

Linear Regression

Gradient Descent



อ. ปรัชญ์ ปิยะวงศ์วิศาล

Pratch Piyawongwisal

Today

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

Recap: Supervised Learning

• Classification

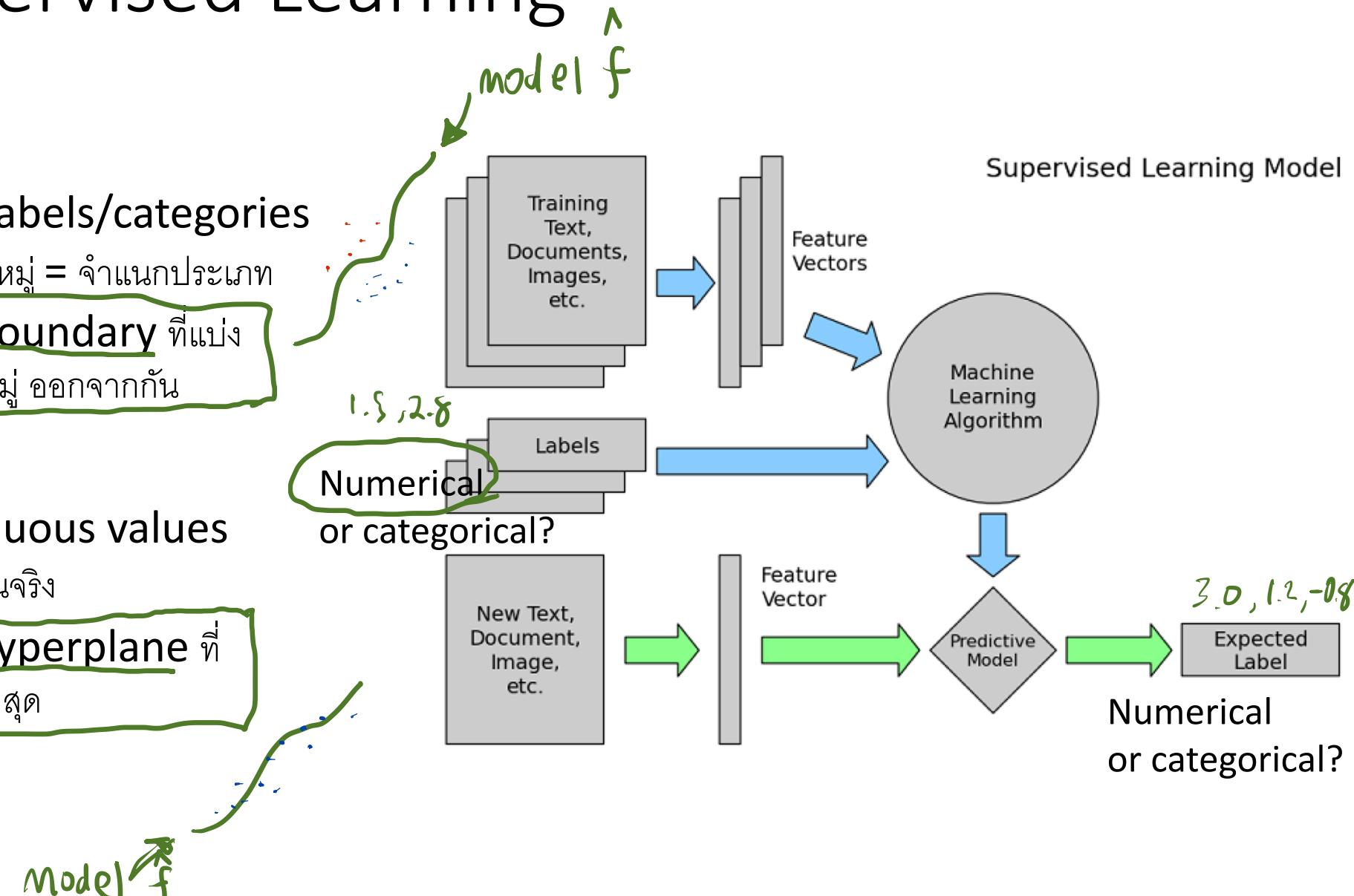
- Predicts class labels/categories
- ทำนายค่าที่เป็นหมวดหมู่ = จำแนกประเภท
- อาจมองเป็นการหา **boundary** ที่แบ่งข้อมูลในแต่ละหมวดหมู่ ออกจากกัน

• Regression

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

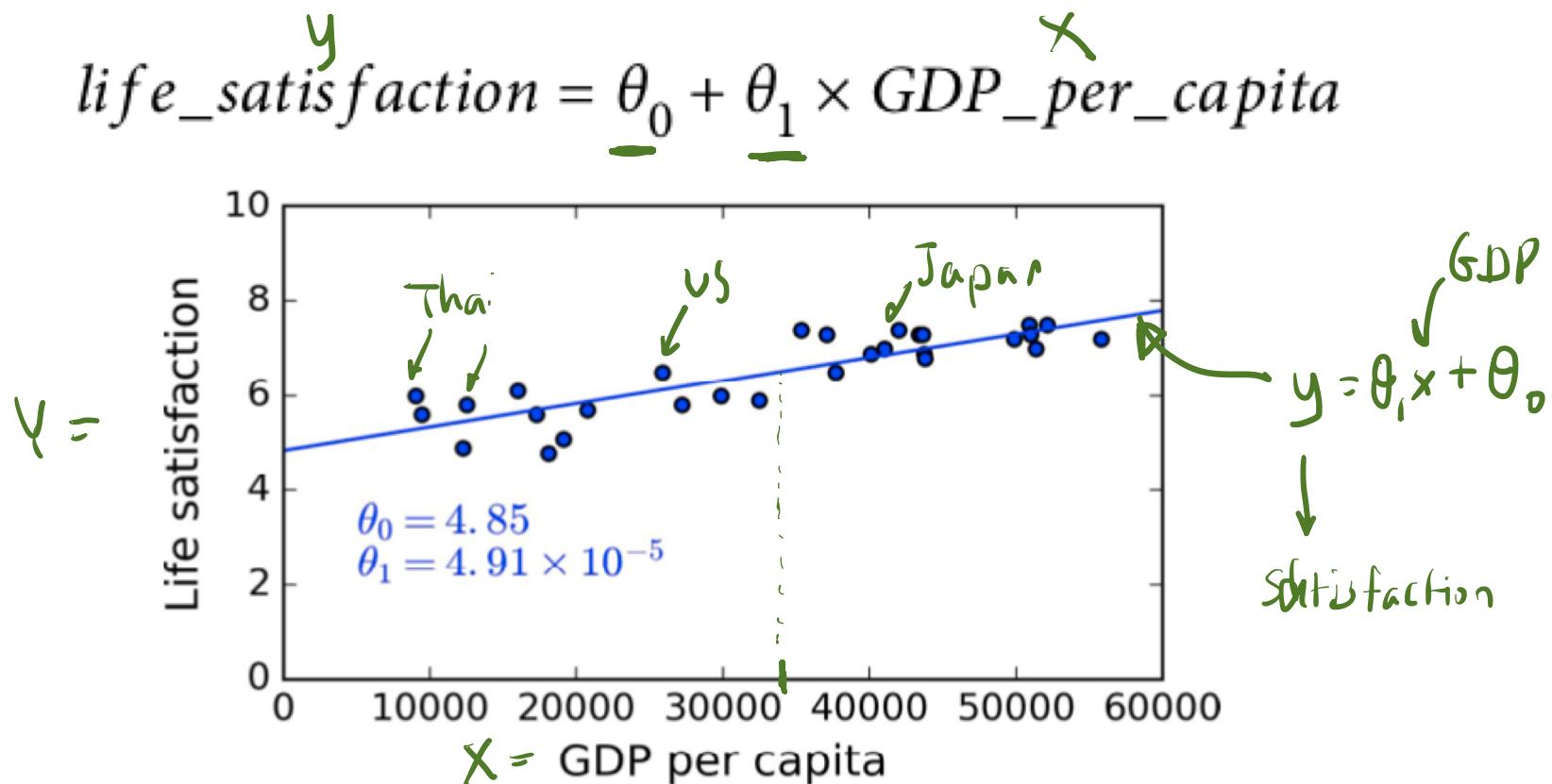
kNN

Linear Regression



Recall: Supervised Learning Lab (slides #2)

- ใช้ Scikit-Learn ในการสร้างโมเดลเพื่อทำนาย Life Satisfaction จาก GDP per capita
 - sklearn.linear_model.LinearRegression()



Linear Regression

- เป็นอัลกอริทึมแบบ

- supervised
- parametric**

ต่างๆ กับ KNN
ที่เป็นแบบ
non-parametric

- ใช้สำหรับทำ regression

- Example

- โมเดล: life_satisfaction = $h(\theta) = \theta_0 + \theta_1 \times \underline{GDP}$
- มี 1 feature คือ GDP
 - มี 2 parameters คือ θ_0, θ_1

มีผู้ช่วย (ใช้ข้อมูล train ที่มี label เฉลย) ในการหา $h: X \rightarrow y$
มี assumption ว่าฟังก์ชัน $h: X \rightarrow y$ เป็นแบบ Linear
โดยที่ฟังก์ชัน h จะมี สัมประสิทธิ์/ตัวแปร (parameter) ที่แสดงถึง
อิทธิพลของแต่ละ feature X_i ต่อค่าของ y
ผลการทำนายเป็นเลขจำนวนจริง

