暑期培训

CS229 Lecture 2

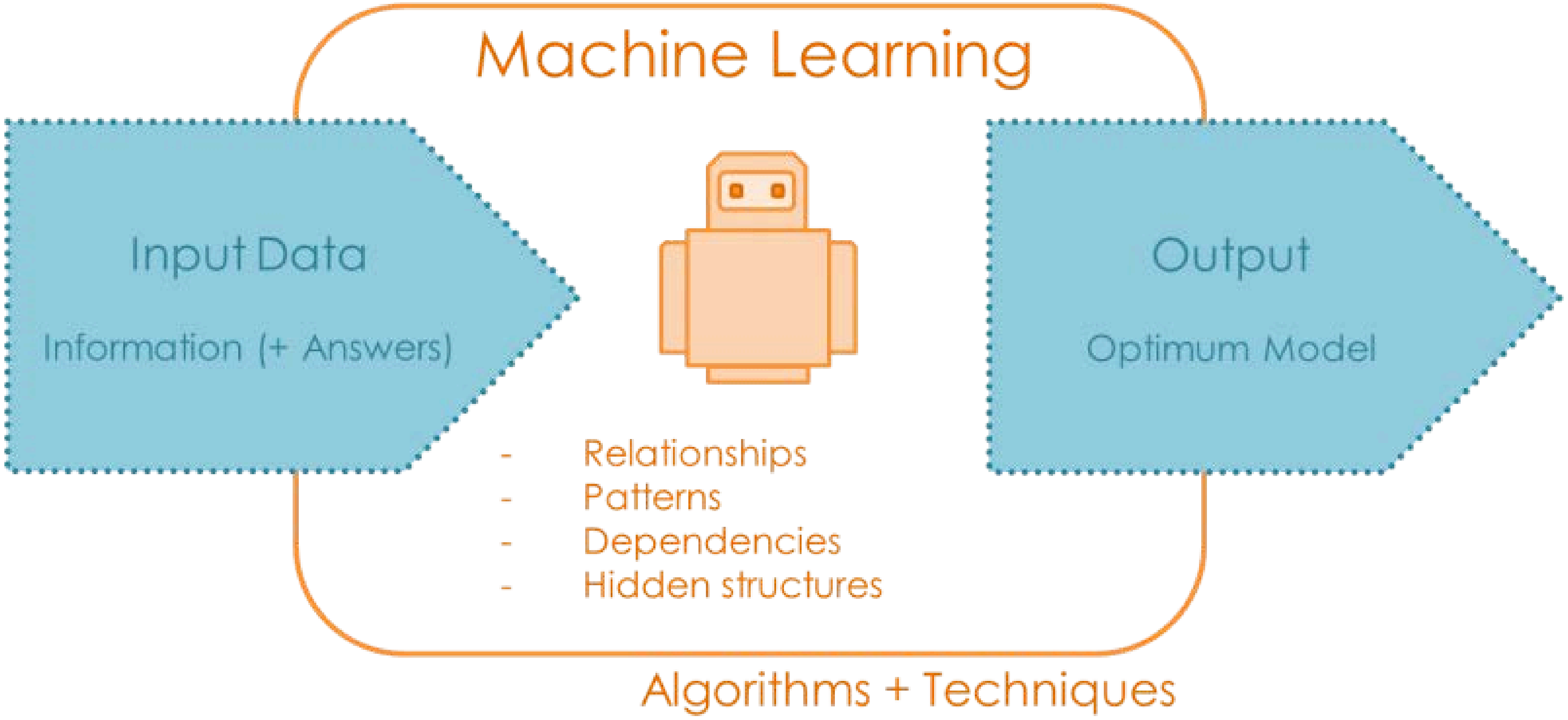
Machine Learning

顾晓玲
guxl@hdu.edu.cn

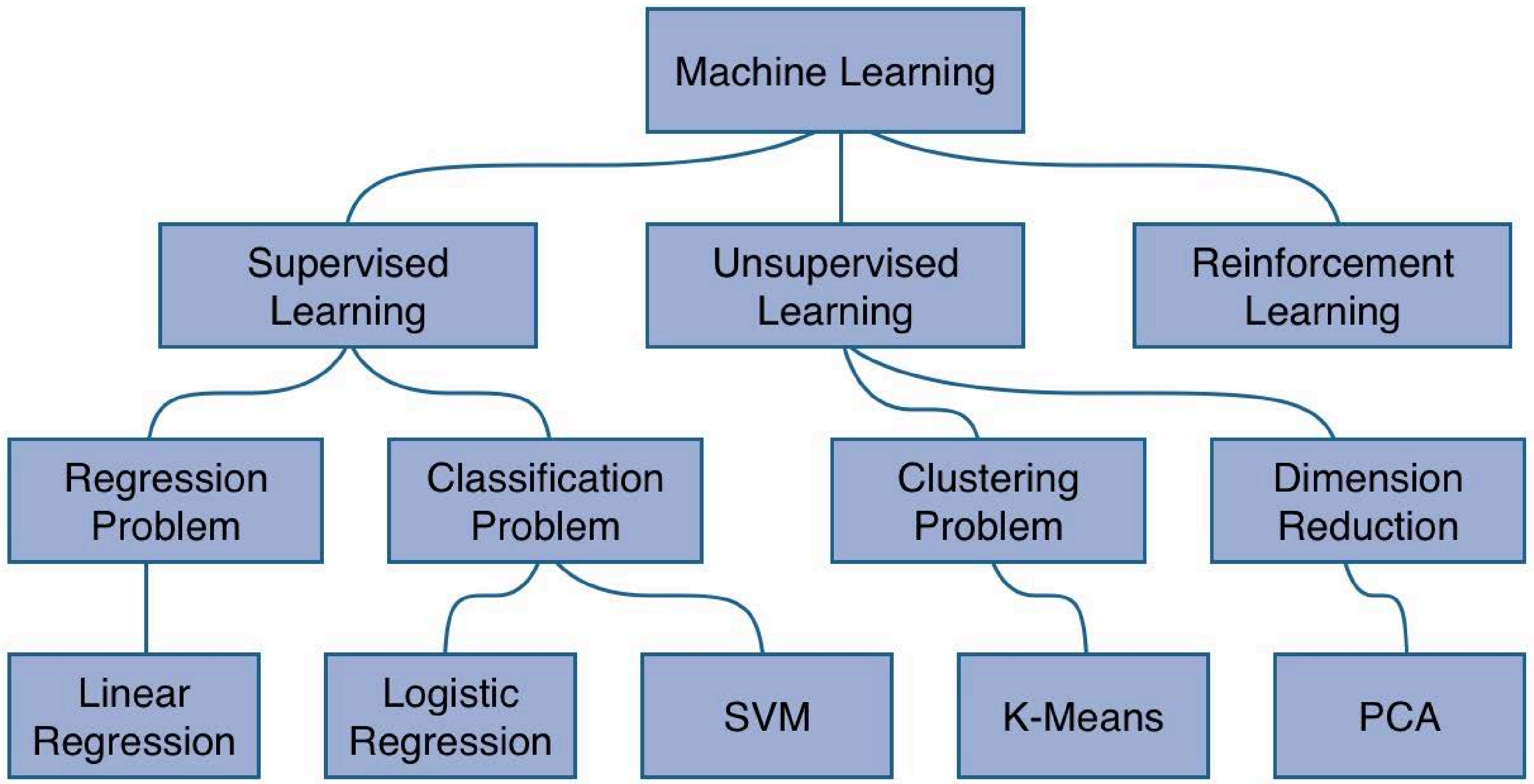
Outlines

- Machine Learning: Overview
- Linear Regression
- Classification
- Logistic Regression
- Regularization
- Perceptron

1 Machine Learning：Overview



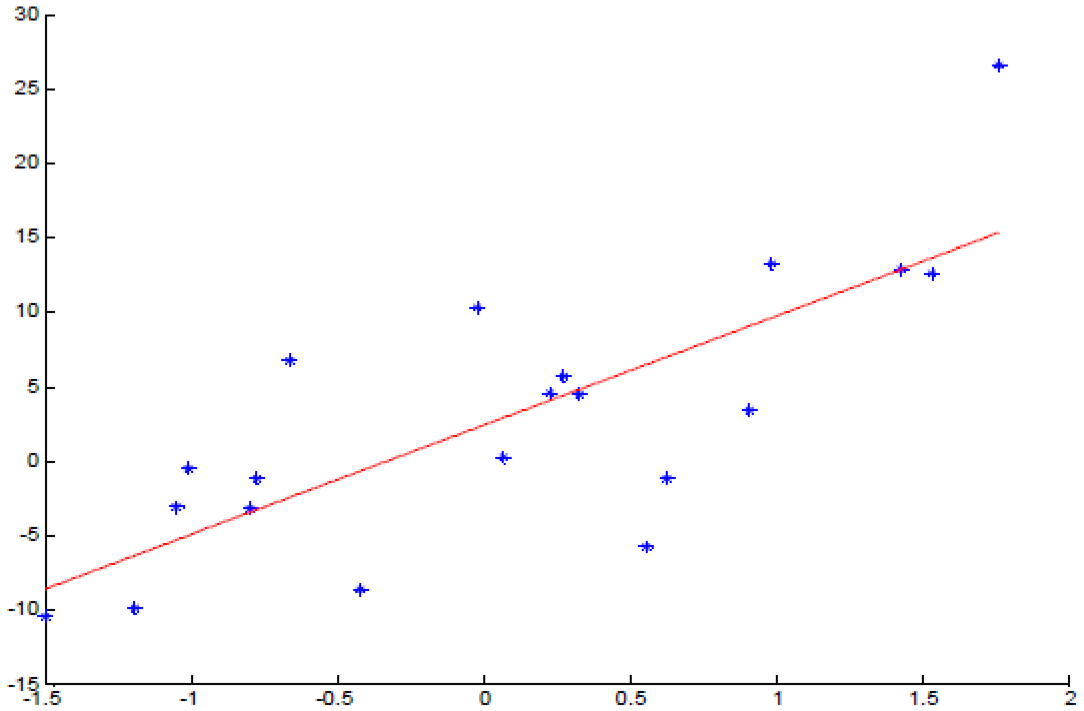
1 Machine Learning: Overview



2 Linear Regression: 1 Dimensional Input

Linear regression. Example

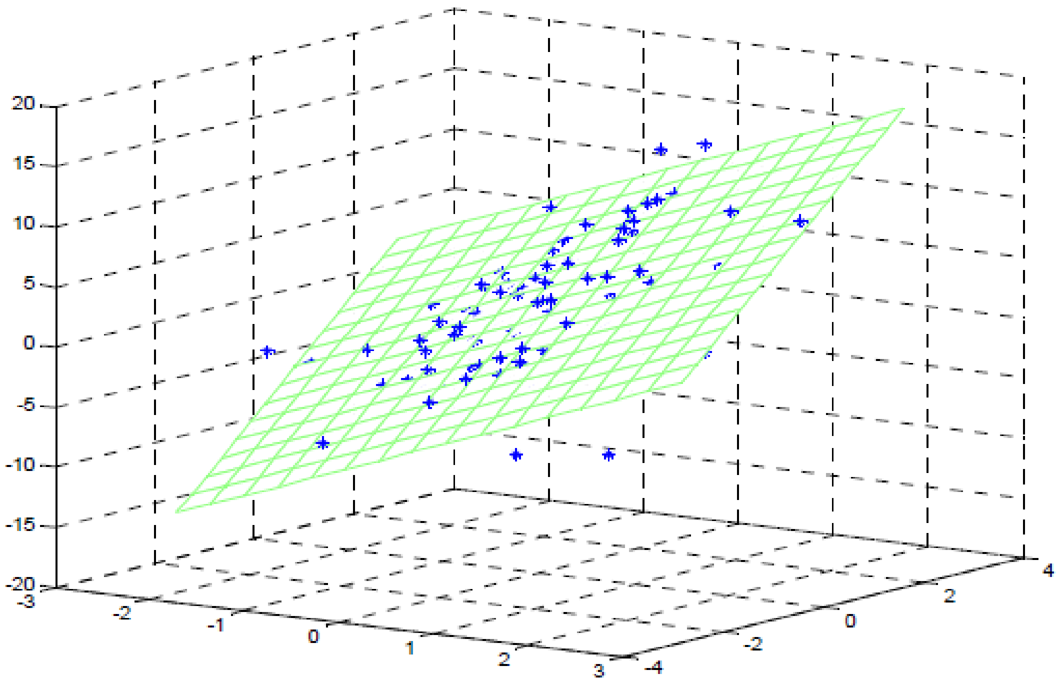
- 1 dimensional input $\mathbf{x} = (x_1)$



2 Linear Regression: 2 Dimensional Input

Linear regression. Example.

