

# Classification

To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

The classification problem is just like the regression problem, except that the values we now want to predict take on only a small number of discrete values. For now, we will focus on the **binary classification problem** in which  $y$  can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.) For instance, if we are trying to build a spam classifier for email, then  $x^{(i)}$  may be some features of a piece of email, and  $y$  may be 1 if it is a piece of spam mail, and 0 otherwise. Hence,  $y \in \{0, 1\}$ . 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+" . Given  $x^{(i)}$ , the corresponding  $y^{(i)}$  is also called the label for the training example.

## Classification

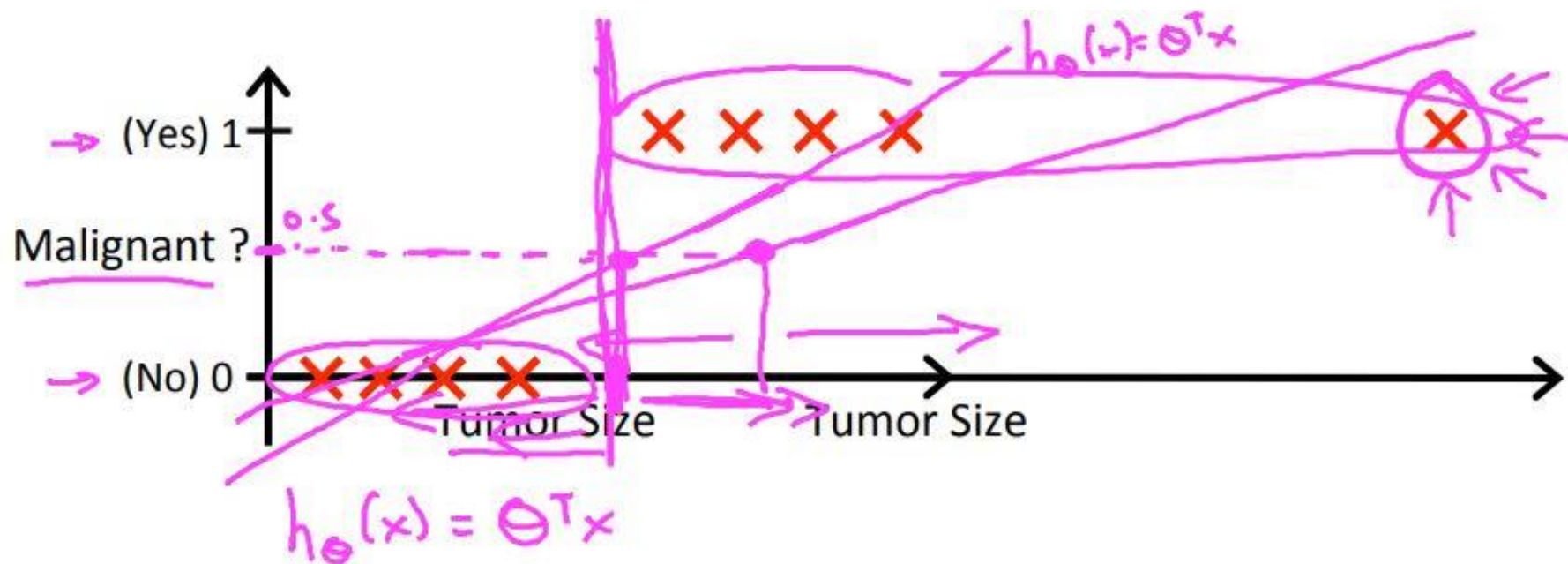
- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

→  $y \in \{0, 1\}$

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

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

→  $y \in \{0, 1, 2, 3\}$



→ 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"



# Hypothesis Representation

We could approach the classification problem ignoring the fact that  $y$  is discrete-valued, and use our old linear regression algorithm to try to predict  $y$  given  $x$ . However, it is easy to construct examples where this method performs very poorly. Intuitively, it also doesn't make sense for  $h_{\theta}(x)$  to take values larger than 1 or smaller than 0 when we know that  $y \in \{0, 1\}$ . To fix this, let's change the form for our hypotheses  $h_{\theta}(x)$  to satisfy  $0 \leq h_{\theta}(x) \leq 1$ . This is accomplished by plugging  $\theta^T x$  into the Logistic Function.

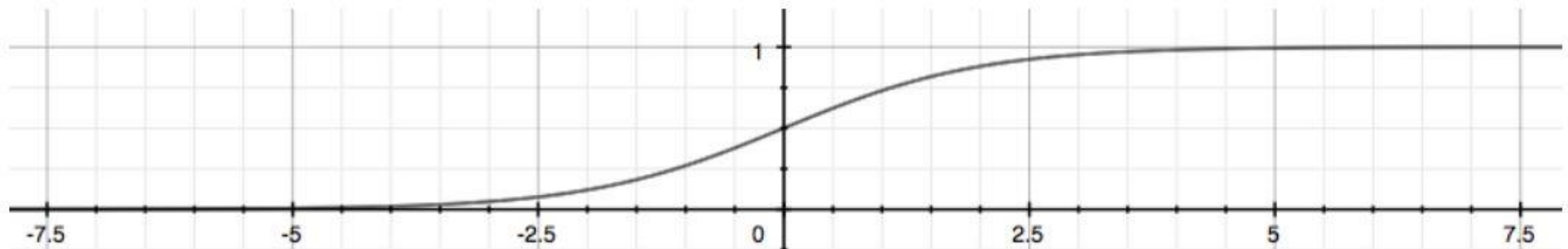
Our new form uses the "Sigmoid Function," also called the "Logistic Function":

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

$$z = \theta^T x$$

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

The following image shows us what the sigmoid function looks like:



The function  $g(z)$ , shown here, maps any real number to the  $(0, 1)$  interval, making it useful for transforming an arbitrary-valued function into a function better suited for classification.



