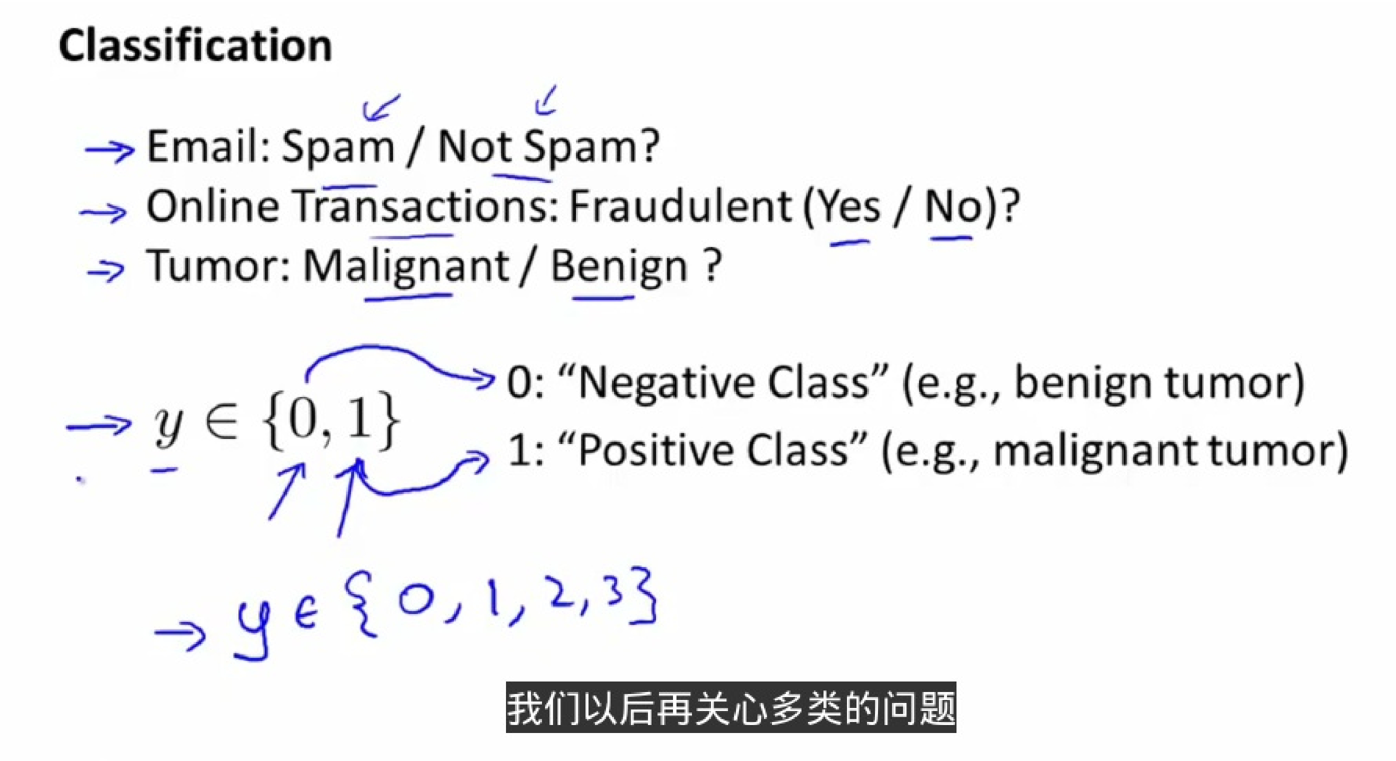
1. 分类问题。首先考虑简单情况，即y取值0,1。y={0,1}。Negative Class/Positive Class



Classification

→ Email: Spam / Not Spam?

→ Online Transactions: Fraudulent (Yes / No)?

→ Tumor: Malignant / Benign ?

→ $y \in \{0, 1\}$

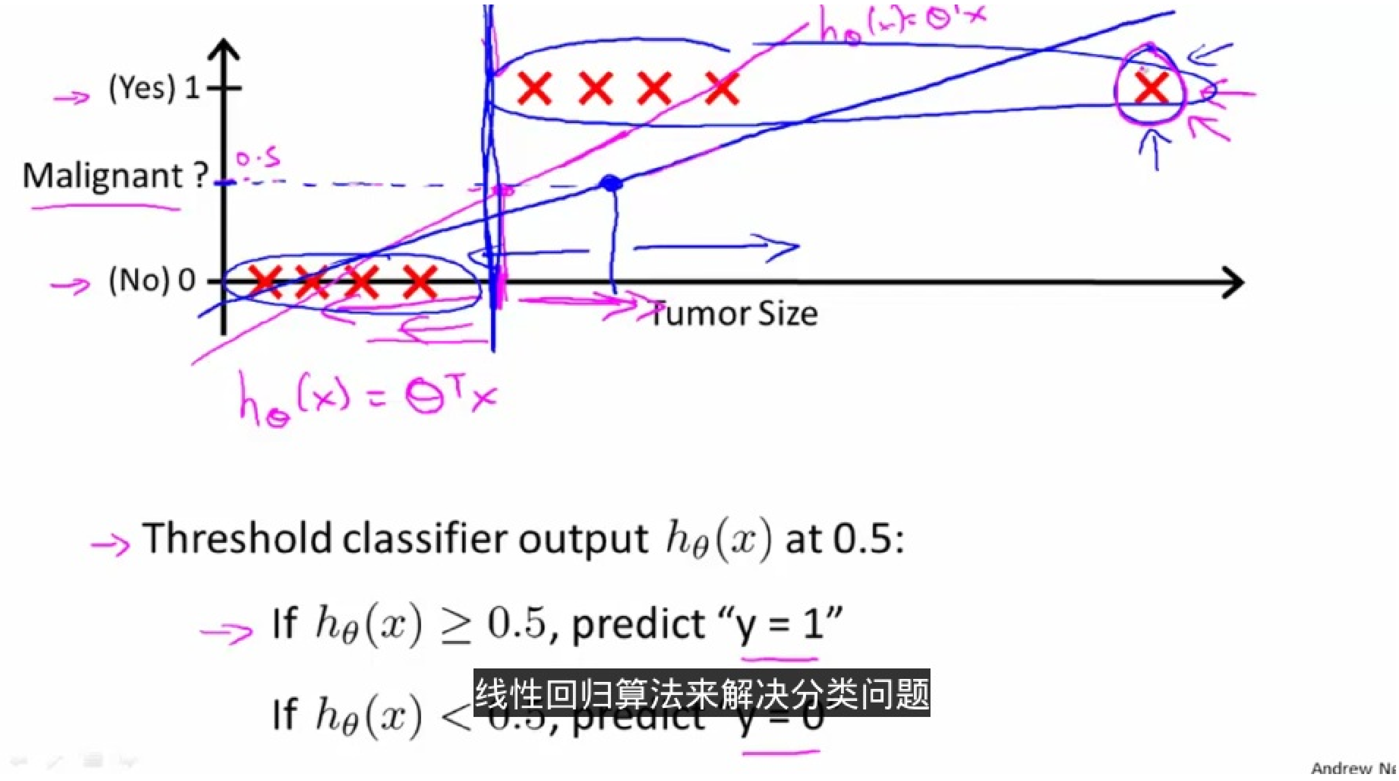
0: "Negative Class" (e.g., benign tumor)

1: "Positive Class" (e.g., malignant tumor)

→ $y \in \{0, 1, 2, 3\}$

我们以后再关心多类的问题

考虑用线性回归来解决分类问题，阈值分类器，根据阈值来划分：



→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

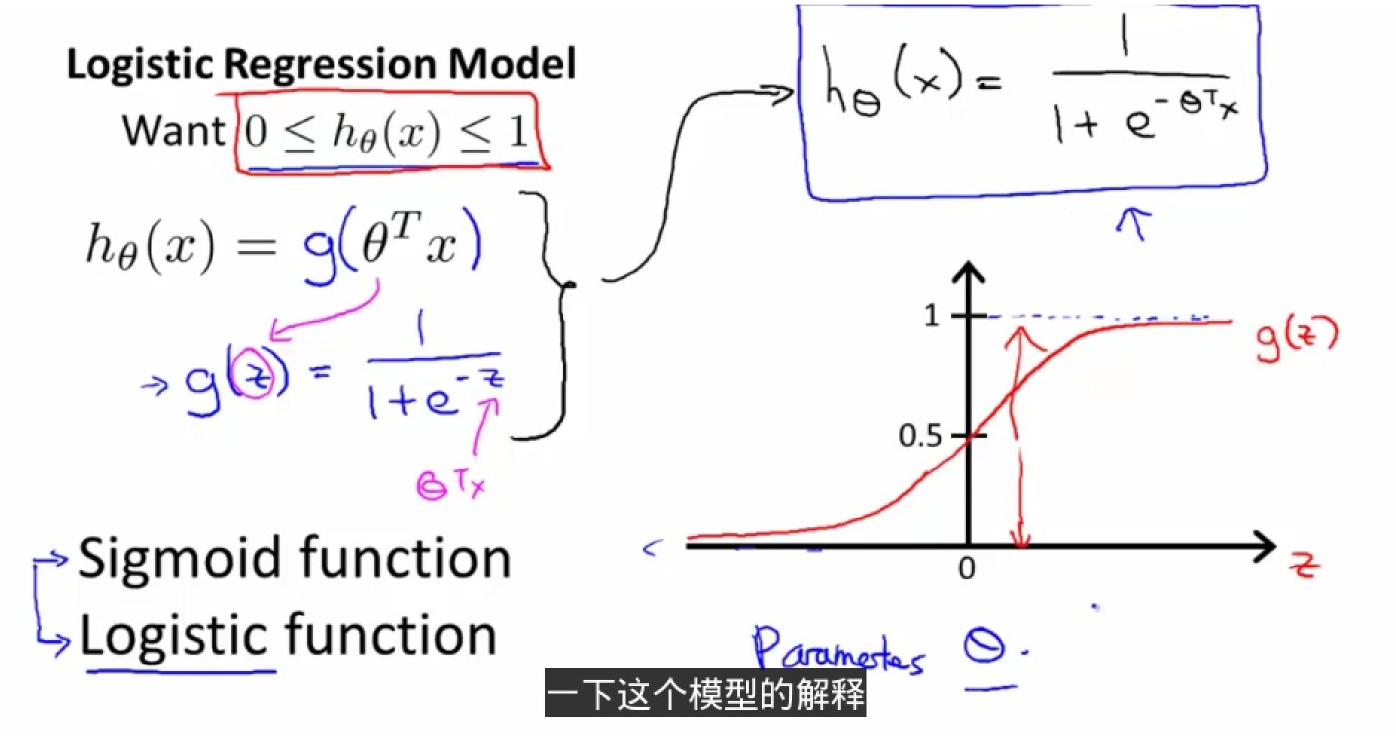
If $h_\theta(x) < 0.5$, predict "y = 0"