อนุญาติ = linear



$$\underline{\theta_0} + \theta_1 \times \underline{GDP}$$

life
sat

↔

$$\theta_0 + \theta_1 \times X_1 \text{ GDP}$$

ความต้องการ

$$\theta_2 \times X_2 \text{ ความต้องการ }$$

ปริมาณ

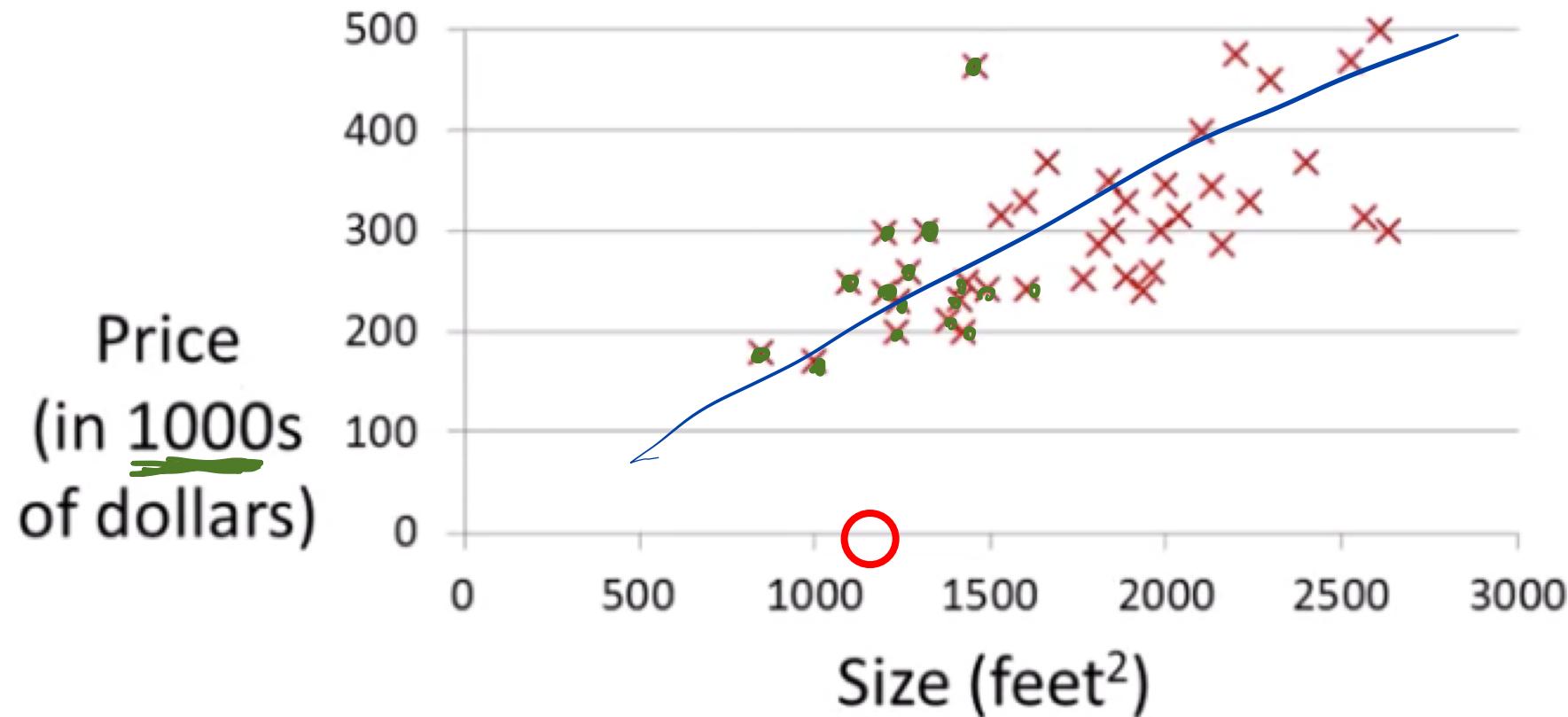
$$\theta_3 \times X_3 \text{ รัฐธรรมศาสตุ }$$



แก้
ไข่วิธีการ
วิธีการ
Parameters
ไม่พบวิธี

$$lifesat = 0.001 X_1 - 20 X_2 + 0 \times X_3 + 100$$

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
...	...

$$n = \# \text{ ฝี ใจ }$$

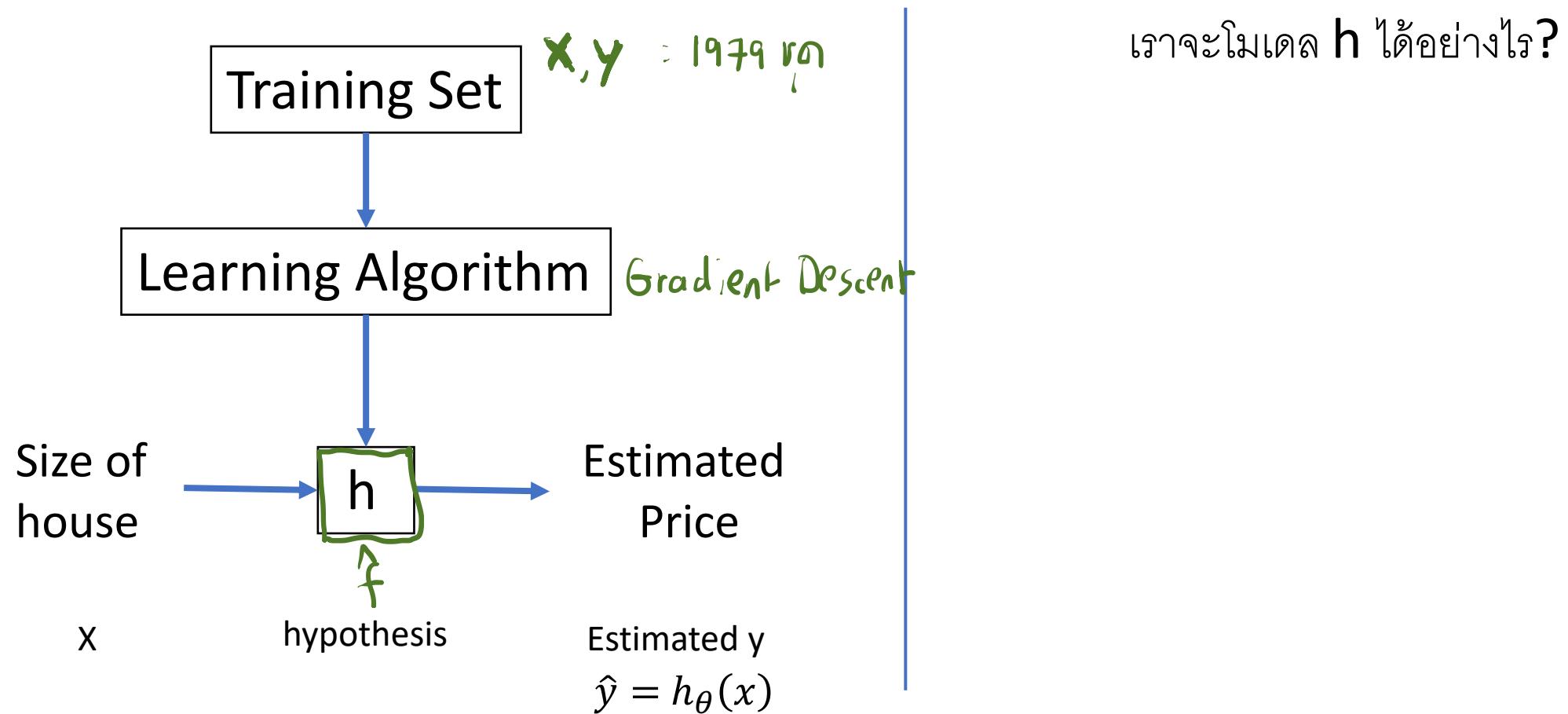
$$\mathbf{x} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(N)} \end{bmatrix}$$

นิยามสัญลักษณ์:

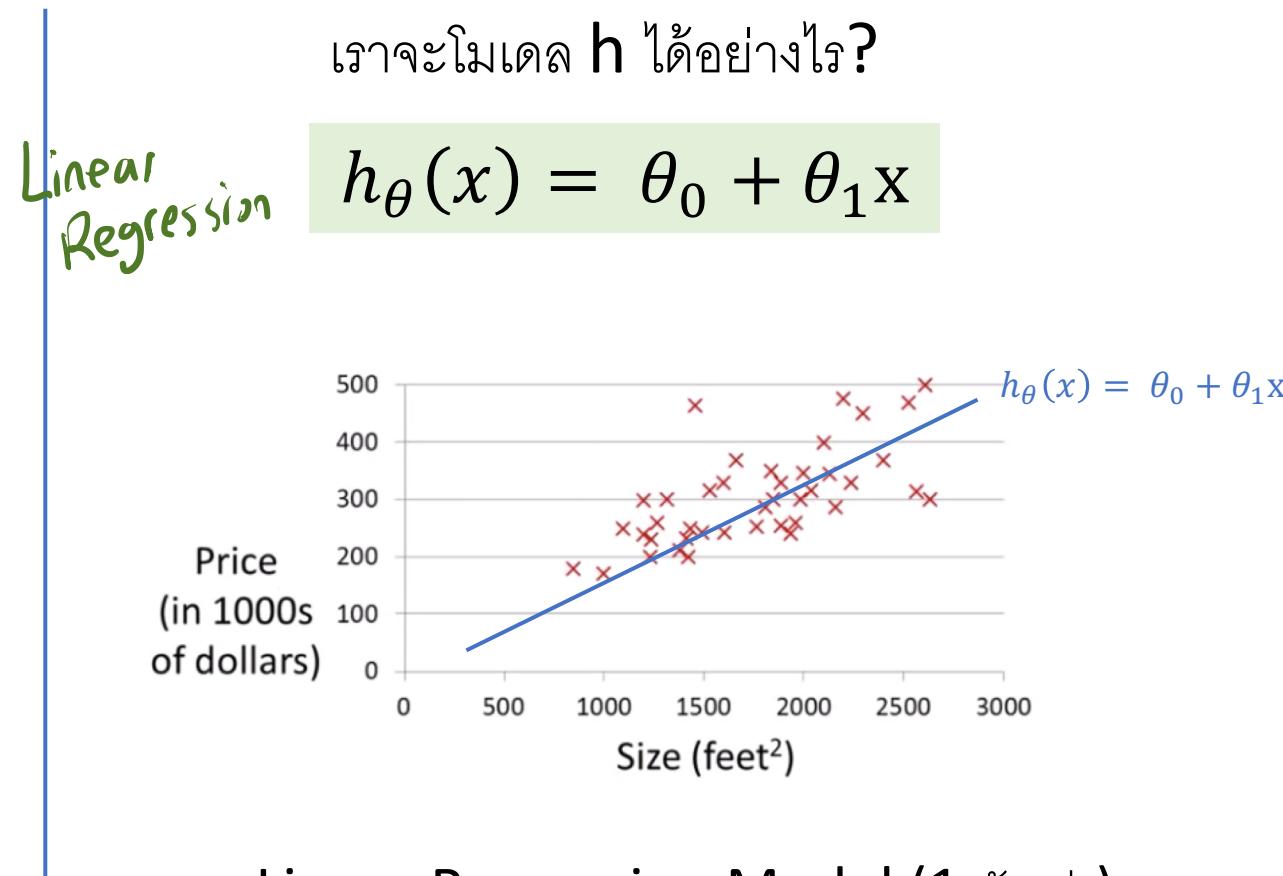
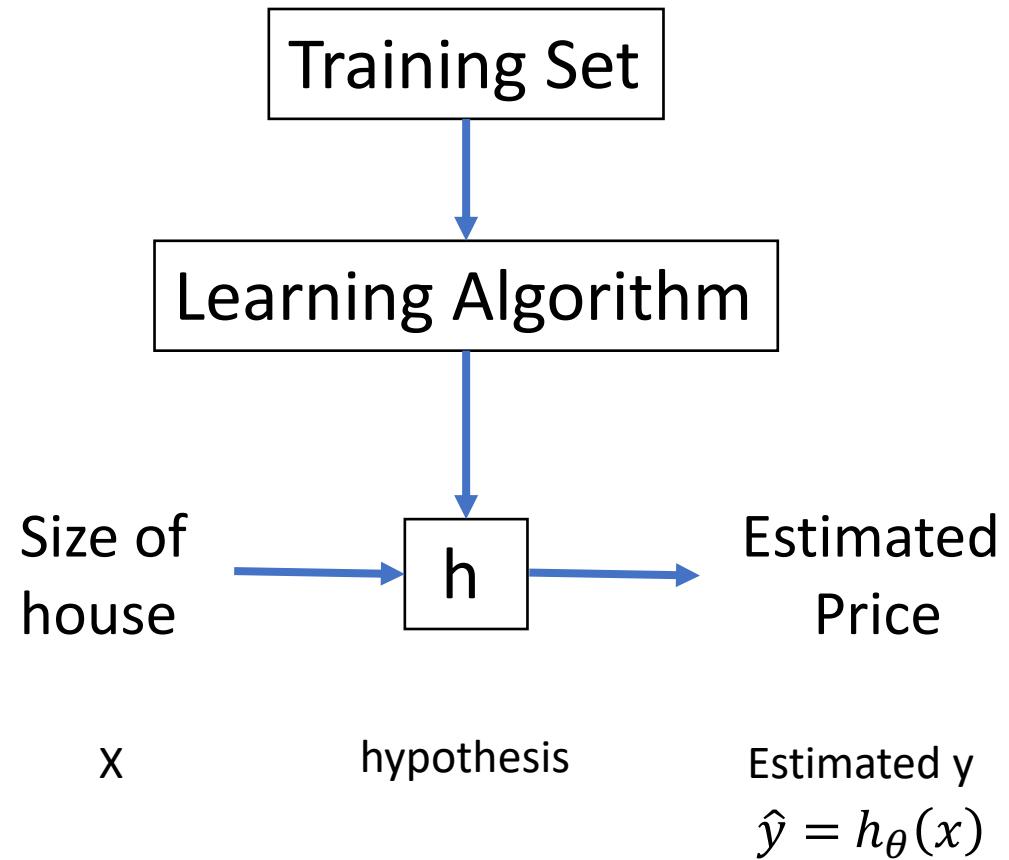
- M = จำนวนข้อมูลชุด training | 77
- x = เวกเตอร์ของ **feature** $M \times 1$ (#ฝี ใจ)
- y = เวกเตอร์ของค่า **target** $M \times 1$
- $(x^{(i)}, y^{(i)})$ = ข้อมูล training ชิ้นที่ i

$$\begin{aligned} x &= \begin{bmatrix} 2104 \\ 1416 \\ \vdots \\ \end{bmatrix} \\ y &= \begin{bmatrix} 460 \\ 232 \\ \vdots \\ \end{bmatrix} \end{aligned}$$

Example: Housing Prices

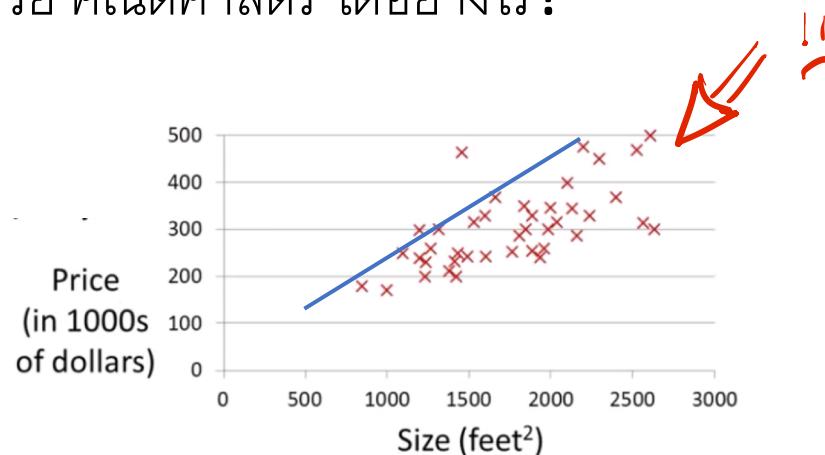
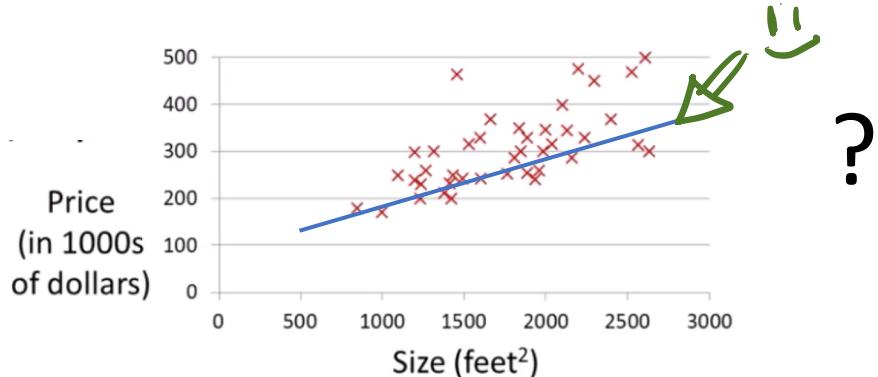


Example: Housing Prices



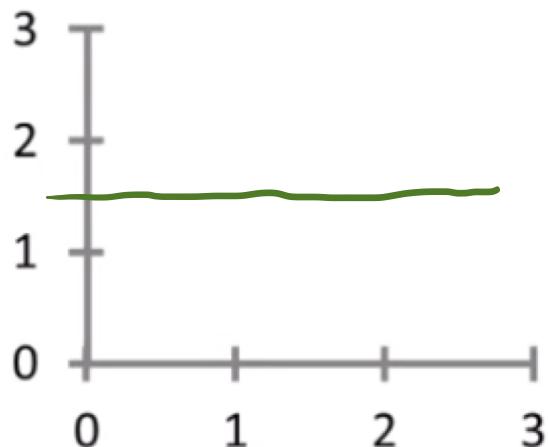
Linear Regression Model

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



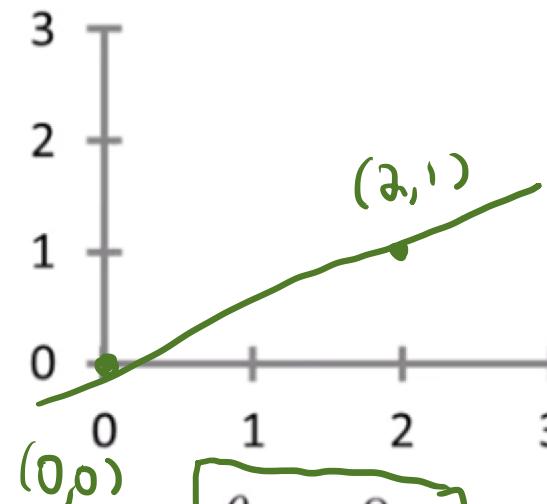
Linear Regression Model

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



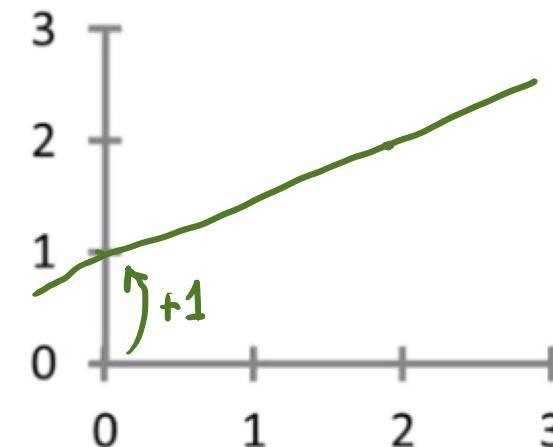
$$\boxed{\begin{array}{l} \theta_0 = 1.5 \\ \theta_1 = 0 \end{array}}$$