- 2 dimensional input $\mathbf{x} = (x_1, x_2)$



2 Linear Regression: An Example

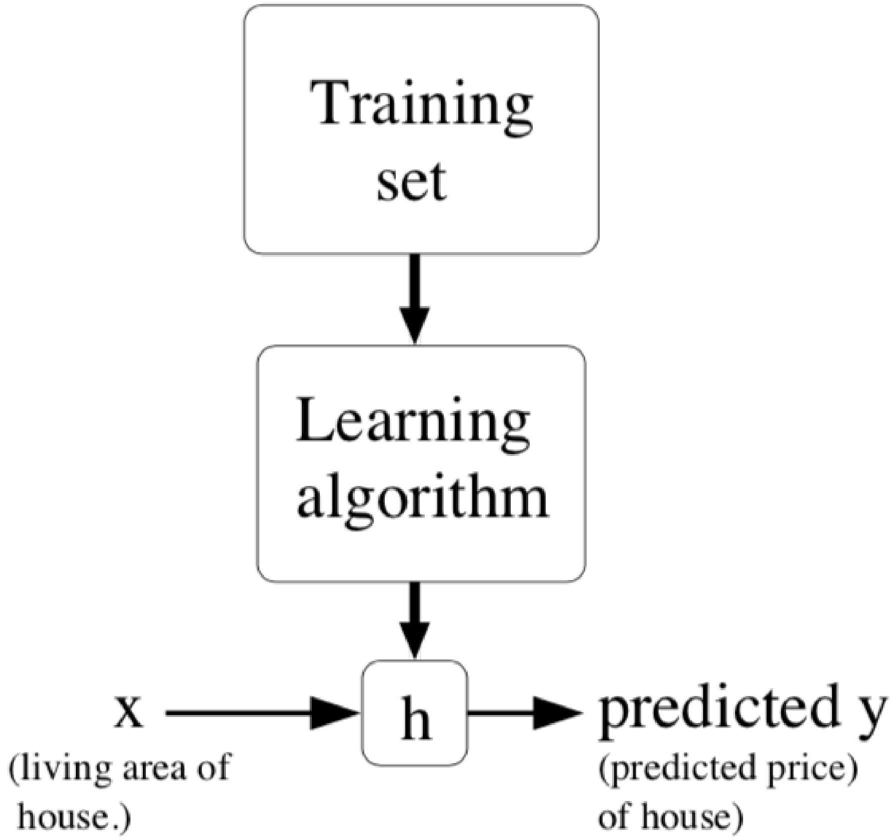
- Suppose we have a dataset giving the living areas, number of bedrooms and prices of 200 houses from a specific region:

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

- Given data like this, how can we learn to predict the prices of other houses, as a function of the size of their living areas and the number of bedrooms?

2 Linear Regression: Terms and Concepts

- Sample, Example
- Feature
- Target
- Hypothesis
- Training Data (Training Set)
- Test data (Test Set)
- Training Error & Test Error



2 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
$\vdots$	$\vdots$	$\vdots$

$x_1^{(i)}$: the living area of the i-th house in the training set

$x_2^{(i)}$: the number of bedrooms of the i-th house in the training set

$y^{(i)}$: the price of the i-th house in the training set

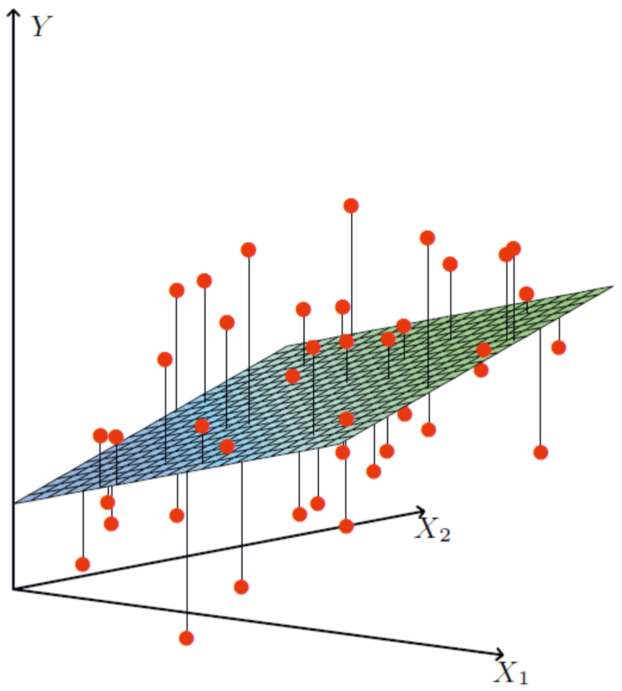
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

θ_i's : **parameters (weights)**

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

2 Linear Regression: Cost Function

- Now, given a training set, how do we learn the parameters θ? One reasonable method seems to be to make h(x) close to y.



- **Cost Function (Loss Function):**

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

2 Linear Regression: Cost Function

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

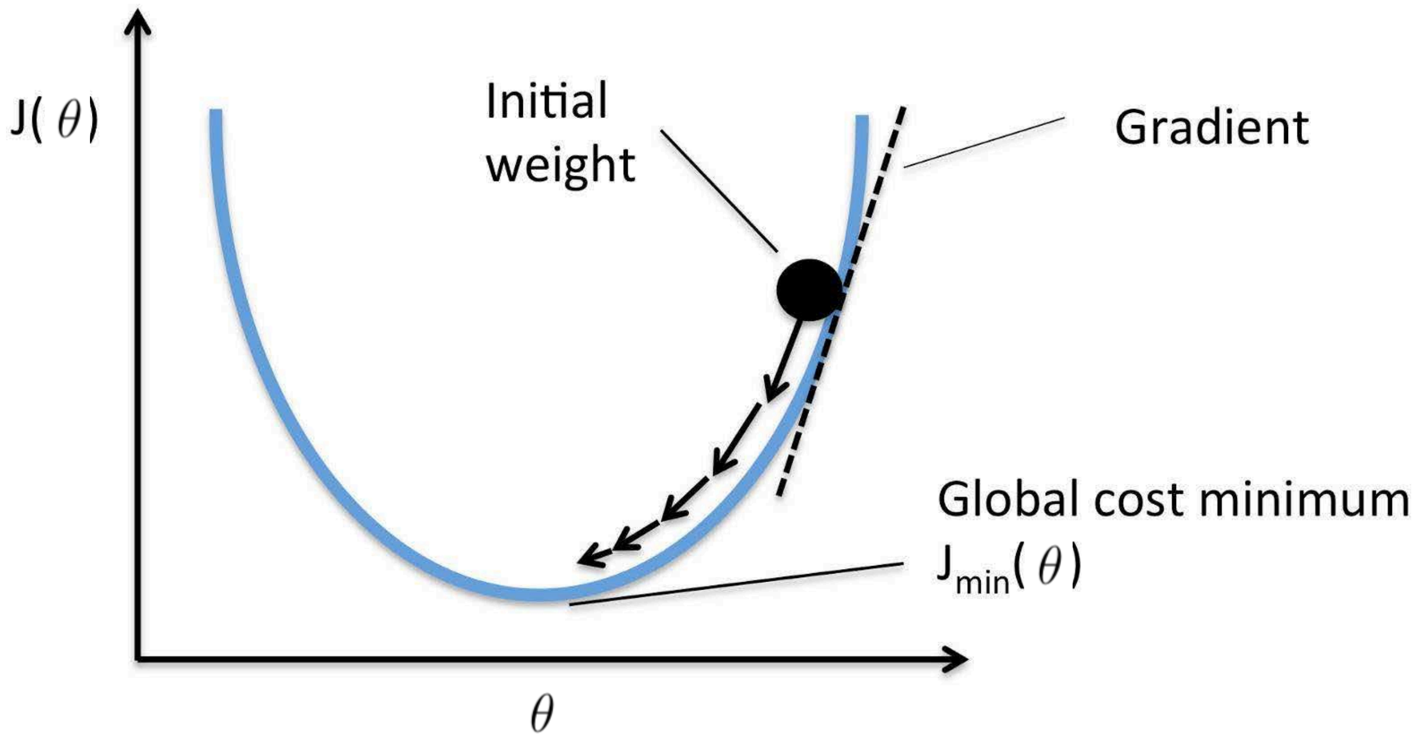
Parameters: θ_0 , θ_1 , θ_2

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

Goal: Minimize $J(\theta)$

2 Linear Regression: Gradient Descent

- **Gradient Descent Algorithm**



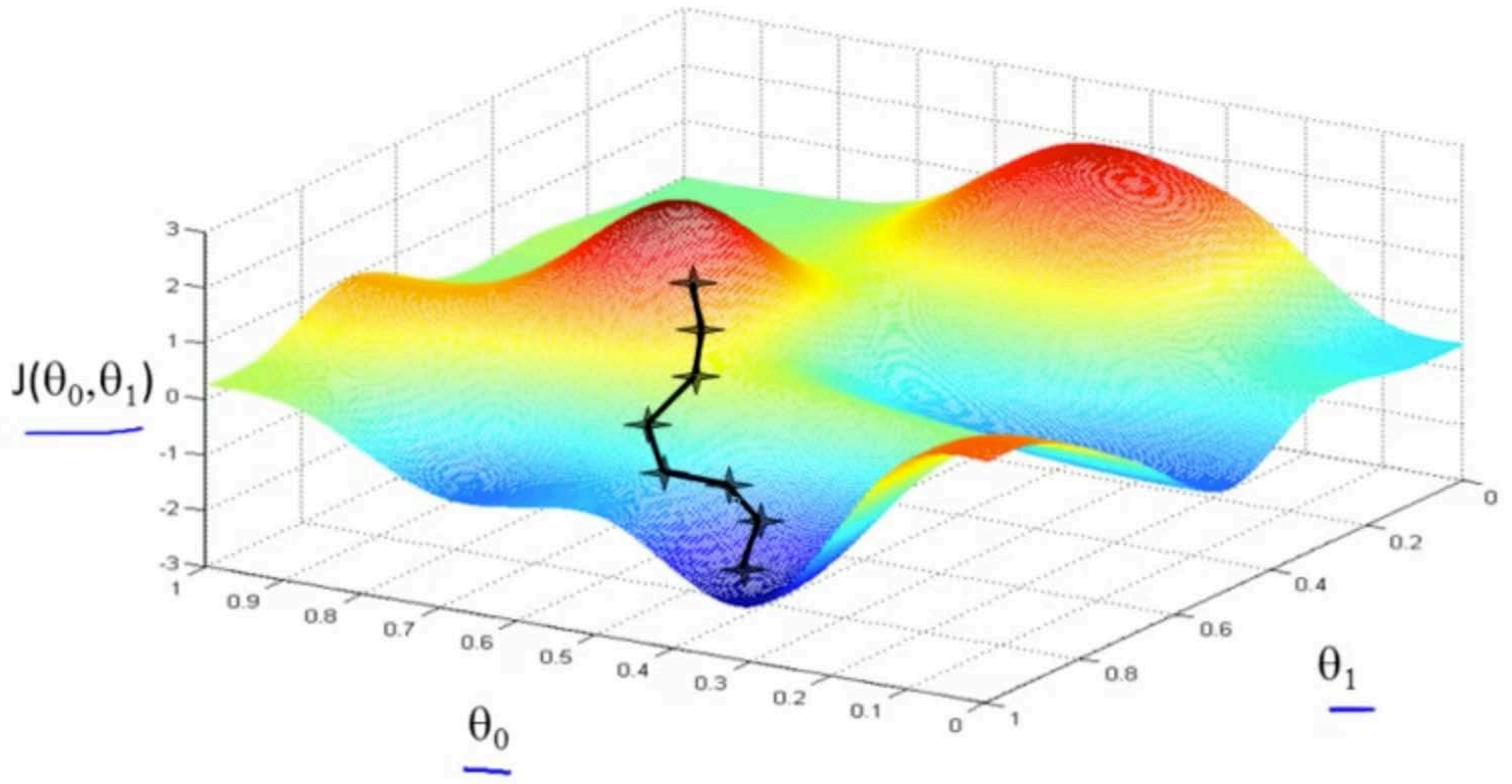
Update Rule:

$$\theta := \theta - \alpha \nabla_{\theta} J(\theta)$$

In Our Case:

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

2 Linear Regression: Gradient Descent



2 Linear Regression : Gradient Descent

- Let’s assume we have only one training example (x, y):

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

- For a single training example, this gives the update rule:

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

2 Linear Regression: Gradient Descent

- **Batch Gradient Descent**: looks at every example in the entire training set on every step.