Activate Windows

$h_\theta(x)$  will give us the **probability** that our output is 1. For example,  $h_\theta(x) = 0.7$  gives us a probability of 70% that our output is 1. Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

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

Classification:  $y = 0 \text{ or } 1$

$h_{\theta}(x)$  can be  $> 1$  or  $< 0$

Logistic Regression:  $0 \leq h_{\theta}(x) \leq 1$

Classification



## Logistic Regression Model

Want  $0 \leq h_{\theta}(x) \leq 1$

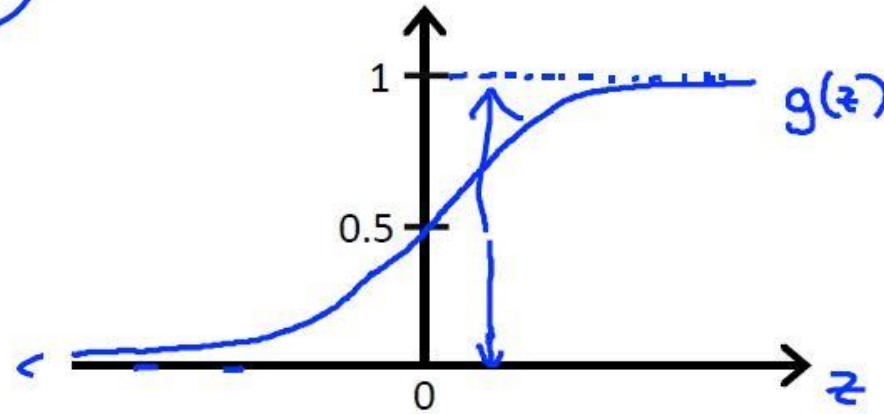
$$h_{\theta}(x) = g(\theta^T x)$$

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

$\theta^T x$

→ Sigmoid function  
→ Logistic function

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



Parameters  $\underline{\theta}$ .



Activate Windows

## Interpretation of Hypothesis Output

$h_{\theta}(x)$

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

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

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

Tell patient that 70% chance of tumor being malignant

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

$$\underline{y = 0 \text{ or } 1}$$

“probability that  $y = 1$ , given  $x$ ,  
parameterized by  $\theta$ ”

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

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

## Logistic regression

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

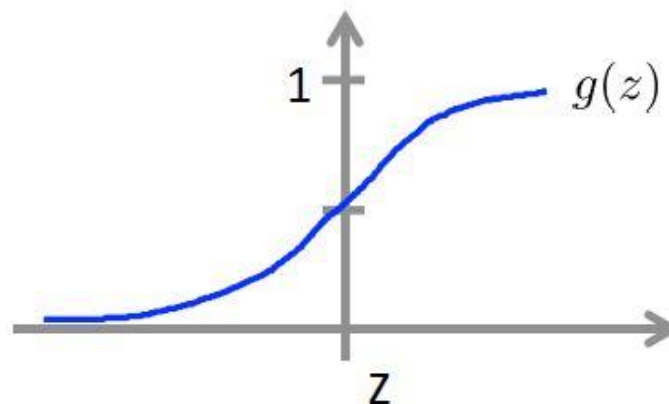
$$g(z) = \frac{1}{1+e^{-z}}$$

Suppose predict “ $y = 1$ ” if  $h_{\theta}(x) \geq 0.5$

$$\theta^T x \geq 0$$

predict “ $y = 0$ ” if  $h_{\theta}(x) < 0.5$

$$\theta^T x < 0$$



$$g(z) \geq 0.5 \\ \text{when } z \geq 0$$

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

$$g(z) < 0.5$$

$$\text{when } z < 0$$

Activate Win  
Go to PC settings

# Decision Boundary

In order to get our discrete 0 or 1 classification, we can translate the output of the hypothesis function as follows:

$$h_{\theta}(x) \geq 0.5 \rightarrow y = 1$$

$$h_{\theta}(x) < 0.5 \rightarrow y = 0$$

The way our logistic function  $g$  behaves is that when its input is greater than or equal to zero, its output is greater than or equal to 0.5:

$$g(z) \geq 0.5$$

*when*  $z \geq 0$

Remember.

$$z = 0, e^0 = 1 \Rightarrow g(z) = 1/2$$
$$z \rightarrow \infty, e^{-\infty} \rightarrow 0 \Rightarrow g(z) = 1$$
$$z \rightarrow -\infty, e^{\infty} \rightarrow \infty \Rightarrow g(z) = 0$$



Activate Windows

Go to PC settings to activate Windows



So if our input to  $g$  is  $\theta^T X$ , then that means:

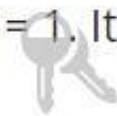
$$h_{\theta}(x) = g(\theta^T x) \geq 0.5$$

when  $\theta^T x \geq 0$

From these statements we can now say:

$$\theta^T x \geq 0 \Rightarrow y = 1$$
$$\theta^T x < 0 \Rightarrow y = 0$$

The **decision boundary** is the line that separates the area where  $y = 0$  and where  $y = 1$ . It is created by our hypothesis function.