线性回归算法来解决分类问题

Andrew N

但这显然不是一个好主意。另外，h(x)的范围有可能也不止在[0, 1]。

2. 逻辑回归（Logistic Regression）。S型函数/逻辑函数。



Logistic Regression Model

Want $0 \le h_\theta(x) \le 1$

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

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

$\theta^T x$

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

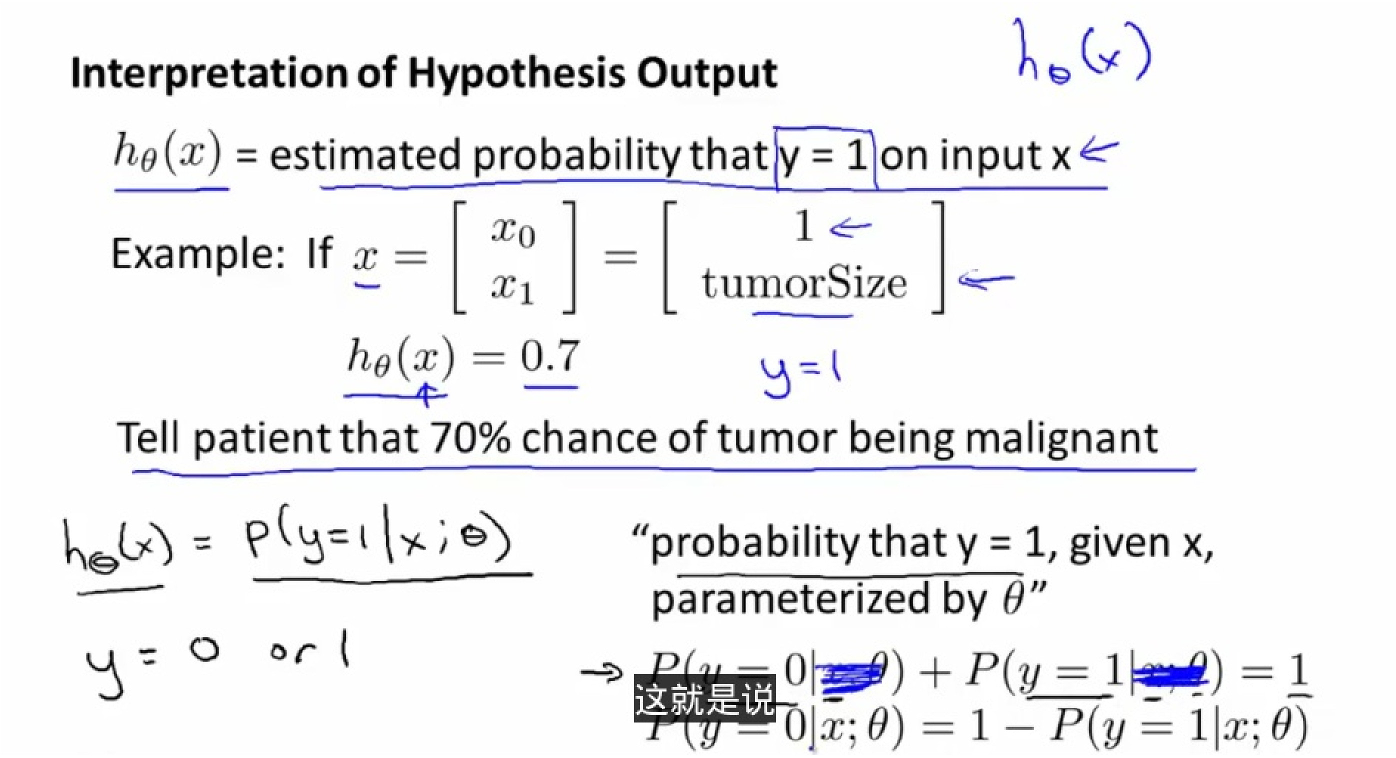
Sigmoid function

Logistic function

Parameters θ.

一下这个模型的解释

逻辑函数h(x)值的意义为：y=1的概率。即：



Interpretation of Hypothesis Output

$h_\theta(x)$

$h_\theta(x)$ = estimated probability that y = 1 on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\theta(x) = 0.7$ $y=1$

Tell patient that 70% chance of tumor being malignant

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

$y = 0$ or 1

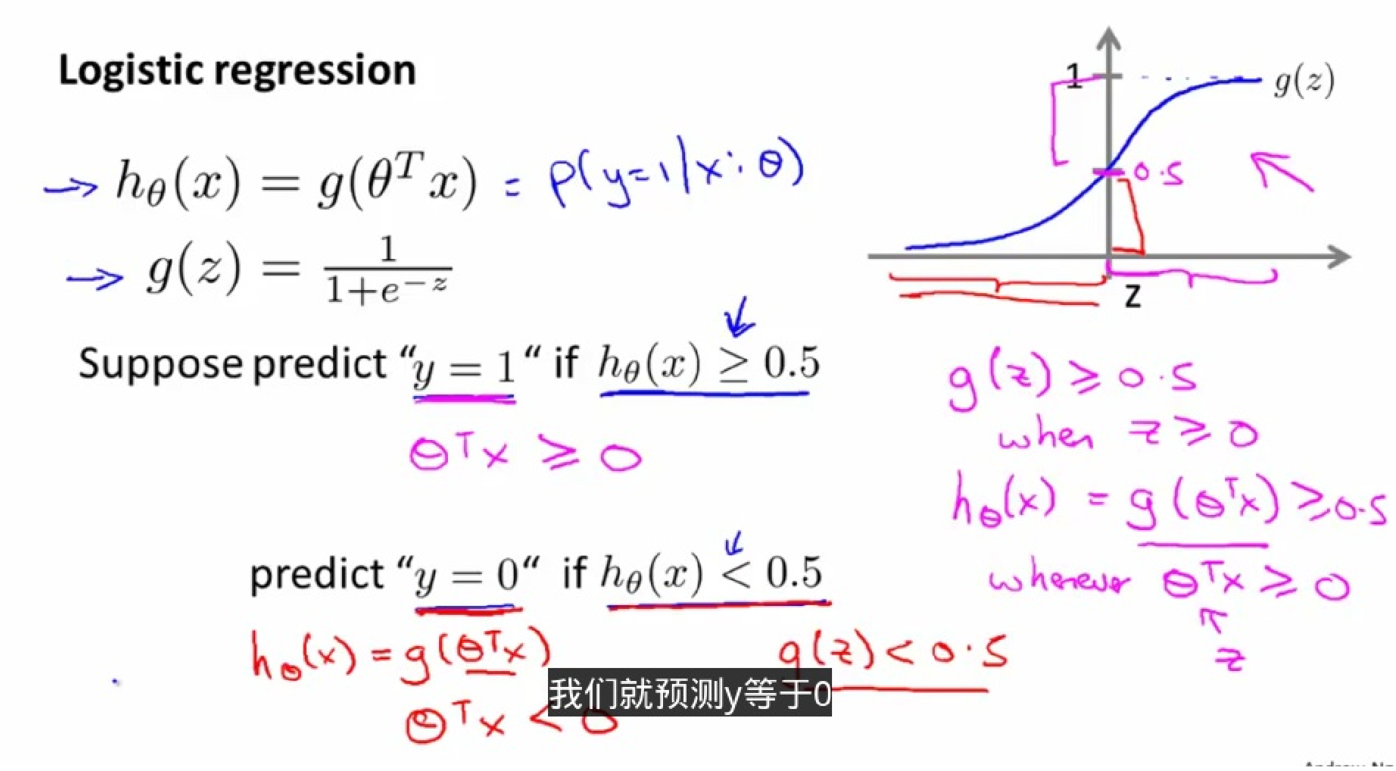
"probability that y = 1, given x, parameterized by θ"