Old Version

$$\text{Repeat until convergence } \{$$
$$\theta_j := \theta_j + \alpha \textstyle\sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{).}$$
$$\}$$

New Version

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

2 Linear Regression: Gradient Descent

- **Stochastic Gradient Descent:** update the parameters according to the gradient of the error with respect to that single training example only.

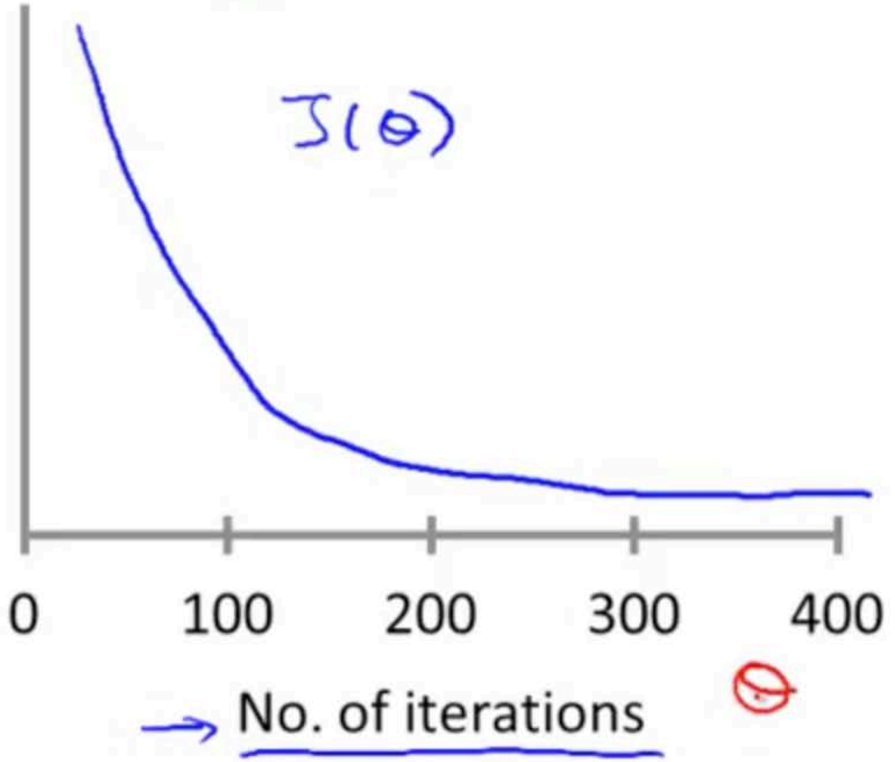
$$
\begin{aligned}
&\text{Loop } \{ \\
&\qquad \text{for i=1 to m, } \{ \\
&\qquad\qquad \theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)} \qquad \text{(for every } j\text{).} \\
&\qquad \} \\
&\}
\end{aligned}
$$

- Often, stochastic gradient descent gets θ “close” to the minimum much faster than batch gradient descent.

2 Linear Regression: Gradient Descent

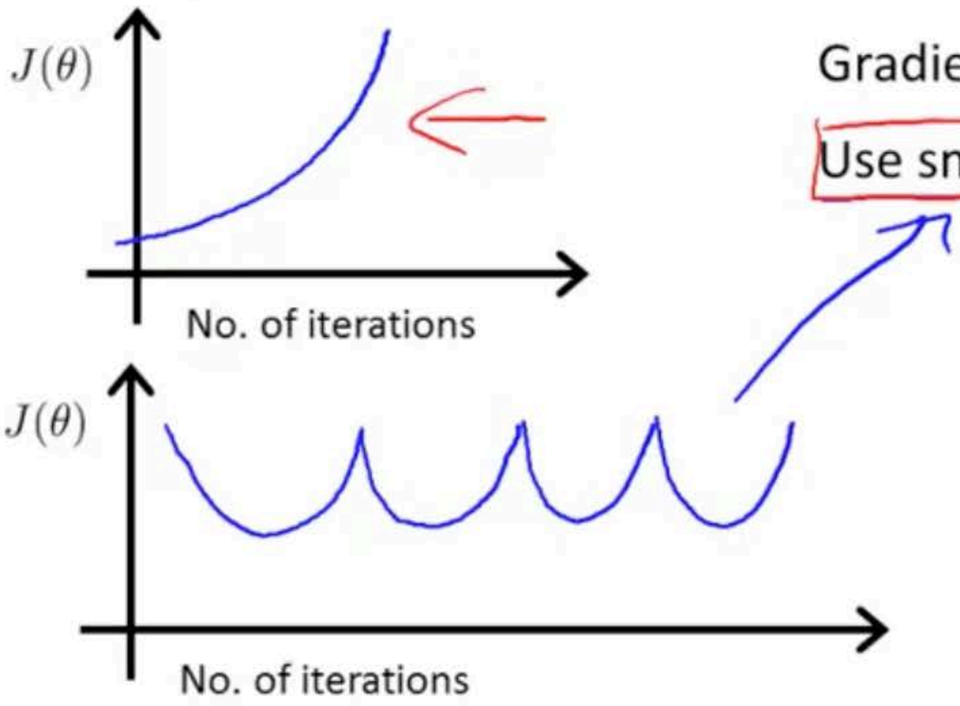
Making sure gradient descent is working correctly.

$$\boxed{\min_{\theta} J(\theta)}$$



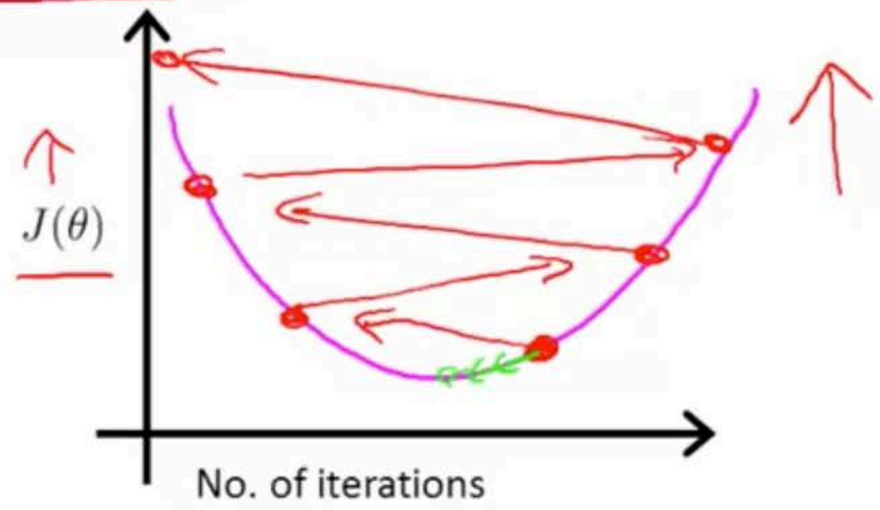
2 Linear Regression: Gradient Descent

Making sure gradient descent is working correctly.



Gradient descent not working.

$\boxed{\text{Use smaller } \alpha.}$



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

2 Linear Regression: Normal Equations

- **Represent J as matrix-vectorial notation**

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

$$J(\theta) = \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

2 Linear Regression: Normal Equations

- **Represent J as matrix-vectorial notation**

$$X = \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix}$$

m-by-n matrix, actually m-by-n + 1

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

m-dimensional vector

2 Linear Regression: Normal Equations

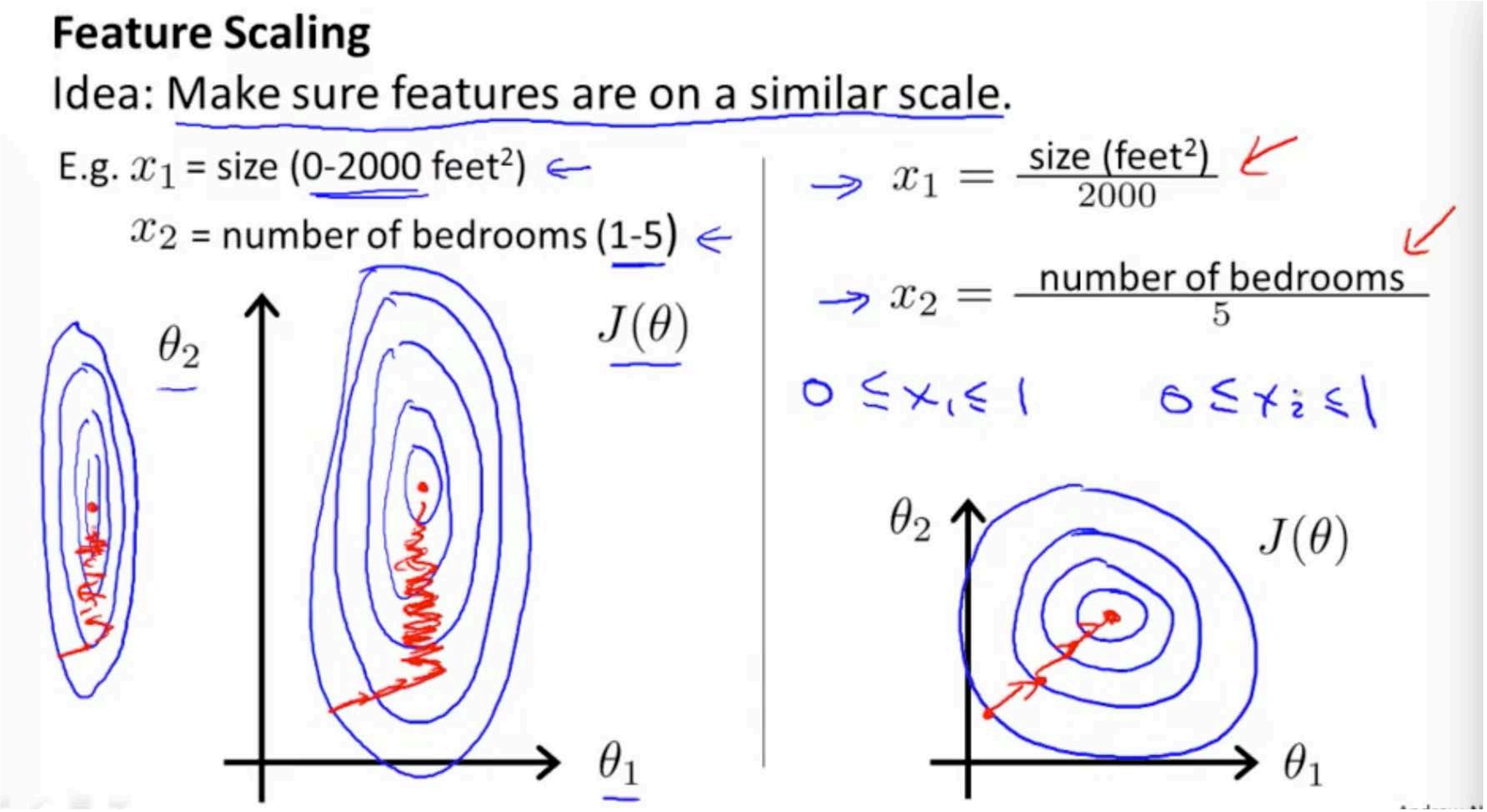
- **For example**

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

$$X = \begin{bmatrix} 1 & 2104 & 3 \\ 1 & 1600 & 3 \\ 1 & 2400 & 3 \end{bmatrix} \qquad \vec{y} = \begin{bmatrix} 400 \\ 330 \\ 369 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

Note: more precisely, the features should be scaled.

