

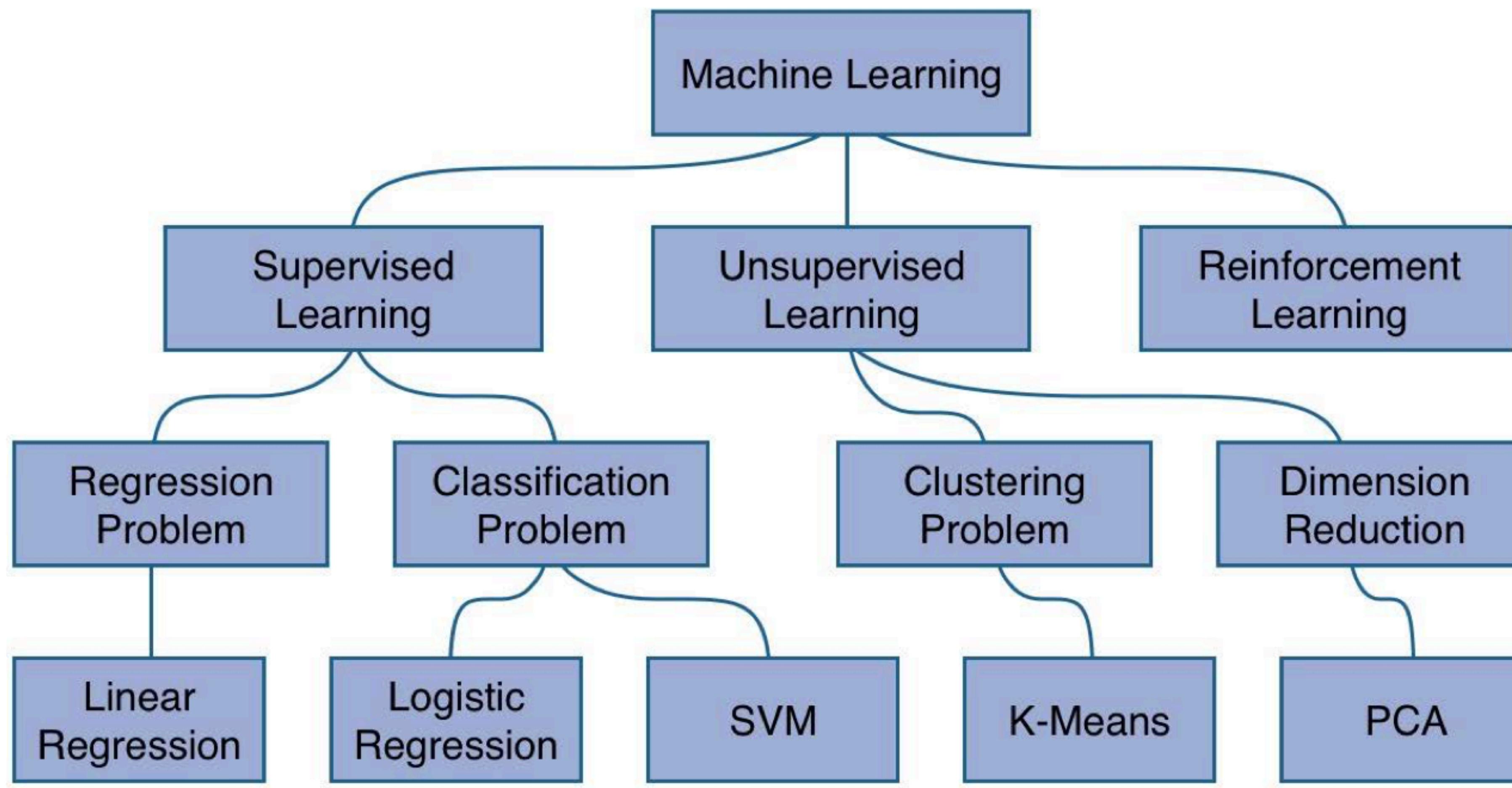
Regression

Bahan Kuliah SD3104 Machine Learning

Sevi Nurafni

**Fakultas Sains dan Teknologi
Universitas Koperasi Indonesia 2024**

Machine Learning: Overview

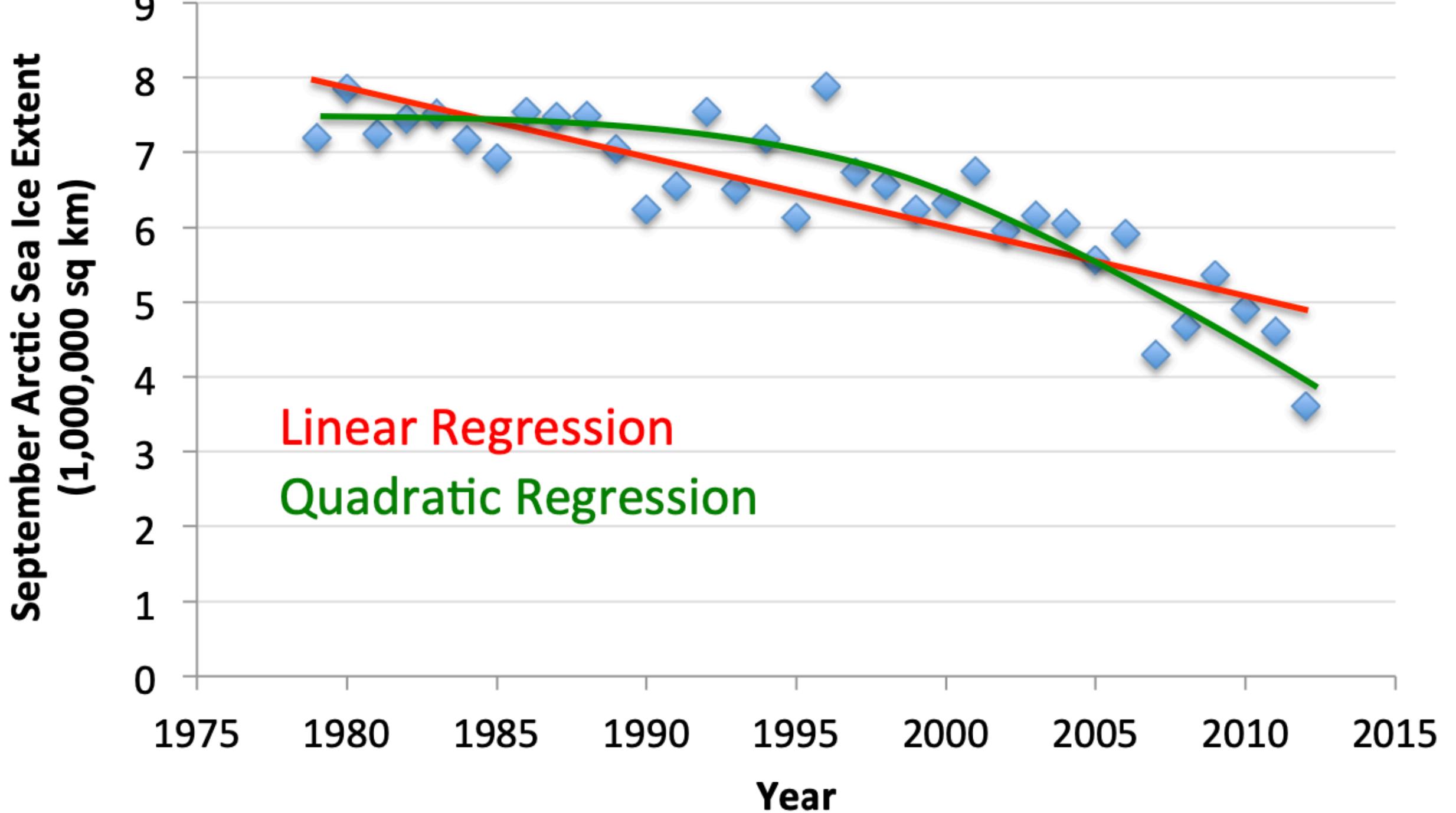


Regression

- Given:

Data $X = \{x_1, \dots, x_n\}$ yang mana $x_i \in \mathbb{R}_d$

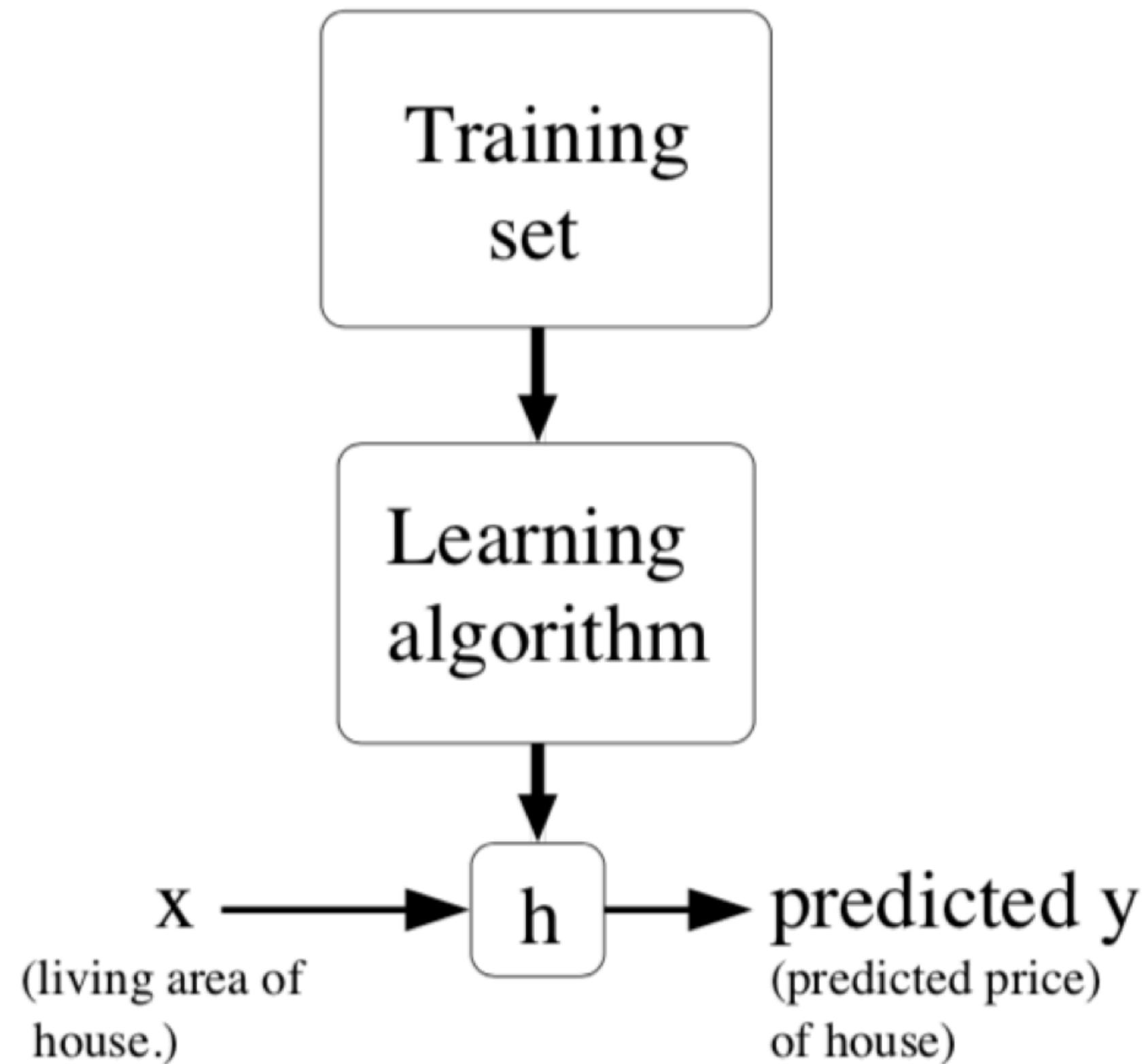
Label $y = \{y_1, \dots, y_n\}$ yang mana $y_i \in \mathbb{R}$



Data from G. WiH. Journal of Statisics Educalon, Volume 21, Number 1 (2013)

Linear Regression: Terms and Concepts

- Sample
- Feature
- Target
- Hypothesis
- Training Data
- Test Data
- Training Error & Test Error



Linear Regression: Hypothesis

Living area (feet ²)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
:	:	:

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

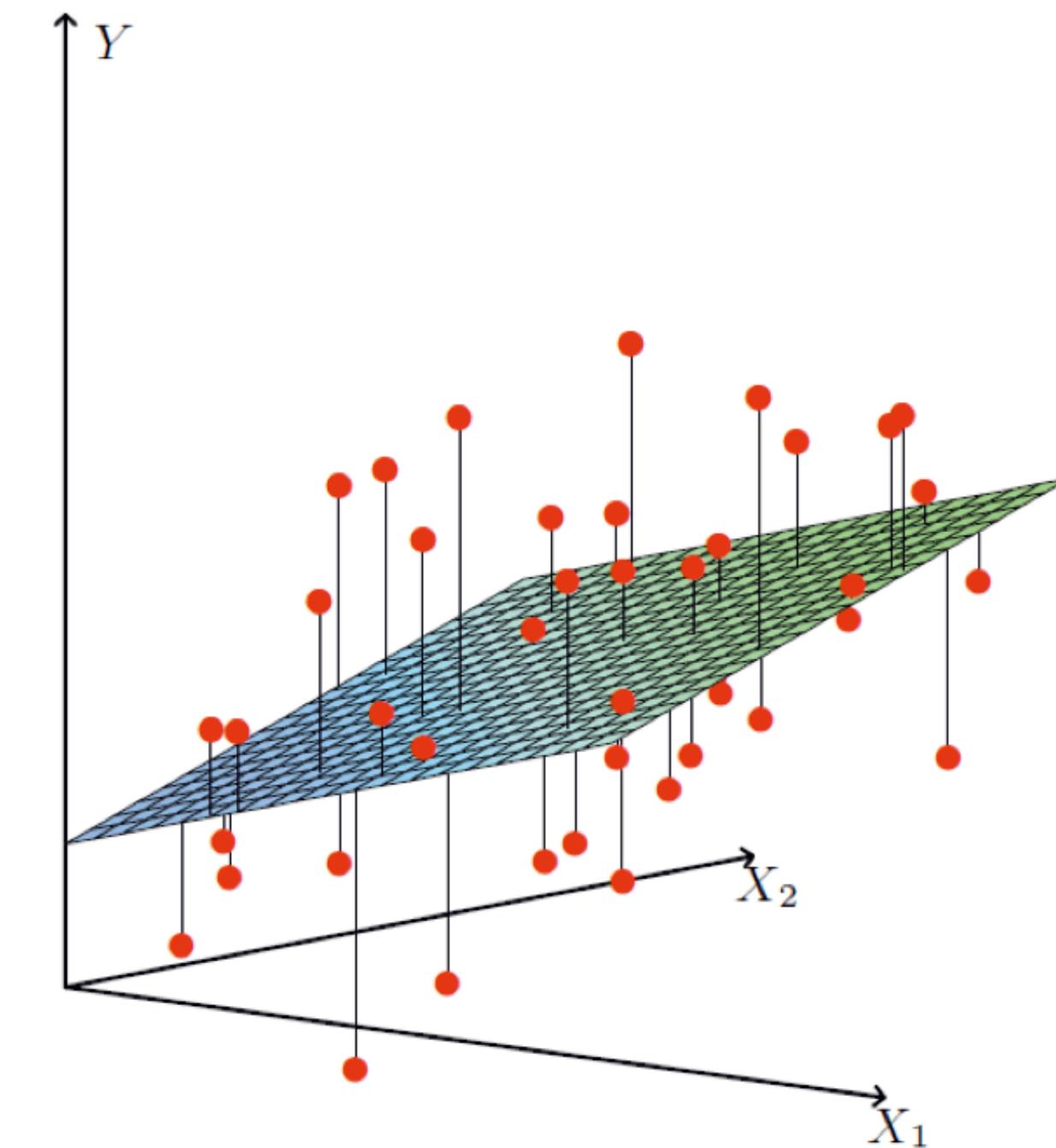
- $h(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$

- $x_1^{(i)}$: ukuran rumah ke- i dalam set
- $x_2^{(i)}$: jumlah kamar tidur dari rumah ke- i dalam training set
- $y^{(i)}$: harga rumah ke- i dalam training set
- θ_i 's: parameter (weights)

Linear Regression: Cost Function

- Sekarang, dengan adanya training set, bagaimana kita mempelajari parameter θ ? Salah satu metode yang masuk akal adalah membuat $h(x)$ mendekati y .
- **Cost Function (Loss Function)**

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$



Linear Regression: Cost Function

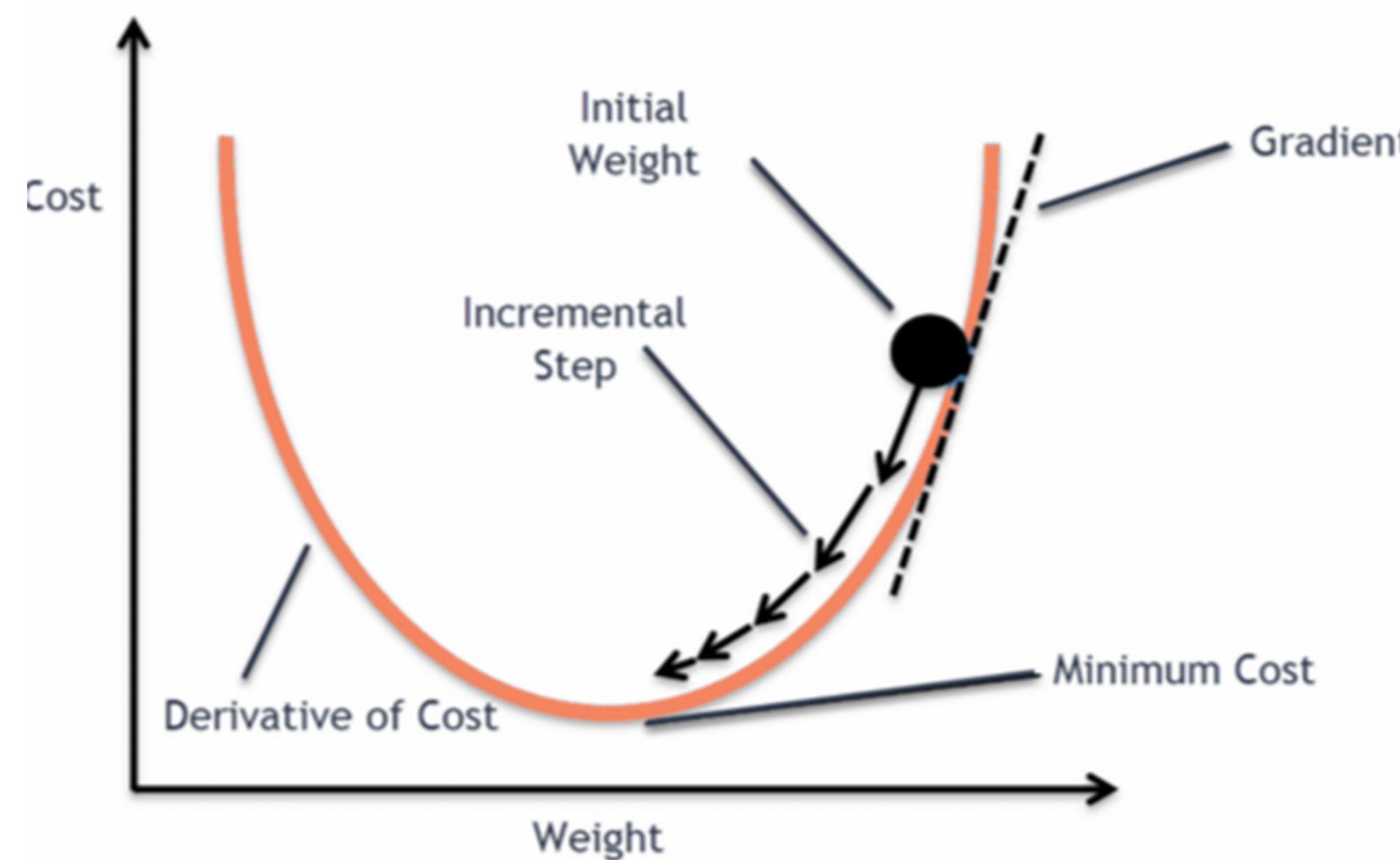
- Hypothesis:
$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$
- Parameters: $\theta_0, \theta_1, \theta_2$
- Cost Function:
$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$
- Goal: Minimize $J(\theta)$