$P(y=0|x;\theta) + P(y=1|x;\theta) = 1$

$P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$

这就是说

3. 决策边界（Decision Boundary）。g(z)>=0 when z>=0; whenever theta'x >=0。



Logistic regression

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

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

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

$\theta^T x \geq 0$

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

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

$\theta^T x < 0$

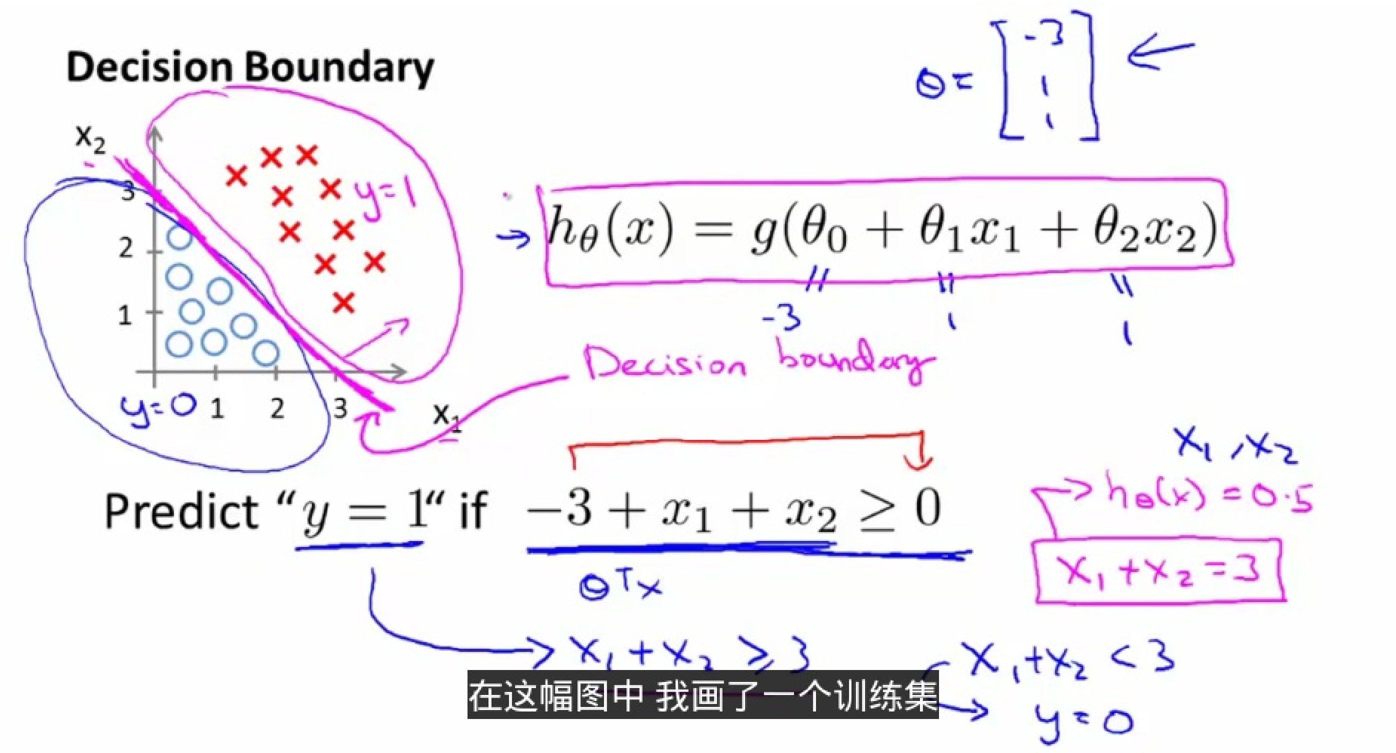
$g(z) \geq 0.5$ when $z \geq 0$

$h_\theta(x) = g(\theta^T x) \geq 0.5$ whenever $\theta^T x \geq 0$

z

我们就预测y等于0

决策边界是参数的性质而不是训练集的性质。事实上，决策边界就是：theta'x = 0。在决策边界右边被预测为1, 左边被预测为0。



Decision Boundary

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

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

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

Decision boundary

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

$\rightarrow x_1 + x_2 \geq 3$

$x_1 + x_2 < 3 \rightarrow y = 0$

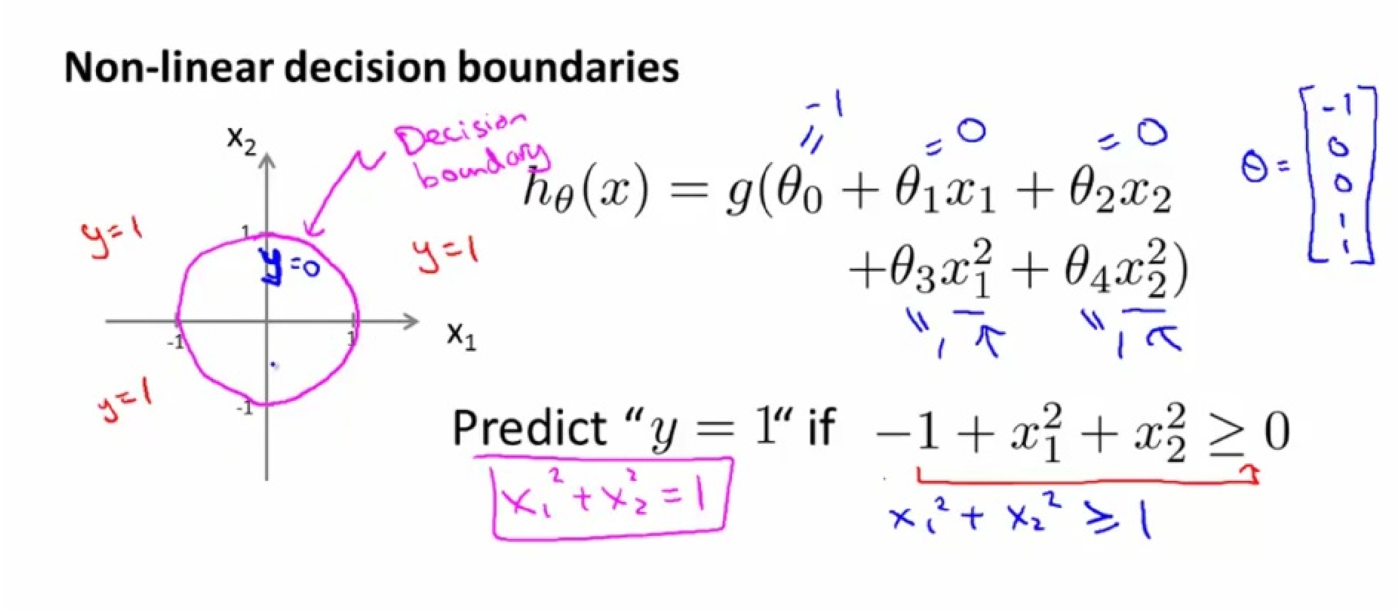
x_1, x_2

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

$\boxed{x_1 + x_2 = 3}$

在这幅图中 我画了一个训练集

非线性决策边界：



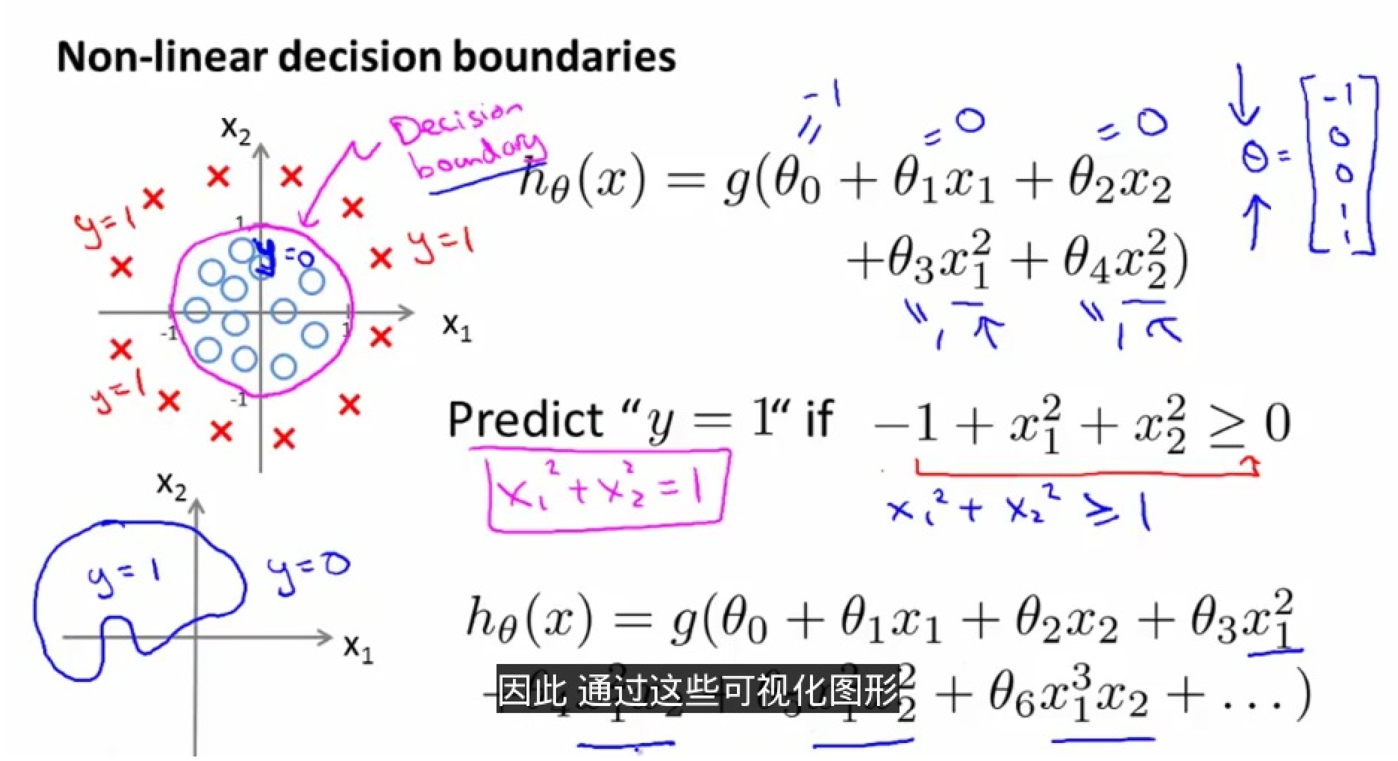
Non-linear decision boundaries

$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2 \geq 0}_{x_1^2 + x_2^2 \geq 1}$