2 Linear Regression: Feature Scaling



2 Linear Regression: Normal Equations

$$J(\theta) = \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y})$$

X: m × $(n+1)$

$\vec{y}$: m × 1

θ : (n + 1) × 1

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

a real number

$$\boxed{\mathrm{tr}A = \mathrm{tr}A^T}$$

$$\boxed{\begin{aligned} \nabla_{A^T}\mathrm{tr}ABA^TC &= B^TA^TC^T + BA^TC \\ \nabla_A \mathrm{tr}AB &= B^T \end{aligned}}$$

2 Linear Regression: Normal Equations

- **Normal Equations**

$$X^T X\theta = X^T\vec{y}$$

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

What if $X^T X$ *is non*−*invertible*?

2 Linear Regression: Normal Equations

What if $X^T X$ is non-invertible?

- Redundant features (linearly dependent).

  E.g. x_1 = size in feet^2

  x_2 = size in m^2

  $x_1 = (3.28)^2 x_2$

  1m = 3.28 feet

  $m = 10$

  $n = 100$

  $\theta \in \mathbb{R}^{101}$

- Too many features (e.g. $m \leq n$).
  - Delete some features, or use regularization.

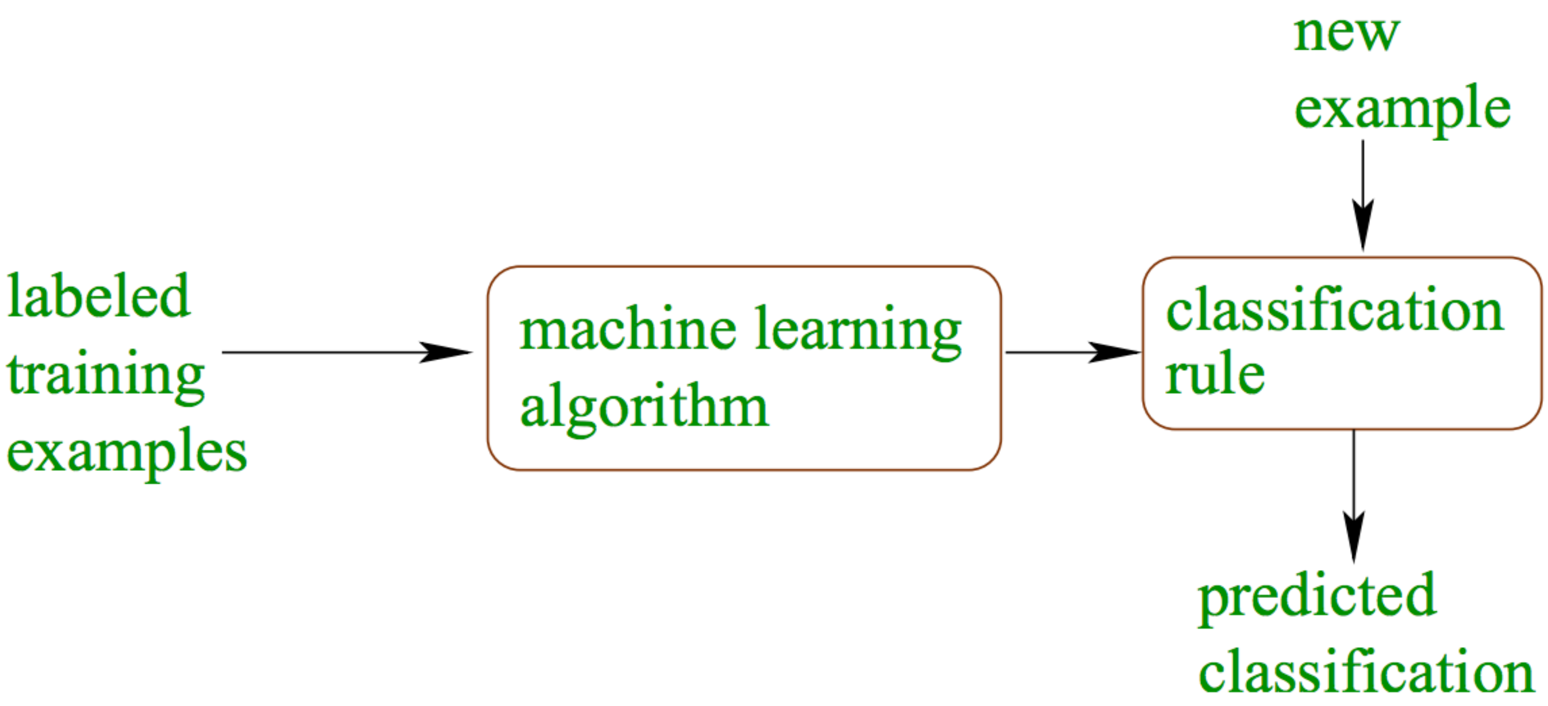
2 Gradient Descent Vs. Normal Equations

m **training examples,** n **features.**

Gradient Descent	Normal Equation
• Need to choose α.	• No need to choose α.
• Needs many iterations.	• Don't need to iterate.
• Works well even when n is large.	• Need to compute $(\boxed{X^T X})^{-1}$ $n \times n$ $O(n^3)$
	• Slow if n is very large.
	$n = 100$

3 Classification: Overview

- Classification: classifies examples into given set of categories



3 Classification: Application

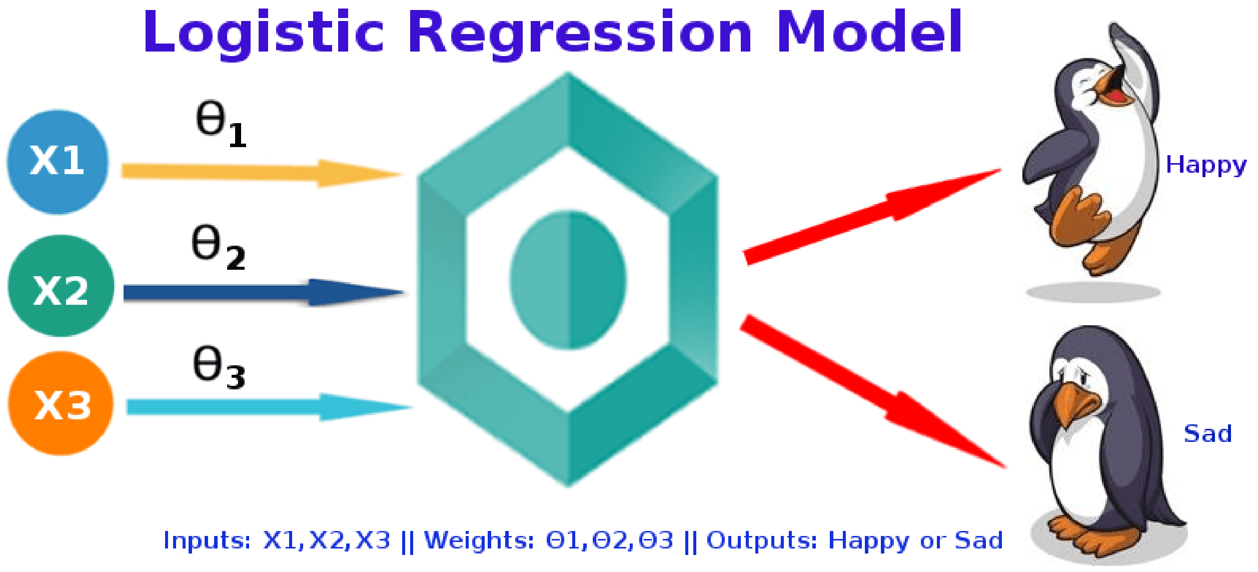
- Image/Document Classification
- Spam Email Filtering
- Optical Character Recognition
- Machine Vision (e.g., face detection)
- Natural Language Processing (e.g., spoken language understanding)
- Fraud Detection (e.g., if a transaction is fraudulent)
- Market Segmentation (e.g.: predict if customer will respond to promotion)
- Bioinformatics (e.g., classify proteins according to their function)

3 Classification: Algorithms

- Logistic Regression
- Support Vector Machine (SVM)
- Naïve Bayes Classifier
- K-Nearest Neighbor Classifier
- Decision Tree, Random Forest
- (Deep) Neural Network

4 Logistic Regression: Overview

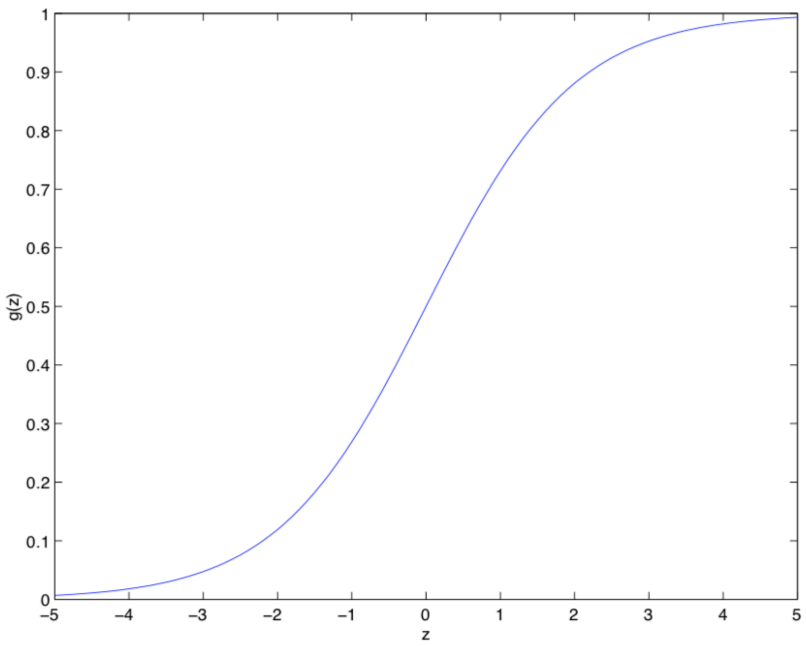
- Logistic Regression was used in the biological sciences in early twentieth century. Logistic Regression is used when the dependent variable(target) is categorical. For example:
  - To predict whether an email is spam (1) or (0)
  - Whether the tumor is malignant (1) or not (0)



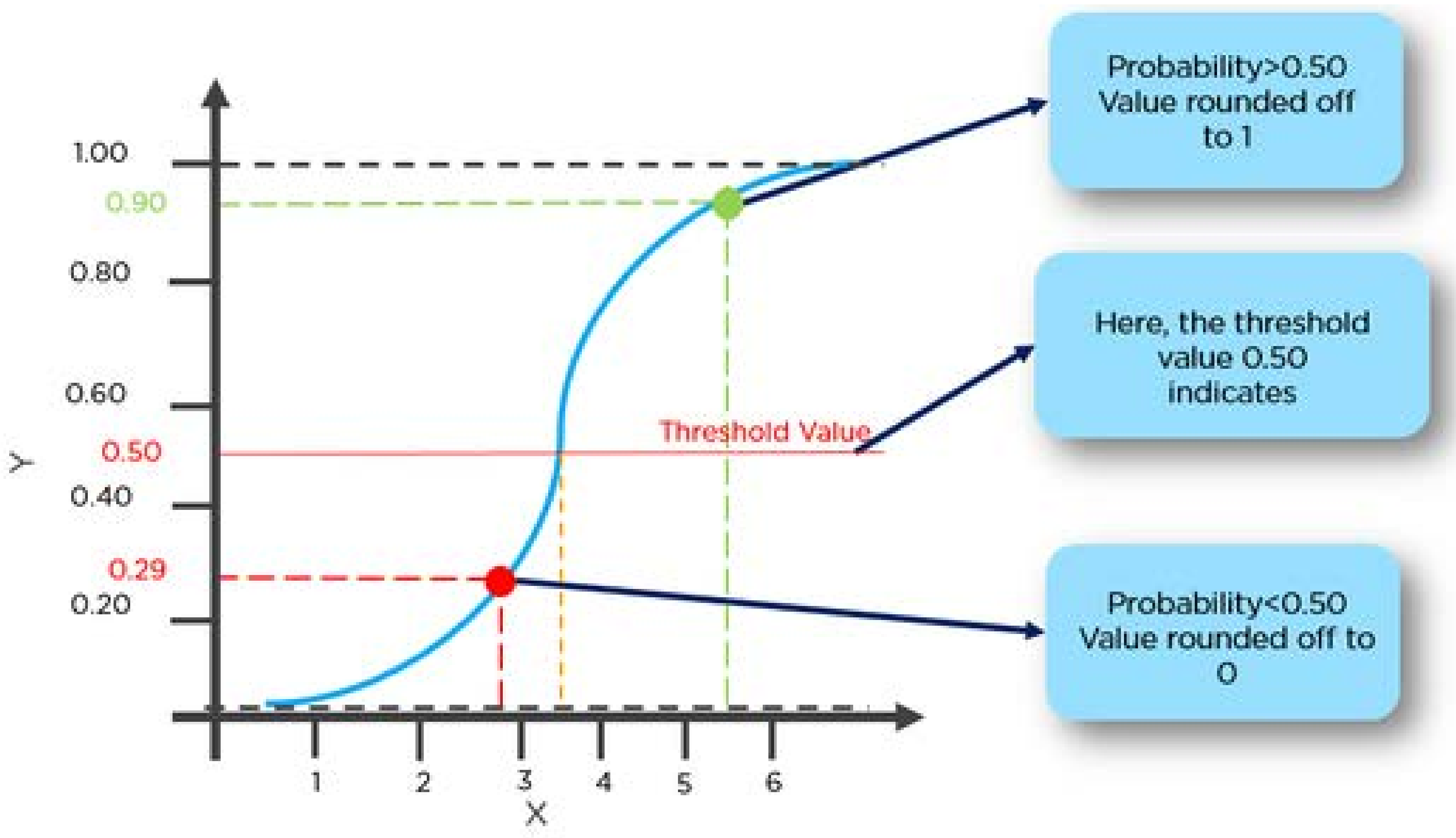
4 Logistic Regression: : Hypothesis

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

$$g(z) = \frac{1}{1 + e^{-z}}$$ is called the logistic function or sigmoid function



