

Linear Regression Gradient Descent

อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

- Recap – kNN, MNIST
- Linear Regression
- Gradient Descent
- Polynomial Regression
- Regularization

Recap: Supervised Learning

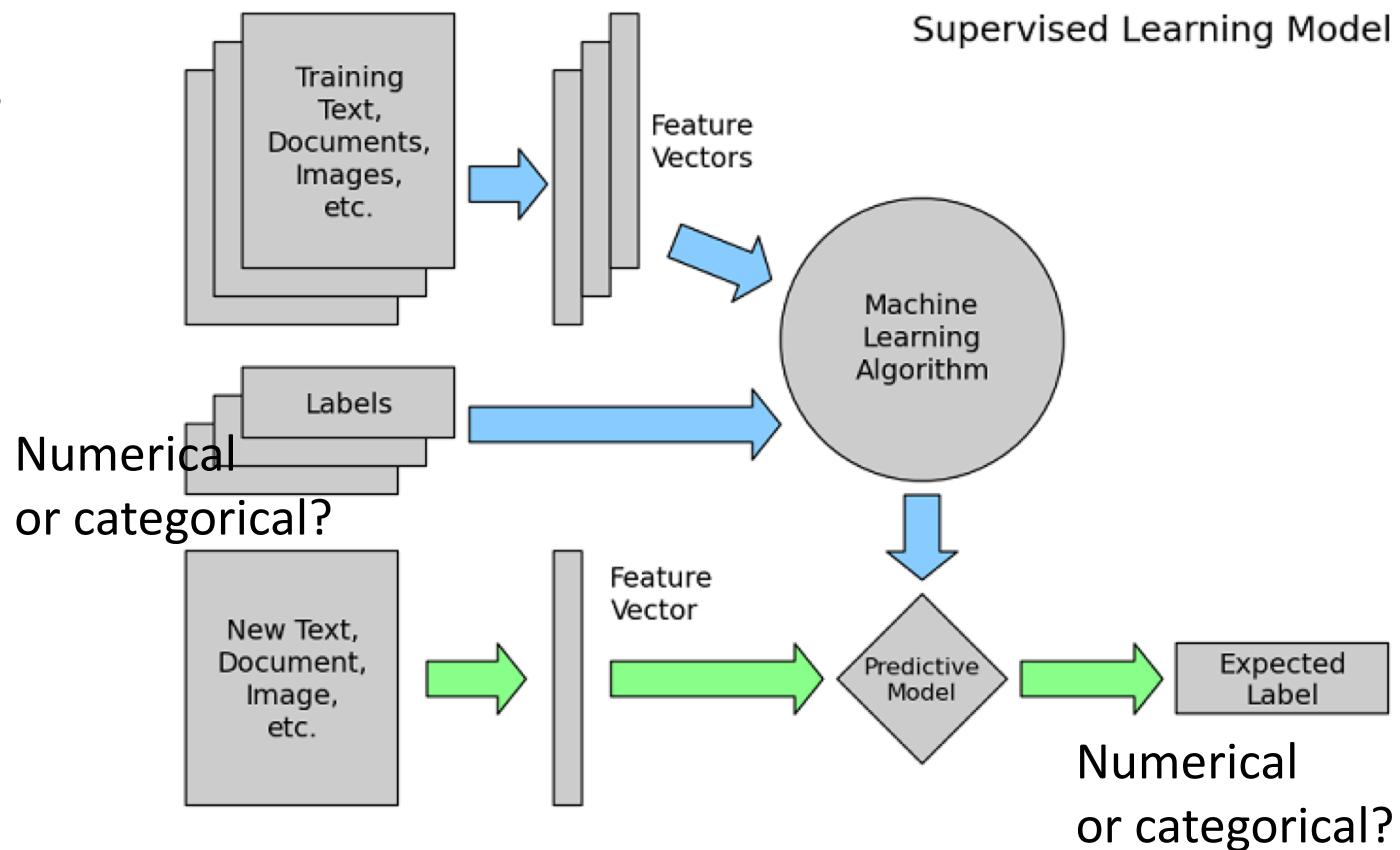
- Classification

kNN

- Predicts class labels/categories
- ทำนายค่าที่เป็นหมวดหมู่ = จำแนกประเภท
- อาจมองเป็นการหา **boundary** ที่แบ่งข้อมูลในแต่ละหมวดหมู่ ออกจากกัน

- Regression

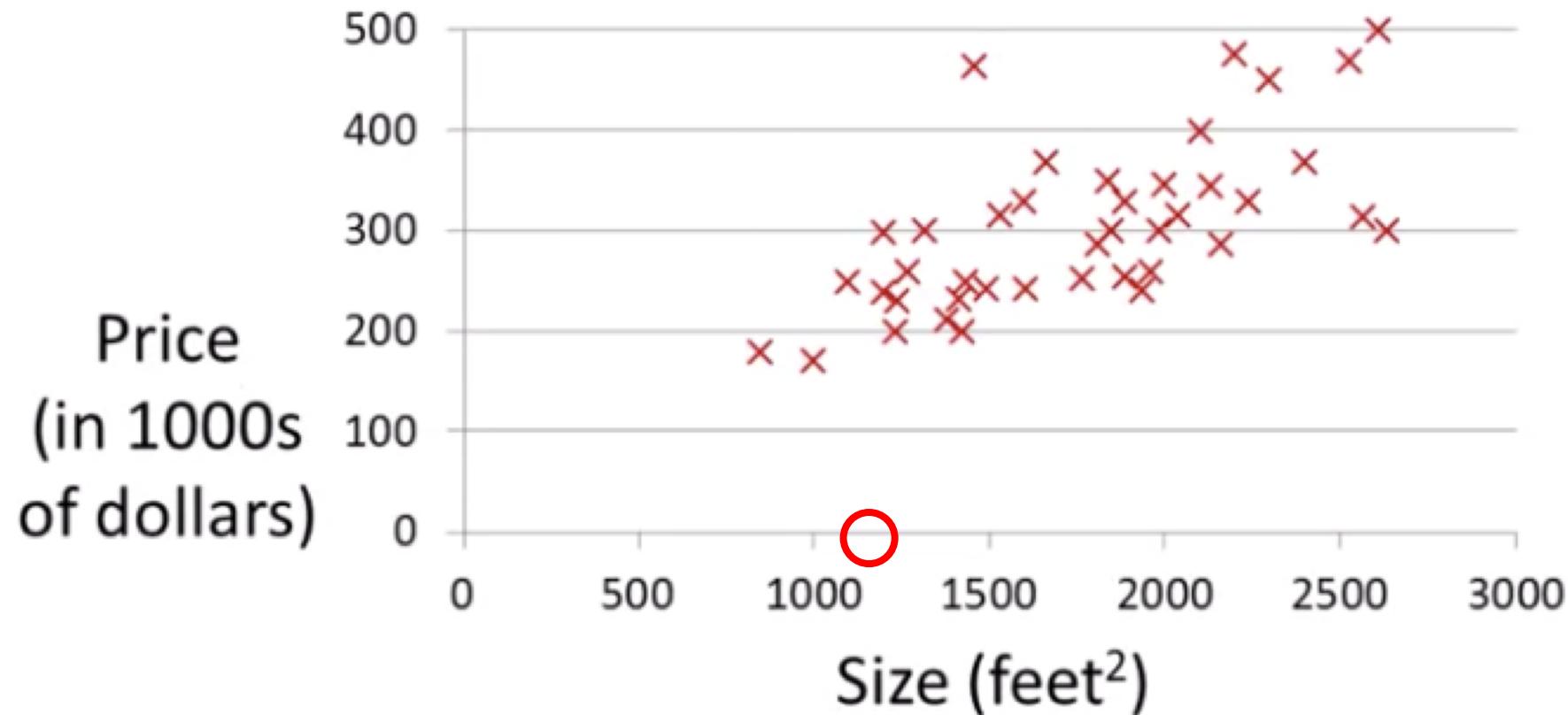
- Linear Regression
- Predicts continuous values
 - ทำนายค่าที่เป็นจำนวนจริง
 - อาจมองเป็นการหา **hyperplane** ที่ fit กับข้อมูลที่มีมากที่สุด



Linear Regression

- เป็นอัลกอริทึมแบบ
 - supervised
 - parametric
 - ใช้สำหรับทำ regression
 - มีผู้ช่วย (ใช้ข้อมูล train ที่มี label เฉลย ในการหา $h: X \rightarrow Y$)
 - มี assumption ว่าฟังก์ชัน $h: X \rightarrow Y$ เป็นแบบ Linear
 - มีตัวแปร (parameter) ในโมเดลของ h
 - ผลทำนายเป็นเลขจำนวนจริง
- Example
 - โมเดล: $\text{life_satisfaction} = h(\theta) = \theta_0 + \theta_1 \times GDP$

Example: Housing Prices



Q: บ้านที่มีพื้นที่ 1200 Sq. ft. จะขายได้ในราคาเท่าใด?