$\boxed{x_1^2 + x_2^2 = 1}$



Non-linear decision boundaries

$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

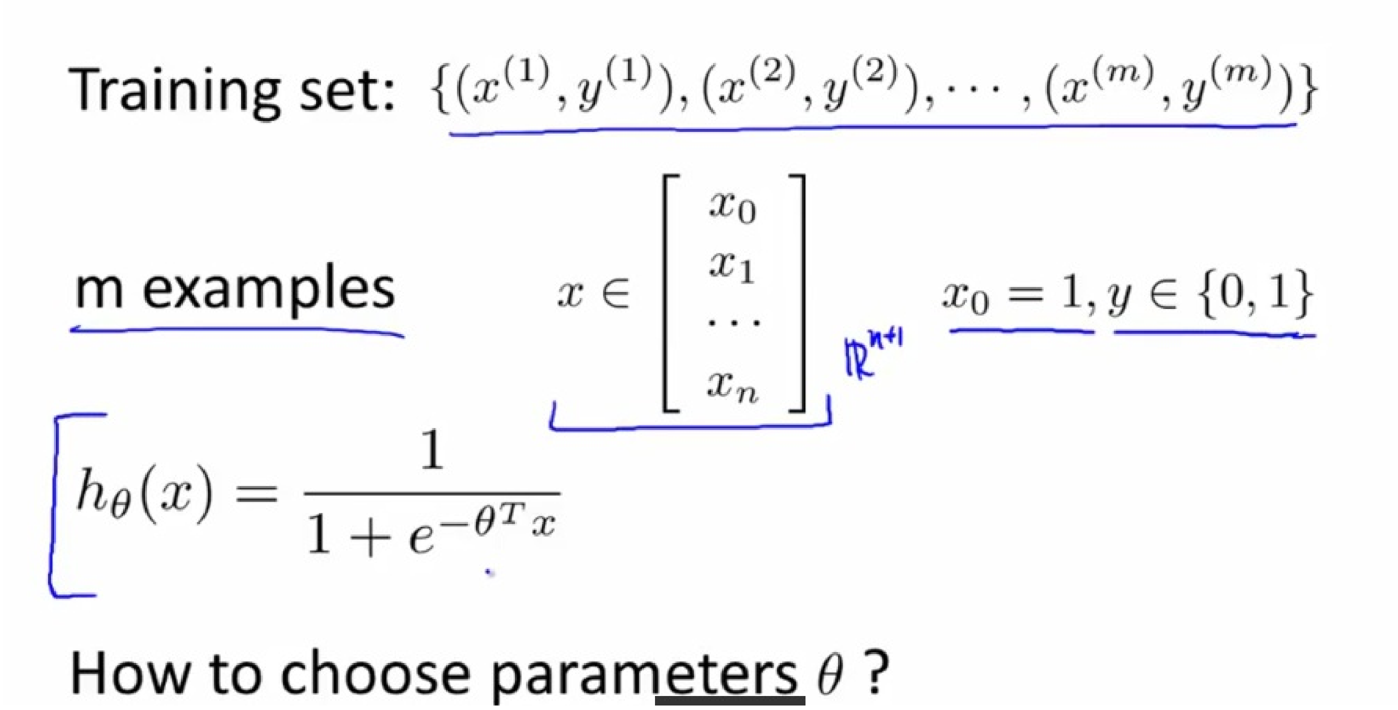
Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2 \geq 0}_{x_1^2 + x_2^2 \geq 1}$

$\boxed{x_1^2 + x_2^2 = 1}$

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 \underline{x_1^2}$$

因此 通过这些可视化图形 $_2 + \theta_6 \underline{x_1^3 x_2} + \ldots)$

但是，如何自动选择参数theta，以便给定一个数据集，可以根据数据自动拟合参数呢？



Training set: $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$

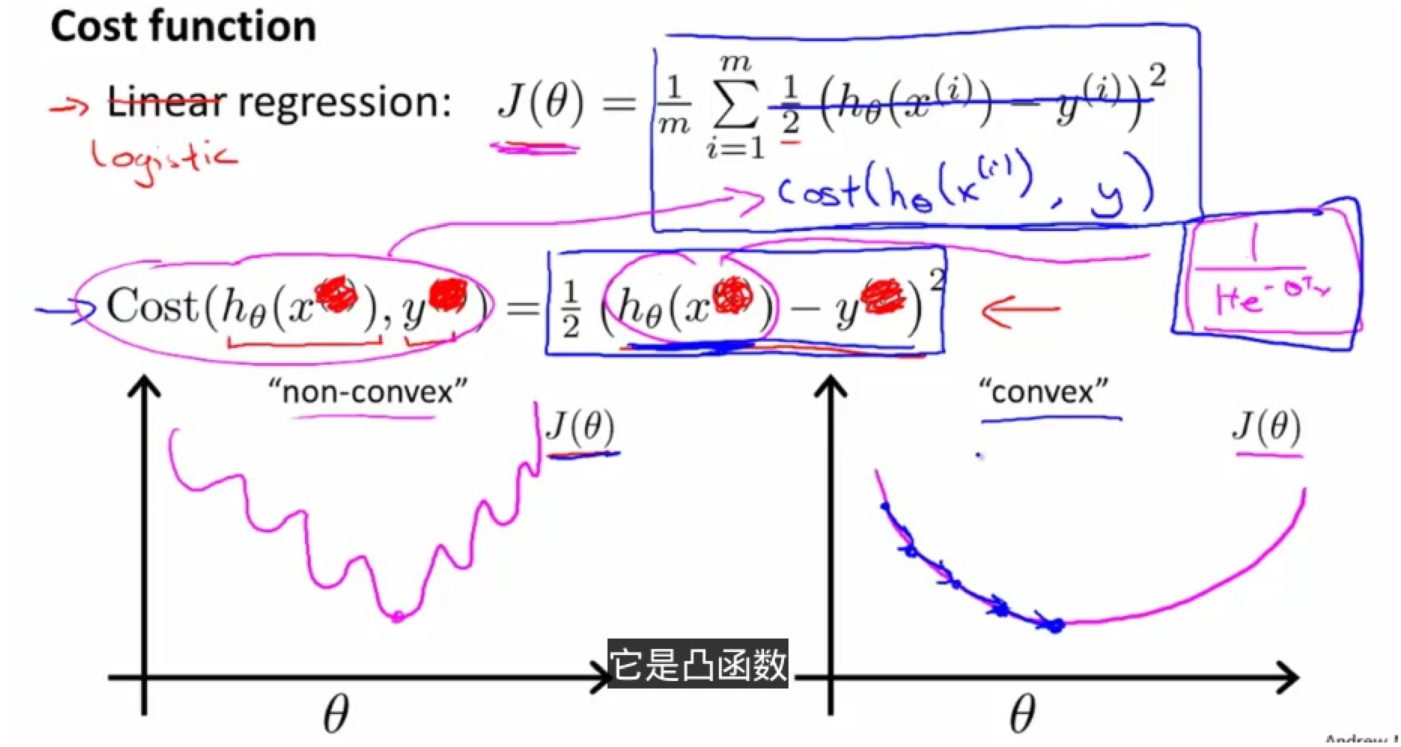
m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad \mathbb{R}^{n+1} \qquad x_0 = 1, y \in \{0,1\}$$

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

How to choose parameters θ ?

4. 代价函数。分类问题代价函数是非凸函数。



Cost function

~~Linear~~ regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

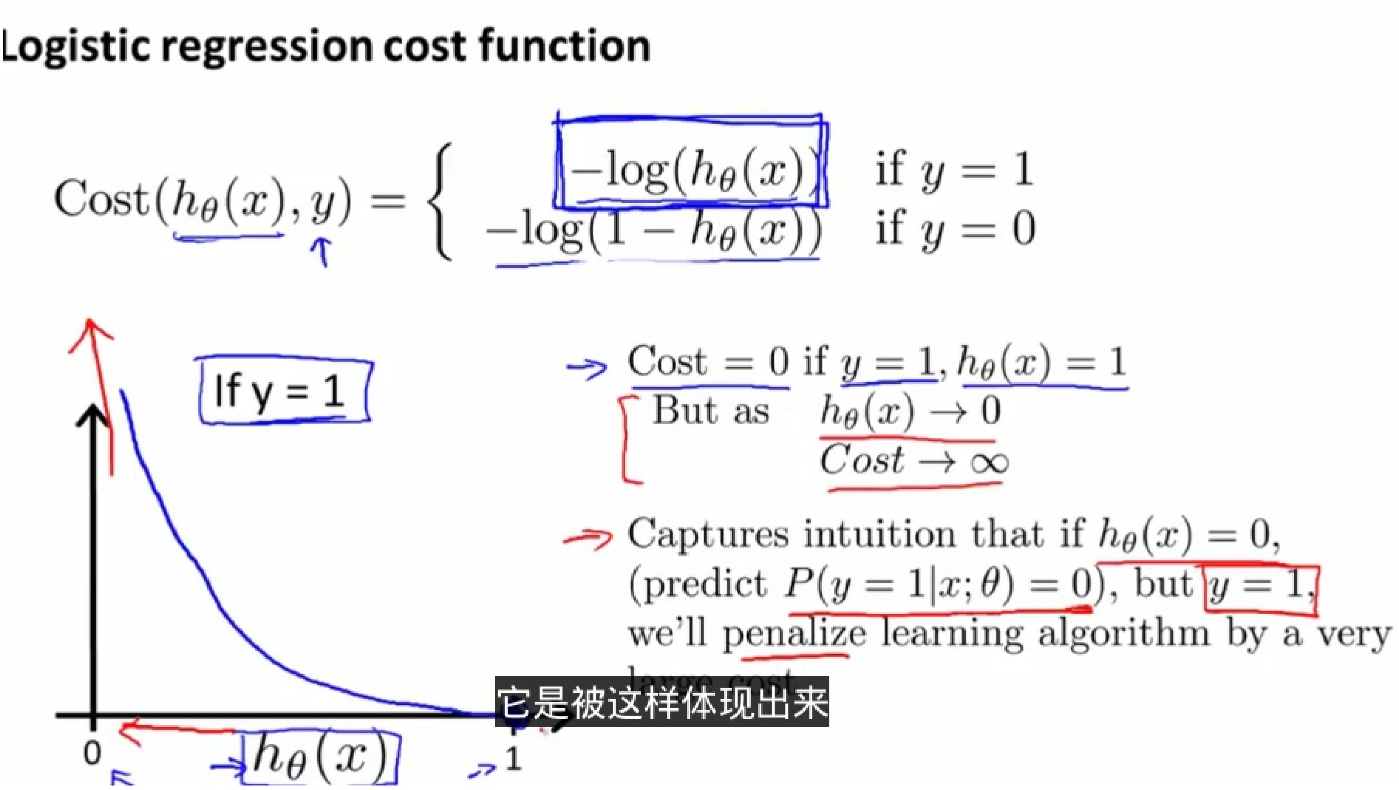
logistic

$\text{cost}(h_\theta(x^{(i)}), y)$

$$\text{Cost}(h_\theta(x), y) = \frac{1}{2}\left(h_\theta(x) - y\right)^2 \leftarrow \frac{1}{1+e^{-\theta^T x}}$$