Linear Regression: Gradient Descent



Gradient descent adalah sebuah algoritma optimisasi yang sering digunakan dalam pembelajaran mesin untuk melatih model dengan fokus pada meminimalkan **cost function** $J(\theta)$, yang berhubungan dengan parameter-parameter θ . Fungsi biaya ini pada dasarnya adalah kesalahan antara prediksi model $h_{\theta}(x)$ dan data sebenarnya y .

Linear Regression: Gradient Descent



- Jika gradien (kemiringan) cost function bernilai positif: Menaikkan nilai parameter akan meningkatkan cost. Jadi, untuk mengurangi cost, kita perlu menurunkan nilai parameter tersebut.
- Sebaliknya, jika gradien cost function bernilai negatif: Menurunkan nilai parameter justru akan menambah cost. Dalam situasi ini, agar cost tetap rendah, nilai parameter sebaiknya tidak dikurangi.

What is Gradient?

- Gradien dari suatu fungsi, yang dilambangkan sebagai ∇f , adalah sekumpulan nilai yang menunjukkan **turunan parsial** dari fungsi tersebut untuk setiap variabelnya.
- Jika fungsi tersebut memiliki banyak variabel, seperti $f(\theta_0, \theta_1, \theta_2, \dots, \theta_n)$ maka gradiennya terdiri dari semua turunan parsial terhadap masing-masing variabel tersebut. Gradien ini membantu kita mengetahui arah perubahan fungsi agar bisa menurunkan nilainya secara optimal.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial \theta_0} \\ \frac{\partial f}{\partial \theta_1} \\ \frac{\partial f}{\partial \theta_2} \\ \vdots \\ \frac{\partial f}{\partial \theta_n} \end{bmatrix}$$

Gradient Descent: Parameters Update Rule



- Kita sudah tahu bahwa parameter akan diperbarui secara bertahap (iteratif) dalam arah berlawanan dengan gradien $\nabla J(\theta)$ selama proses pembelajaran berlangsung. Dengan kata lain, kita memperbaiki parameter dengan bergerak ke arah yang mengurangi nilai cost function.
- aturan pembaruan untuk parameter:

$$\theta := \theta - \alpha \nabla J(\theta)$$

- α adalah learning rate atau tingkat pembelajaran, yang menentukan seberapa besar langkah yang diambil setiap kali parameter diperbarui.
- Nilai α **berkisar antara 0 hingga 1** ($0 < \alpha < 1$). Artinya, semakin kecil nilai α , semakin kecil perubahan setiap langkah, sedangkan nilai yang lebih besar membuat perubahan parameter lebih cepat.

Gradient Descent: Parameters Update Rule

- Update Rule in our case:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

Misal kita hanya memiliki satu contoh data training yang terdiri dari pasangan (x, y) :

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2 \\ &= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\ &= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^n \theta_i x_i - y \right) \\ &= (h_\theta(x) - y) x_j\end{aligned}$$

- Untuk satu contoh data pelatihan, ini memberikan aturan pembaruan

$$\theta_j := \theta_j + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}$$

Linear Regression: Gradient Descent

Batch Gradient Descent adalah metode pembaruan parameter di mana setiap langkah pembaruan menggunakan seluruh data training.

Pros:

- Karena menggunakan seluruh data, perhitungan gradien lebih **akurat** dan stabil.
- Cocok untuk dataset berukuran kecil hingga sedang

Cons:

- Memakan **waktu dan sumber daya** yang besar jika dataset sangat besar, karena harus melihat **semua contoh** pada setiap langkah.
- **Kurang efisien** dibandingkan metode lain seperti **Stochastic Gradient Descent (SGD)**, yang memperbarui parameter setelah melihat satu contoh saja.

Linear Regression: Gradient Descent

Batch Gradient Descent

```
repeat until convergence {  
     $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})$   
     $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$   
}
```

Linear Regression: Gradient Descent

Stochastic Gradient Descent adalah cara untuk memperbarui parameter model dengan hanya menggunakan satu contoh data pelatihan pada setiap langkah.

Pros:

- **Lebih cepat** mencapai nilai minimum dibandingkan dengan **Batch Gradient Descent**, terutama saat bekerja dengan dataset besar.
- **Efisien** karena tidak perlu menunggu semua data diproses untuk melakukan pembaruan.
- **Adaptif** untuk data streaming atau dataset dinamis yang terus bertambah.

Cons:

- Karena hanya melihat satu contoh data pada setiap langkah, **arah gradien mungkin tidak selalu tepat** dan bisa **berfluktuasi** di sekitar minimum.
- **Tidak stabil**: SGD mungkin tidak selalu menemukan nilai minimum global secara pasti, hanya mendekati (atau “close” to the minimum).

Linear Regression: Gradient Descent

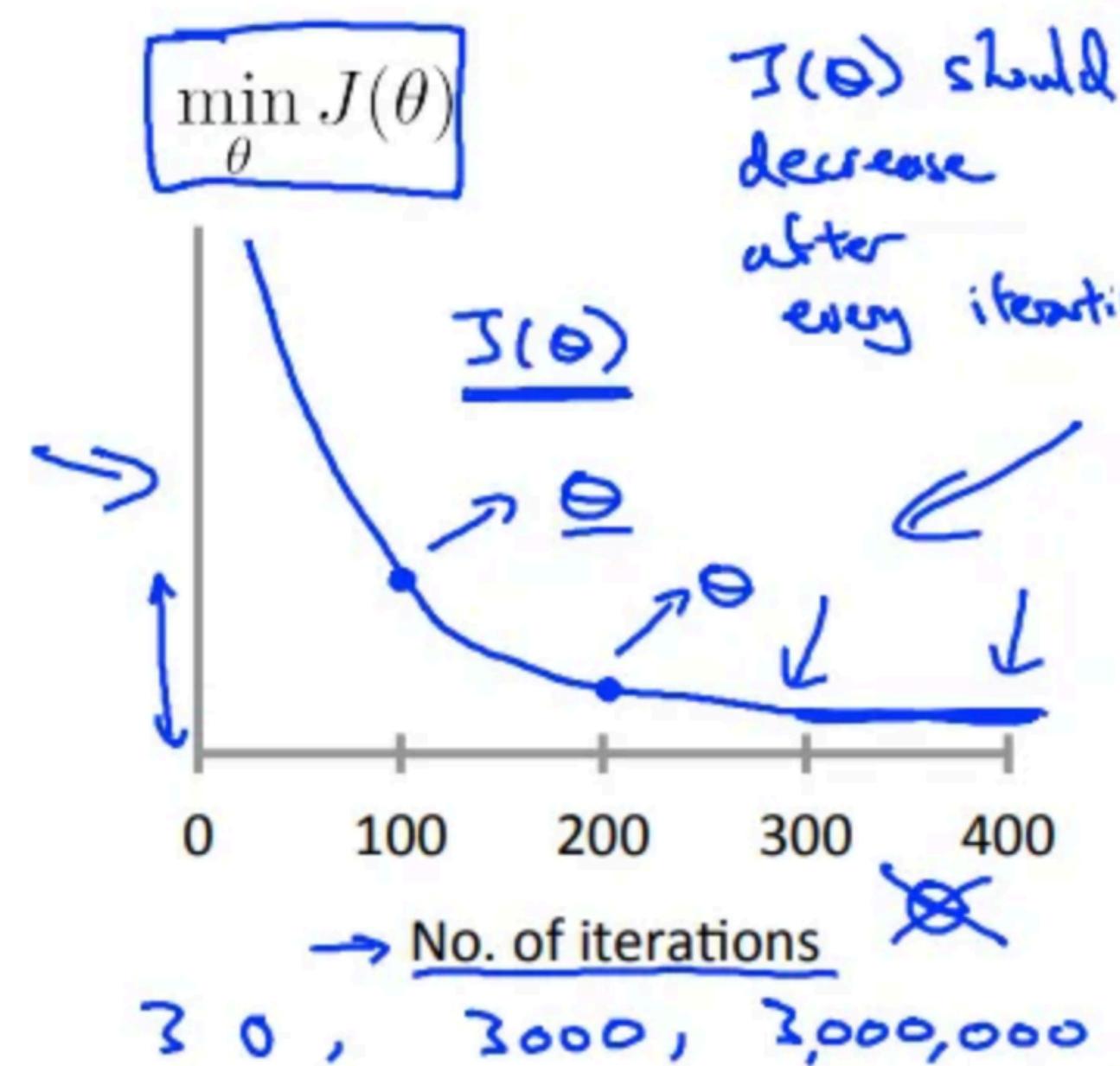


Stochastic Gradient Descent

```
Loop {  
    for i=1 to m, {  
         $\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}$       (for every j).  
    }  
}
```