4 Logistic regression: Hypothesis



4 Logistic Regression: Sigmoid Function

A useful property of the derivative of the sigmoid function:

$$\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}$$

4 Logistic Regression: How to fit θ?

Let's assume that:

$$\begin{aligned} P(y=1 \mid x;\theta) &= h_\theta(x) \\ P(y=0 \mid x;\theta) &= 1-h_\theta(x) \end{aligned}$$

This can be written as:

$$p(y \mid x;\theta) = (h_\theta(x))^y \, (1-h_\theta(x))^{1-y}$$

4 Logistic Regression: Maximize the Likelihood

- Assuming that the m training examples were generated independently, we can then write down the likelihood of the parameters as:

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}} \end{aligned}$$

- It will be easier to maximize the log likelihood:

$$\begin{aligned} \ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)})) \end{aligned}$$

4 Logistic Regression: Maximize the likelihood

- **Maximize the likelihood**

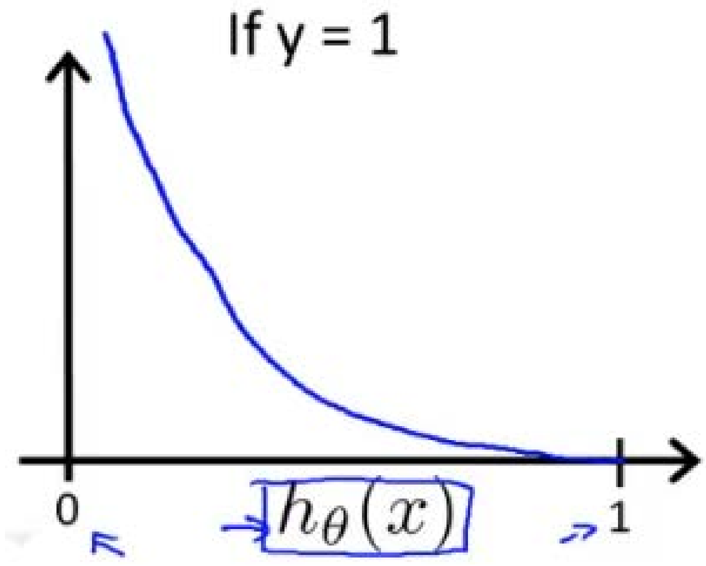
$$
\begin{aligned}
\frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial \theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\
&= (y - h_\theta(x))\, x_j
\end{aligned}
$$

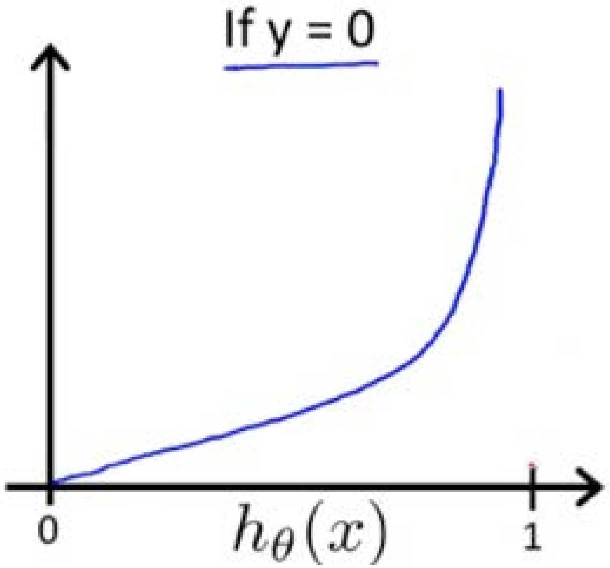
- **The stochastic gradient ascent rule**

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

4 Logistic Regression: Minimize Cross-Entropy Loss

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





4 Logistic Regression: Minimize Cross-Entropy Loss

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

- **Cross-Entropy Loss**

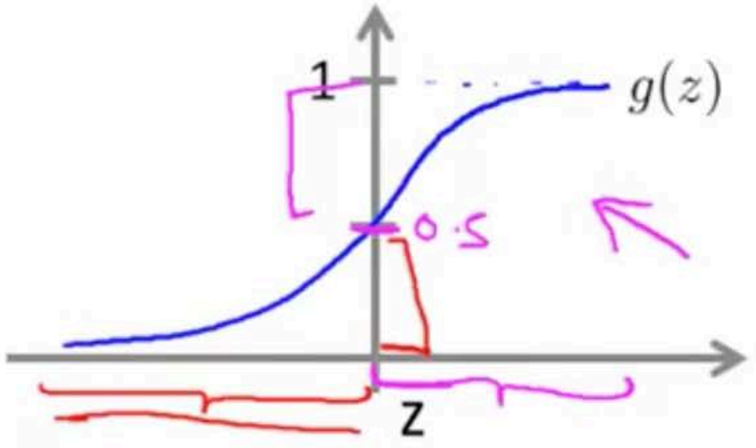
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost\left(h_\theta\left(x^{(i)}\right), y^{(i)}\right) = -\frac{1}{m}[y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log(1 - h_\theta\left(x^{(i)}\right))]$$

4 Logistic regression: Decision Boundary

Logistic regression

$$h_\theta(x) = g(\theta^T x) = P(y=1|x;\theta)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Suppose predict "$y = 1$" if $h_\theta(x) \geq 0.5$

$\theta^T x \geq 0$

$g(z) \geq 0.5$

when $z \geq 0$

$h_\theta(x) = g(\theta^T x) \geq 0.5$

whenever $\theta^T x \geq 0$

z

predict "$y = 0$" if $h_\theta(x) < 0.5$

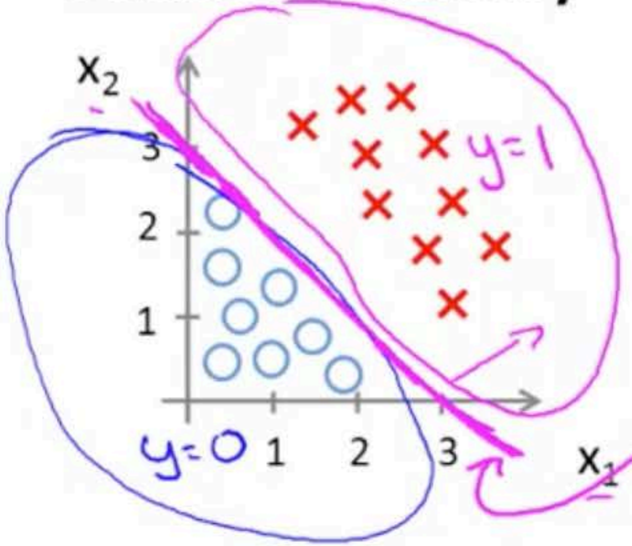
$h_\theta(x) = g(\theta^T x)$

$\theta^T x < 0$

$g(z) < 0.5$

4 Logistic regression: Decision Boundary

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$\rightarrow h_\theta(x) = g(\underset{\overset{\shortparallel}{-3}}{\theta_0} + \underset{\overset{\shortparallel}{1}}{\theta_1} x_1 + \underset{\overset{\shortparallel}{1}}{\theta_2} x_2)$$