寻找代价函数为凸函数：

在y=1的时



Logistic regression cost function

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

If y = 1

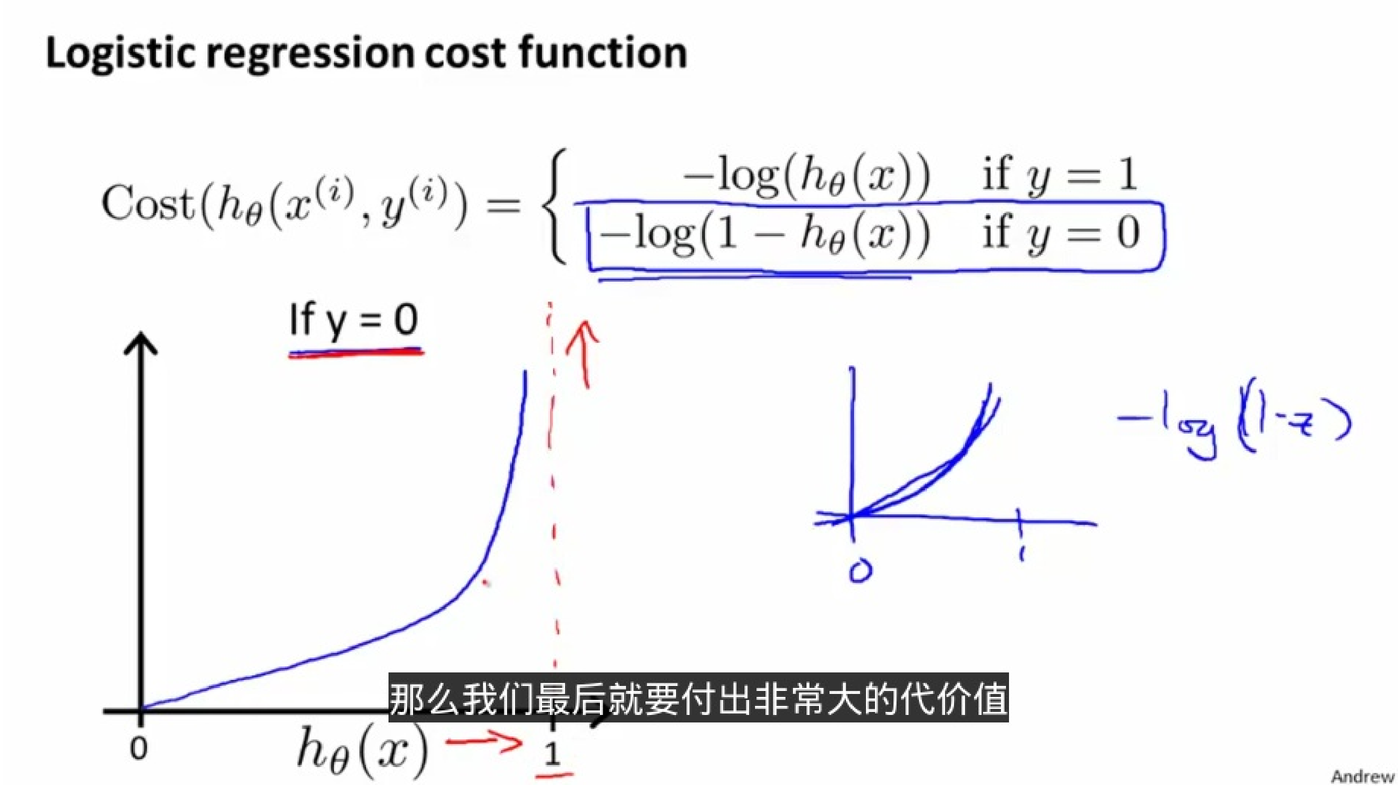
→ Cost $=0$ if $y=1, h_\theta(x)=1$
But as $h_\theta(x)\rightarrow 0$
$Cost \rightarrow \infty$

→ Captures intuition that if $h_\theta(x)=0$, (predict $P(y=1|x;\theta)=0$), but $\boxed{y=1}$, we'll penalize learning algorithm by a very large cost.

它是被这样体现出来

候：

在y=0的时候：



Logistic regression cost function

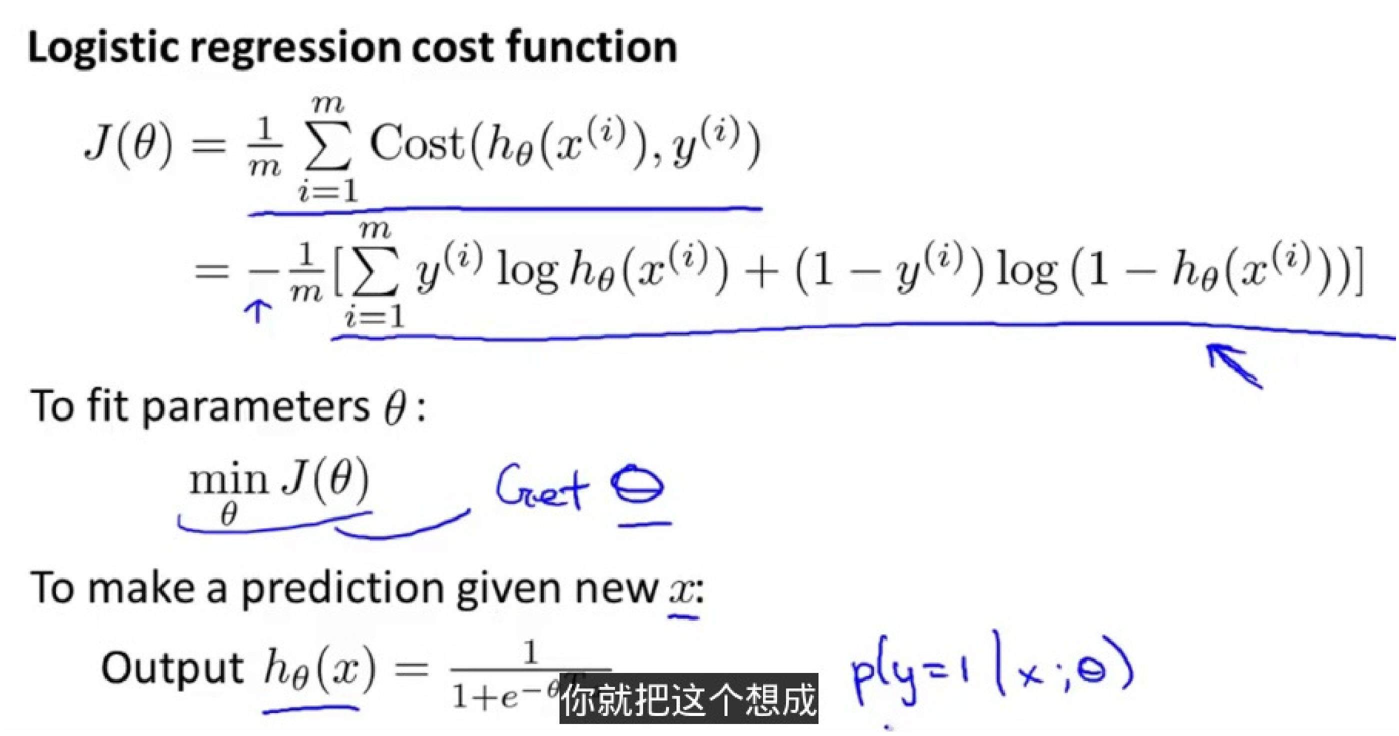
$$\mathrm{Cost}(h_\theta(x^{(i)},y^{(i)})=\begin{cases}-\log(h_\theta(x)) & \text{if } y=1\\ \boxed{-\log(1-h_\theta(x)) \quad \text{if } y=0}\end{cases}$$

If y = 0

那么我们最后就要付出非常大的代价值

显而易见，这样的代价函数是凸函数，没有局部最优值。

5. 实现逻辑回归。



Logistic regression cost function

$$J(\theta) = \underline{\frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})}$$

$$= \underset{\uparrow}{-}\frac{1}{m}\underline{[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))]}$$

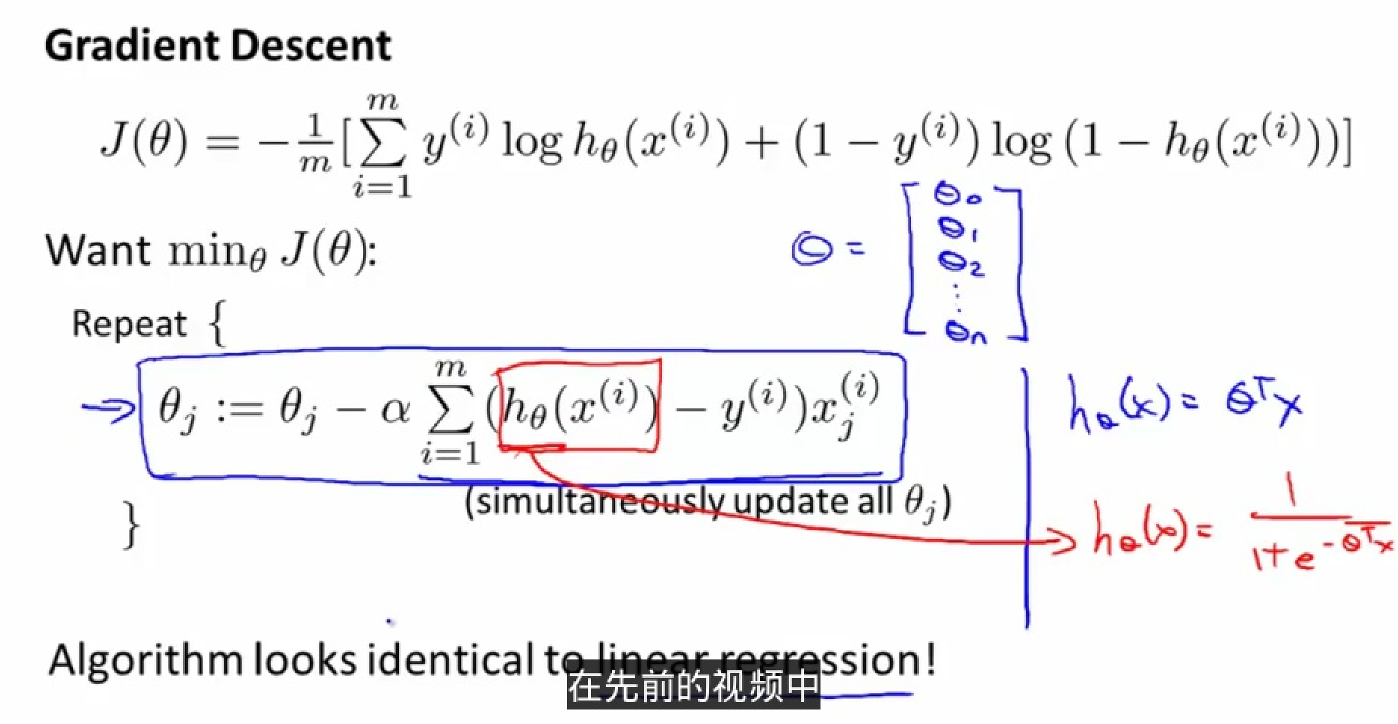
To fit parameters θ:

$$\underline{\min_\theta J(\theta)}$$ Get $\underline{\theta}$

To make a prediction given new $\underline{x}$:

Output $\underline{h_\theta(x)} = \frac{1}{1+e^{-\theta^T x}}$ 你就把这个想成 $p(y=1|x;\theta)$

求偏导，计算代价函数，发现和线性回归梯度下降更新规则相同！



Gradient Descent

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

Want $\min_\theta J(\theta)$: $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$

Repeat {

$$\rightarrow \boxed{\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (\boxed{h_\theta(x^{(i)})} - y^{(i)}) x_j^{(i)}}$$

(simultaneously update all θ_j)

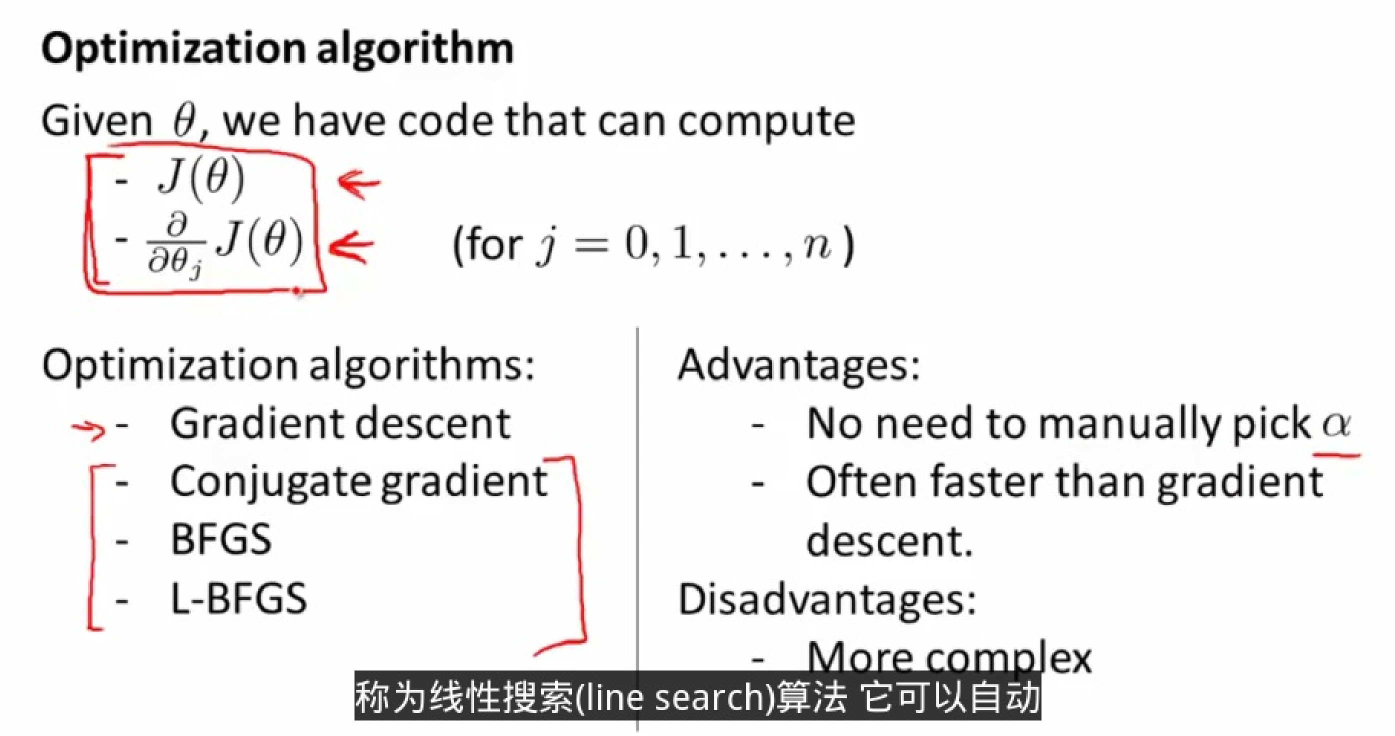
}

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

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

Algorithm looks identical to linear regression! 在先前的视频中

6. 优化算法。



Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

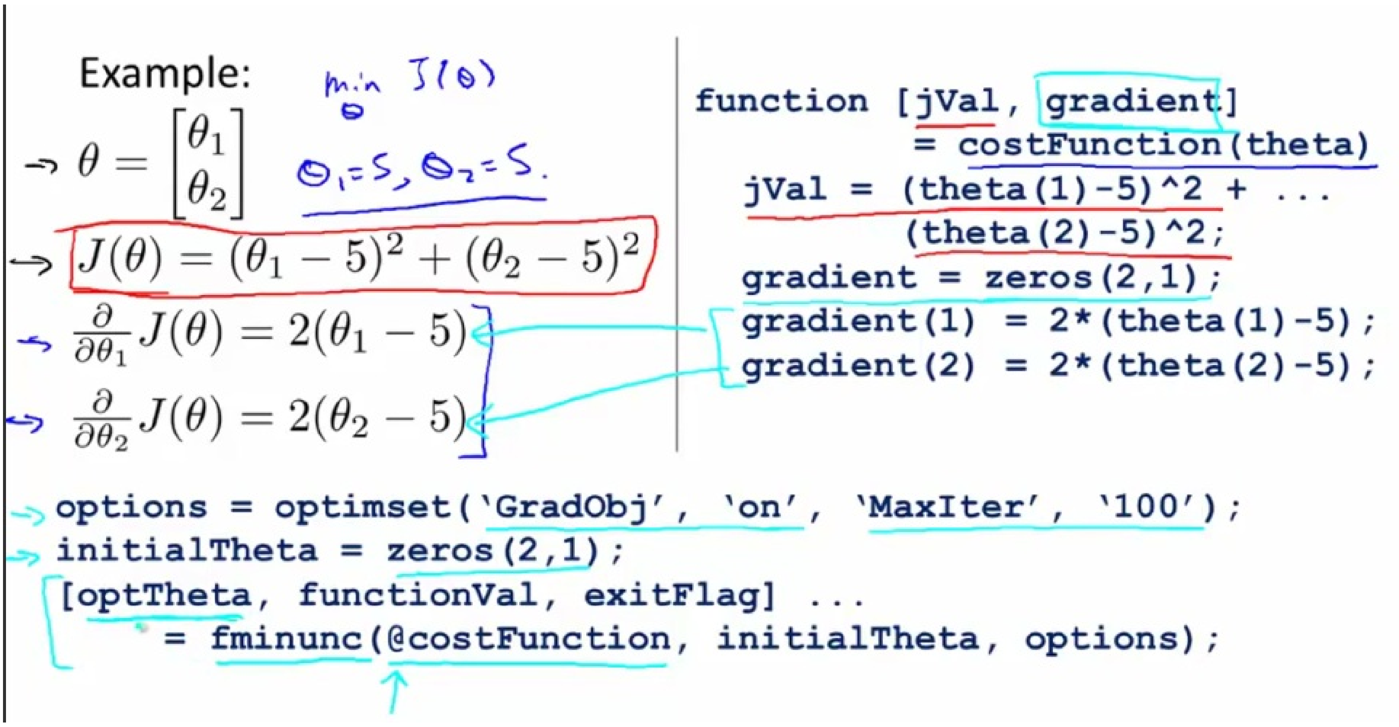
Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

称为线性搜索(line search)算法 它可以自动



Example: $\min_{\theta} J(\theta)$

$\theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}$ $\theta_1=5, \theta_2=5.$