Activate Windows  
Go to PC settings to activate

---

**Example:**

$$\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$

$$y = 1 \text{ if } 5 + (-1)x_1 + 0x_2 \geq 0$$

$$5 - x_1 \geq 0$$

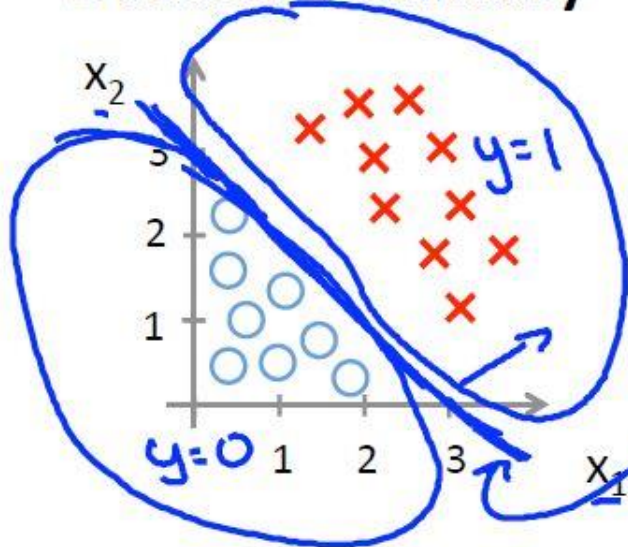
$$-x_1 \geq -5$$

$$x_1 \leq 5$$

In this case, our decision boundary is a straight vertical line placed on the graph where  $x_1 = 5$ , and everything to the left of that denotes  $y = 1$ , while everything to the right denotes  $y = 0$ .

Again, the input to the sigmoid function  $g(z)$  (e.g.  $\theta^T X$ ) doesn't need to be linear, and could be a function that describes a circle (e.g.  $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$ ) or any shape to fit our data.

## Decision Boundary



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

$$h_{\theta}(x) = g(\underbrace{\theta_0}_{-3} + \underbrace{\theta_1}_{1}x_1 + \underbrace{\theta_2}_{1}x_2)$$