Decision boundary

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2 \geq 0}_{\theta^T x}$

$\rightarrow x_1 + x_2 \geq 3$

x_1, x_2

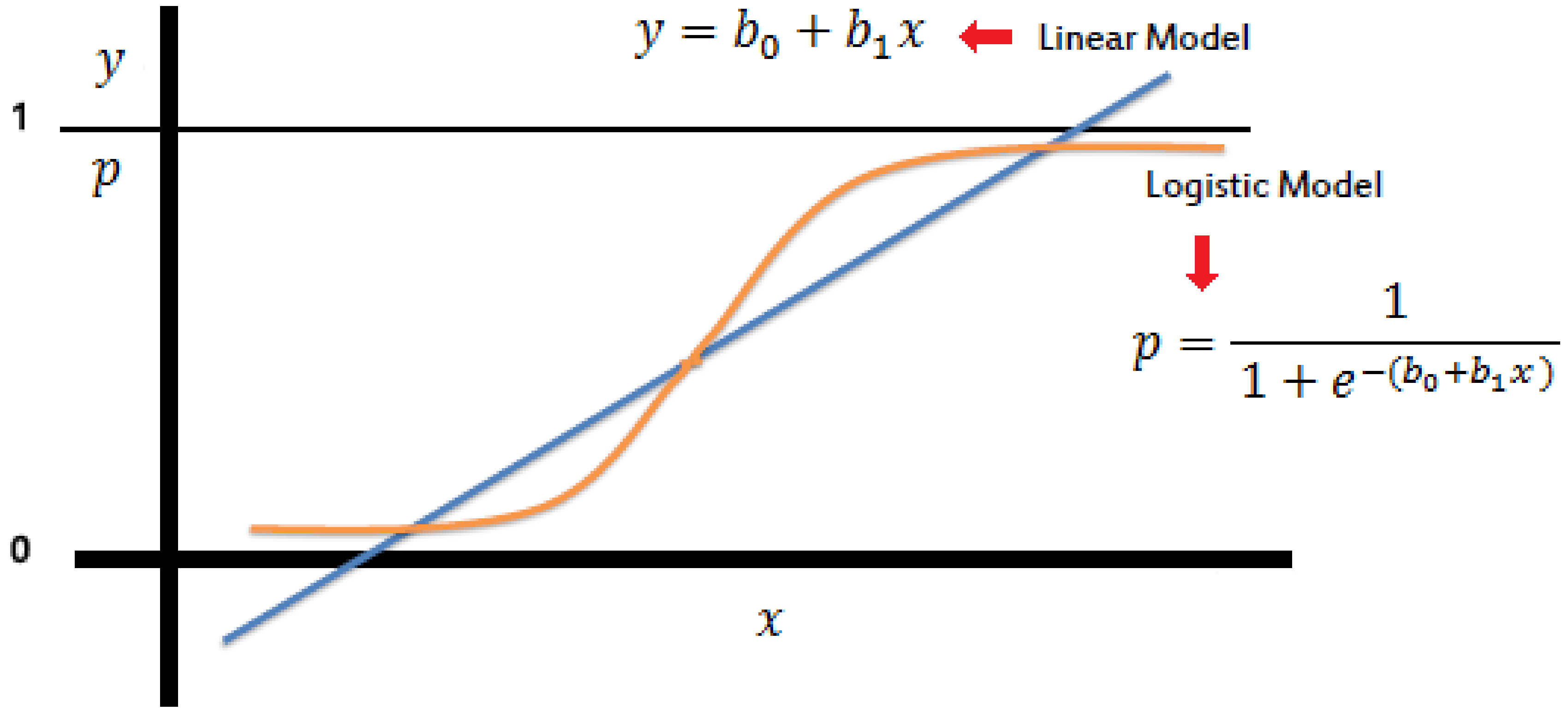
$\rightarrow h_\theta(x) = 0.5$

$\boxed{x_1 + x_2 = 3}$

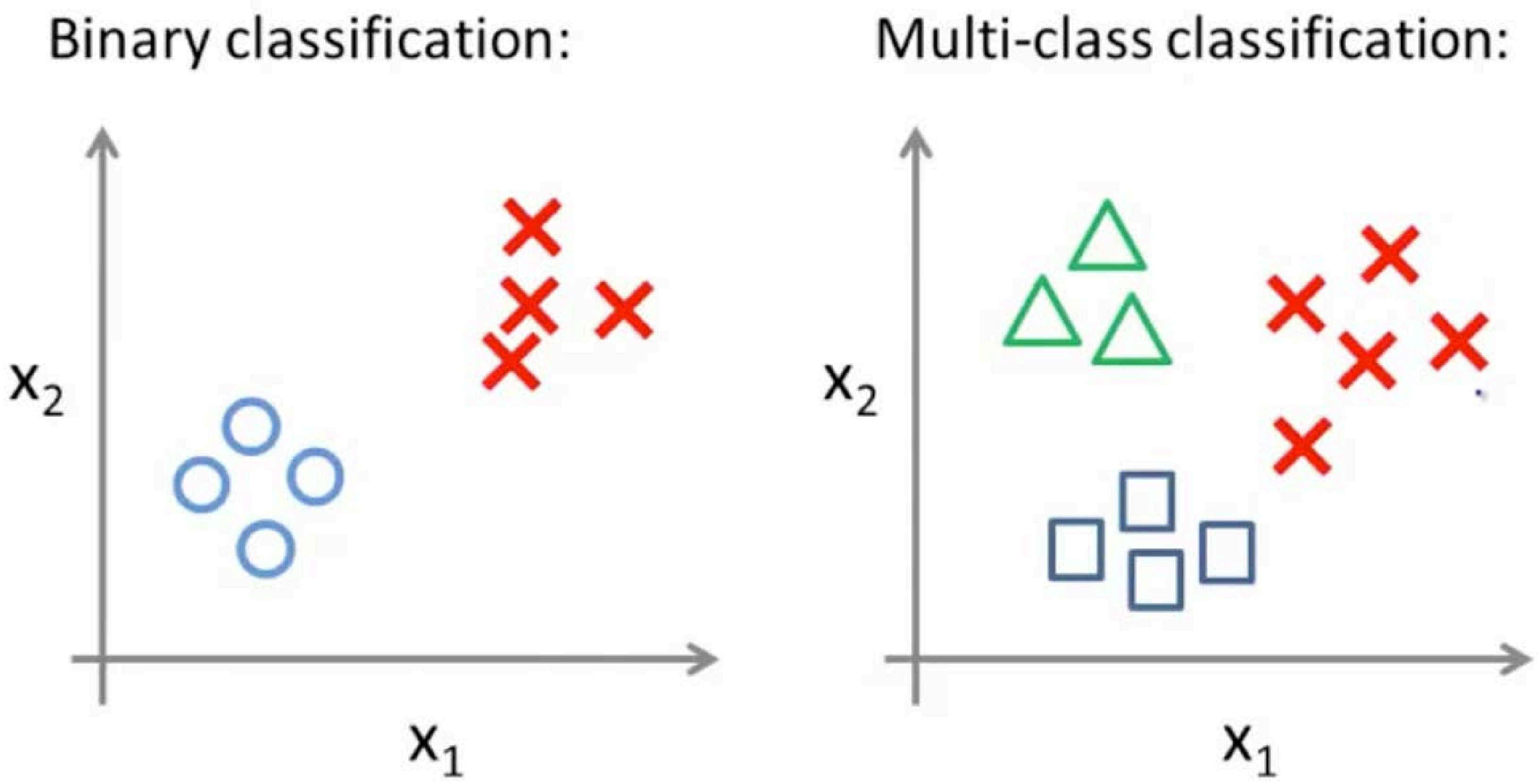
$x_1 + x_2 < 3$

$\rightarrow y = 0$

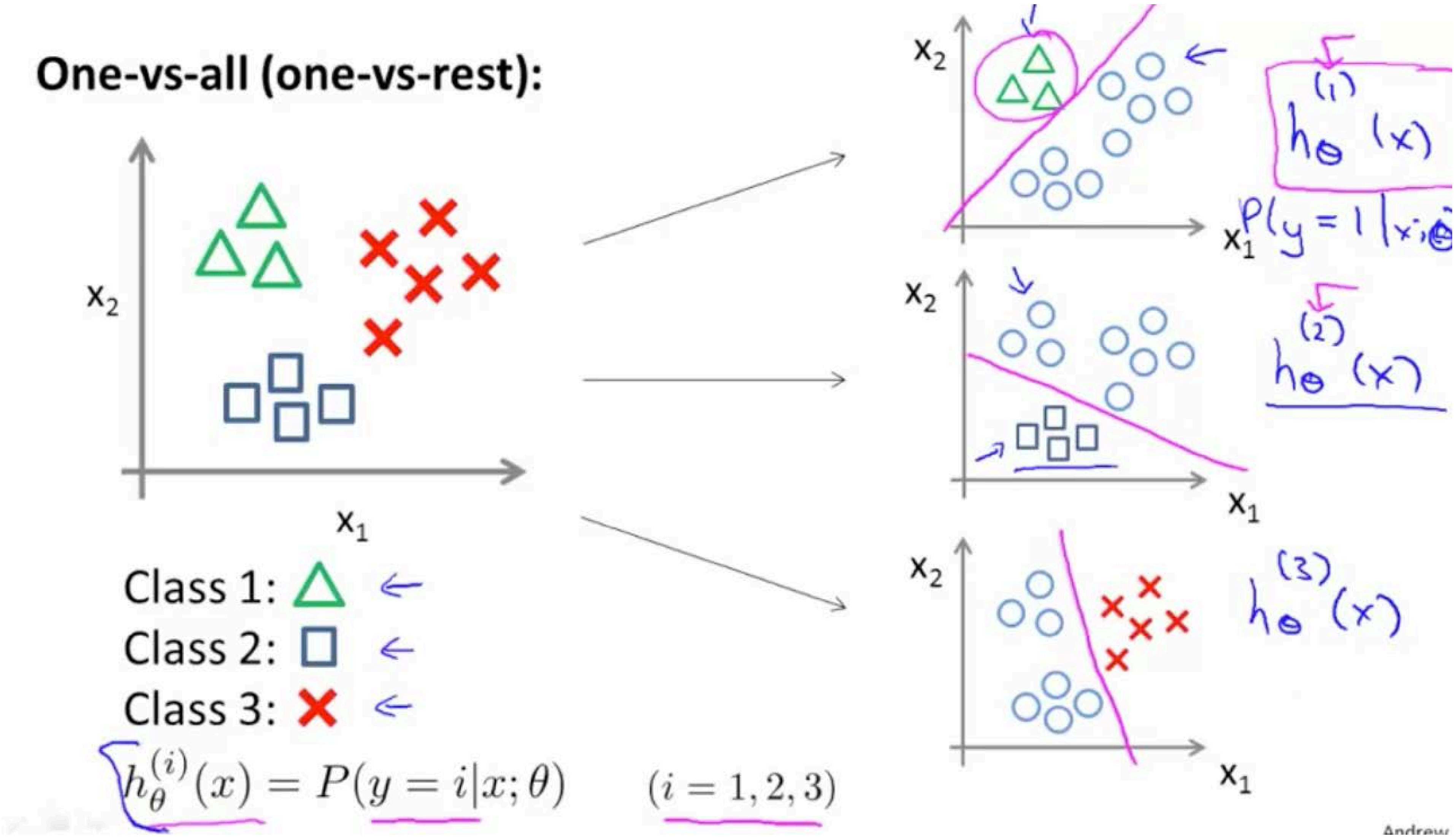
4 Logistic regression vs Linear regression



4 Logistic regression: Multiclass Classification



4 Logistic regression: Multiclass Classification



4 Logistic regression: Multiclass Classification

One-vs-all

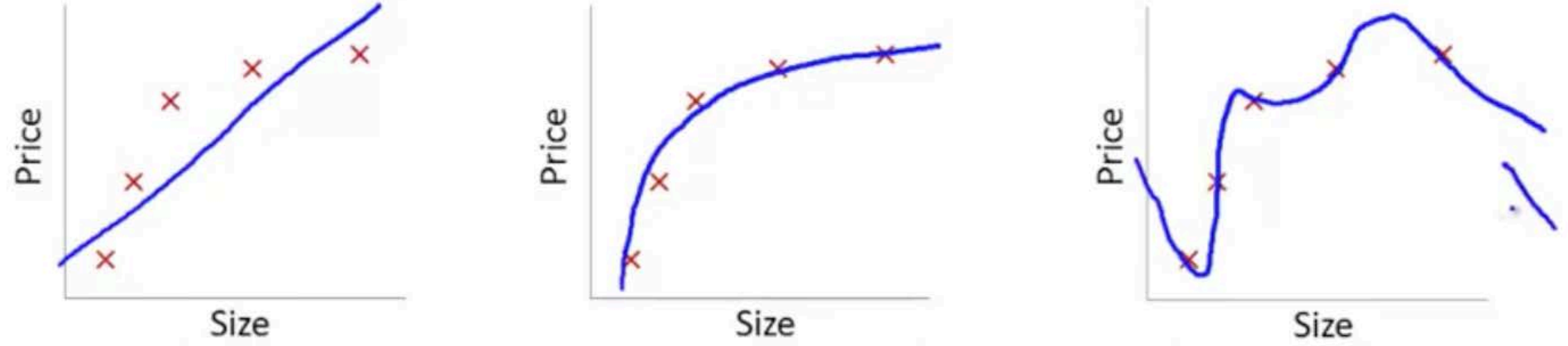
Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

5 Regularization: Overfitting

Example: Linear regression (housing prices)



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

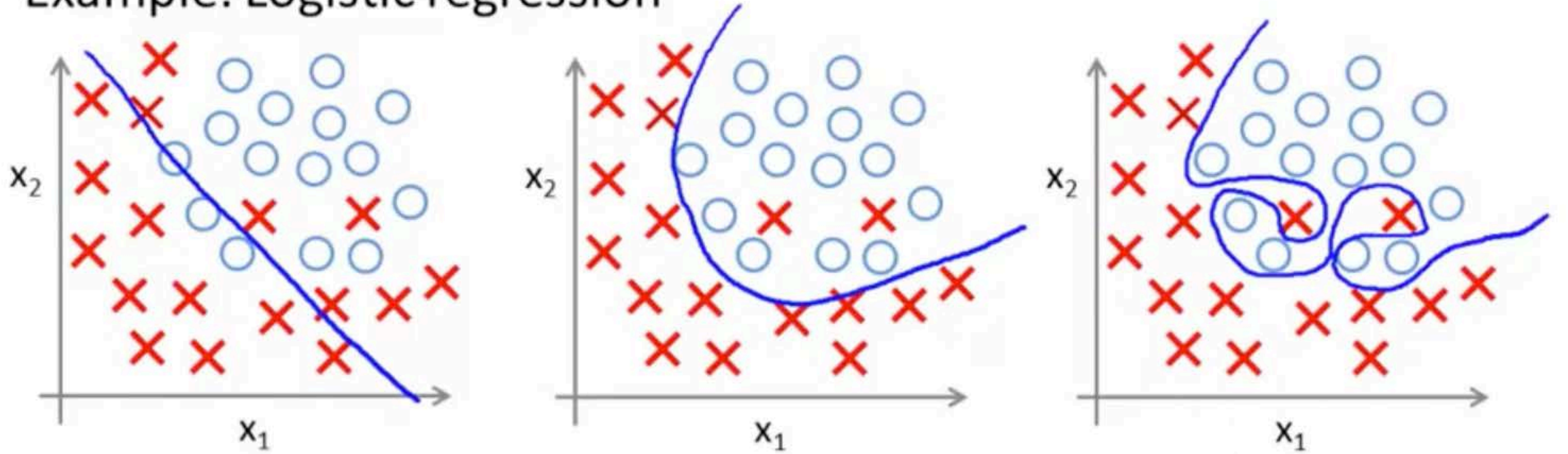
$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$ "Just right"

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

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

5 Regularization: Overfitting

Example: Logistic regression



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

(g = sigmoid function)

"Underfit"

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

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

5 Regularization: Overfitting

Addressing overfitting:

Options:

1. Reduce number of features.
   - Manually select which features to keep.
   - Model selection algorithm (later in course).
2. Regularization.
   - Keep all the features, but reduce magnitude/values of parameters θ_j.
   - Works well when we have a lot of features, each of which contributes a bit to predicting y.