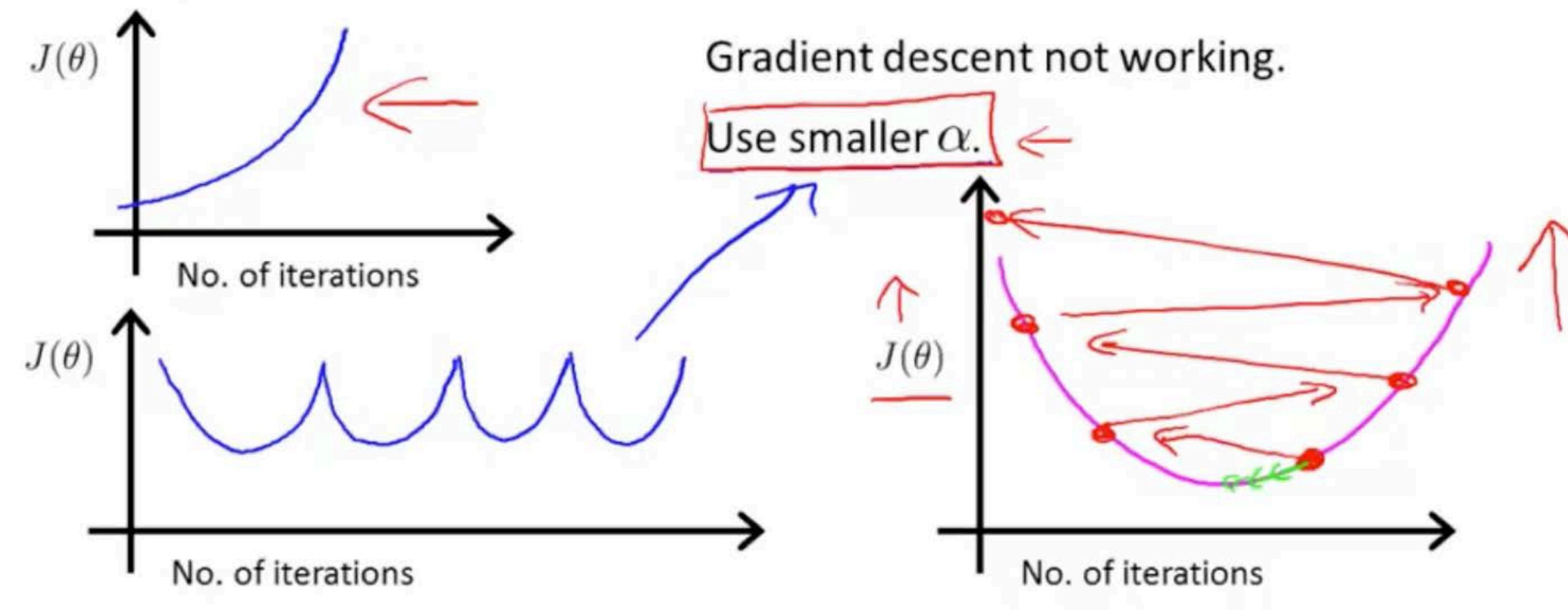
Linear Regression: Gradient Descent

- Pastikan Gradient Descent bekerja dengan tepat!



Linear Regression: Gradient Descent

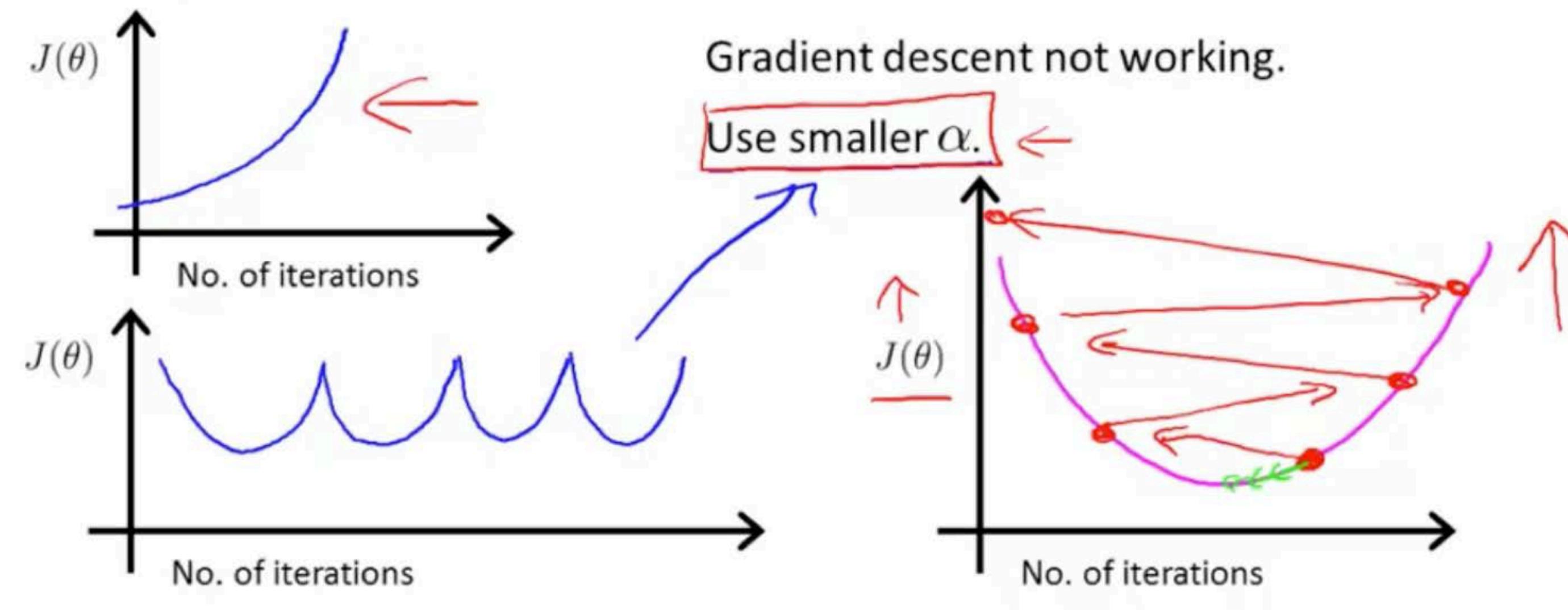
- Pastikan Gradient Descent bekerja dengan tepat!



- For sufficiently small α , $J(\theta)$ should decrease on every iteration. ←
- But if α is too small, gradient descent can be slow to converge.