Example: Housing Prices

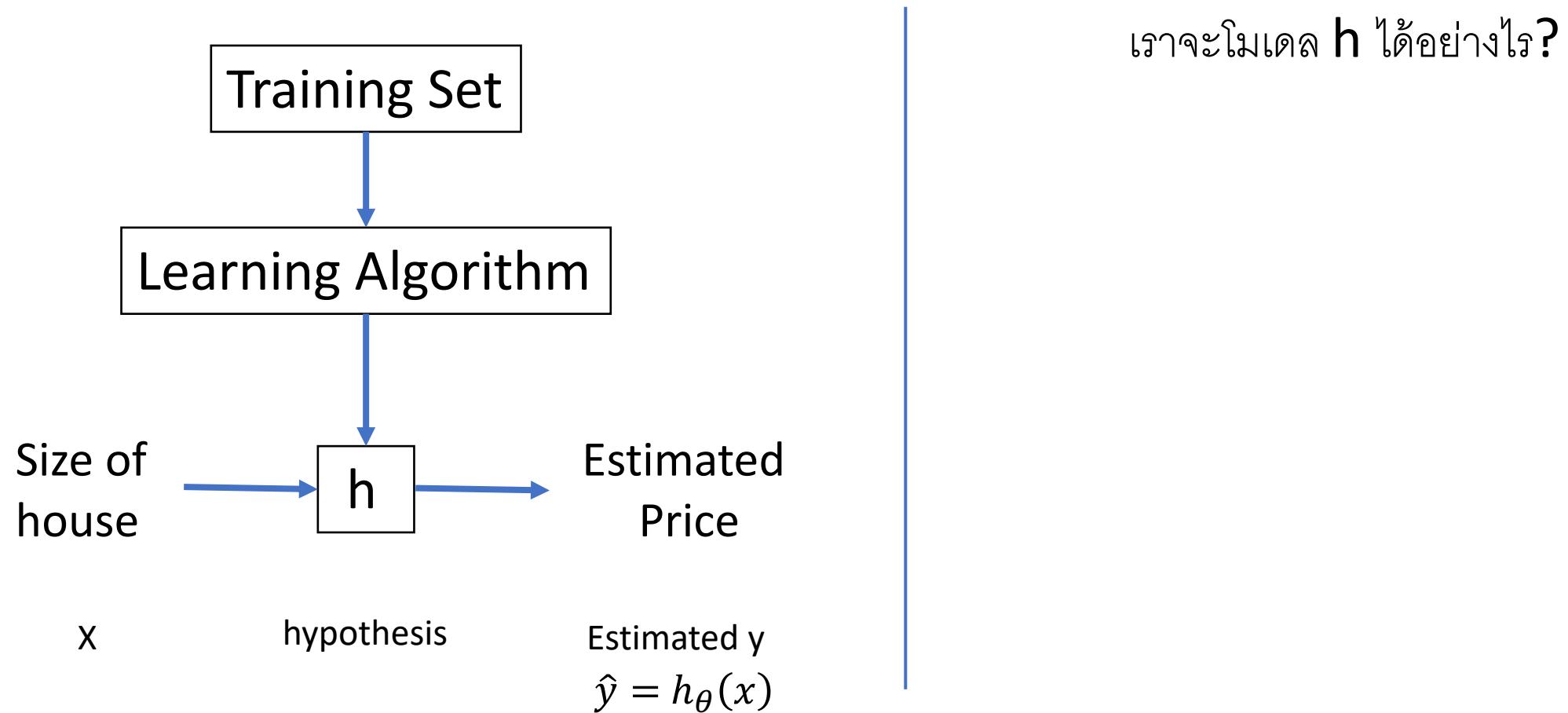
Training Set:

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

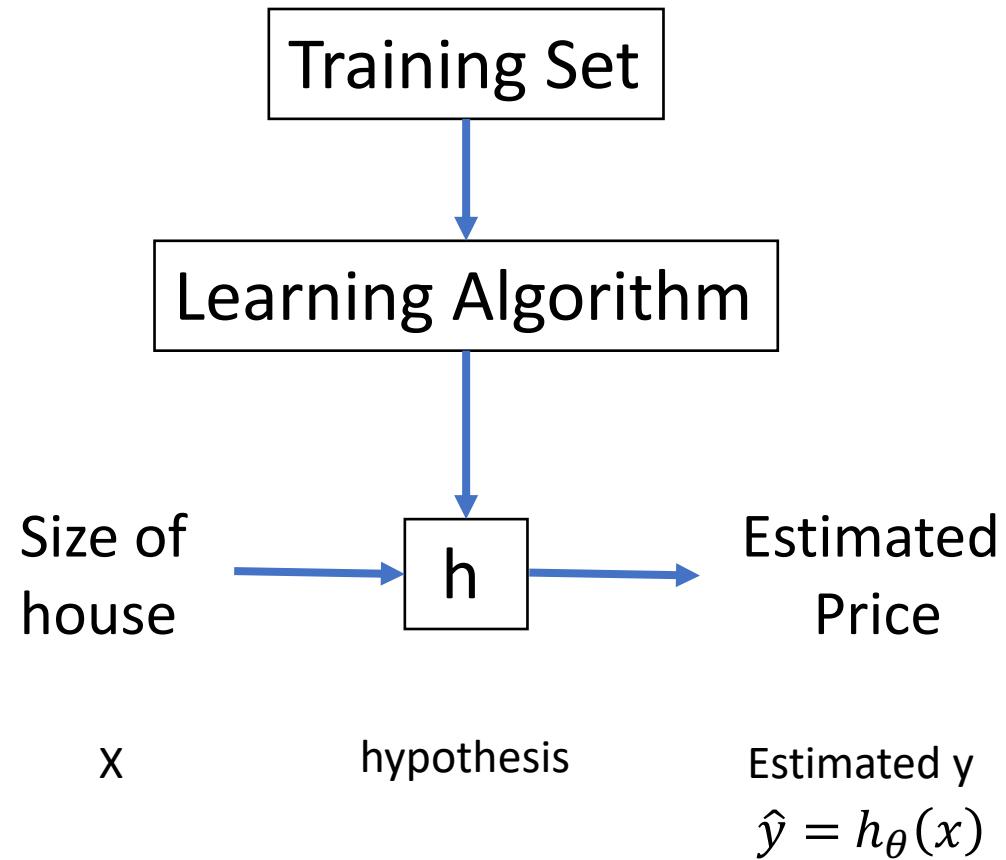
นิยามสัญลักษณ์:

- M = จำนวนข้อมูลชุด training
- x = เวกเตอร์ของ feature
- y = เวกเตอร์ของค่า target
- $(x^{(i)}, y^{(i)})$ = ข้อมูล training ชั้นที่ i

Example: Housing Prices

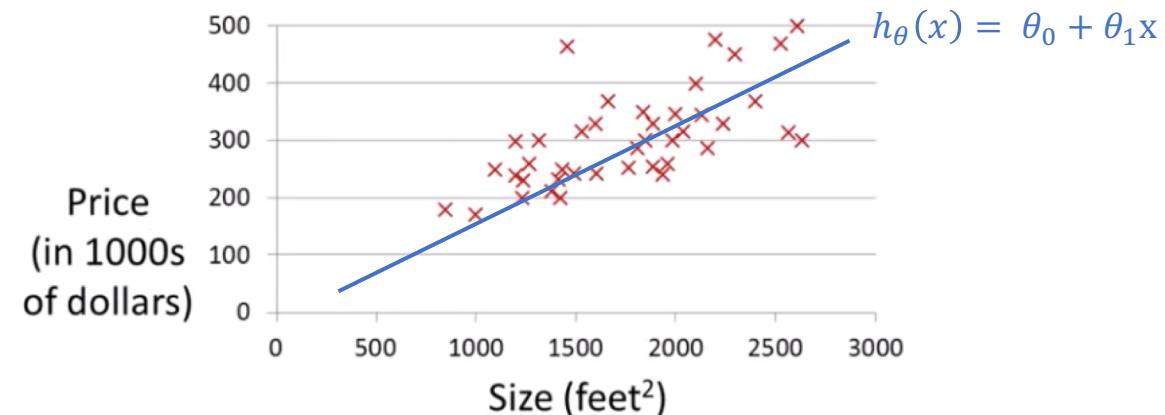


Example: Housing Prices



เราจะโมเดล h ได้อย่างไร?