Decision boundary

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

$$\theta^T x$$

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

$x_1, x_2$

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

$$\boxed{x_1 + x_2 = 3}$$

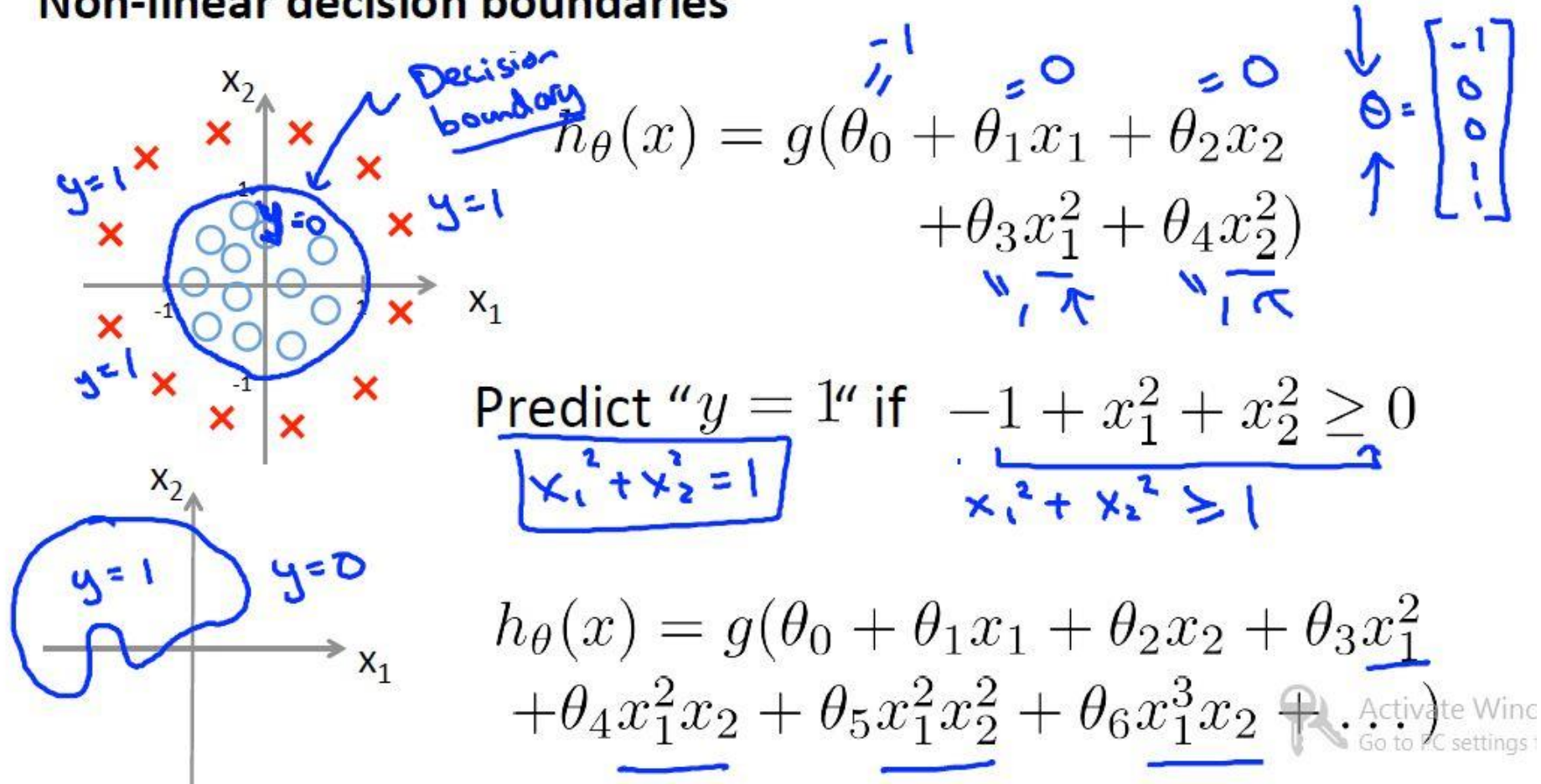
$$\rightarrow x_1 + x_2 < 3$$

$$y = 0$$



Activate  
Go to PC's

## Non-linear decision boundaries





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

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} \mathbb{R}^{n+1}$$

$$\underline{x_0 = 1}, \underline{y \in \{0, 1\}}$$

$$\left[ h_{\theta}(x) = \frac{1}{1 + e^{-\underline{\theta^T x}}} \right]$$

How to choose parameters  $\theta$  ?



Activate W  
Go to PC setting

# Cost Function

We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.

Instead, our cost function for logistic regression looks like:

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

$$\text{Cost}(h_{\theta}(x), y) = -\log(h_{\theta}(x)) \quad \text{if } y = 1$$

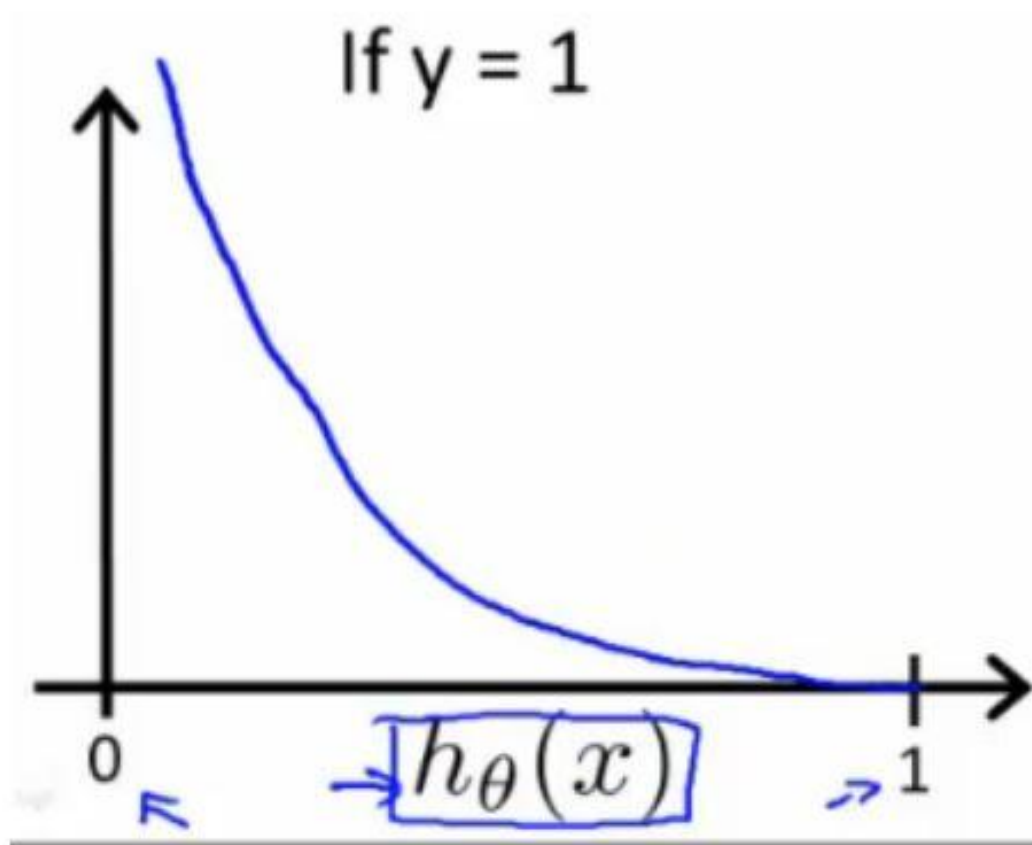
$$\text{Cost}(h_{\theta}(x), y) = -\log(1 - h_{\theta}(x)) \quad \text{if } y = 0$$