$$y = 1.5$$



$$\boxed{\begin{array}{l} \theta_0 = 0 \\ \theta_1 = 0.5 \end{array}}$$

$$y = 0.5x$$

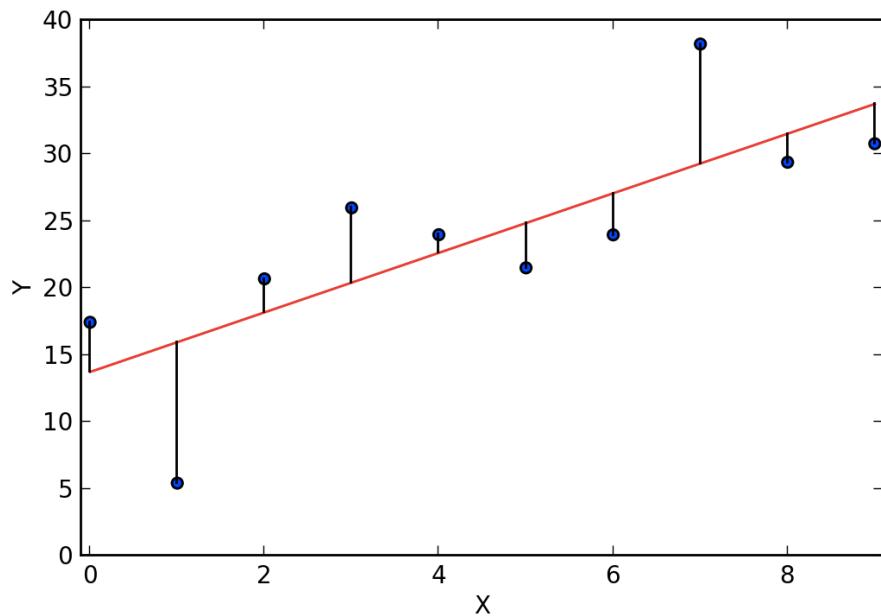


$$\boxed{\begin{array}{l} \theta_0 = 1 \\ \theta_1 = 0.5 \end{array}}$$

$$y = 0.5x + 1$$

Linear Regression Model

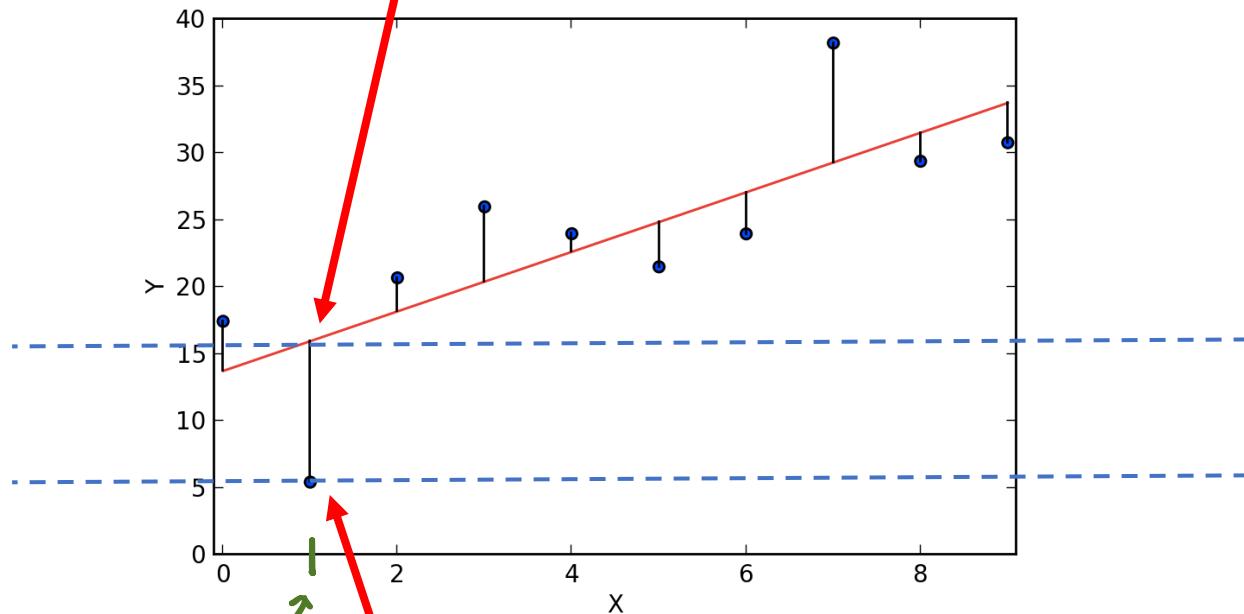
- Idea: เราจะเลือก θ_0, θ_1 ที่ทำให้ $h_\theta(x)$ มีค่าใกล้กับ y ในข้อมูลชุด training ที่สุด



Linear Regression Model

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

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x^{(i)}$$



$$x^{(i)} = 1$$

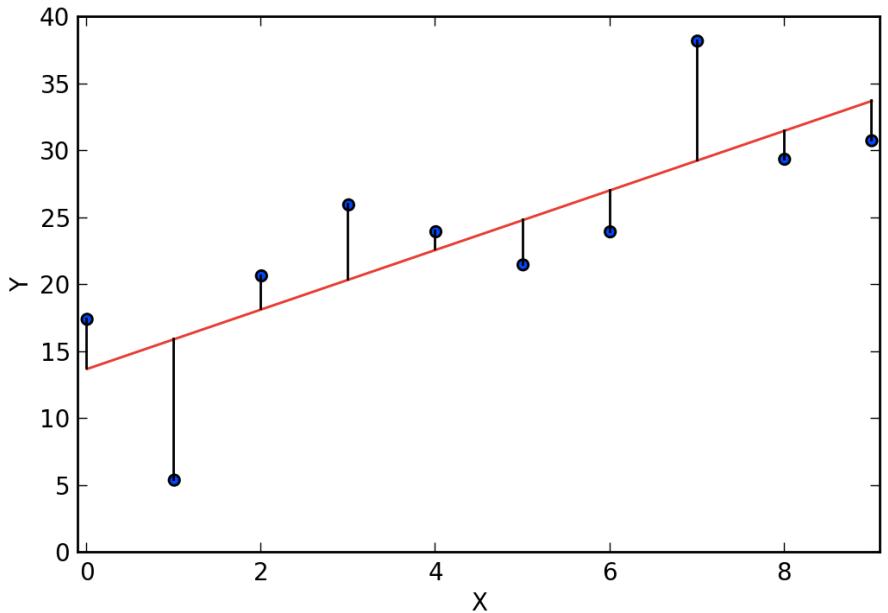
y (ข้อมูล train) $y^{(i)}$

} ระยะห่าง = $h_\theta(x^{(i)}) - y^{(i)}$

y_{pred} ค่าที่คำนวณ
 y_{train} ค่าเดิม

Linear Regression Model

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



ចងមុខទូ train

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

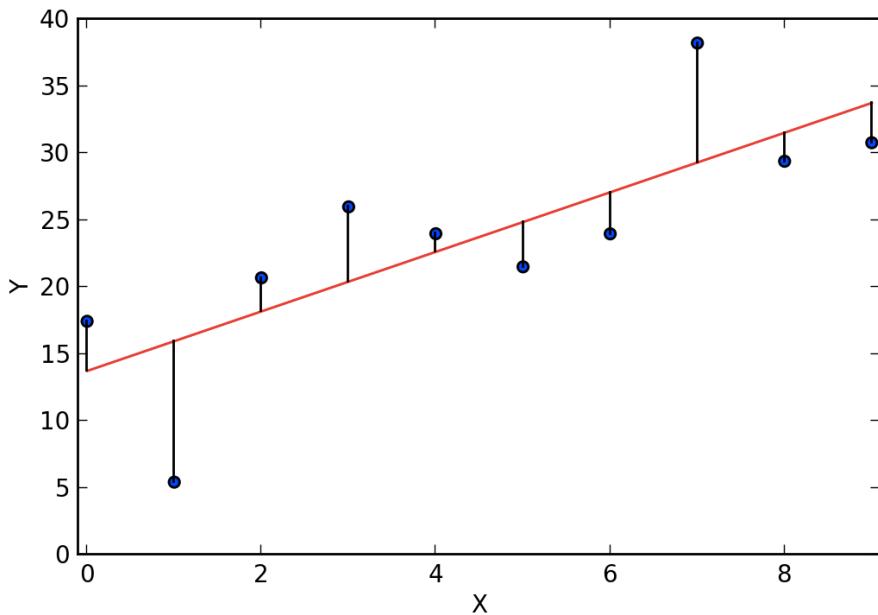
សម្រាប់លើកវិនិច្ឆ័យ

រួចរាល់ការពិនិត្យ

sum (ផលរាម)
ទុកា គុណធម្មតា

Linear Regression Model

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



* * * สำคัญมาก

“Cost Function”