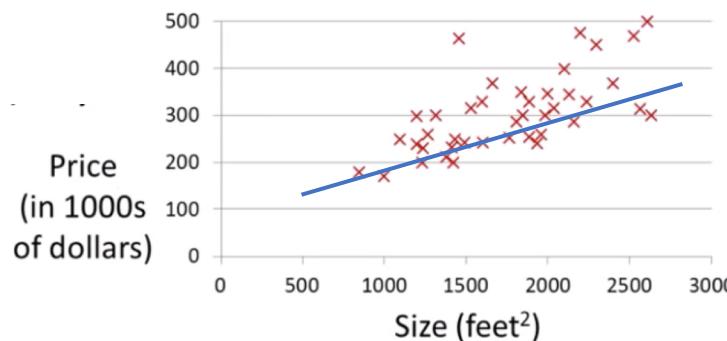
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



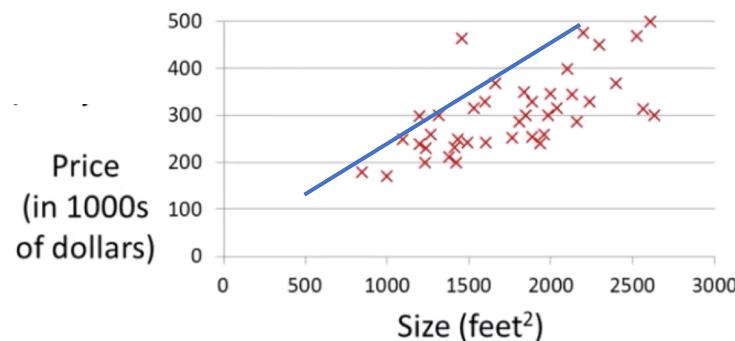
Linear Regression Model (1 ตัวแปร)

Linear Regression Model

- โมเดลการทำนาย: $\hat{y} = h_{\theta}(x) = \theta_0 + \theta_1 x$
 - * θ_i คือ “พารามิเตอร์” (parameter) ของโมเดล => เป็นค่าคงที่ใดๆ
- เราจะเลือกค่า θ_i ที่ดีที่สุดได้อย่างไร?
- กล่าวคือ เราสามารถหาโมเดลที่ fit ข้อมูลได้ดีที่สุดด้วย คณิตศาสตร์ ได้อย่างไร?

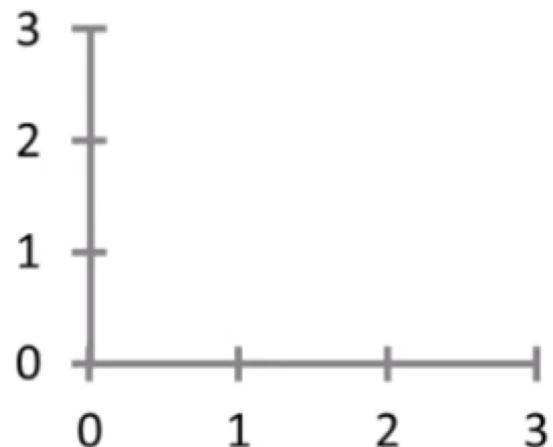


?

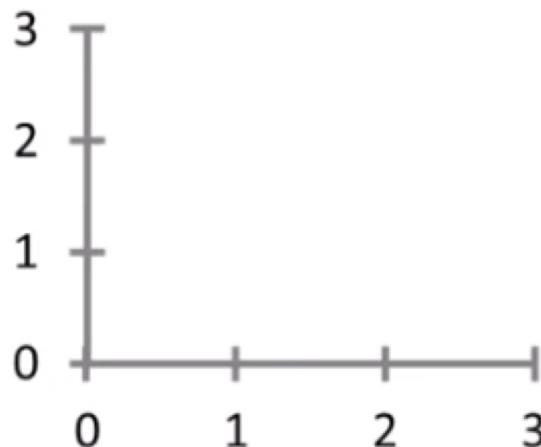


Linear Regression Model

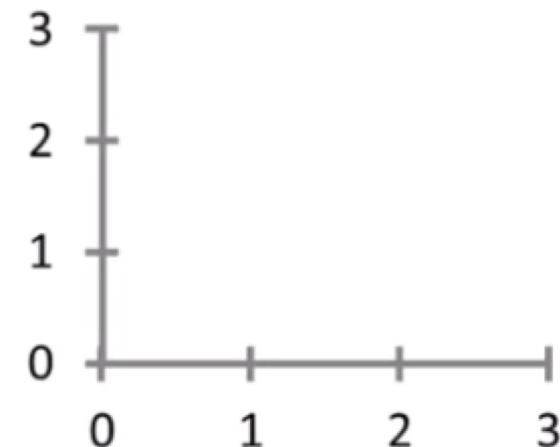
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$



$$\begin{aligned}\theta_0 &= 1.5 \\ \theta_1 &= 0\end{aligned}$$



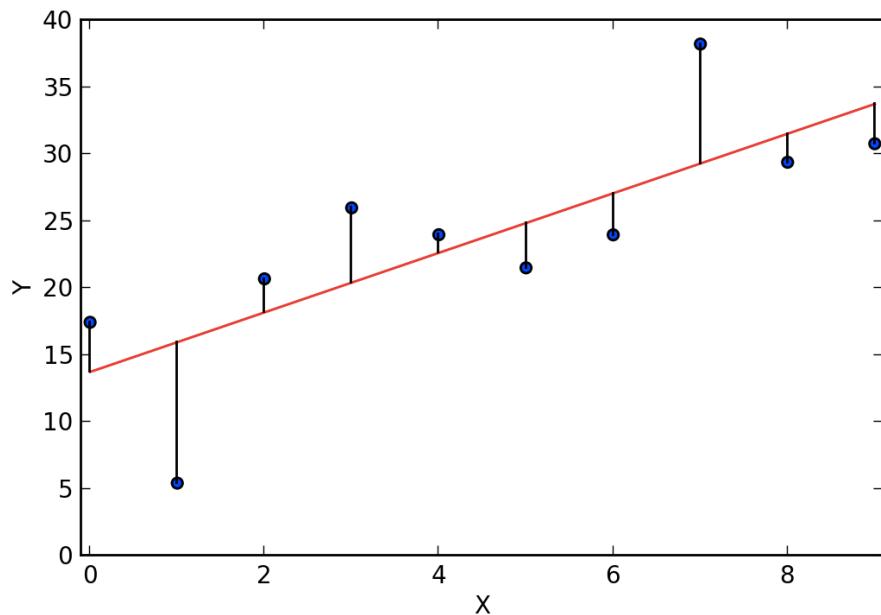
$$\begin{aligned}\theta_0 &= 0 \\ \theta_1 &= 0.5\end{aligned}$$