Linear Regression: Gradient Descent

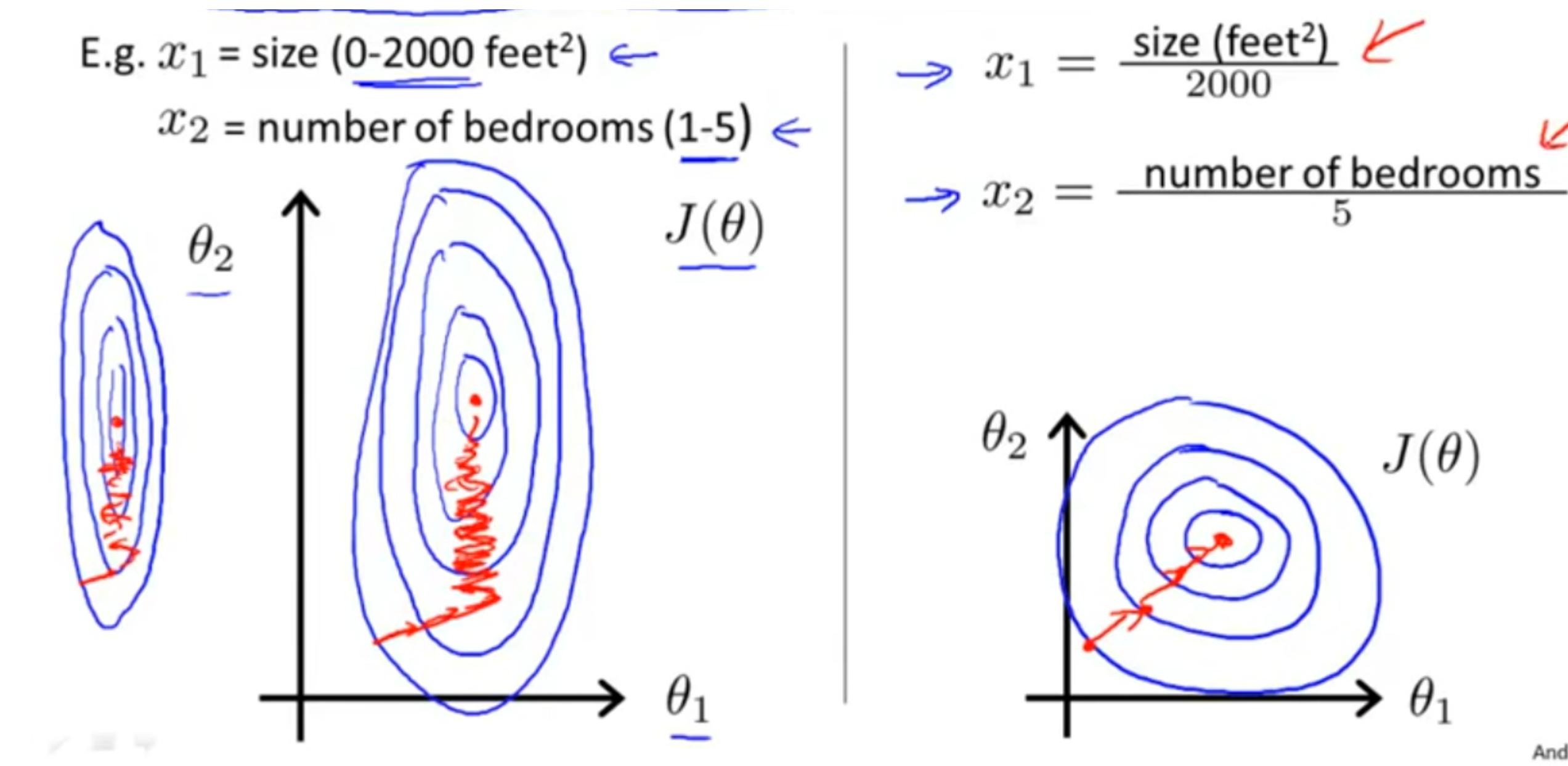
- Pastikan Gradient Descent bekerja dengan tepat!



- For sufficiently small α , $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Linear Regression: Feature Scaling

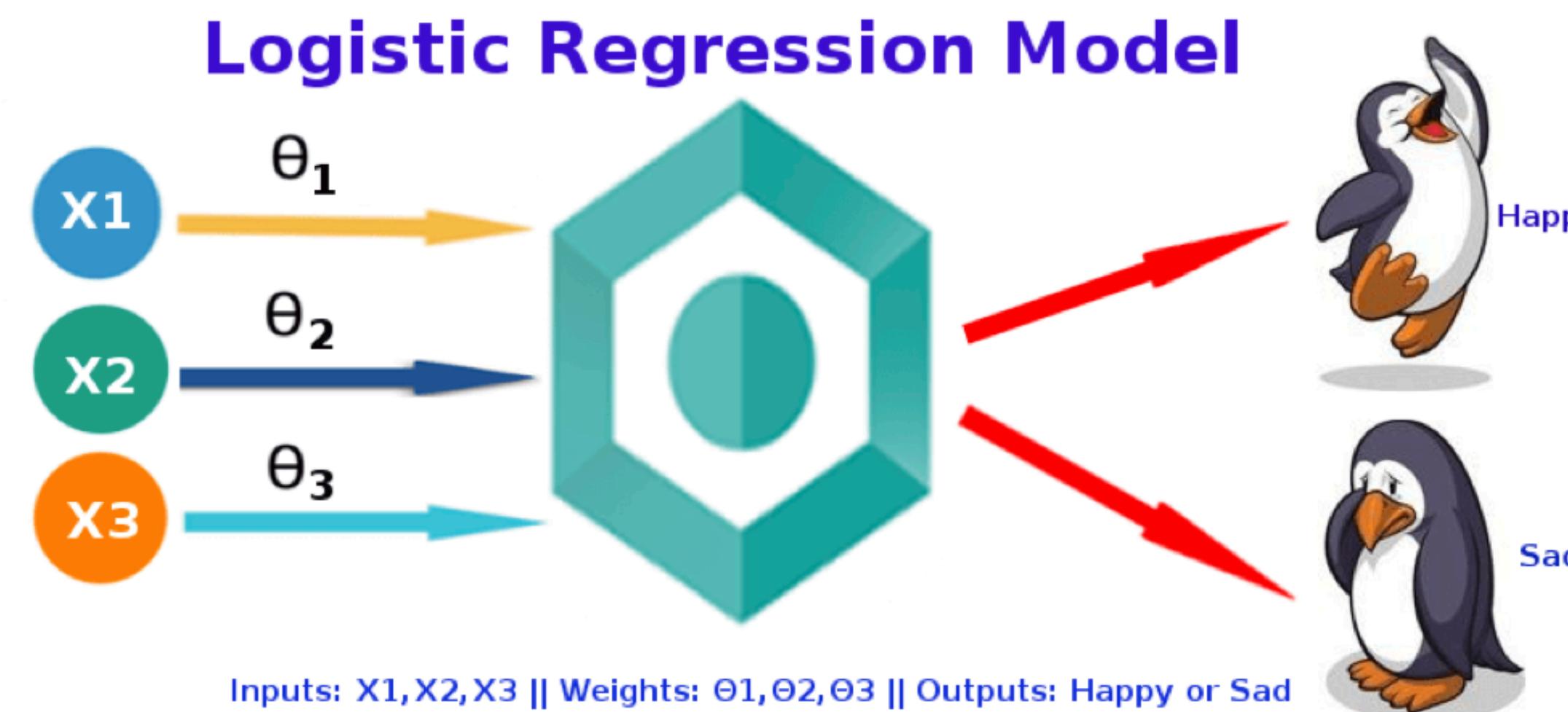
Feature scaling adalah cara menyesuaikan skala data sehingga semua fitur (variabel) memiliki rentang nilai yang sama. **Tujuan utamanya** adalah agar algoritma ML bekerja lebih efisien dan menemukan solusi lebih cepat.



Logistic Regression

Logistic Regression

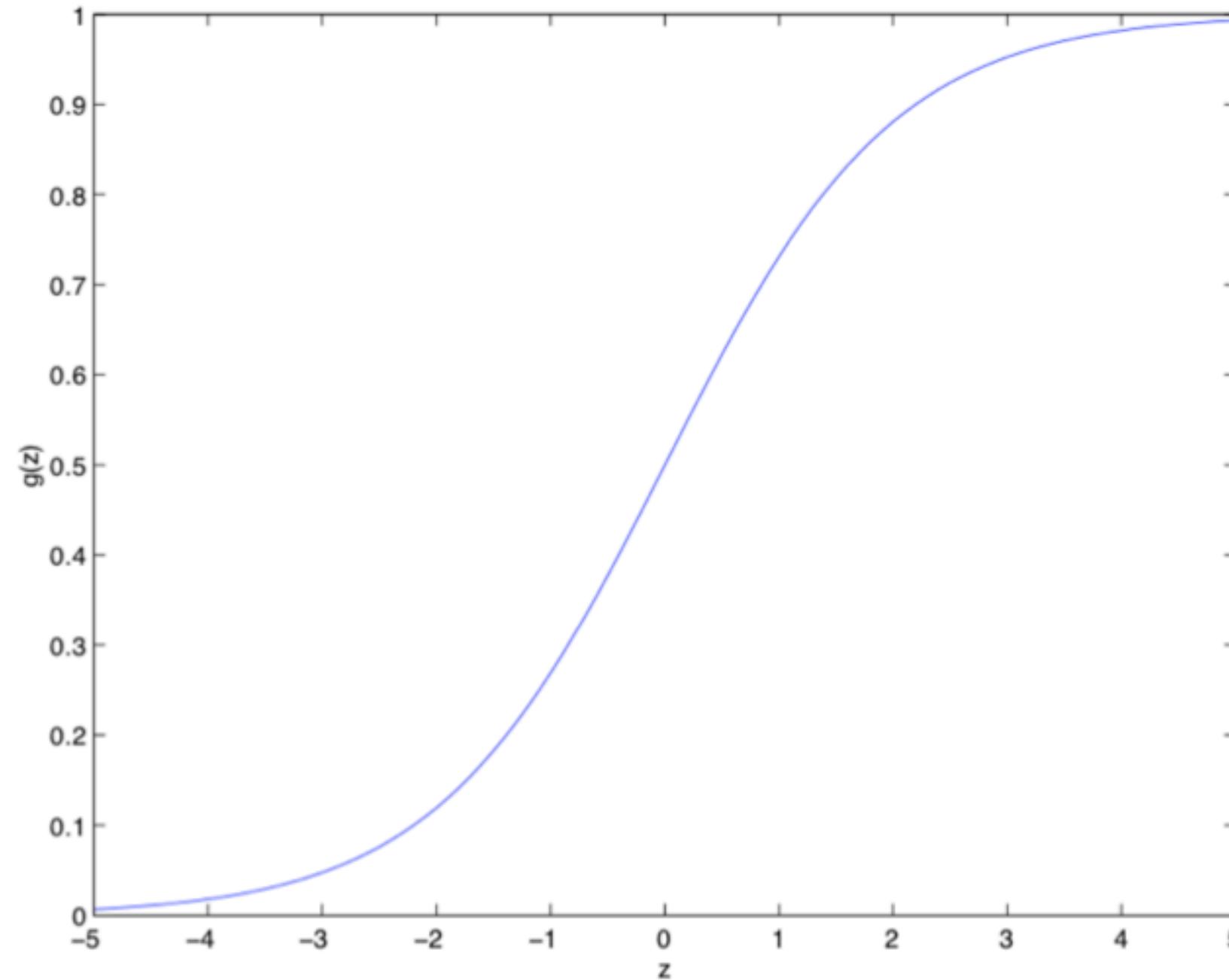
- **Logistic Regression** adalah metode yang mulai digunakan dalam ilmu biologi pada awal abad ke-20. Metode ini digunakan **ketika hasil yang ingin diprediksi berbentuk kategori** (misalnya ya atau tidak, benar atau salah).