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

ลดจำนวนข้อผิดพลาด

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

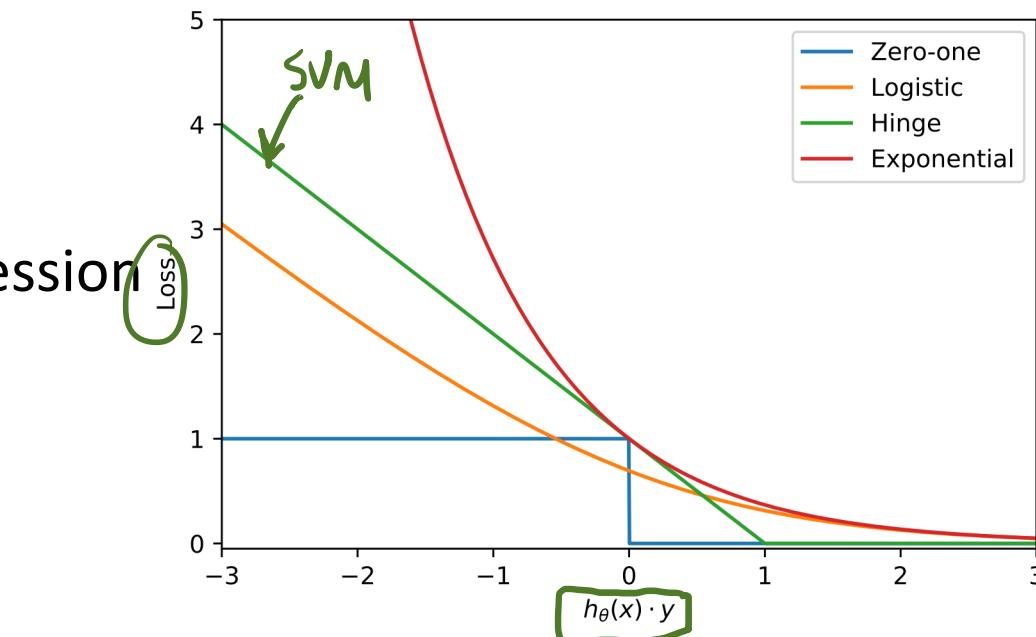
หาค่า θ_0, θ_1 ที่ทำให้ J น้อยที่สุด

Cost Function

ความแย่ๆ ของ模

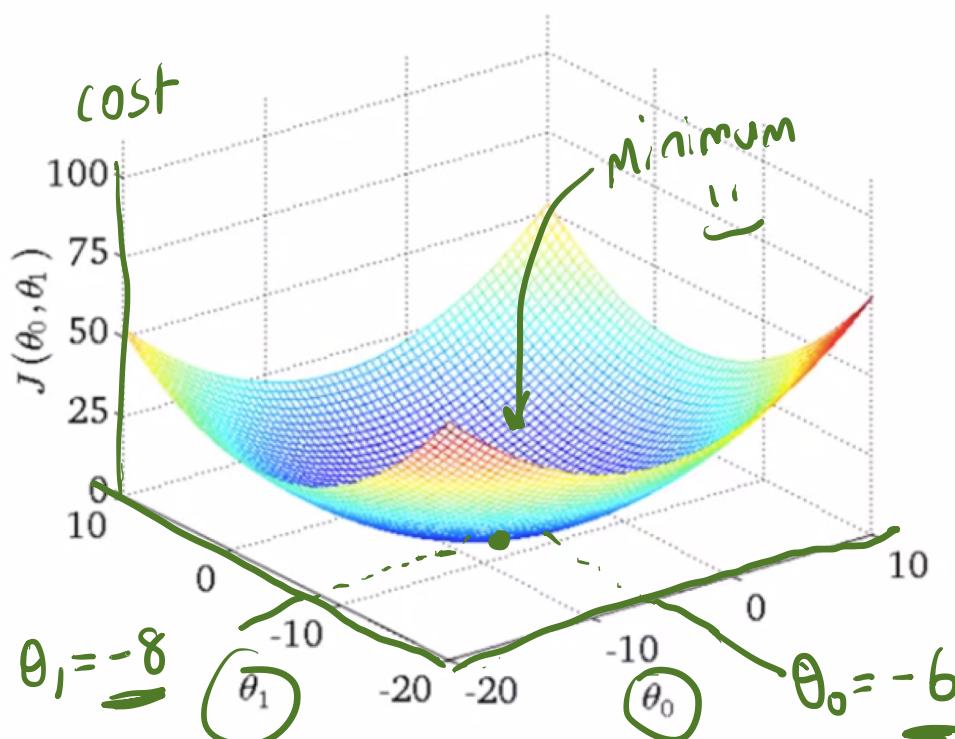
- Cost Function หรือ Loss Function คือฟังก์ชันที่บอกถึง Error ว่า ค่า \hat{y} ที่คำนวณได้ต่างจากค่า y จริงโดยเฉลี่ยเท่าไหร่
- ใน การ train ML algorithm เราจะพยายาม minimize loss ผู้ทำนาย \leftrightarrow ข้อมูลใน train set
บ่งชี้ว่า ค่า \hat{y} ดีมาก
- (*ไม่ออกสอบ*) มีหลากหลายชนิดขึ้นกับงาน เช่น
 - L1, L2 loss $\sum |y - f(x)|$
 - 0-1 loss
 - ⇒ **Hinge loss**
 - Cross-entropy loss (logloss)
 - Kullback-Leibler loss

Regression
Classification
SVM
Logistic Regression
VAE

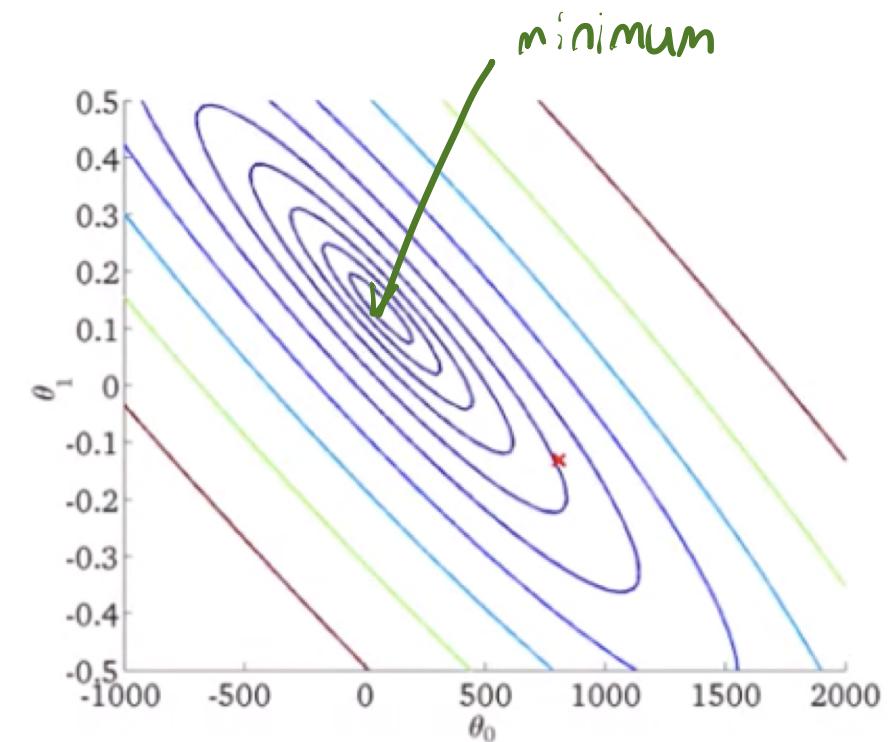


Cost Function - Visualized

- 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 ↪
- เราสามารถเขียนโมเดล **linear regression** ที่มี หลายตัวแปร ได้ในรูป

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

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

$$\theta^T x = [\theta_0 \ \theta_1 \ \dots \ \theta_n] \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$= \theta_0 + \theta_1 x_1 + \dots$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$= \boxed{\theta^T x}$$

ก ด้า # ฝึก

โดยที่ n คือจำนวนฟีเจอร์; θ เป็นเวกเตอร์ของพารามิเตอร์; x เป็นเวกเตอร์ของฟีเจอร์บ้านหลังหนึ่ง

Linear Regression - Summary

Q: เราสามารถหาโมเดลที่ fit ข้อมูลได้ดีที่สุดด้วย คณิตศาสตร์ ได้อย่างไร?