$$\begin{aligned}\theta_0 &= 1 \\ \theta_1 &= 0.5\end{aligned}$$

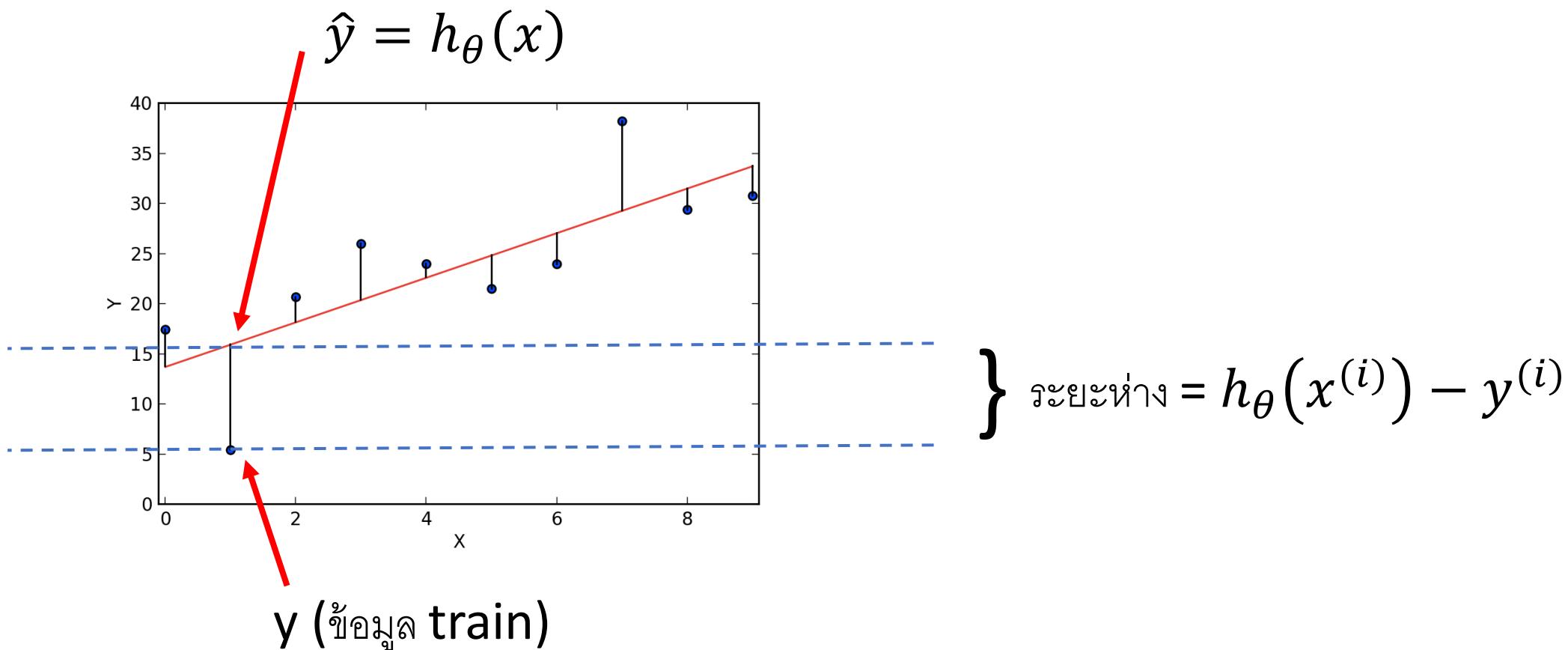
Linear Regression Model

- Idea: เราจะเลือก θ_0, θ_1 ที่ทำให้ $h_\theta(x)$ มีค่าใกล้กับ y ในข้อมูลชุด training ที่สุด



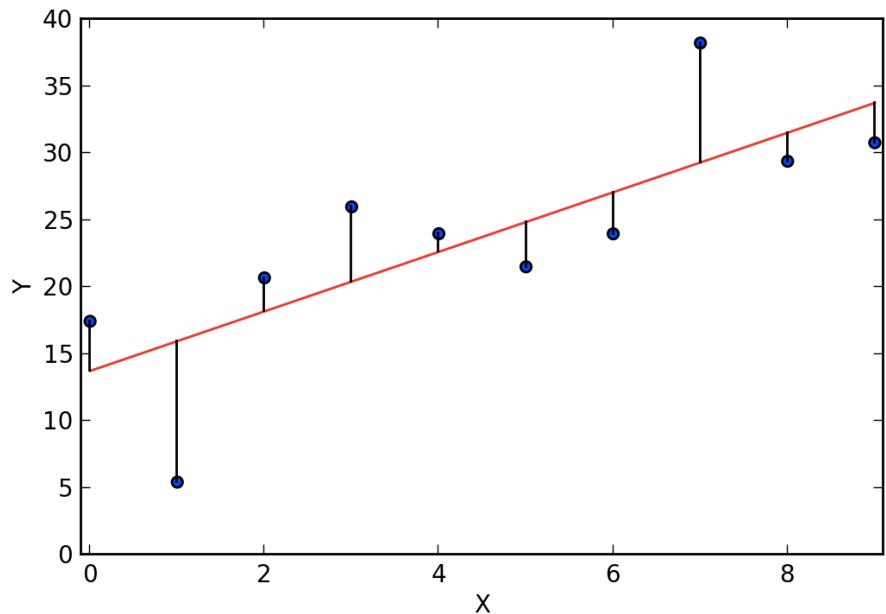
Linear Regression Model

- Idea: เราจะเลือก θ_0, θ_1 ที่ทำให้ $h_\theta(x)$ มีค่าใกล้กับ y ในข้อมูลชุด training ที่สุด



Linear Regression Model

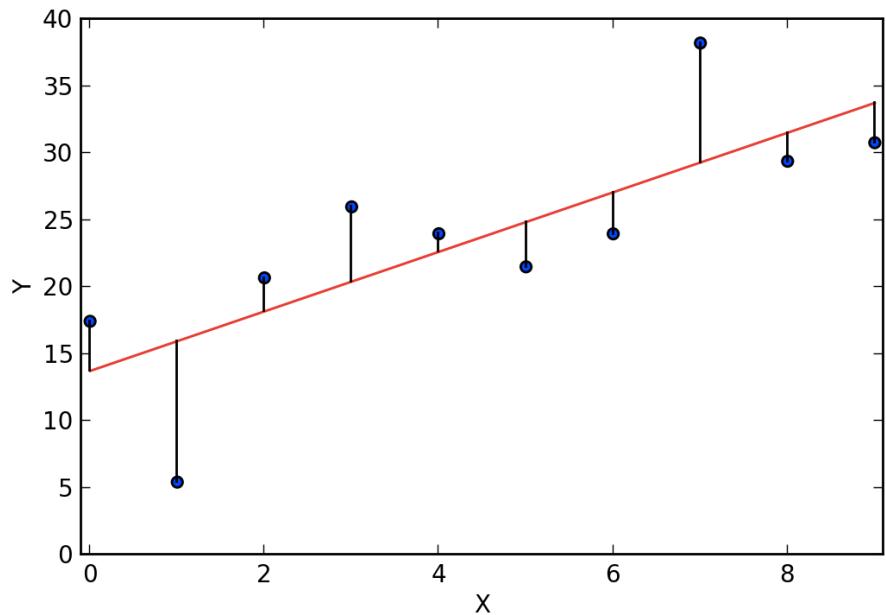
- Idea: เราจะเลือก θ_0, θ_1 ที่ทำให้ $h_\theta(x)$ มีค่าใกล้กับ y ในข้อมูลชุด training ที่สุด



ระยะห่างกำลังสอง
Minimize $\sum_{i=1}^M (h_\theta(x^{(i)}) - y^{(i)})^2$
sum (ผลรวม)
ทุกๆ จุดข้อมูล

Cost Function

- Idea: เราจะเลือก θ_0, θ_1 ที่ทำให้ $h_\theta(x)$ มีค่าใกล้กับ y ในข้อมูลชุด training ที่สุด



“Cost Function”

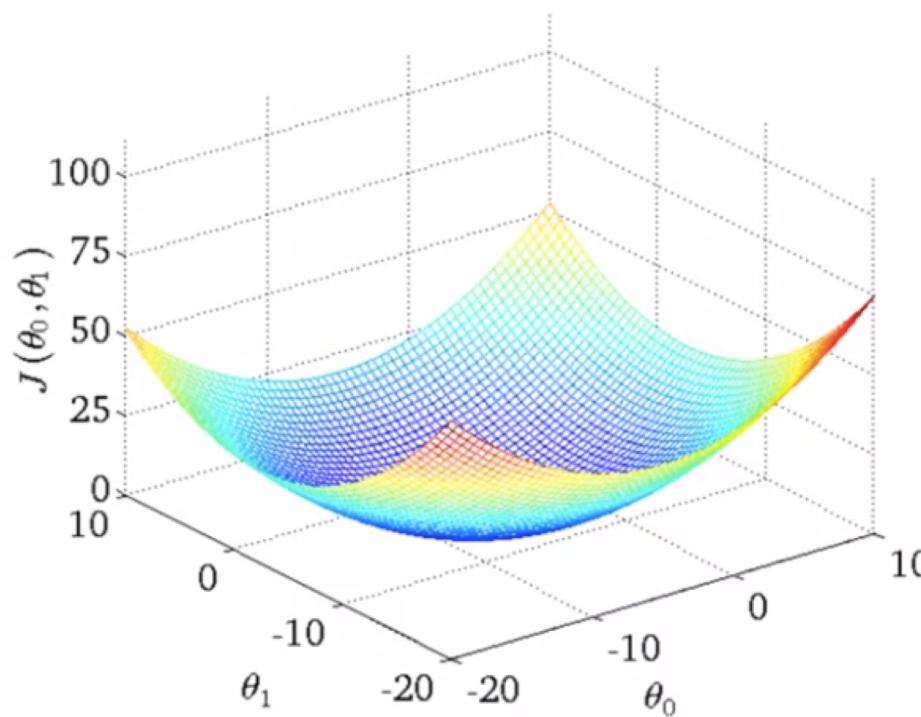
$$J(\theta_0, \theta_1) = \frac{1}{M} \sum_{i=1}^M (h_\theta(x^{(i)}) - y^{(i)})^2$$