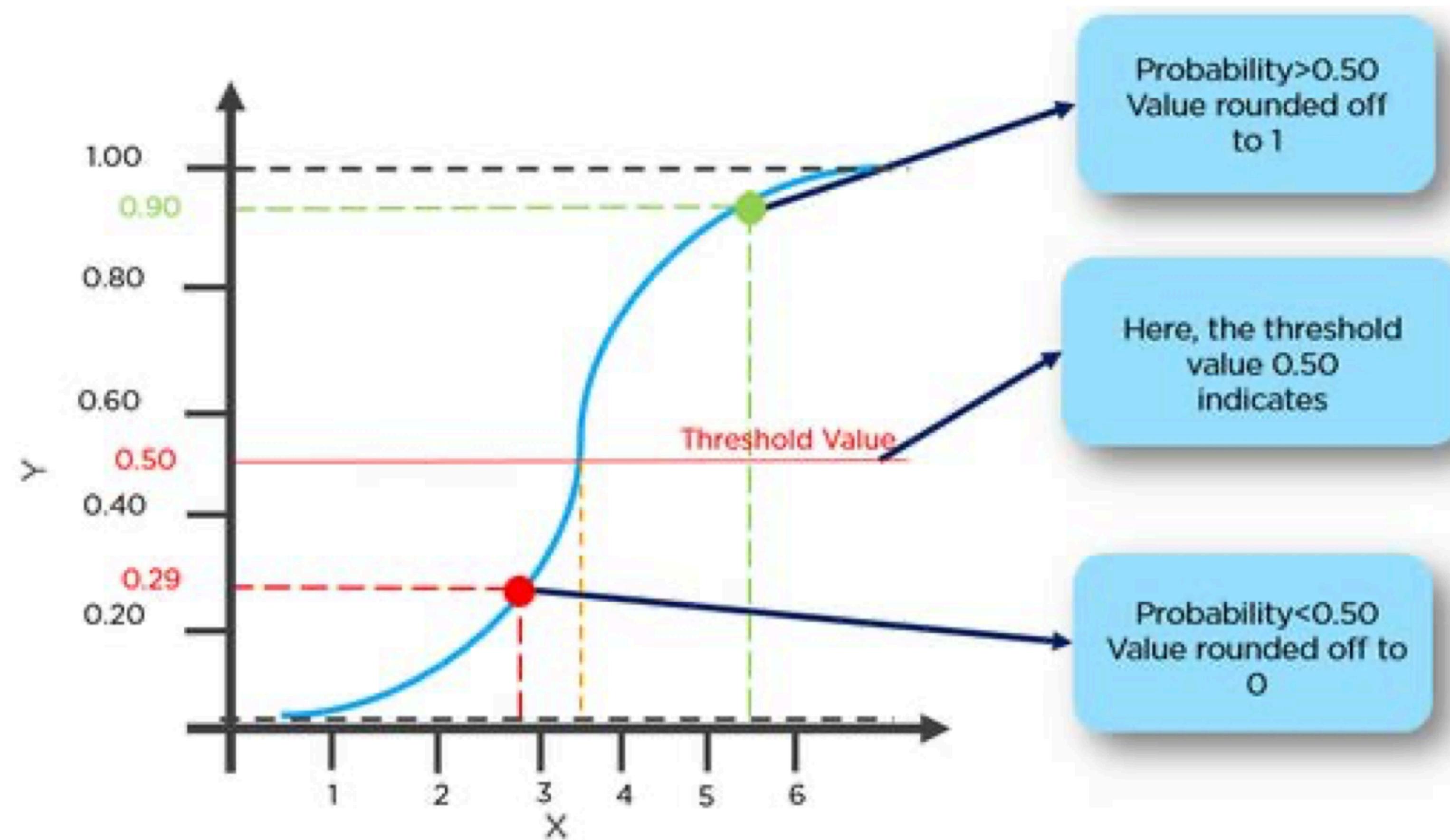
Logistic Regression: Hypothesis

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

• $g(z) = \frac{1}{1 + e^{-z}}$ disebut logistik function atau sigmoid function



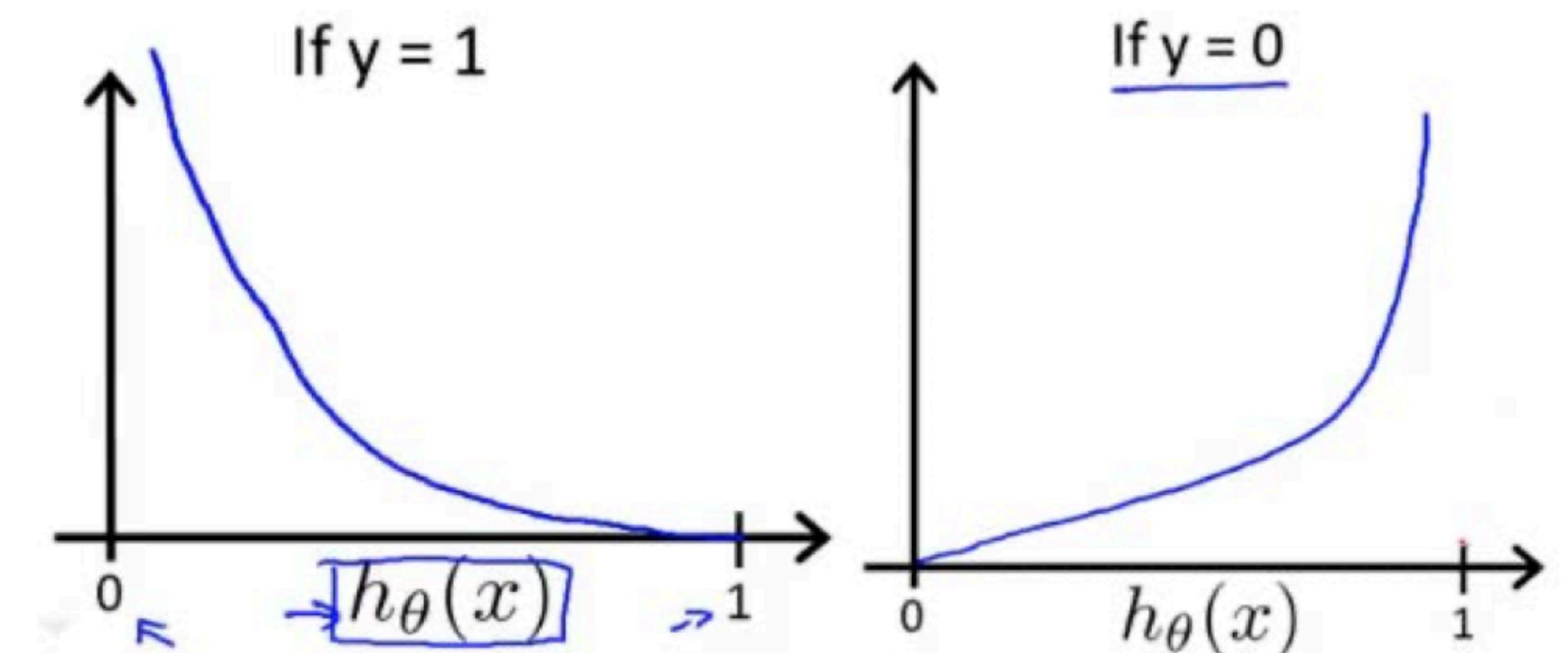
Logistic Regression: Hypothesis



Logistic Regression: Cost Function

- Cross-Entropy Loss adalah **cost function** yang digunakan dalam **Logistic Regression** untuk mengukur seberapa baik model memprediksi hasil yang benar.
- Tujuan dari fungsi ini adalah **meminimalkan kesalahan** antara prediksi model dan nilai yang sebenarnya.

$$Cost(h_{\theta}(x^{(i)}), y^{(i)}) = \begin{cases} -\log(h_{\theta}(x^{(i)})), & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x^{(i)})), & \text{if } y = 0 \end{cases}$$