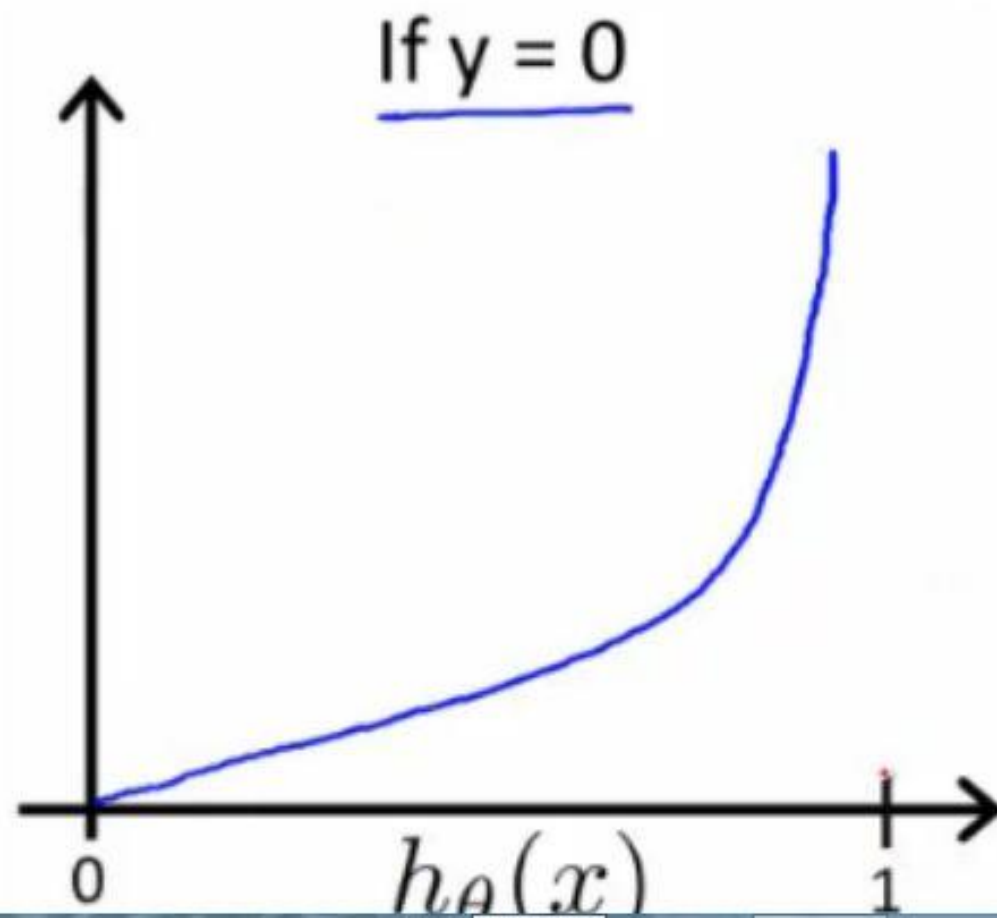
Activate Windows

Go to PC settings to activate

When  $y = 1$ , we get the following plot for  $J(\theta)$  vs  $h_{\theta}(x)$ :



Similarly, when  $y = 0$ , we get the following plot for  $J(\theta)$  vs  $h_\theta(x)$ :



$\text{Cost}(h_{\theta}(x), y) = 0$  if  $h_{\theta}(x) = y$

$\text{Cost}(h_{\theta}(x), y) \rightarrow \infty$  if  $y = 0$  and  $h_{\theta}(x) \rightarrow 1$

$\text{Cost}(h_{\theta}(x), y) \rightarrow \infty$  if  $y = 1$  and  $h_{\theta}(x) \rightarrow 0$

If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.

Note that writing the cost function in this way guarantees that  $J(\theta)$  is convex for logistic regression.