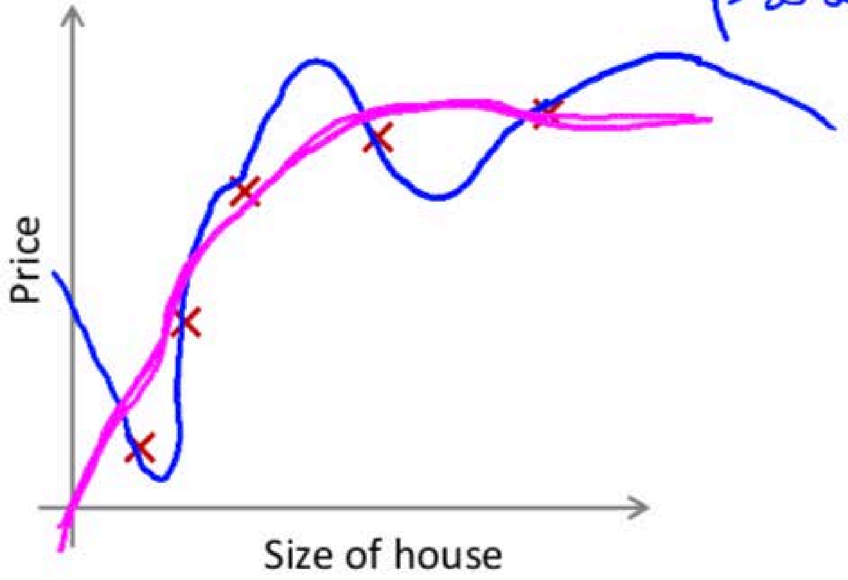
5 Regularization

Regularization.

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

regularization parameter

$$\min_\theta J(\theta)$$



5 Regularization: Linear Regression

In regularized linear regression, we choose θ to minimize

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

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting λ to be very large can't hurt it
- Algortihm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

5 Regularization: Linear Regression

Gradient descent

$\theta_0 \quad \theta_1, \theta_2, \ldots, \theta_n$

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \boxed{\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}} \qquad \frac{\partial}{\partial \theta_0} J(\theta)$$

$$\theta_j := \theta_j - \boxed{\alpha} \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} - \frac{\lambda}{m} \theta_j \right]$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

}

$$\theta_j := \boxed{\theta_j (1 - \alpha \frac{\lambda}{m})} - \boxed{\alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}} \qquad \to J(\theta)$$

$$1 - \alpha \frac{\lambda}{m} < 1 \qquad 0.99 \qquad \theta_j \times 0.99 \qquad \boxed{\theta_j^{2}}$$

5 Regularization: Linear Regression

Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \mathbb{R}^m$$

$m \times (n+1)$

$$\min_\theta J(\theta)$$

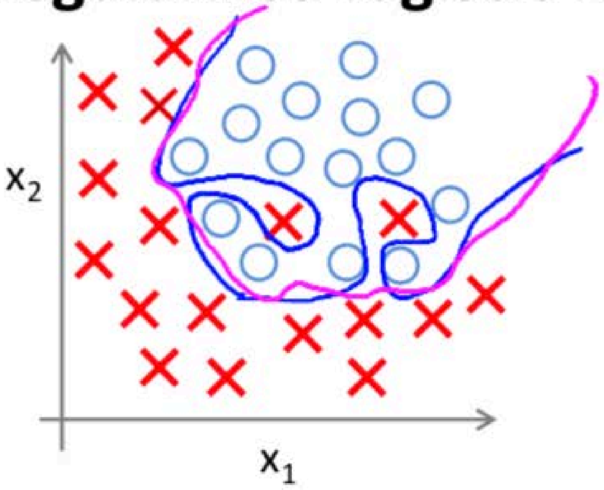
$$\frac{\partial}{\partial \theta_j} J(\theta) \stackrel{\text{set}}{=} 0$$

$$\theta = \Big(X^T X + \lambda \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & 0 & 1 & \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{(n+1)\times(n+1)} \Big)^{-1} X^T y$$

E.g. $n=2$ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

5 Regularization: Logistic Regression

Regularized logistic regression.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

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

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

$\boxed{\theta_1, \theta_2, \dots, \theta_n}$

Andrew Ng

5 Regularization: Logistic Regression

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

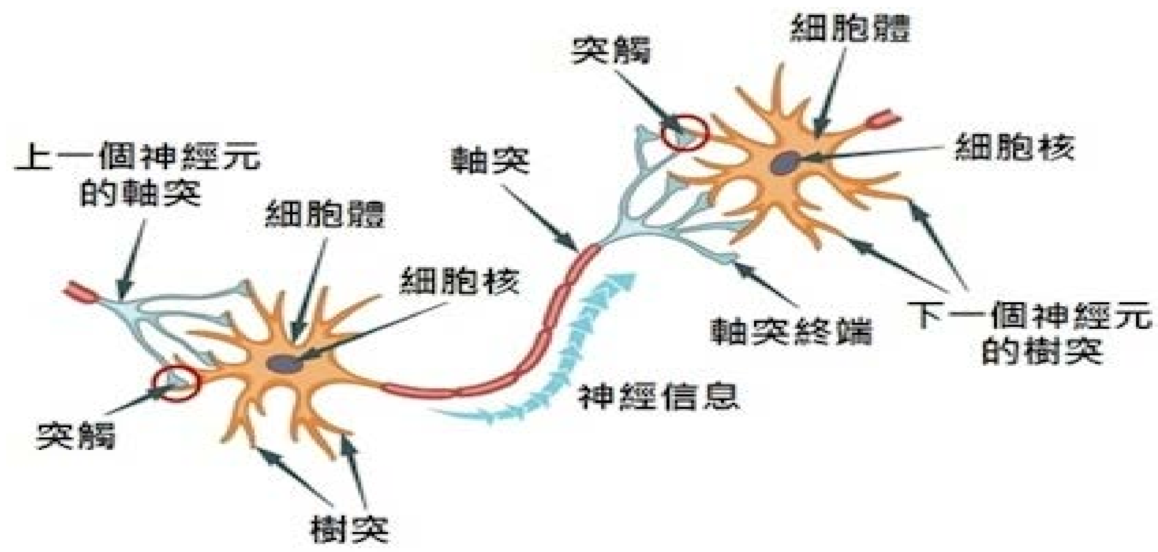
$$\theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} - \frac{\lambda}{m} \theta_j \right]}_{\frac{\partial}{\partial \theta_j} J(\theta)}$$

$$(j = \cancel{0}, \underbrace{1, 2, 3, \ldots, n}_{\theta_1, \ldots, \theta_n})$$

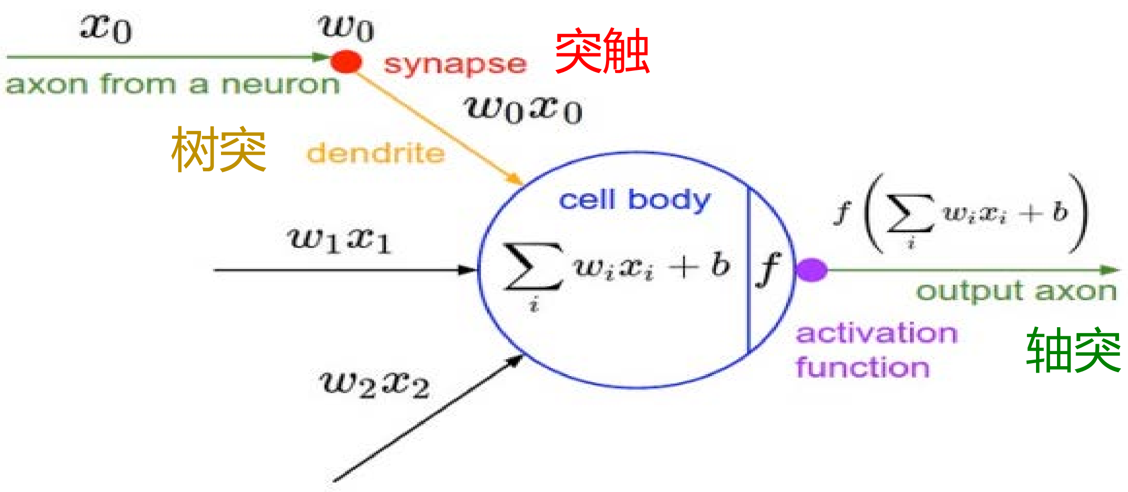
}

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

6 Perceptron: Artificial Neuron



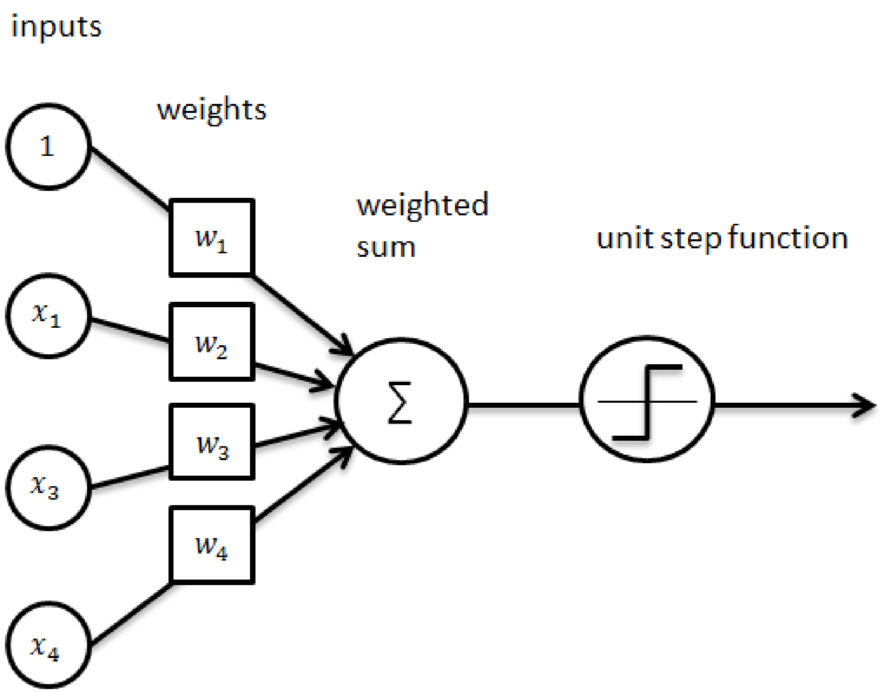
生物神经元的基本结构

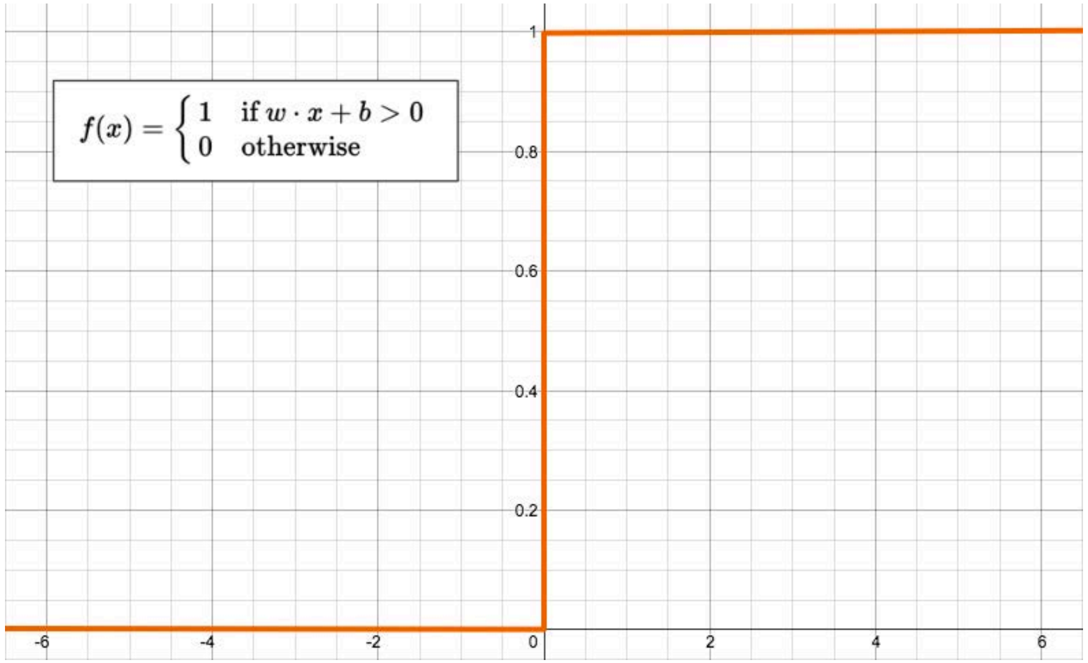


人工神经元的结构模型

6 Perceptron: Model

- **Perceptron**
  - First compute a weighted sum of the inputs from other neurons.
  - Then output a 1 if the weighted sum exceeds the threshold.