เป้าหมาย: $\underset{\theta_0, \theta_1}{\text{Minimize}} J(\theta_0, \theta_1)$

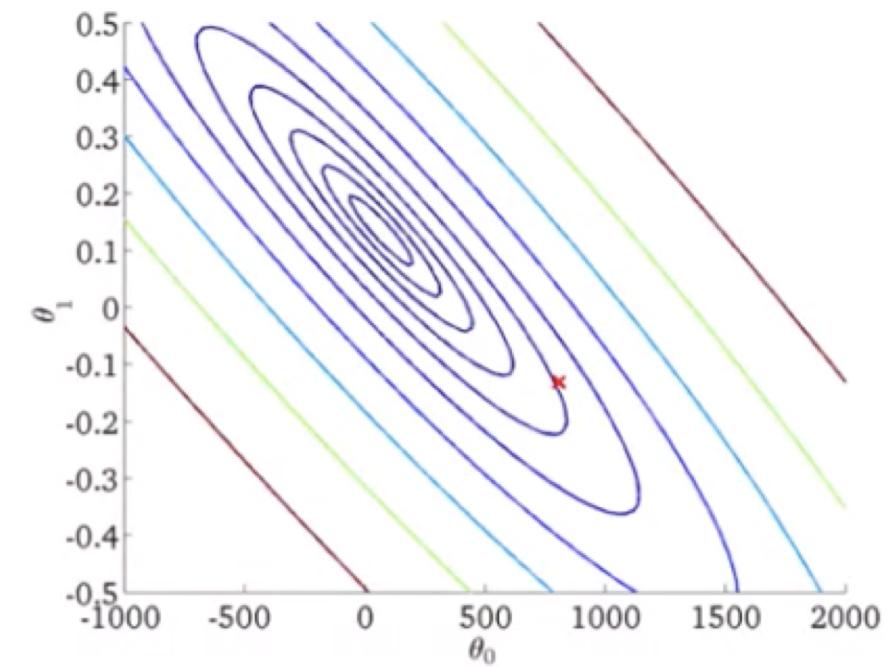
หาค่า θ_0, θ_1 ที่ทำให้ J น้อยที่สุด

Cost Function - Visualized

- ในกรณี 2 พารามิเตอร์ θ_0, θ_1
- หาก plot $J(\theta_0, \theta_1)$ จะได้กราฟ 3 มิติดังรูป



หรือแสดงเป็น **contour plot** จะได้ดังรูป



Multivariate Case (กรณีหลายตัวแปร)

- สังเกตว่าที่ผ่านมาเรามีแค่ 1 ตัวแปร/**feature** คือ x = ขนาดบ้าน (Sq. ft.)
- แต่จริงๆ ข้อมูลชุด **train** อาจมีมากกว่า 1 feature ที่สนใจได้ เช่น
 - x_1 = ขนาดบ้าน (Sq. ft.)
 - x_2 = จำนวนห้องนอน
 - x_3 = ระยะทางไปห้างสรรพสินค้าที่ใกล้ที่สุด
- เรายสามารถเขียนโมเดล **linear regression** ที่มีหลายตัวแปรได้ในรูป

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \\ = \boldsymbol{\theta}^T \boldsymbol{x}$$

โดยที่ n คือจำนวนฟีเจอร์; $\boldsymbol{\theta}$ เป็นเวกเตอร์ของพารามิเตอร์; \boldsymbol{x} เป็นเวกเตอร์ของฟีเจอร์บ้านหลังหนึ่ง

Linear Regression - Summary

- เราสามารถหาโมเดลที่ **fit** ข้อมูลได้ดีที่สุดด้วย คณิตศาสตร์ ได้อย่างไร?
- โมเดลการทำนาย: $\hat{y} = h_{\theta}(x) = \boldsymbol{\theta}^T \mathbf{x}$
- กำหนดเป้าหมาย $J(\theta)$  เรียกว่า **cost function**
บ่งบอกถึงความ~~แย่~~ของโมเดล
- **cost function** ของโมเดล linear regression คือ mean-square error (MSE)
$$J(\theta) = MSE(\theta) = \frac{1}{M} \sum_{i=1}^M (\theta^T \mathbf{x}^{(i)} - y^{(i)})^2$$
- การ **train** โมเดล คือ การหาพารามิเตอร์ $\boldsymbol{\theta}$ ที่ทำให้ **cost** $J(\theta)$ ต่ำที่สุด:

$$\hat{\boldsymbol{\theta}}_{MSE} = \operatorname{argmin}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Solution 1 – Closed-form solution

- $\hat{\theta}_{MSE} = \underset{\theta}{\operatorname{argmin}} J(\theta) = \underset{\theta}{\operatorname{argmin}} \frac{1}{M} \sum_{i=1}^M (\theta^T x^{(i)} - y^{(i)})^2$
- Solution: หากใช้ matrix calculus แก้สมการตรง ๆ จะได้ว่า

$$\hat{\theta}_{MSE} = (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{y}$$

สมการนี้มีชื่อว่า **“Normal Equation”**

ไม่ต้องจำ พิสูจน์ Normal Equation ด้วย Matrix Calculus

$$\begin{aligned}\frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) &= \frac{1}{2} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \\ &= J(\theta)\end{aligned}$$

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\ &= \frac{1}{2} \nabla_\theta (\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y}) \\ &= \frac{1}{2} \nabla_\theta \text{tr} (\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y}) \\ &= \frac{1}{2} \nabla_\theta (\text{tr} \theta^T X^T X \theta - 2 \text{tr} \vec{y}^T X \theta) \\ &= \frac{1}{2} (X^T X \theta + X^T X \theta - 2 X^T \vec{y}) \\ &= X^T X \theta - X^T \vec{y}\end{aligned}$$

ให้พจน์นี้เป็น **0** ก็จะได้ **Normal Equation**

Exercise: Housing Price Prediction

- ข้อมูล training

ขนาด (Sq. ft.)	จำนวน ห้องนอน	ระยะทางไป ห้างสรรพสินค้า	ราคา (ล้านบาท)
1000	2	5	9.5
1500	3	30	8.0
2000	5	40	12.5
1700	1	5	9.0
1200	2	30	5.5

กำหนดให้โมเดล Linear Regression คือ

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

โดยที่

x_1 = ขนาดบ้าน (Sq. ft.)

x_2 = จำนวนห้องนอน

x_3 = ระยะทางไปห้างสรรพสินค้าที่ใกล้ที่สุด

- จงหา $\hat{\theta}_{MSE}$ ด้วยวิธี Normal Equation
- จงคำนวณราคาร้านที่มี $X = (1200, 3, 8)$

Exercise: Housing Price Prediction

- ข้อมูล training

ขนาด (Sq. ft.)	จำนวน ห้องนอน	ระยะทางไป ห้างสรรพสินค้า	ราคา (ล้านบาท)
1000	2	5	9.5
1500	3	30	8.0
2000	5	40	12.5
1700	1	5	9.0
1200	2	30	5.5

เราสามารถเขียนข้อมูลในรูป matrix ได้ดังนี้

$$X = \begin{bmatrix} 1 & 1000 & 2 & 5 \\ 1 & 1500 & 3 & 30 \\ 1 & 2000 & 5 & 40 \\ 1 & 1700 & 1 & 5 \\ 1 & 1200 & 30 & 5.5 \end{bmatrix}$$

$$y = \begin{bmatrix} 9.5 \\ 8.0 \\ 12.5 \\ 9.0 \\ 5.5 \end{bmatrix}$$

Exercise: Housing Price Prediction

Q: ทำไมต้องเติม colum นี้ช้ายเป็น 1 ?

