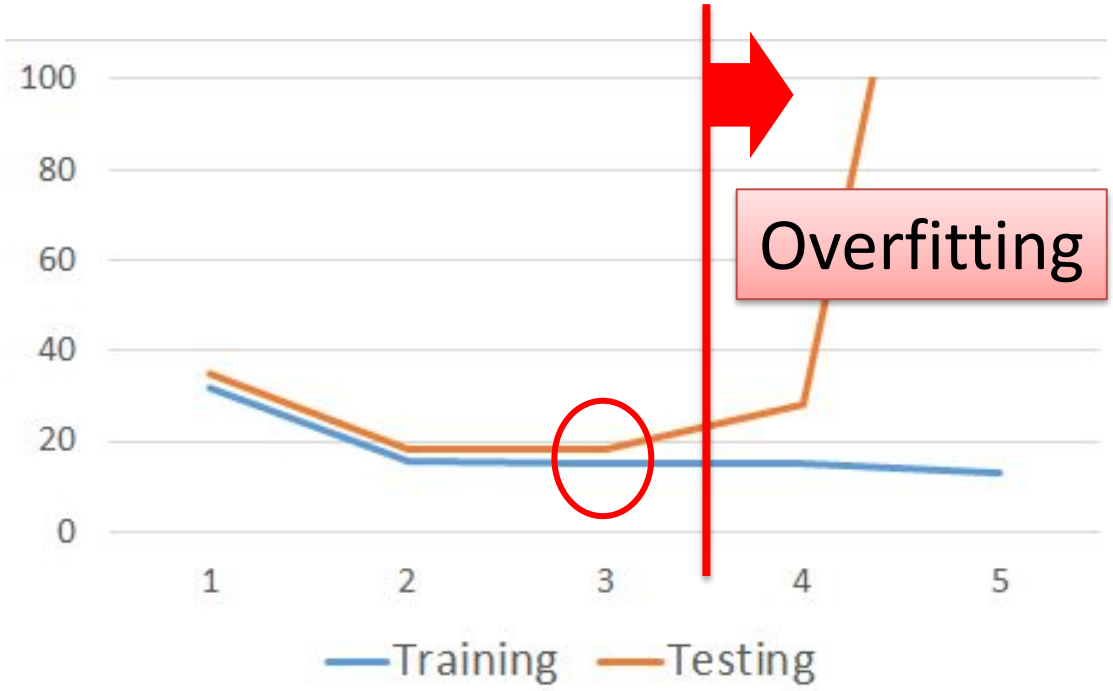


A more complex model yields lower error on training data.

If we can truly find the best function

Source: Machine Learning,李宏毅,台大電機,2016.

Model Selection



	Training	Testing
1	31.9	35.0
2	15.4	18.4
3	15.3	18.1
4	14.9	28.2
5	12.8	232.1

A more complex model does not always lead to better performance on ***testing data***

This is ***Overfitting*** → Select suitable model

Source: Machine Learning,李宏毅,台大電機,2016.

Conclusion

Linear Regression

- Model the linear relations between X and Y
  - SSE (Sum of Square Error)
- Two algorithms to find the linear relation
  - Least square (LS) : matrix operation
  - Gradient descent (GD) : iterative procedure

Overfitting

- **Least training error ≠ least test error**
- Overfitting:
  Higher-order nonlinear regression may cause higher test error
- Model selection:
  Select the best order of nonlinear regression with least test error