Logistic Regression: Minimize Cost Function

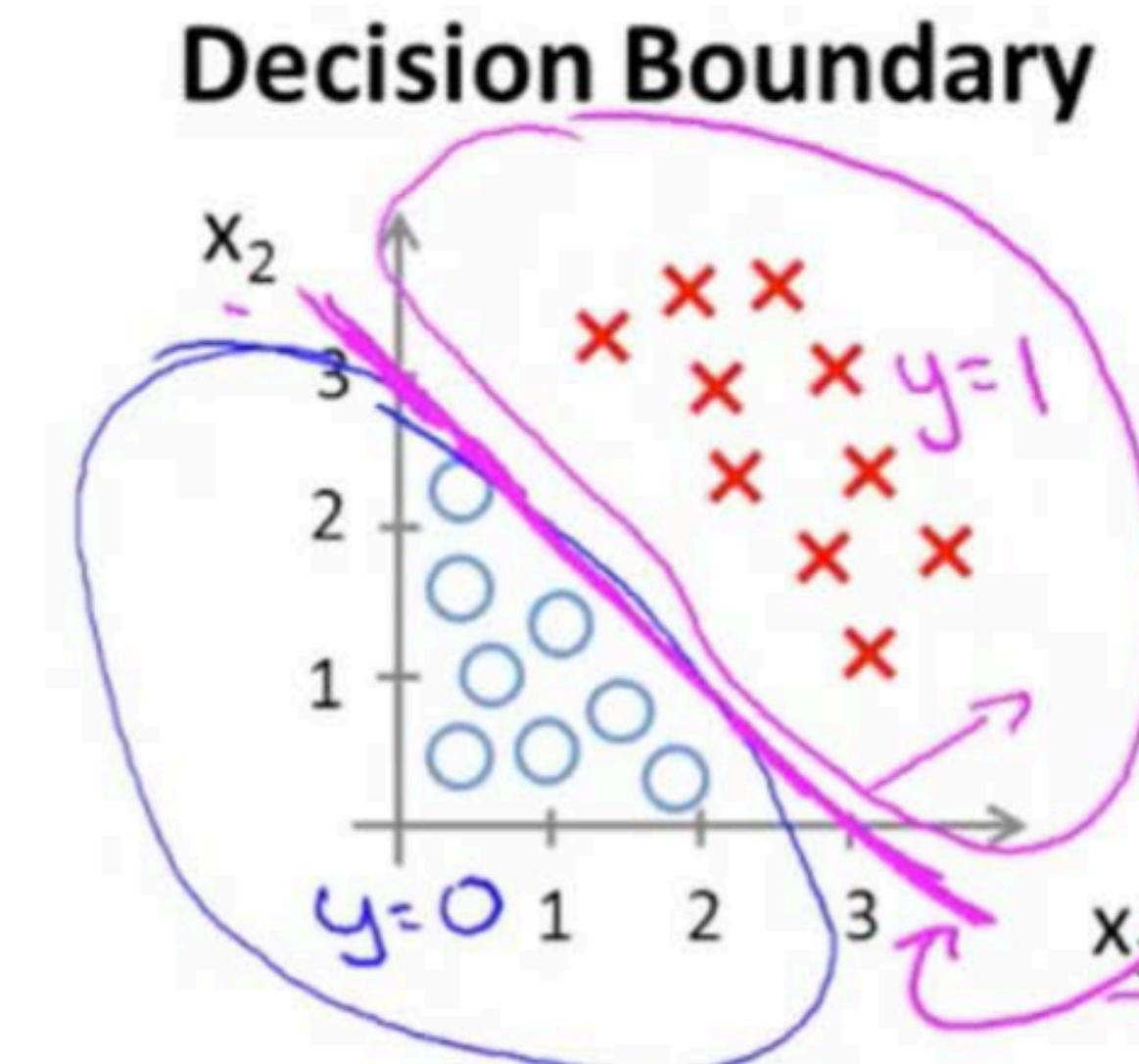


$$Cost(h_{\theta}(x^{(i)}), y^{(i)}) = \begin{cases} -\log(h_{\theta}(x^{(i)})), & \text{if } y = 1 \\ -\log(1 - h_{\theta}(x^{(i)})), & \text{if } y = 0 \end{cases}$$

- **Cross-Entropy Loss**

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m Cost(h_{\theta}(x^{(i)}), y^{(i)}) = -\frac{1}{m} [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

Logistic Regression: Decision Boundary



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

$$\Theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} \leftarrow$$

Predict " $y = 1$ " if $\underline{-3 + x_1 + x_2 \geq 0}$

$$\Theta^T x$$

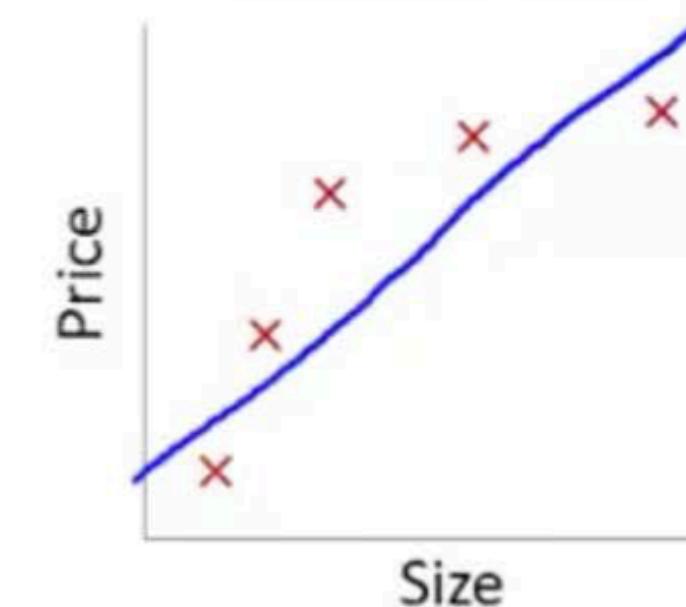
$$\underline{x_1 + x_2 \geq 3}$$

$$\begin{aligned} & x_1, x_2 \\ & \rightarrow h_{\theta}(x) = 0.5 \\ & x_1 + x_2 = 3 \end{aligned}$$

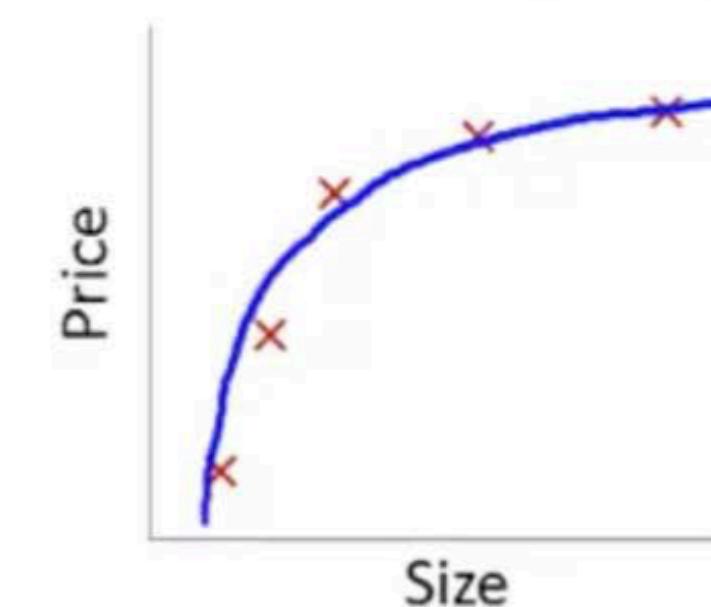
$$\begin{aligned} & x_1 + x_2 < 3 \\ & \rightarrow y = 0 \end{aligned}$$

Regularization: Overfitting

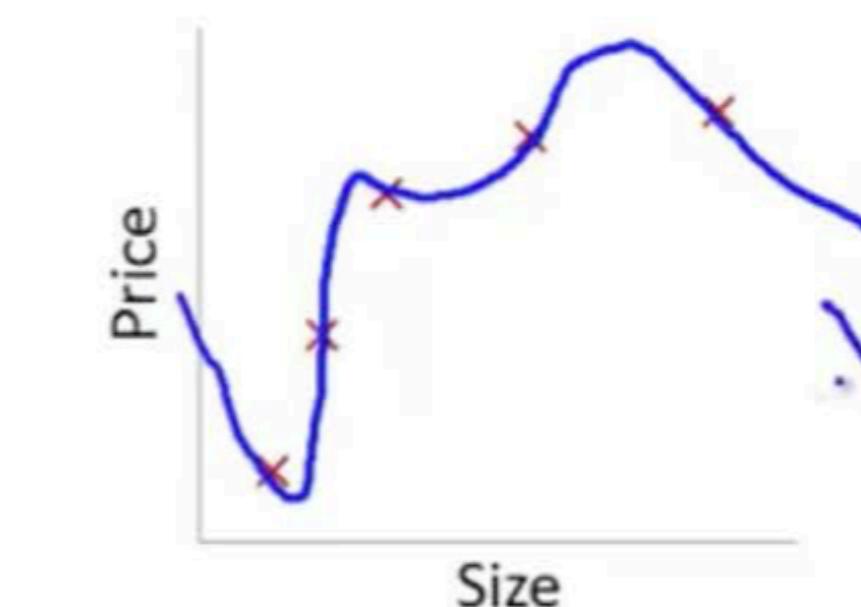
Example: Linear regression (housing prices)



$\rightarrow \theta_0 + \theta_1 x$
 "Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$
 "Just right"