$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

$\frac{\partial}{\partial\theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial\theta_2} J(\theta) = 2(\theta_2 - 5)$

```
function [jVal, gradient]
                = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
            (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

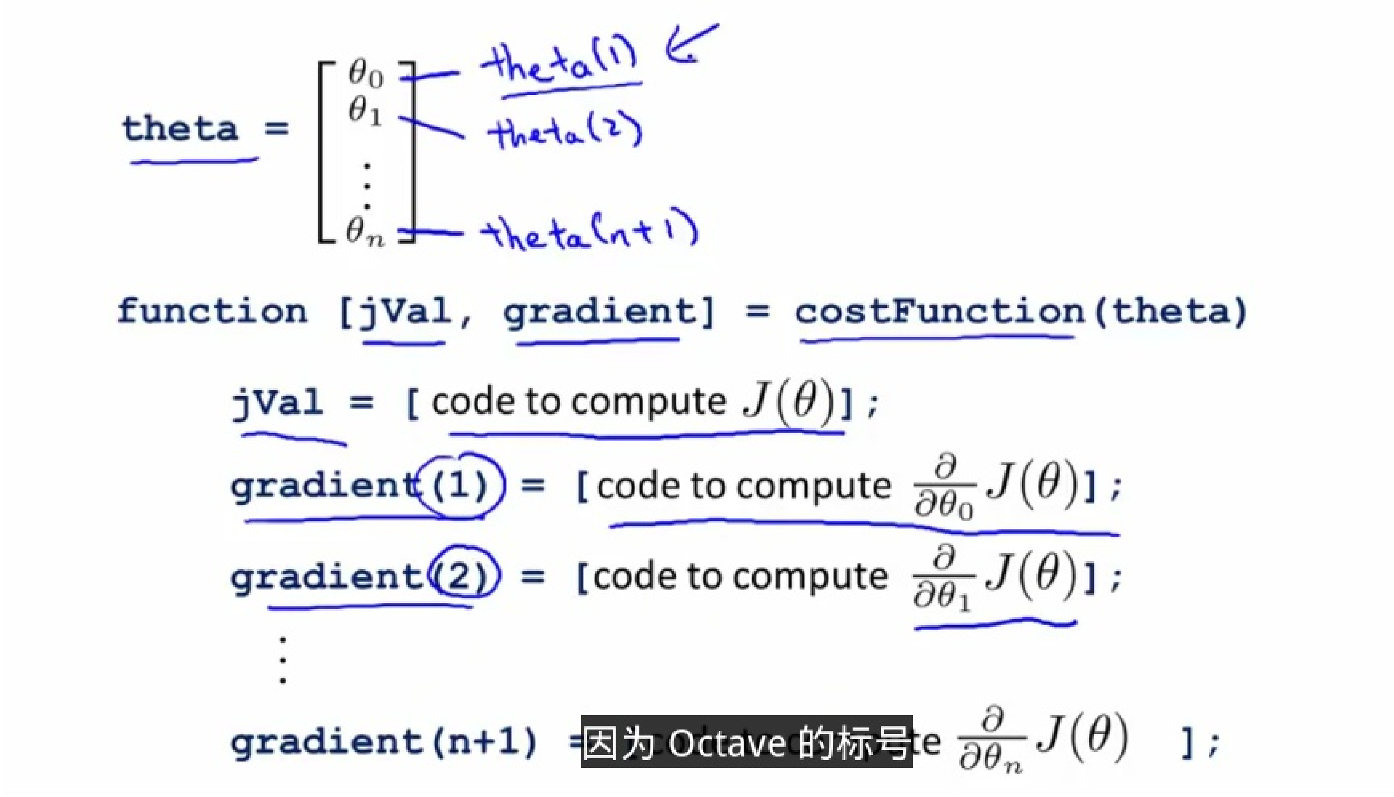
```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
        = fminunc(@costFunction, initialTheta, options);
```

```
function [jVal, gradient] = costFunction(theta)
jVal = (theta(1)-5)^2+(theta(2)-5)^2;
gradient = zeros(2,1);
gradient(1) = 2*(theta(1)-5);
gradient(2) = 2*(theta(2)-5);

octave:1> options = optimset('GradObj', 'on', 'MaxIter', '100');
octave:2> initialTheta = zeros(2,1);
octave:3> [optTheta, functionVal, exitFlag]=fminunc(@costFunction, initialTheta, options)
optTheta =

   5.0000
   5.0000

functionVal =    7.8886e-31
exitFlag =  1
```



theta = $\begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$

```
function [jVal, gradient] = costFunction(theta)
```

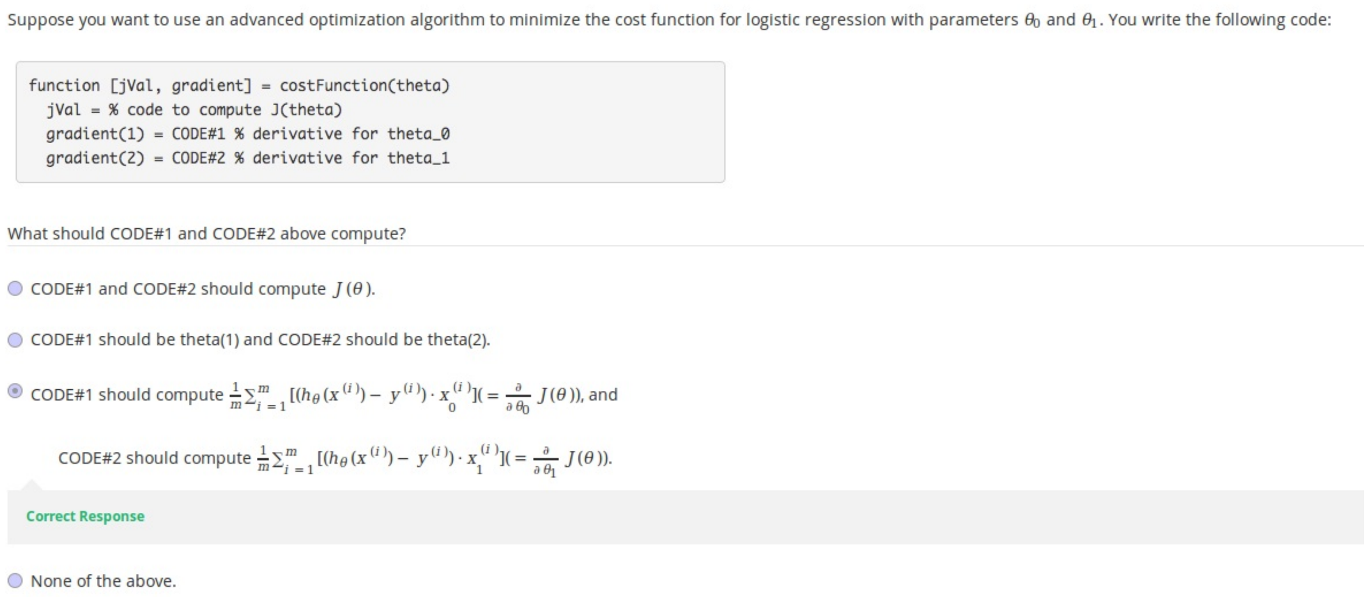
jVal = [code to compute $J(\theta)$] ;

gradient(1) = [code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$] ;

gradient(2) = [code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$] ;

⋮

gradient(n+1) = 因为 Octave 的标号te $\frac{\partial}{\partial\theta_n}J(\theta)$] ;



Suppose you want to use an advanced optimization algorithm to minimize the cost function for logistic regression with parameters θ_0 and θ_1. You write the following code:

```
function [jVal, gradient] = costFunction(theta)
  jVal = % code to compute J(theta)
  gradient(1) = CODE#1 % derivative for theta_0
  gradient(2) = CODE#2 % derivative for theta_1
```

What should CODE#1 and CODE#2 above compute?

○ CODE#1 and CODE#2 should compute $J(\theta)$.

○ CODE#1 should be theta(1) and CODE#2 should be theta(2).

● CODE#1 should compute $\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)})-y^{(i)})\cdot x_0^{(i)}]$(= $\frac{\partial}{\partial\theta_0}J(\theta)$), and

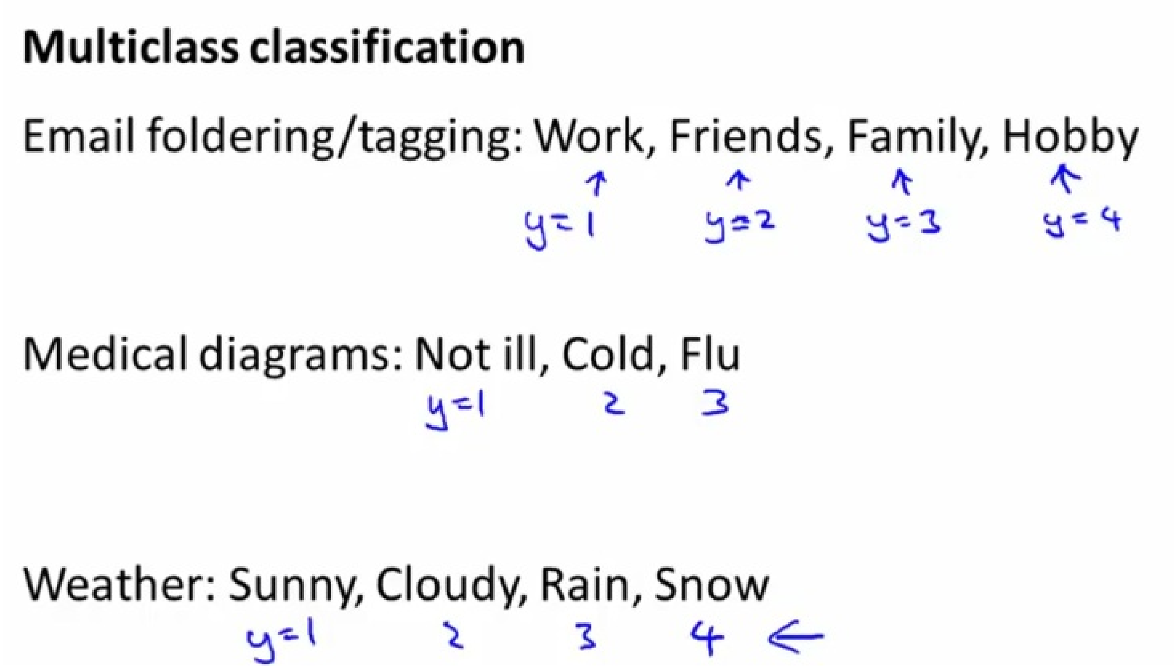
CODE#2 should compute $\frac{1}{m}\sum_{i=1}^{m}[(h_\theta(x^{(i)})-y^{(i)})\cdot x_1^{(i)}]$(= $\frac{\partial}{\partial\theta_1}J(\theta)$).

Correct Response

○ None of the above.

因此可以自己写代价函数，从而使用octave高级优化的方法。

7. 逻辑回归的多分类方法。y取值不再是0,1,而可以是离散的多个。



Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

↑ ↑ ↑ ↑

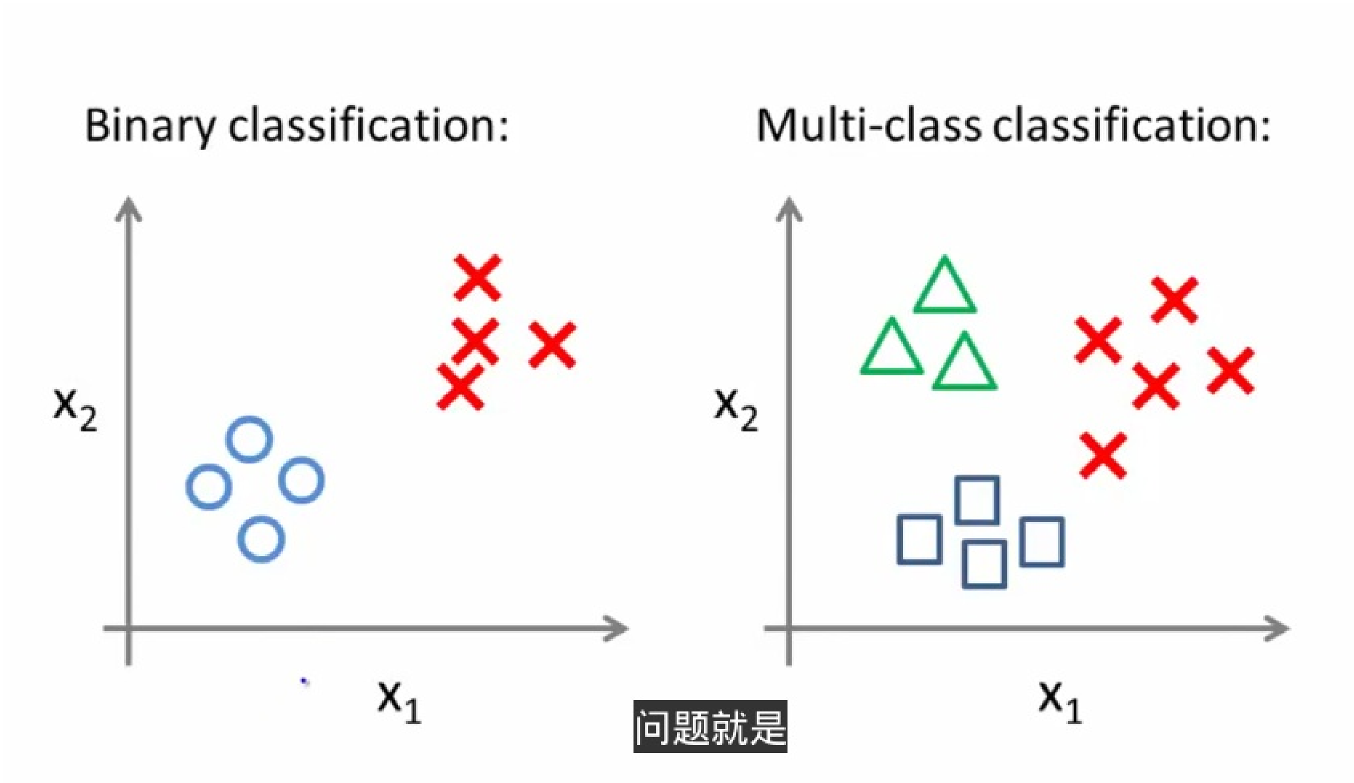
y=1 y=2 y=3 y=4

Medical diagrams: Not ill, Cold, Flu

y=1 2 3

Weather: Sunny, Cloudy, Rain, Snow

y=1 2 3 4 ←

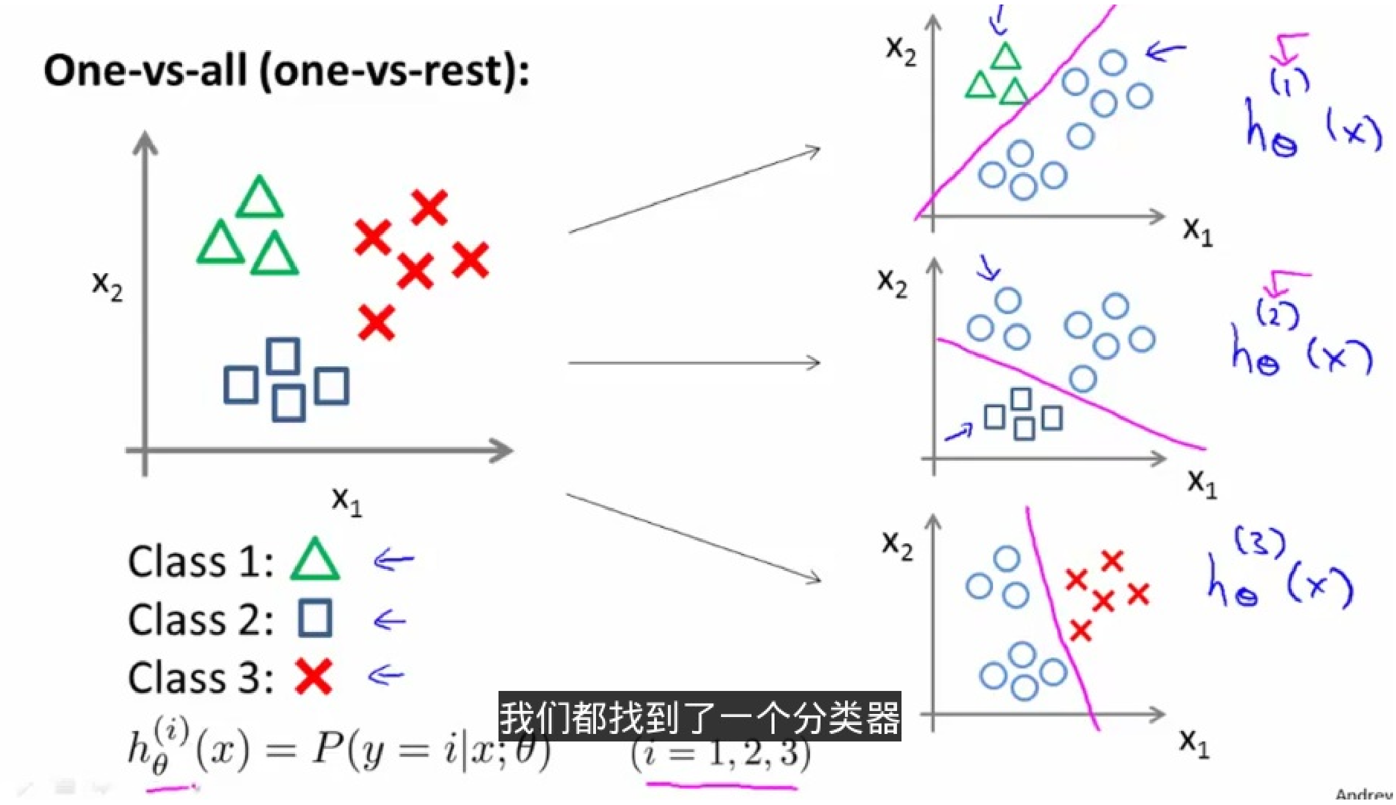


Binary classification:

Multi-class classification:

问题就是

这种问题可以转化成y取二值的逻辑回归分析：即从one-vs-all -> one-vs-rest。



One-vs-all (one-vs-rest):

Class 1: △

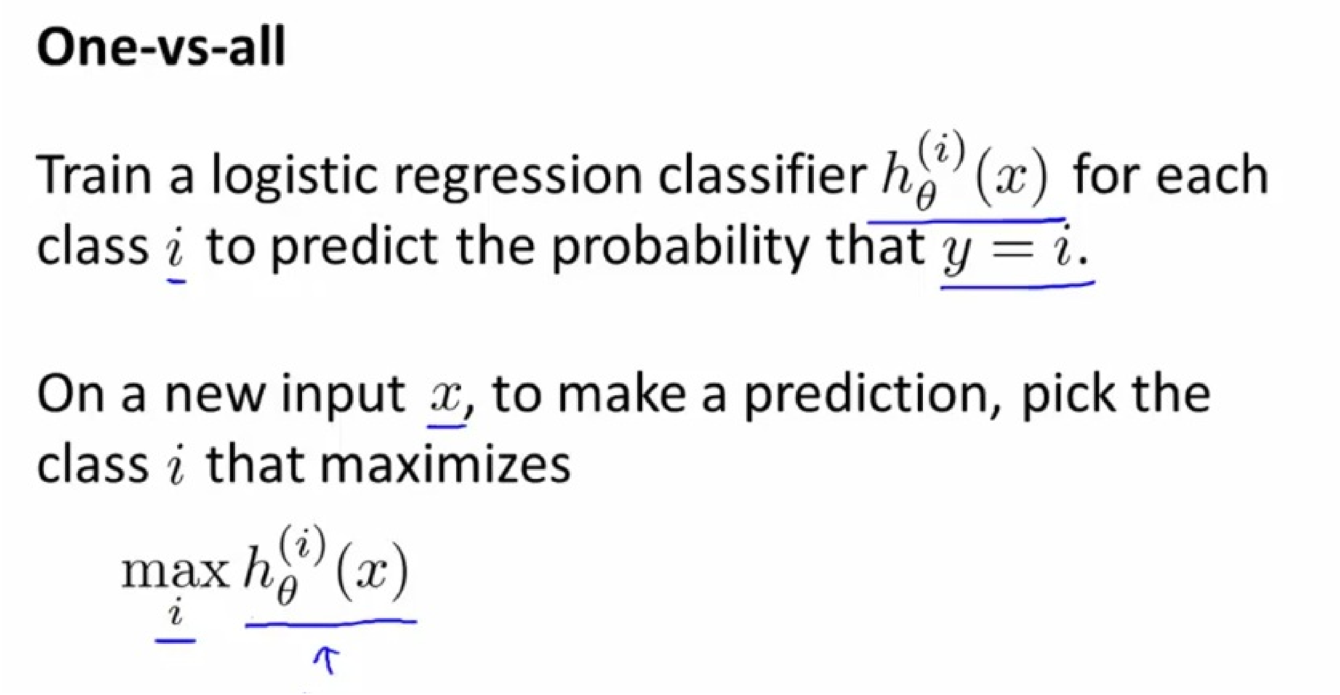
Class 2: □

Class 3: ✕

$h_\theta^{(i)}(x) = P(y = i|x;\theta)$ $(i = 1, 2, 3)$

我们都找到了一个分类器

从而将一个三元分类问题转换成三次一元分类。



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

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