Model
⇒

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

- กำหนดเป้าหมาย $J(\theta)$ → เรียกว่า cost function

ปัจบุกถึงความ แย่ ของโมเดล
เฉลี่บ ผิดกันมาก

- cost function ของโมเดล linear regression คือ mean-square error (MSE)

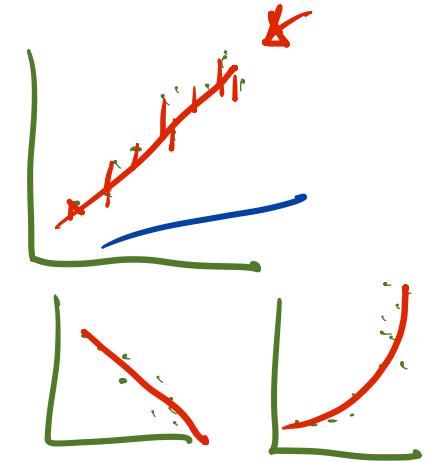
$$J(\theta) = MSE(\theta) = \frac{1}{M} \sum_{i=1}^M (\theta^T x^{(i)} - y^{(i)})^2$$

$y^{(i)}$
 y_{pred} $y^{(i)}$
 y_{true}

- การ train โมเดล คือ การหาพารามิเตอร์ θ ที่ทำให้ cost $J(\theta)$ ต่ำที่สุด:

$$\hat{\theta}_{MSE} = \underset{\theta}{\operatorname{argmin}} J(\theta) \Rightarrow \text{ปัญหา optimization}$$

vector



Solution 1 – Closed-form solution

ເກົ່ານມາຮວດ

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

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

ສມກາຣນີ້ມີຢືນວ່າ “**Normal Equation**”

x_{train} y_{train}

ไม่ต้องจำ พิสูจน์ Normal Equation ด้วย Matrix Calculus

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

$$\nabla_{\theta} J(\theta) = \vec{0}$$

trace

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \nabla_{\theta} \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\ &= \frac{1}{2} \nabla_{\theta} (\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y}) \\ &= \frac{1}{2} \nabla_{\theta} \text{tr}(\theta^T X^T X \theta - \theta^T X^T \vec{y} - \vec{y}^T X \theta + \vec{y}^T \vec{y}) \\ &= \frac{1}{2} \nabla_{\theta} (\text{tr} \theta^T X^T X \theta - 2 \text{tr} \vec{y}^T X \theta) \\ &= \frac{1}{2} (X^T X \theta + X^T X \theta - 2 X^T \vec{y}) \\ \vec{0} &= X^T X \theta - X^T \vec{y}\end{aligned}$$

$$\nabla_{\theta} (\text{tr} \dots)$$

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

1200 3 8 ?

กำหนดให้โมเดล Linear Regression คือ

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

โดยที่

x_1 = ขนาดบ้าน (Sq. ft.)

x_2 = จำนวนห้องนอน

x_3 = ระยะทางไปห้างสรรพสินค้าที่ใกล้ที่สุด

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

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

$$\begin{aligned} M &= 5 && \text{จำนวน} \\ n &= 3 && \text{ฝั่งขวา} \end{aligned}$$

เรารสามารถเขียนข้อมูลในรูป matrix ได้ดังนี้

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

$M \times (n+1)$
 5×4

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

$M \times 1$

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

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

(x ในที่นี่คือ feature ของบ้าน 1 หลัง)

Exercise: Housing Price Prediction

$$h_{\theta}(x) = \underline{\theta}^T X$$

Q: ทำไมต้องเติมคอลัมน์ซ้ายเป็น 1 ?

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

$$= \begin{bmatrix} \theta^T x^{(1)} \\ \theta^T x^{(2)} \\ \vdots \\ \theta^T x^{(M)} \end{bmatrix}$$

ทำให้เราสามารถลื้งฟรีบน x กัน

โมเดลใน Matrix Form:

$$\widehat{y} = h_{\theta}(x) = \underline{x} \theta$$

feature ปล่อยไว้

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

(X ในที่นี่เก็บ feature ของบ้าน 5 หลัง)

รวมข้อมูลทุกอัน

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

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

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 

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

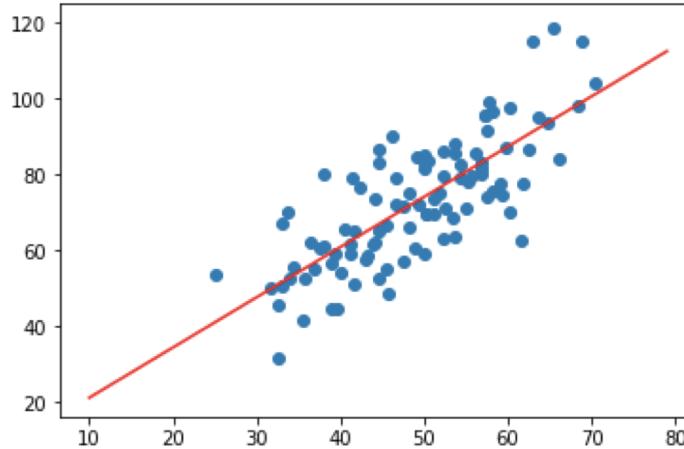
predict 

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

จำนวนผู้เช่า $(1, 1200, 3, 8) \times \begin{bmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \\ \hat{\theta}_2 \\ \hat{\theta}_3 \end{bmatrix}$

Solution 1 – Closed-form solution

ผลลัพธ์:



เส้นสีแดงเป็นตามสมการ

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

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

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

- ข้อดี

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

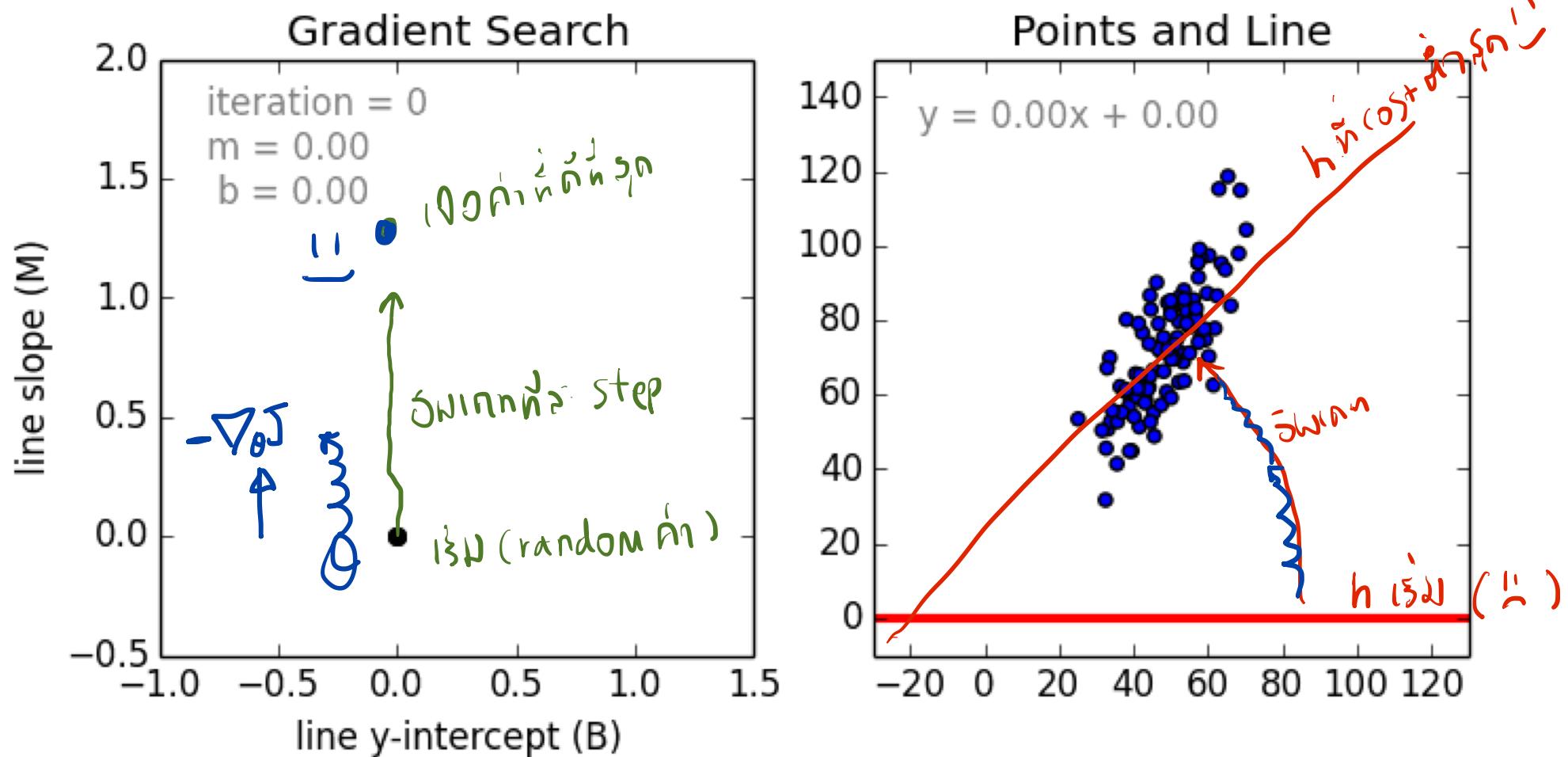
Solution 2 - Gradient Descent

- 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 มัวๆ
- แล้วค่อยปรับ θ ที่ลงนิด ให้ **cost** ลดที่ลงนิด จนเราพอใจ
ที่สุด

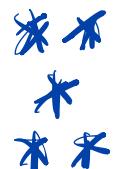
Solution 2 – Gradient Descent



ເຕີໂທວຽກນັບ ດົງເພື່ອປ່ຽນກຸງກາເຫຼາ

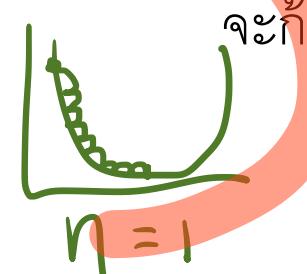
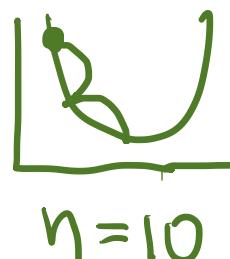
Solution 2 - Gradient Descent