6 Perceptron: Minimizing Squared Errors

$$h_\theta(x) = g(\theta^T x) = g\left(\sum_{i=0}^{n} \theta_i x_i\right) \qquad g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

- **The squared error for a single training example with input x and true output y is:**

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

- **Gradient Descent**

undefined and omitted

$$\frac{\partial E}{\partial \theta_j} = \left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) \times (-1) \times \boxed{g'(in)} \times x_j^{(i)}$$

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

- **Update Rule**

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

Homework (1)

- Task: Regression
- Dataset: Housing (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#housing)
- Algorithms: Linear Regression

housing

- Source: UCI / Housing (Boston)
- # of data: 506
- # of features: 13
- Files:
  - housing
  - housing_scale (scaled to [-1,1])

```
24.0 1:0.00632 2:18.00 3:2.310 4:0 5:0.5380 6:6.5750 7:65.20 8:4.0900 9:1 10:296.0 11:15.30 12:396.90 13:4.98
21.6 1:0.02731 2:0.00 3:7.070 4:0 5:0.4690 6:6.4210 7:78.90 8:4.9671 9:2 10:242.0 11:17.80 12:396.90 13:9.14
34.7 1:0.02729 2:0.00 3:7.070 4:0 5:0.4690 6:7.1850 7:61.10 8:4.9671 9:2 10:242.0 11:17.80 12:392.83 13:4.03
33.4 1:0.03237 2:0.00 3:2.180 4:0 5:0.4580 6:6.9980 7:45.80 8:6.0622 9:3 10:222.0 11:18.70 12:394.63 13:2.94
36.2 1:0.06905 2:0.00 3:2.180 4:0 5:0.4580 6:7.1470 7:54.20 8:6.0622 9:3 10:222.0 11:18.70 12:396.90 13:5.33
28.7 1:0.02985 2:0.00 3:2.180 4:0 5:0.4580 6:6.4300 7:58.70 8:6.0622 9:3 10:222.0 11:18.70 12:394.12 13:5.21
```

Homework (2)

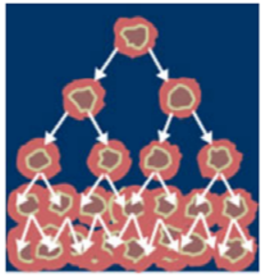
- Task: Classification
- Dataset: Mnist (http://yann.lecun.com/exdb/mnist/). The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples.
- Algorithms: Logistic Regression

Homework (3)

- Task: Classification
- Dataset: Breast Cancer Wisconsin (Diagnostic) Data Set (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29)
- Algorithms: Perceptron

Breast Cancer Wisconsin (Diagnostic) Data Set
Download: Data Folder, Data Set Description

Abstract: Diagnostic Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	569	**Area:**	Life
Attribute Characteristics:	Real	**Number of Attributes:**	32	**Date Donated**	1995-11-01
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	692187

Tips for Homework

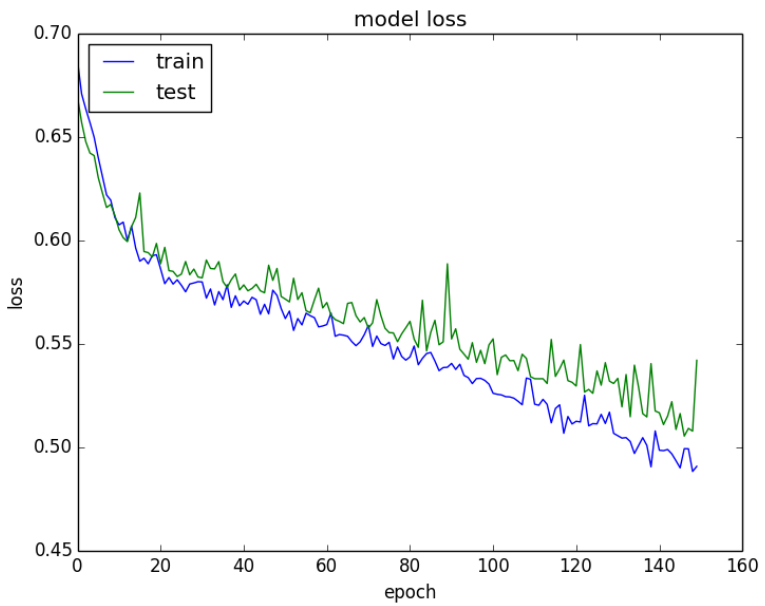
- **Evaluation Metrics:**

✓ Regression: Root Mean Squared Error

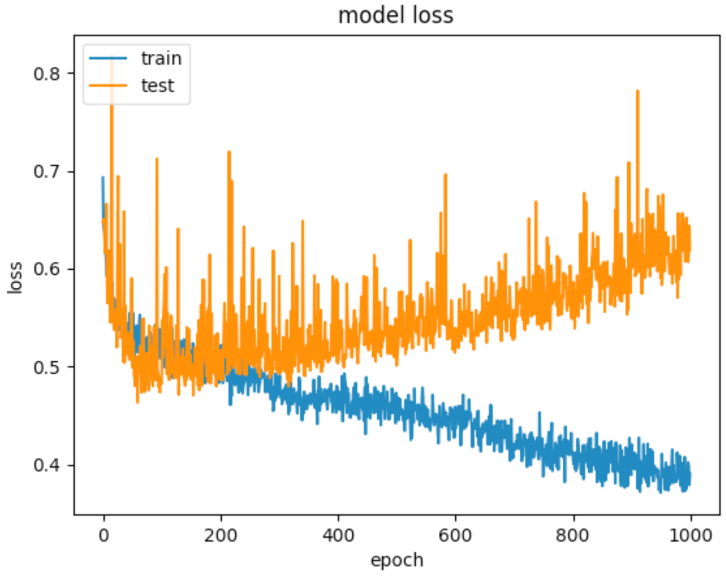
✓ Classification: Accuracy

Tips for Homework

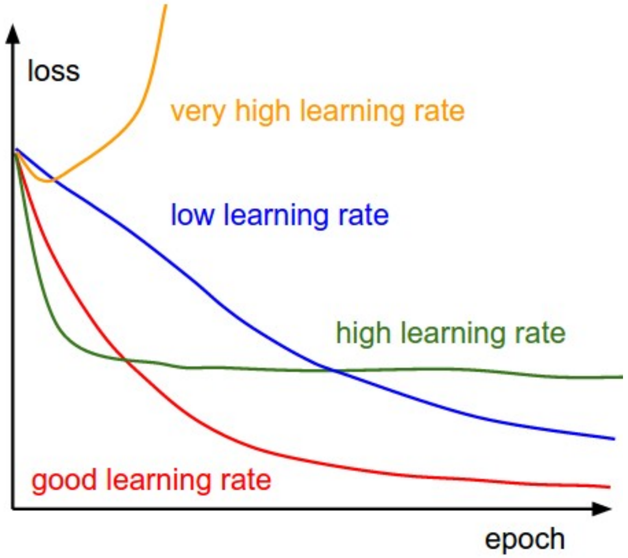
- **Plot Loss Function Figure**



Normal



Overfitting

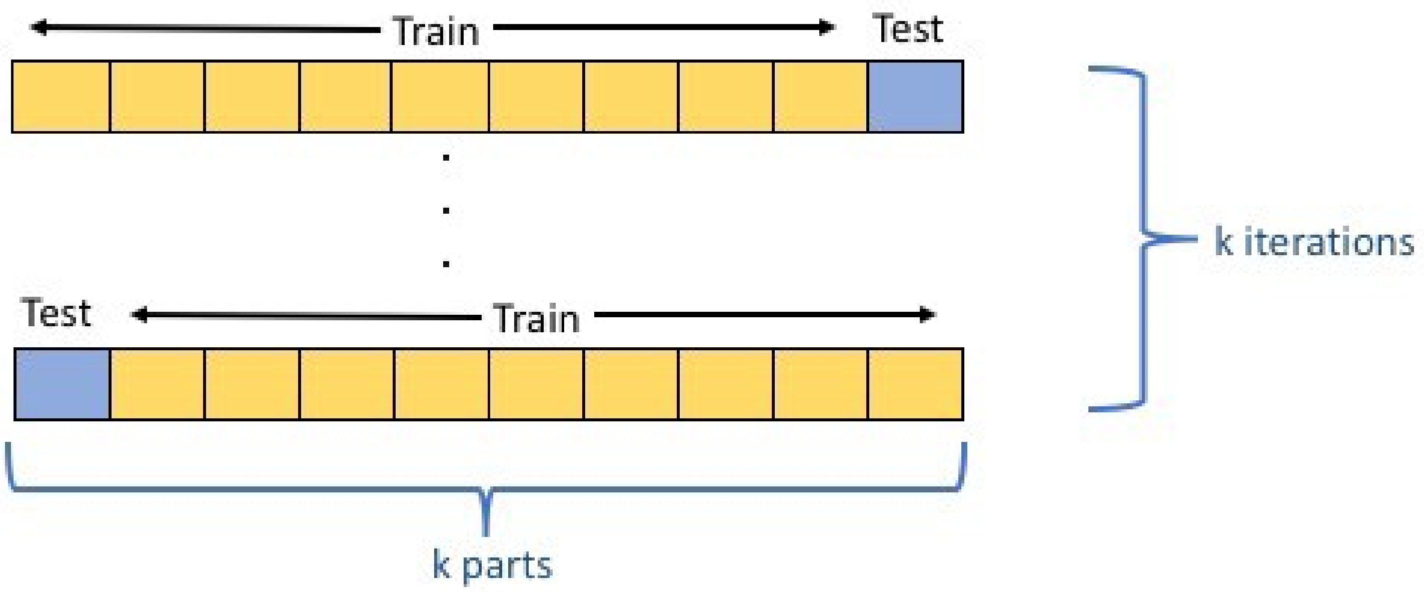


Learning Rate

Tips for Homework

- **K-Folds Cross Validation**

1. Divide the sample data into k parts.
2. Use k-1 of the parts for training, and 1 for testing.
3. Repeat the procedure k times, rotating the test set.
4. Determine an expected performance metric (mean square error, misclassification error rate, confidence interval, or other appropriate metric) based on the results across the iterations



More Materials

- **1. Deep Learning, deeplearning.ai (Andrew Ng et al.) @ Coursera 网易云课堂**
- **2. Andrew Ng. Machine Learning. Coursera，网易公开课**
- **3. 机器学习基石, Hsuan-Tien Lin, 林軒田，台湾大学 @ Coursera**
- **4. 机器学习技法, Hsuan-Tien Lin, 林軒田，台湾大学 @ Coursera**
- **5.CS231n: Convolutional Neural Networks for Visual Recognition Feifei Li et al., University of Stanford**
- **6.Ian Goodfellow, Yoshua Bengio, and Aaron Courville, Deep Learning, Book in preparation for MIT Press, 2016, http://www.deeplearningbook.org/**
- **7. 台大，李宏毅，http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML16.html**
- **MIT公开课，Gilbert Strang ,《线性代数》**

Programming Skills

- **Python**
- **Ubuntu**
- **Keras**
- **Pytorch**
- **Tensorflow**

暑期培训

END

CS229 Lecture 2

Machine Learning

顾晓玲
guxl@hdu.edu.cn