## Cost function

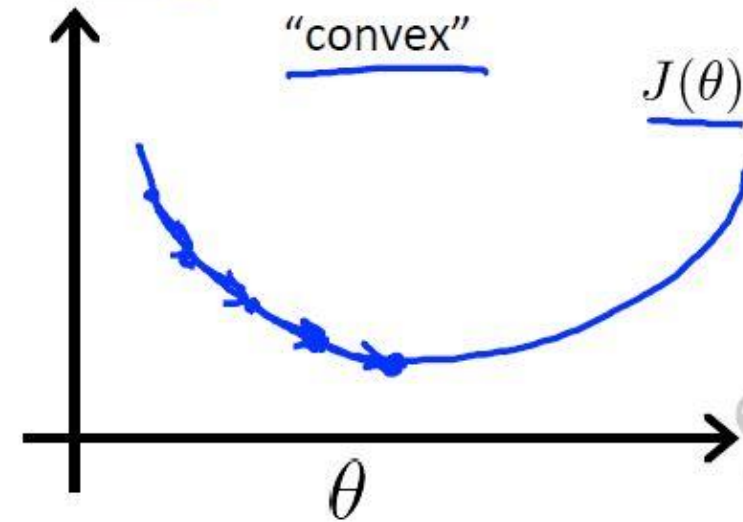
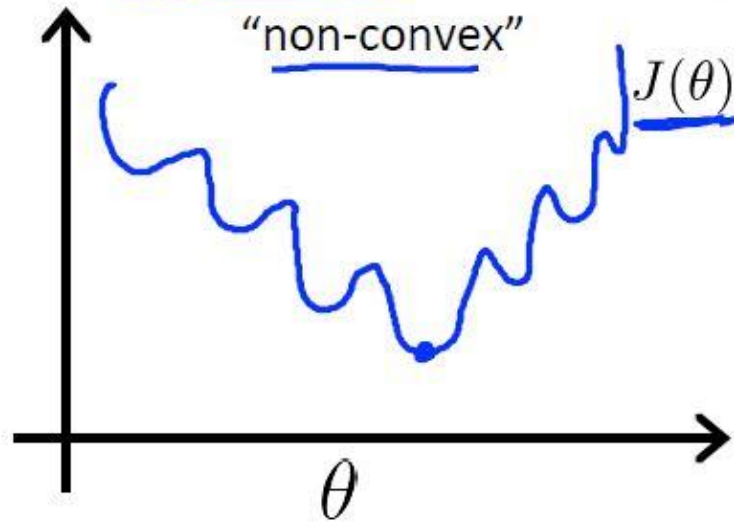
→ Linear regression:  
logistic

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

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

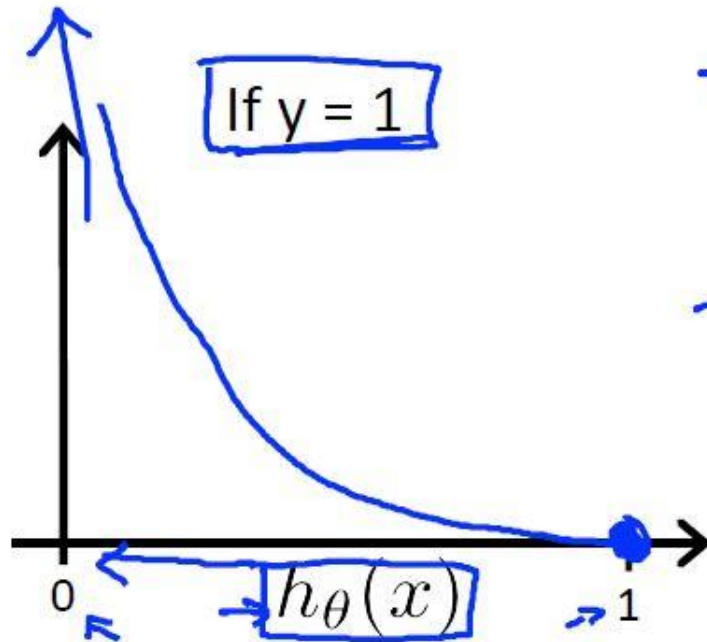
$$\text{Cost}(h_{\theta}(x), y) = \frac{1}{2} (h_{\theta}(x) - y)^2$$

$\frac{1}{1 + e^{-\theta^T x}}$



## Logistic regression cost function

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

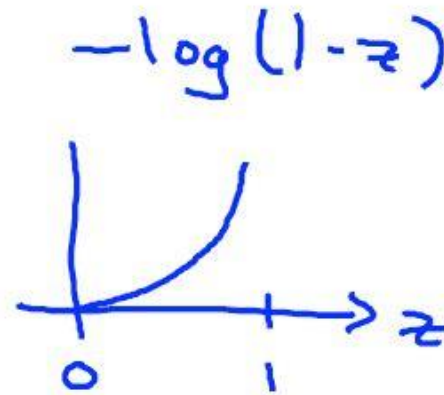
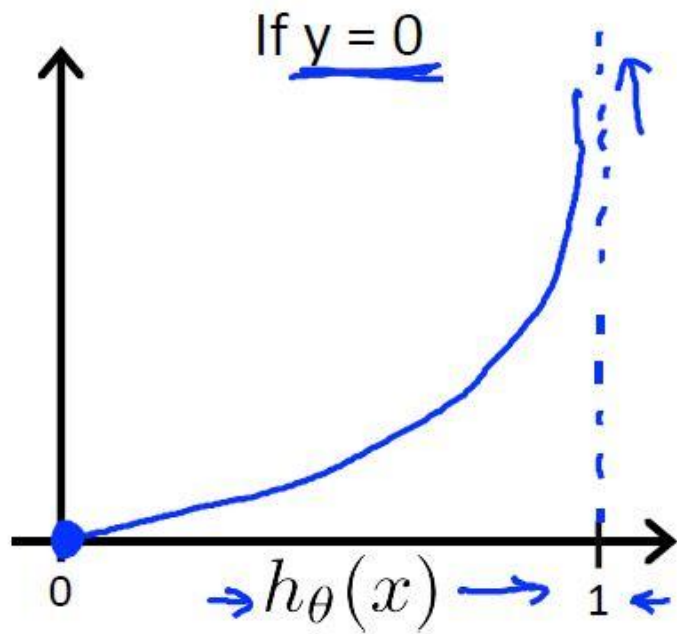