$$X = \begin{bmatrix} 1 & 1000 & 2 & 5 \\ 1 & 1500 & 3 & 30 \\ 1 & 2000 & 5 & 40 \\ 1 & 1700 & 1 & 5 \\ 1 & 1200 & 30 & 5.5 \end{bmatrix}$$

เพราะไม่เดลคือ

$$\begin{aligned} h_{\theta}(x) &= \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \\ &= [1 \ x_1 \ x_2 \ x_3] \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \end{aligned}$$

(x ในที่นี่คือ feature ของบ้าน 1 หลัง)

Exercise: Housing Price Prediction

Q: ทำไมต้องเติม colum นี้ช้ายเป็น 1 ?

$$X = \begin{bmatrix} 1 & 1000 & 2 & 5 \\ 1 & 1500 & 3 & 30 \\ 1 & 2000 & 5 & 40 \\ 1 & 1700 & 1 & 5 \\ 1 & 1200 & 30 & 5.5 \end{bmatrix}$$

โมเดลใน Matrix Form:

$$\hat{y} = h_{\theta}(x) = X\theta$$

$$\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \hat{y}^{(3)} \\ \hat{y}^{(4)} \\ \hat{y}^{(5)} \end{bmatrix} = \begin{bmatrix} 1 & 1000 & 2 & 5 \\ 1 & 1500 & 3 & 30 \\ 1 & 2000 & 5 & 40 \\ 1 & 1700 & 1 & 5 \\ 1 & 1200 & 30 & 5.5 \end{bmatrix} \times \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

(X ในที่นี่เก็บ feature ของบ้าน 5 หลัง)

Exercise: Housing Price Prediction

- ข้อมูล training

ขนาด (Sq. ft.)	จำนวน ห้องนอน	ระยะทางไป ห้างสรรพสินค้า	ราคา (ล้านบาท)
1000	2	5	9.5
1500	3	30	8.0
2000	5	40	12.5
1700	1	5	9.0
1200	2	30	5.5

- เมื่อได้ X, y แล้วก็สามารถหา $\hat{\theta}_{MSE}$ ด้วยวิธี Normal Equation ได้

- เมื่อได้ $\hat{\theta}_{MSE}$ แล้วก็สามารถนำโมเดลไป predict ราคางบ้านหลังใหม่ๆ ได้

Solution 1 – Closed-form solution

- ลองทำ linear regression ด้วยวิธี Normal Equation ใน Jupyter

```
data = genfromtxt('data.csv', delimiter=',')
X = data[:,0]
Y = data[:,1]
Xb = c_[ones((100,1)), X]
theta_mse = linalg.inv(Xb.T.dot(Xb)).dot(Xb.T).dot(Y)
x_test = arange(10,80).reshape((-1,1))
x_testb = c_[ones((len(x_test),1)), x_test]
y_hat = x_testb.dot(theta_mse)
```

train

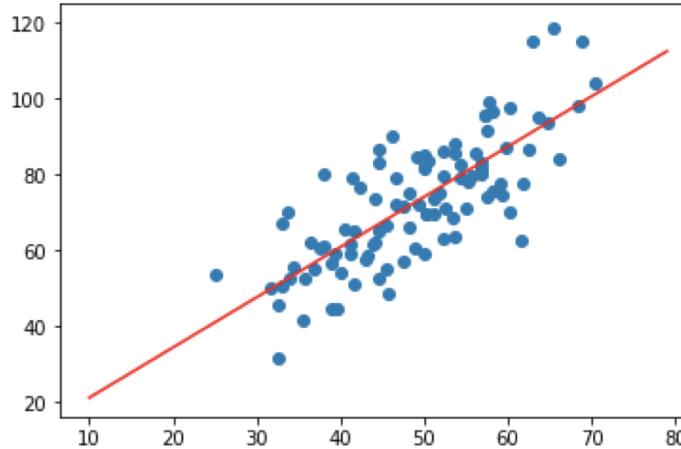
$$\hat{\theta}_{MSE} = (X^T \cdot X)^{-1} \cdot X^T \cdot Y$$

predict

$$\hat{y} = h_{\theta}(x) = \theta^T x$$

Solution 1 – Closed-form solution

ผลลัพธ์:



เส้นสีแดงเป็นตามสมการ

$$\hat{y} = h_{\theta}(x) = \theta^T x$$

- ข้อเสียของวิธี **Normal Equation/ Closed-form**

- การคำนวณ matrix inverse $(\mathbf{X}^T \cdot \mathbf{X})^{-1}$ มี complexity เป็น $O(n^{2.4}) \sim O(n^3)$
- หากจำนวน feature เэкอ \mathbf{X} จะใหญ่และทำให้คำนวณช้า

- ข้อดี

- แม้มีจำนวน training data มาก ก็ไม่ทำให้ช้า **$O(m)$**

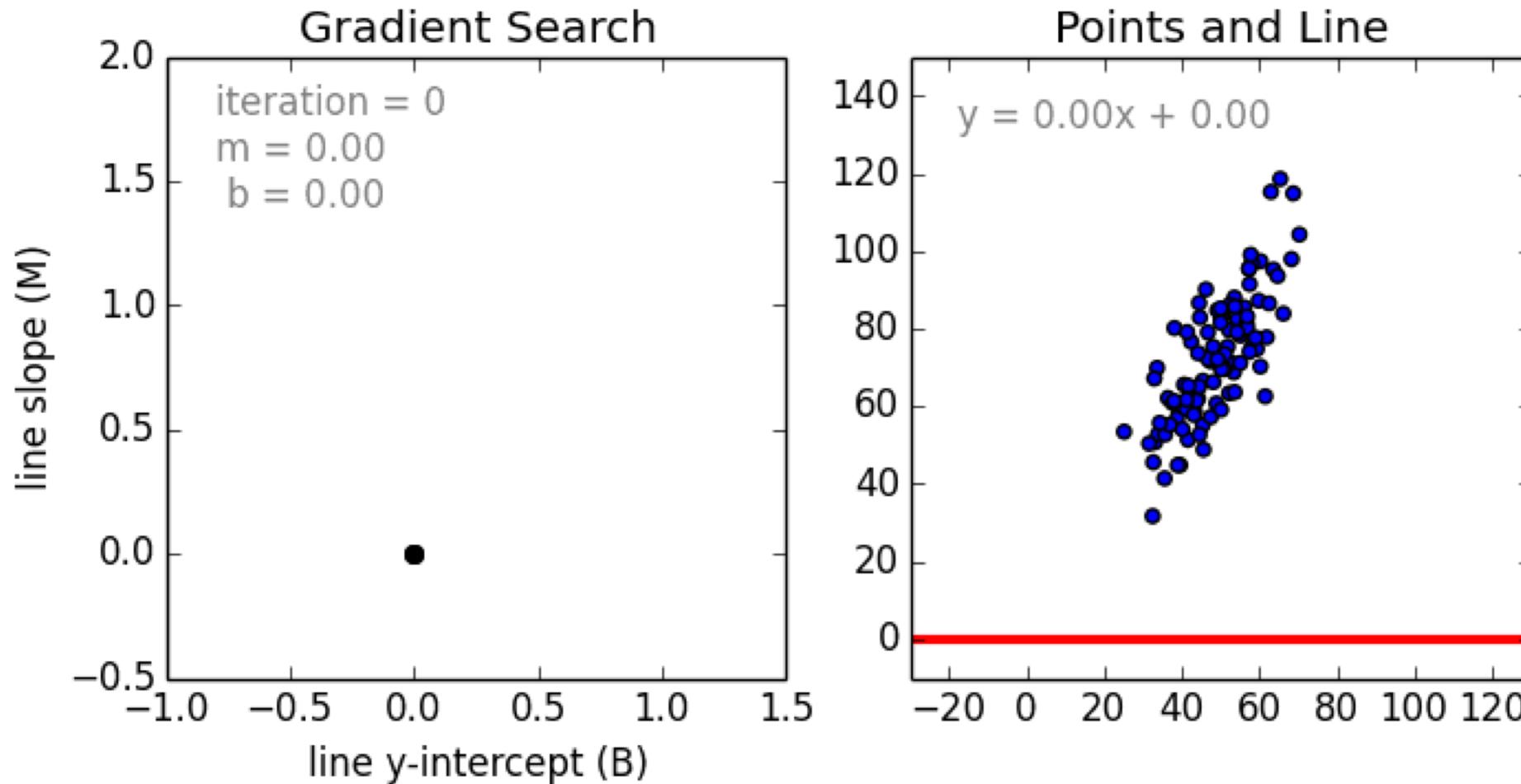
Solution 2 - Gradient Descent

- ในกรณีที่เรามี **feature** เยอะ (น้ำหนัก, ส่วนสูง, ความดัน, ... เป็นพันๆ **feature**) หรือ **training data** ใหญ่มาก เราอาจจะต้องหันมาใช้วิธี **Gradient Descent** แทนวิธี **normal equation**
- เป้าหมายเดิมคือ ต้องการหา θ ที่ทำให้ **MSE cost** ต่ำที่สุด

$$\hat{\theta}_{MSE} = \operatorname{argmin}_{\theta} J(\theta) = \operatorname{argmin}_{\theta} \frac{1}{M} \sum_{i=1}^M (\theta^T x^{(i)} - y^{(i)})^2$$