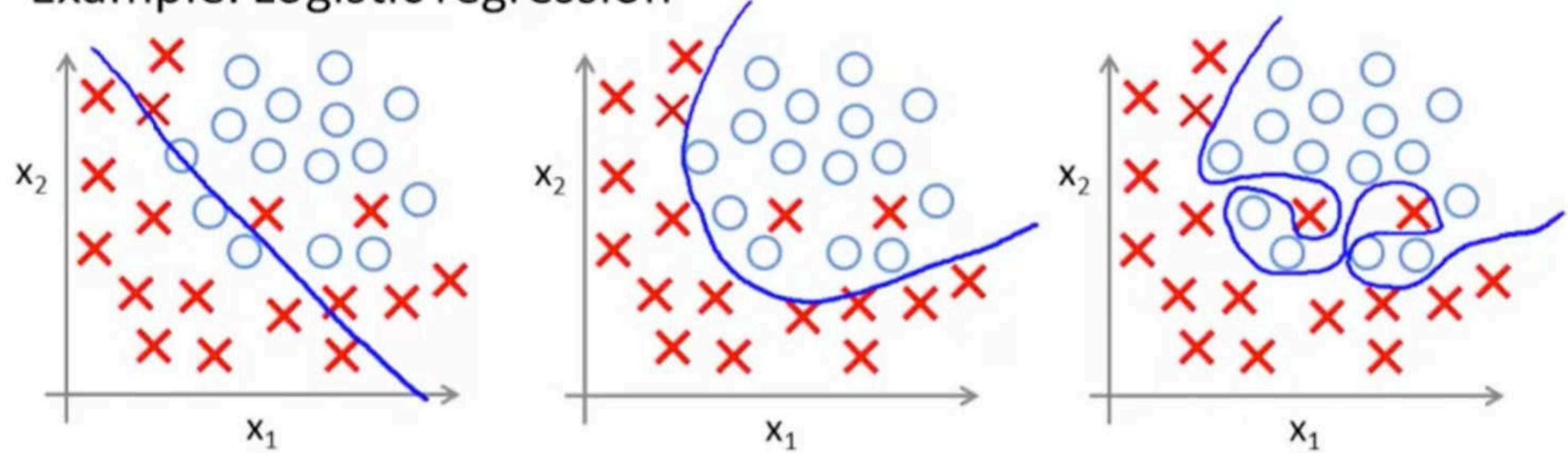
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
 "Overfit" "High variance"

- **Overfitting** terjadi ketika model terlalu fokus pada data training. Artinya, model bisa sangat baik dalam menyesuaikan pola pada data yang digunakan untuk belajar, tetapi gagal saat memprediksi data baru yang belum pernah dilihat sebelumnya.
- **Masalahnya:** Meski model tampil sempurna pada data pelatihan, ketika mencoba **memprediksi data baru** (contohnya harga rumah baru), hasilnya **tidak akurat**.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \approx 0$$

Regularization: Overfitting

Example: Logistic regression



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

(g = sigmoid function)



"Underfit"

$$g(\underline{\theta_0 + \theta_1 x_1 + \theta_2 x_2} \\ + \underline{\theta_3 x_1^2} + \underline{\theta_4 x_2^2} \\ + \underline{\theta_5 x_1 x_2}) \rightarrow$$

$$g(\underline{\theta_0 + \dot{\theta}_1 x_1 + \theta_2 x_1^2} \leftarrow \\ + \underline{\theta_3 x_1^2 x_2} + \underline{\theta_4 x_1^2 x_2^2} \\ + \underline{\theta_5 x_1^2 x_2^3} + \underline{\theta_6 x_1^3 x_2} + \dots)$$

Logistic Regression: Mengatasi Overfitting

1. Mengurangi jumlah feature

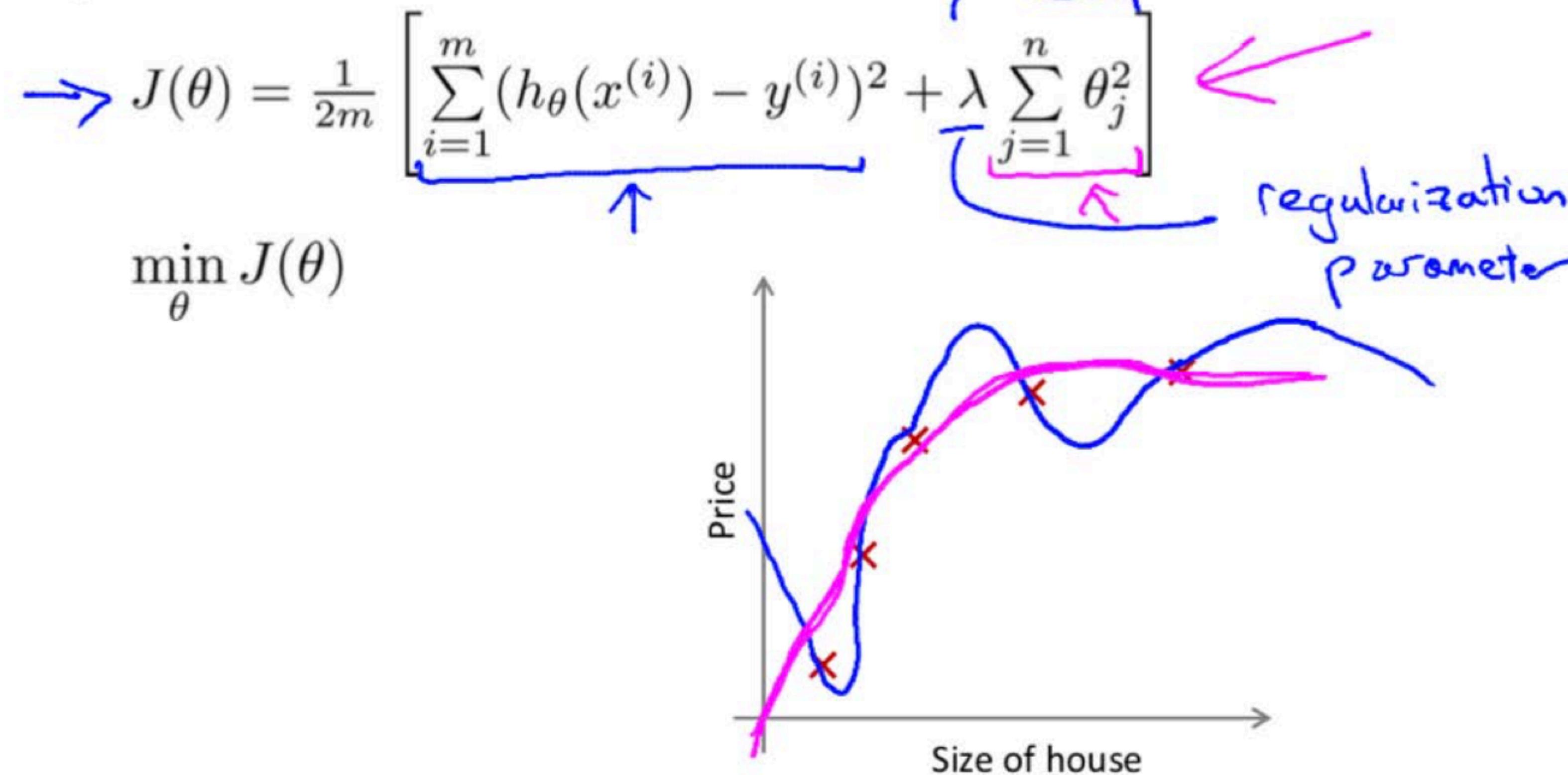
- **Pilih fitur secara manual:** Tentukan sendiri fitur mana yang paling relevan untuk digunakan.
- **Menggunakan algoritma pemilihan model:** Algoritma ini membantu memilih fitur yang tepat (akan dipelajari lebih lanjut).

2. Regularisasi

- Tetap gunakan semua fitur, tetapi kurangi nilai/magnitudo parameter θ_j . Ini mencegah model terlalu bergantung pada beberapa fitur tertentu.
- **Regularisasi bekerja dengan baik** jika ada banyak fitur, dan setiap fitur hanya memberikan kontribusi kecil dalam memprediksi hasil y .

Regularisasi

Regularization.



1. λ kecil: Model lebih fleksibel dan dapat mengikuti pola rumit dalam data, tapi berisiko **overfitting**.
2. λ besar: Parameter model semakin kecil, sehingga model menjadi lebih sederhana dan **kurang mungkin overfit**. Namun, jika λ terlalu besar, model bisa menjadi terlalu sederhana dan tidak cukup akurat (underfitting).

Regularisasi: Linear Regression

- Dalam regularized linear regression, kita memilih parameter θ untuk meminimalkan cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\theta}(x^{(i)}) - y^{(i)} \right)^2 + \lambda \sum_{j=1}^n \theta_j^2$$

- **Apa yang Terjadi Jika λ Terlalu Besar?**

Misalnya, kita menetapkan $\lambda = 10^{10}$ yaitu sangat besar. Apa dampaknya?

1. **Algoritma tetap berjalan**, tetapi memilih λ yang terlalu besar **bukan ide yang baik**.
2. Algoritma gagal menghilangkan overfitting dengan baik.
3. Algoritma **menghasilkan underfitting** (model menjadi terlalu sederhana sehingga bahkan tidak bisa memprediksi data pelatihan dengan baik).
4. **Gradient descent tidak akan konvergen** (proses pencarian solusi berhenti atau berjalan sangat lambat).

Regularisasi: Logistic Regression

- Dalam regularized logistic regression, kita memilih parameter θ untuk meminimalkan cost function:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

- **Mengurangi kompleksitas model** dengan **membatasi nilai parameter θ_j** , membantu mencegah **overfitting**.
- Jika λ terlalu besar, model bisa menjadi **underfit** (terlalu sederhana). Jika terlalu kecil, model bisa **overfit** (terlalu rumit).

**SELAMAT
BELAJAR**