→ Cost = 0 if  $y = 1, h_\theta(x) = 1$   
But as  $h_\theta(x) \rightarrow 0$   
 $\text{Cost} \rightarrow \infty$

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

## Logistic regression cost function

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



# Simplified Cost Function and Gradient Descent

We can compress our cost function's two conditional cases into one case:

$$\text{Cost}(h_{\theta}(x), y) = -y \log(h_{\theta}(x)) - (1 - y) \log(1 - h_{\theta}(x))$$

Notice that when  $y$  is equal to 1, then the second term  $(1 - y) \log(1 - h_{\theta}(x))$  will be zero and will not affect the result. If  $y$  is equal to 0, then the first term  $-y \log(h_{\theta}(x))$  will be zero and will not affect the result.

We can fully write out our entire cost function as follows:

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

A vectorized implementation is:

$$h = g(X\theta)$$
$$J(\theta) = \frac{1}{m} \cdot \left( -y^T \log(h) - (1 - y)^T \log(1 - h) \right)$$



## Logistic regression cost function

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

To fit parameters  $\theta$  :

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

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

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

$$\underline{p(y=1 | x; \theta)}$$



## Gradient Descent

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

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

Repeat {

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

}

(simultaneously update all  $\theta_j$ )

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{n} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$



## Gradient Descent

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

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

Repeat {

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

} (simultaneously update all  $\theta_j$ )

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad \leftarrow \text{for } i = 0 \text{ to } n$$

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

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

Algorithm looks identical to linear regression!