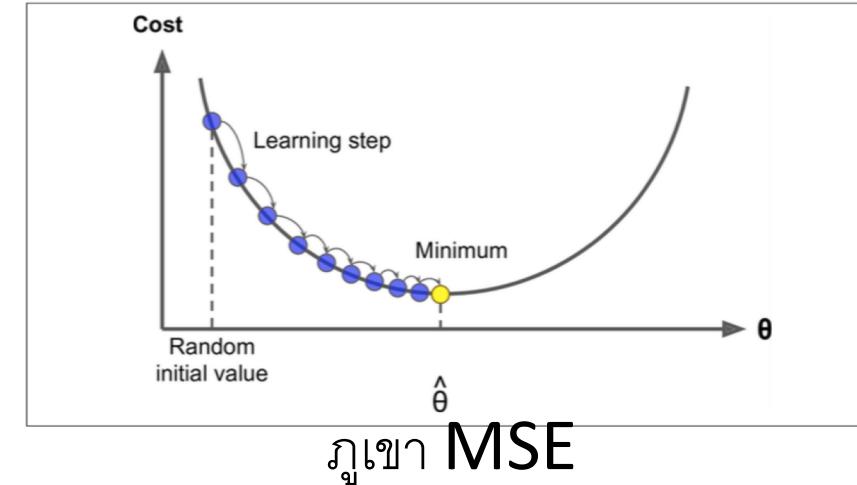
- แต่แทนที่จะแก้สมการตรงๆ เราจะเริ่มจากเดาค่า θ มาสักค่า มั่วๆ
- แล้วค่อยปรับ θ ทีละนิด ให้ **cost** ลดทีละนิด จนเราพอใจ

Solution 2 – Gradient Descent



Solution 2 - Gradient Descent

- อัลกอริทึม: วนทำสมการนี้ซ้ำไปเรื่อยๆ จนกว่า MSE จะเล็กมากพอดี



$$\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \text{MSE}(\theta)$$

θ เริ่มต้นเป็นค่า random

learning rate
จะก้าวให้ญี่่แค่ไหน

gradient ของ $\text{MSE}(\theta)$ w.r.t. θ

เป็นตัวบวกทิศทางลงภูเขามาก

$$\nabla_{\theta} \text{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\theta) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

Solution 2 - Gradient Descent

- ข้อดี: ไม่ต้องหา matrix inverse = เร็วกว่าวิธีแก้เมื่อฟีเจอร์เยอะ

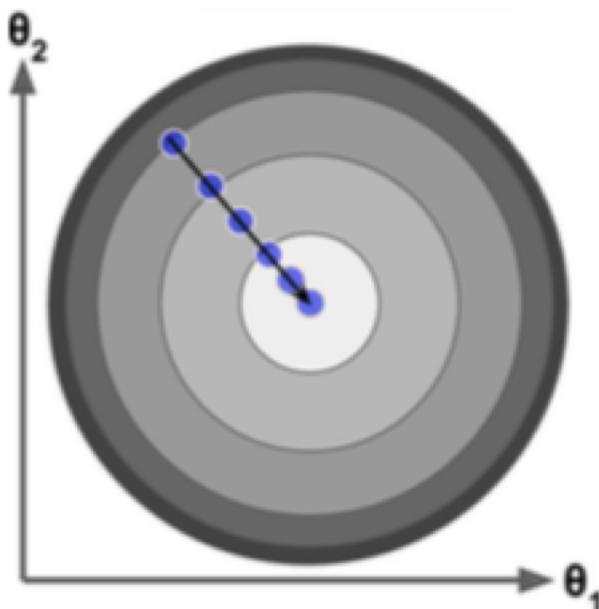
$$\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \text{MSE}(\theta)$$

$$\nabla_{\theta} \text{MSE}(\theta) = \begin{pmatrix} \frac{\partial}{\partial \theta_0} \text{MSE}(\theta) \\ \frac{\partial}{\partial \theta_1} \text{MSE}(\theta) \\ \vdots \\ \frac{\partial}{\partial \theta_n} \text{MSE}(\theta) \end{pmatrix} = \frac{2}{m} \mathbf{X}^T \cdot (\mathbf{X} \cdot \theta - \mathbf{y})$$

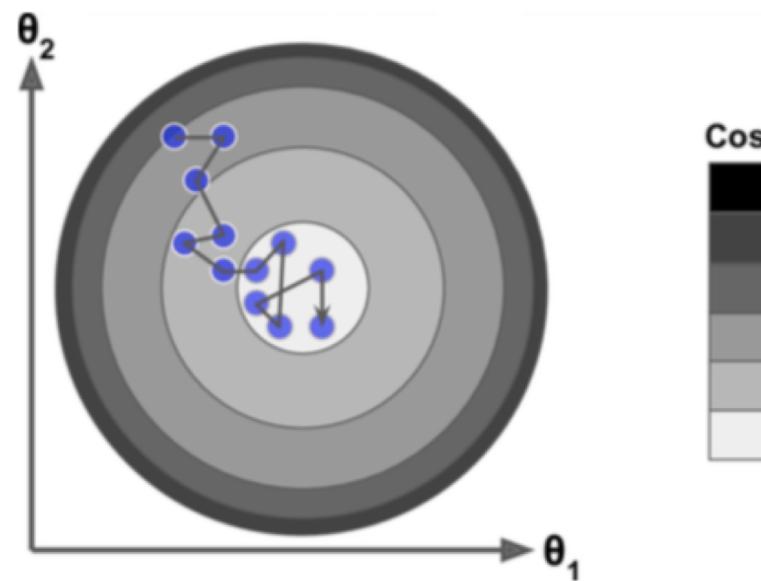
- ข้อเสีย: ทุกๆ step ต้องคำนวณ gradient โดยใช้ \mathbf{X} (training data ทั้งชุด) = ช้า

Solution 3 – Stochastic Gradient Descent

- แทนที่จะใช้ X ทั้งชุด ในแต่ละ step
- ให้เลือกข้อมูล X_i มาบางตัวโดยสุ่ม เพื่อคำนวณ gradient = เร็วขึ้นมาก แต่ converge ช้าหน่อย