- ອັດກອົງ: ວິທີທີ່ໃຫຍ້ໄວ້ມີຄວາມສຳເນົາໃນການຈົບເລີກຂອງ **MSE** ຈະເລື້ອນມາກພອ

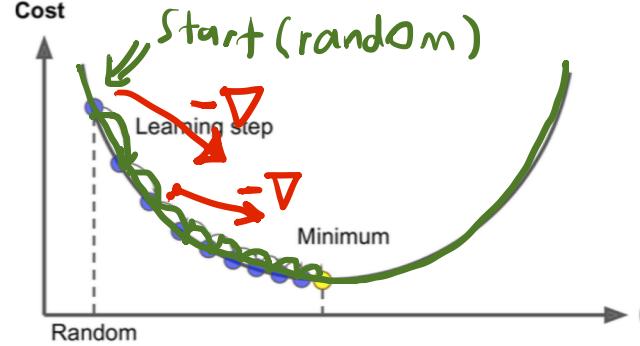


for θ ໃຫຍ້
 $\theta^{(\text{next step})} = \theta - \eta \nabla_{\theta} \text{MSE}(\theta)$

θ ເຮັດວຽກ ເປັນຄ່າ random
learning rate



ກົດກົງ ນີ້ມີກຳນົດໃຫຍ້
ກູ້ເຂົາ MSE



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

ເປັນຕົວບອກທີ່ສທາງລົງກູ້ເຂົາ

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

Batch

Solution 2 - ^V Gradient Descent

- ข้อดี: ไม่ต้องหา matrix inverse = เร็วกว่าวิธีแก้เมื่อฟีเจอร์เยอะ

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

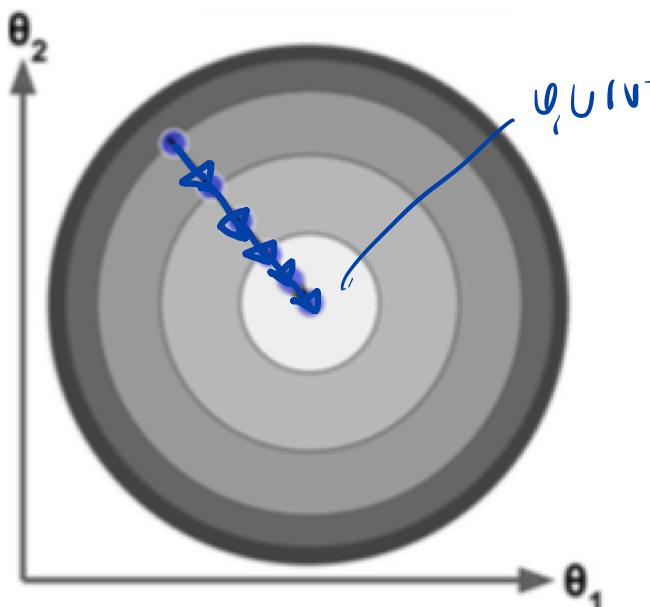
- ข้อเสีย: ทุกๆ step ต้องคำนวณ gradient โดยใช้ \mathbf{X} (training data ทั้งชุด) = ช้า

Solution 3 – Stochastic Gradient Descent

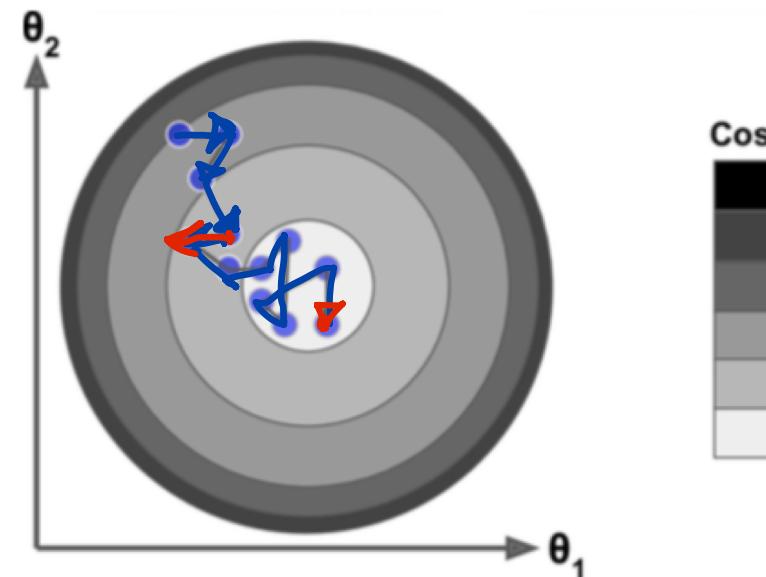
SGD

randomized

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



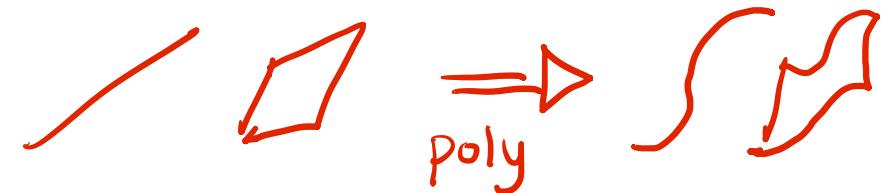
Solution 2: Batch GD (จงจราจร, บีบีด)
X



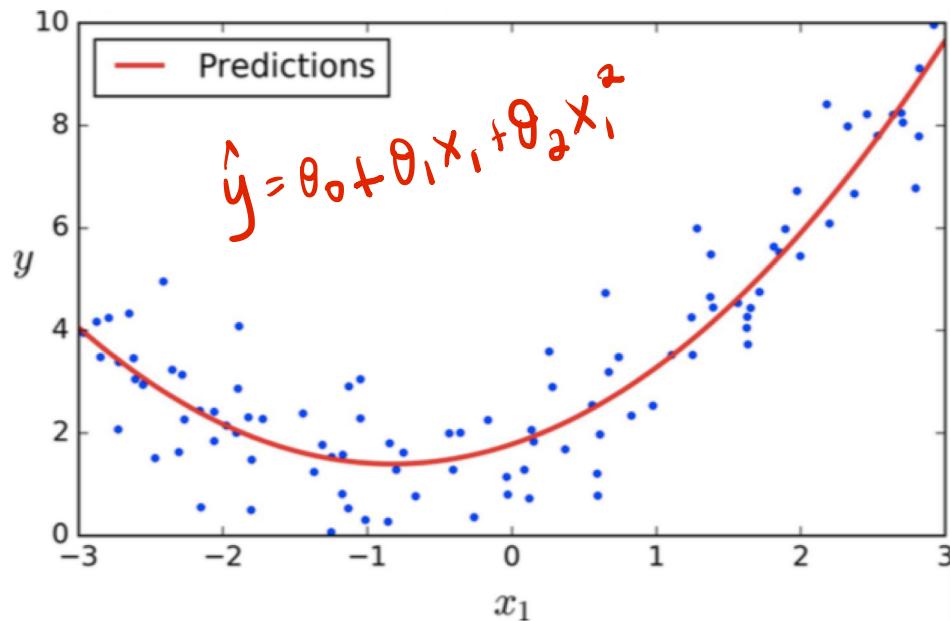
Solution 3: Stochastic GD

Polynomial Regression

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$



- โมเดล linear regression นี้สามารถใช้ทำนายความสัมพันธ์แบบ non-linear อย่าง x^2 ได้ด้วย
- เพียงแค่เปลี่ยนข้อมูล training X ให้มีค่าของ x^2 เพิ่มขึ้นมาด้วย เช่น
- $\Rightarrow x_i$ เดิม = [น้ำหนัก, ส่วนสูง, อายุ] = [70, 150, 30]
- x_i ใหม่ = [70, 150, 30, 4900, 22500, 900] \rightarrow กล้ายเป็น 6 ฟีเจอร์ 7 พารามิเตอร์



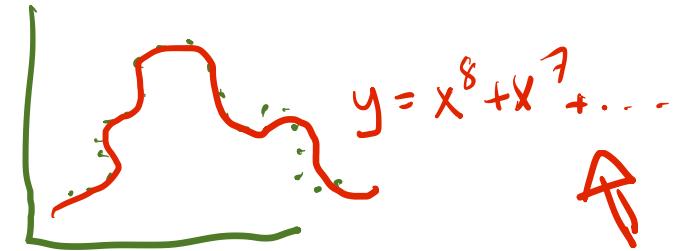
ก็ต่อไป $J(\theta) = \frac{1}{M} \sum (\hat{y} - y)^2$

ต้องแก้ SCD, $\arg \min_{\theta} J$ จึง \Rightarrow คุณสมบัติ

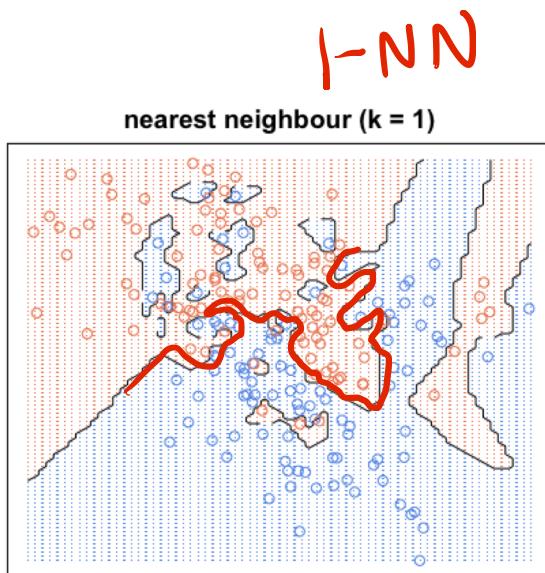
```
from sklearn.preprocessing import PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)
```

model[1]: $\hat{f}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_1^2 + \theta_5 x_2^2 + \theta_6 x_3^2$