## Multiclass classification

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

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

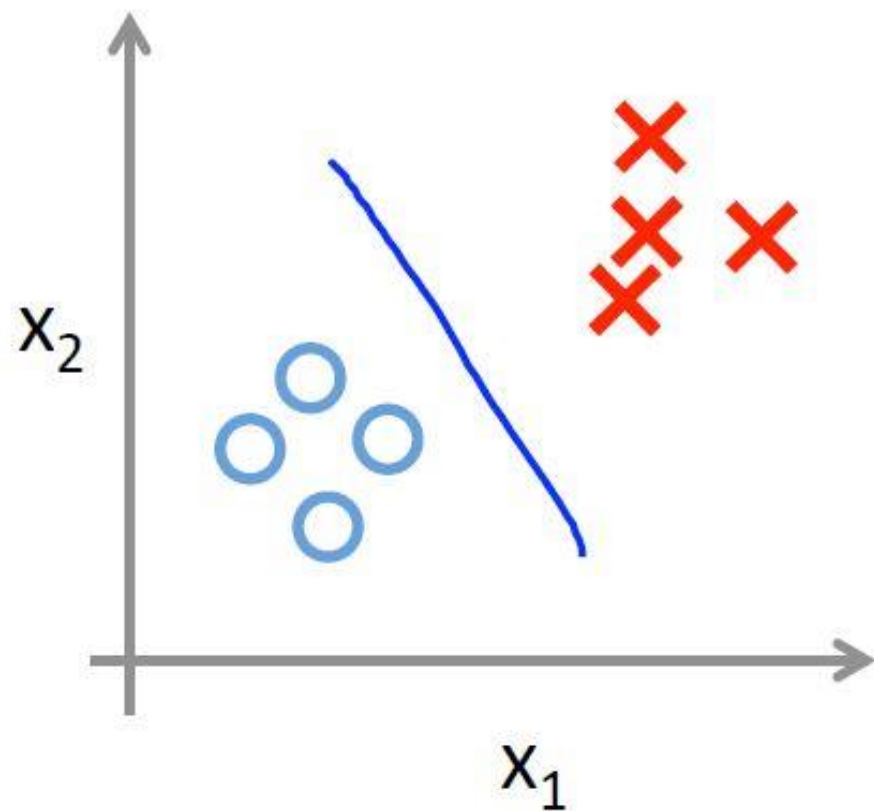
Medical diagrams: Not ill, Cold, Flu

$y=1$  2 3

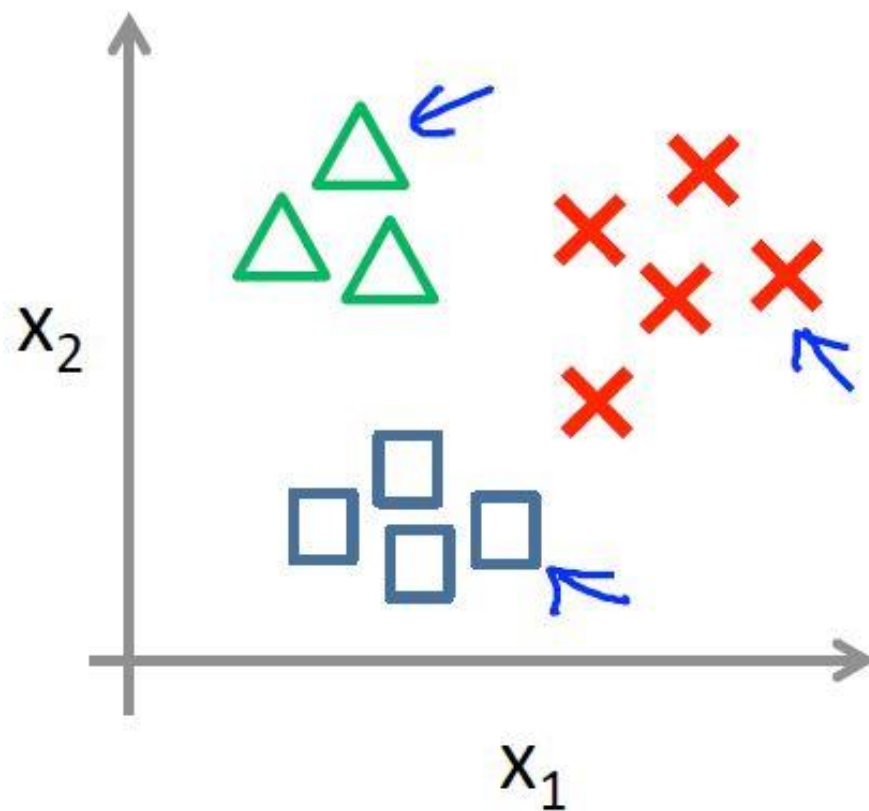
Weather: Sunny, Cloudy, Rain, Snow

$y=1$  2 3 4  $\leftarrow$

Binary classification:

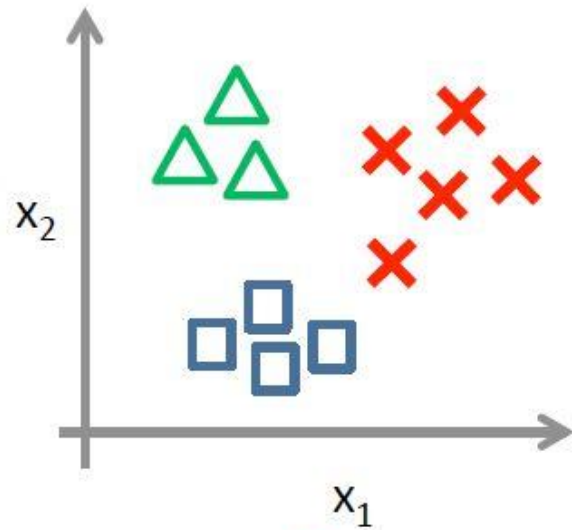




Multi-class classification:








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

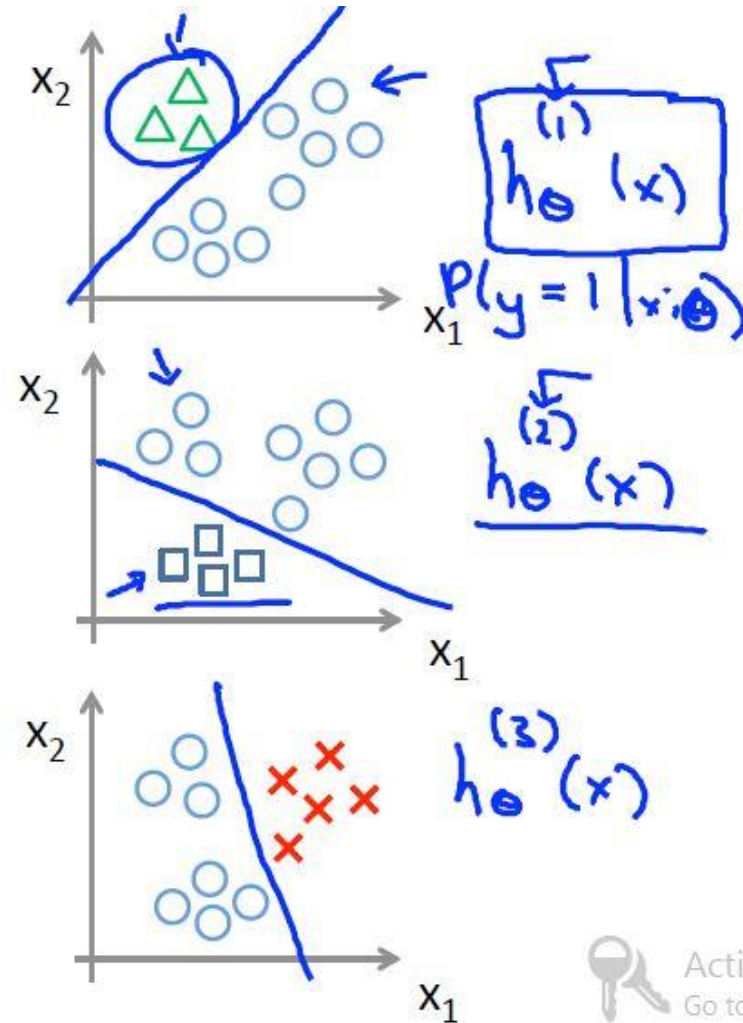


Class 1:  

Class 2:  

Class 3:  

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





## One-vs-all

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

On a new input  $\underline{x}$ , to make a prediction, pick the class  $i$  that maximizes

$$\max_{\underline{i}} \underline{h_{\theta}^{(i)}(x)}$$