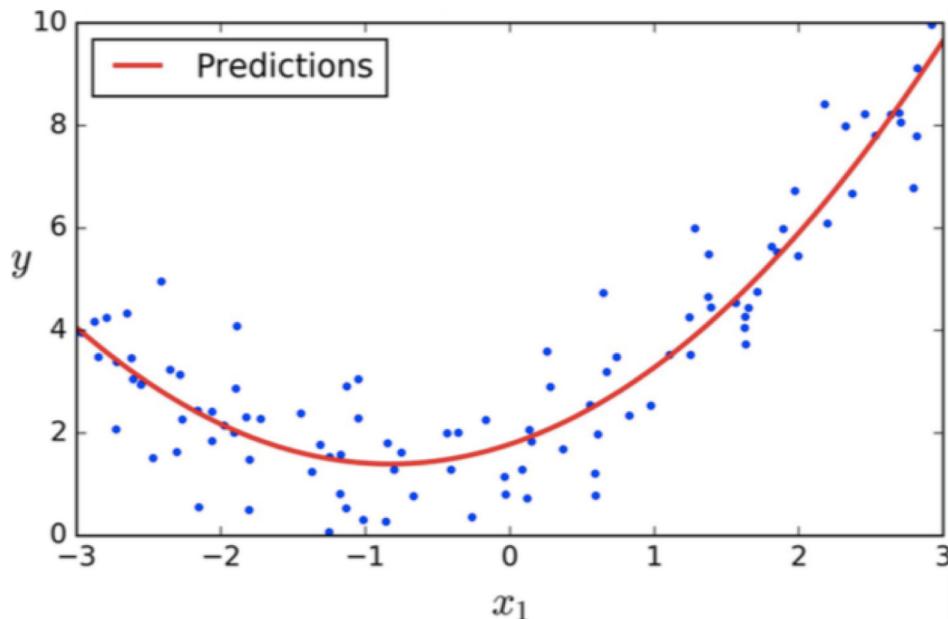
Solution 2: Batch GD



Solution 3: Stochastic GD

Polynomial Regression

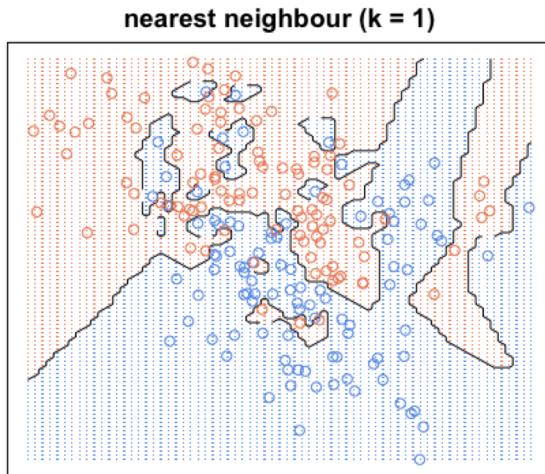
- ไม่เดล linear regression นี่สามารถใช้ทำนายความสัมพันธ์แบบ non-linear อย่าง x^2 ได้ด้วย
- เพียงแค่เปลี่ยนข้อมูล training X ให้มีค่าของ x^2 เพิ่มขึ้นมาด้วย เช่น
 - x_i เดิม = [น้ำหนัก, ส่วนสูง, อายุ] = [70, 150, 30]
 - x_i ใหม่ = [70, 150, 30, 4900, 22500, 900]  กลายเป็น 6 ฟีเจอร์ 7 พารามิเตอร์



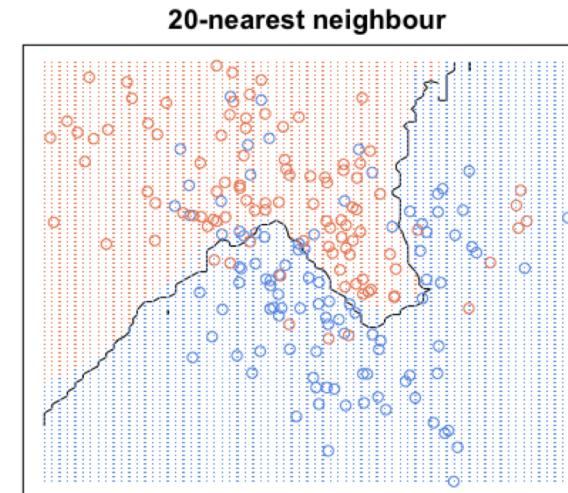
```
from sklearn.preprocessing import PolynomialFeatures  
poly_features = PolynomialFeatures(degree=2, include_bias=False)
```

Problem: Overfitting

- ប័ណ្ណហាសុទ្ធតែម polynomial regression អាជាំងក្នុងការ overfit ខ្លួន



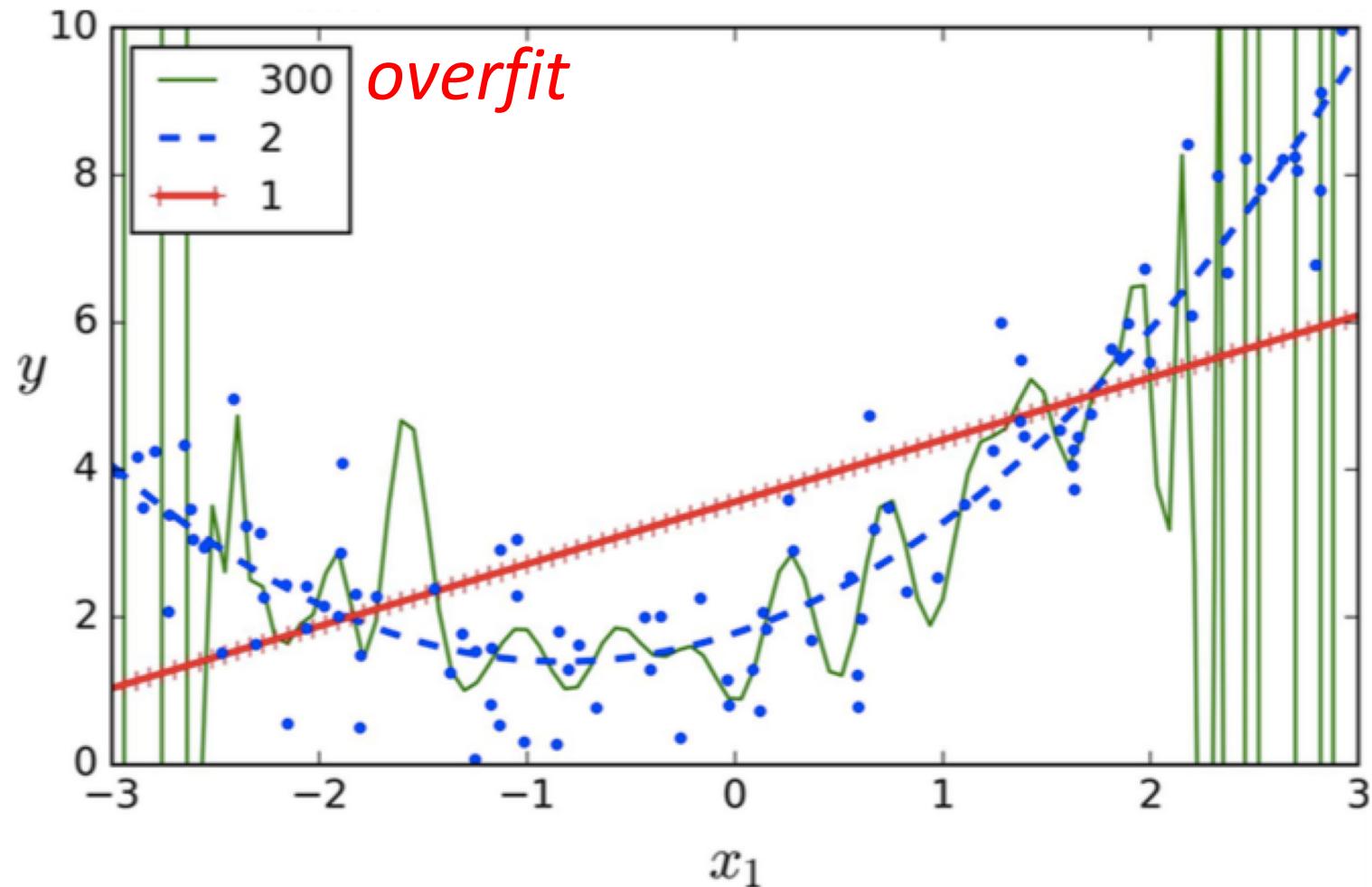
low bias ☺
high variance 😞



high bias 😞
low variance ☺

remember???

Problem: Overfitting



Solution to Overfitting: Regularization

- ทางออกคือเพิ่มพจน์ regularization ใน cost function

$$J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2$$

Ridge (L2) regularization

- พยายามให้ MSE น้อย แต่ก็พยายามให้ θ เล็กด้วย
- θ ที่เล็ก จะช่วยลดอิทธิพลของ polynomial degree สูงๆ อย่าง x^4, x^5 = ลด overfitting

Solution to Overfitting: Regularization

- โค้ด regularize สำหรับวิธี Normal Equation

```
>>> from sklearn.linear_model import Ridge  
>>> ridge_reg = Ridge(alpha=1, solver="cholesky")  
>>> ridge_reg.fit(X, y)  
>>> ridge_reg.predict([[1.5]])  
array([[ 1.55071465]])
```

- โค้ด regularize สำหรับวิธี SGD

```
>>> sgd_reg = SGDRegressor(penalty="l2")  
>>> sgd_reg.fit(X, y.ravel())  
>>> sgd_reg.predict([[1.5]])  
array([[ 1.13500145]])
```

Recap

- Linear Regression
- Minimizing cost function
- Solutions
 - Normal Equation
 - (Stochastic) Gradient Descent
- Polynomial Regression
- Regularization -> solves overfitting