Problem: Overfitting



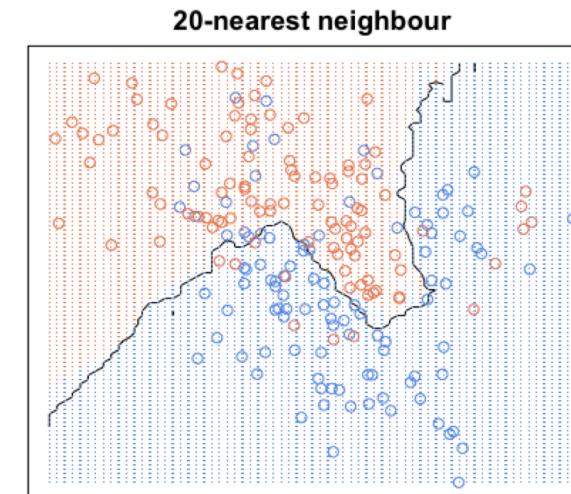
- ปัญหาสำคัญ polynomial regression อาจทำให้เกิดการ overfit ข้อมูล



Overfit

Variance สูง

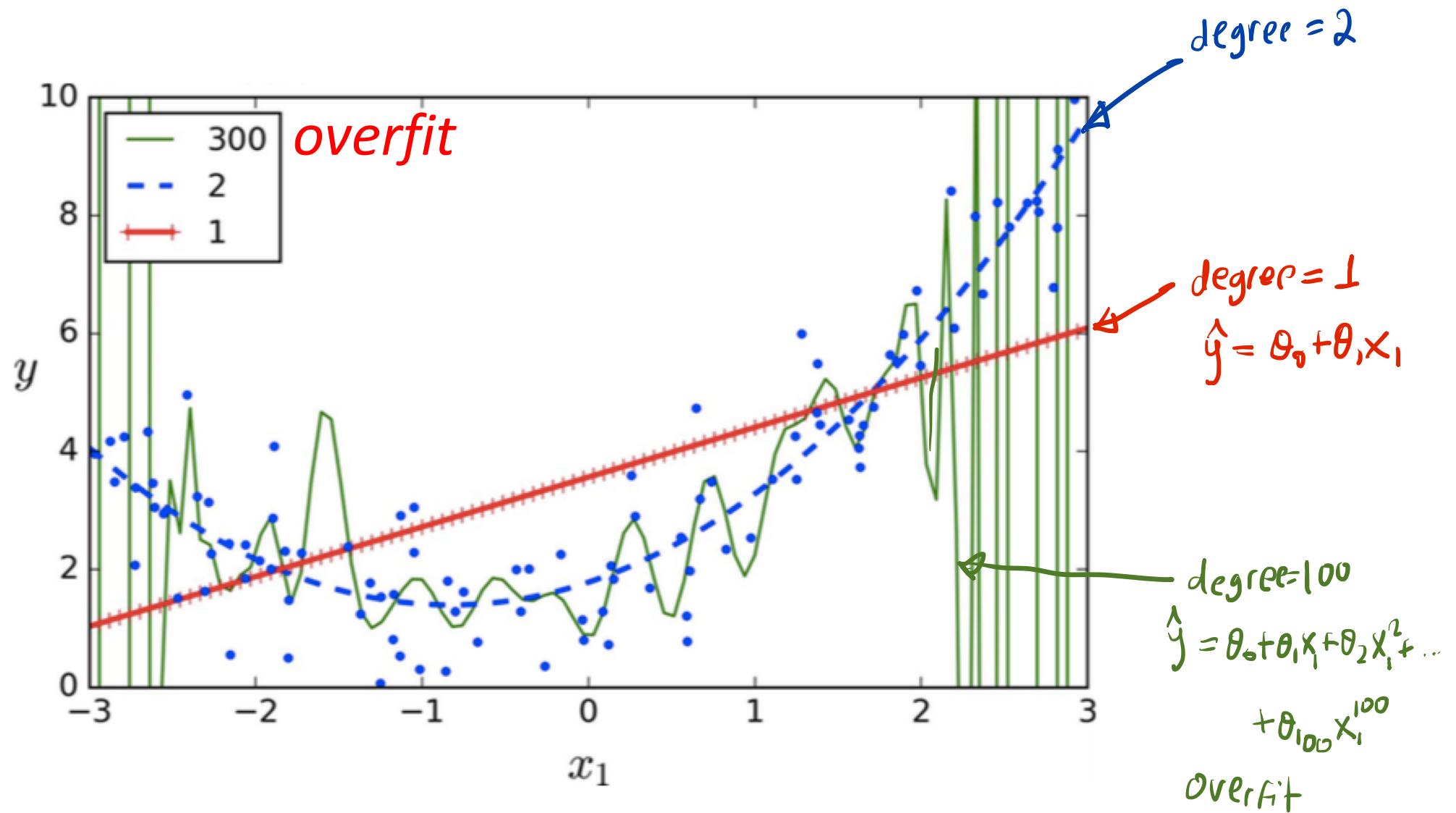
low bias 😊
high variance 😞



high bias 😞
low variance 😊

remember???

Problem: Overfitting



Solution to Overfitting: Regularization

- ทางออกคือเพิ่มพจน์ regularization ใน cost function

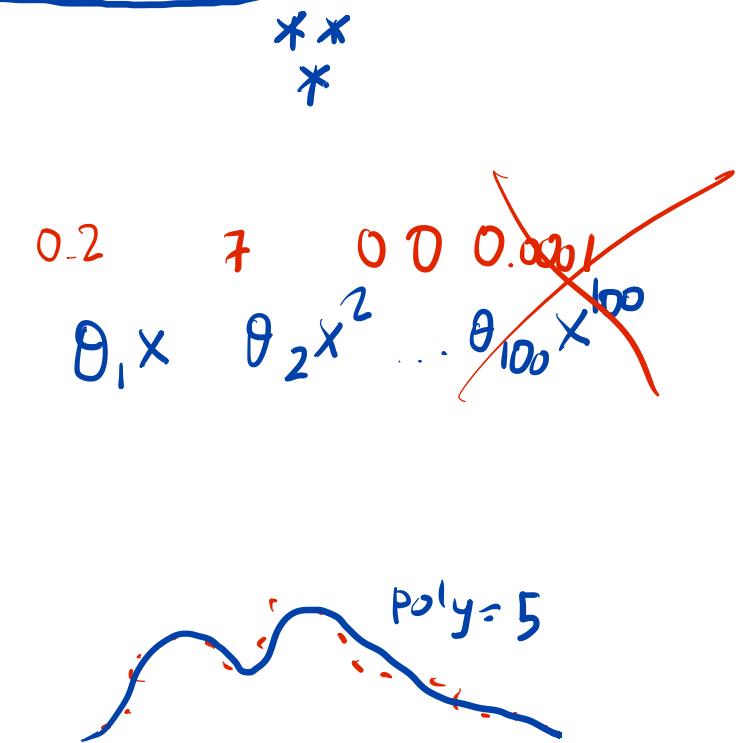
minimize $J(\theta) = \text{MSE}(\theta) + \alpha \frac{1}{2} \sum_{i=1}^n \theta_i^2$

ค่าที่ต้องการ
รักษาให้คงที่

hyperparam

cost

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]])
```

$\alpha = 1$ ใช้ลด overfit เกิน
คาด poly ต้องน้อยที่สุด

- โค้ด regularize สำหรับวิธี SGD

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

regularization

L2 = Ridge

Recap

- Linear Regression
- Minimizing cost function J
- Solutions

- Normal Equation $\theta_{\text{NE}} = (X^T X)^{-1} X^T y$

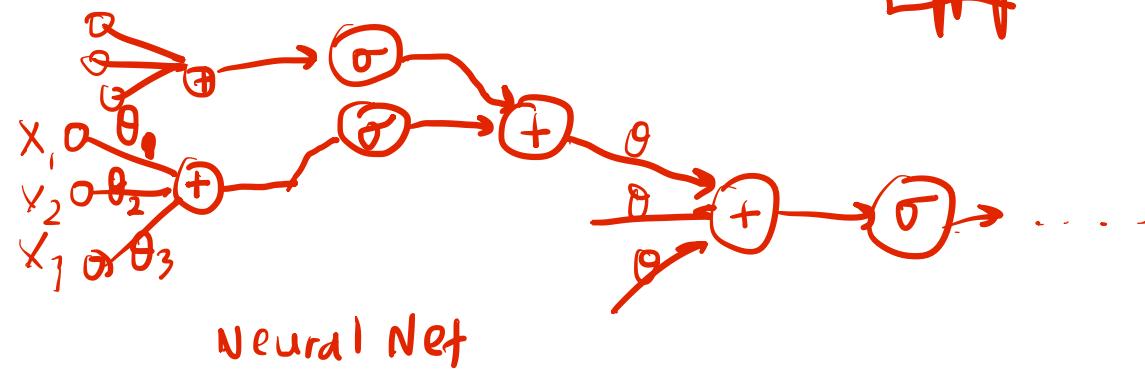
- (Stochastic) Gradient Descent $\theta_{\text{new}} = \theta_{\text{old}} - \eta \nabla_{\theta} J$

- Polynomial Regression



- Regularization -> solves overfitting

$\nabla_{\theta} J = \text{---}$



Probabilistic Interpretation of Linear Regression

- (whiteboard)