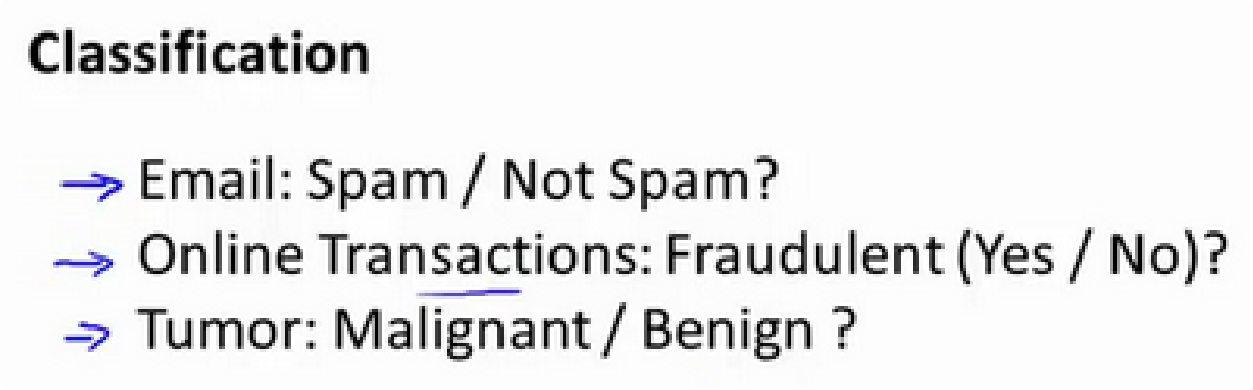
六、逻辑回归

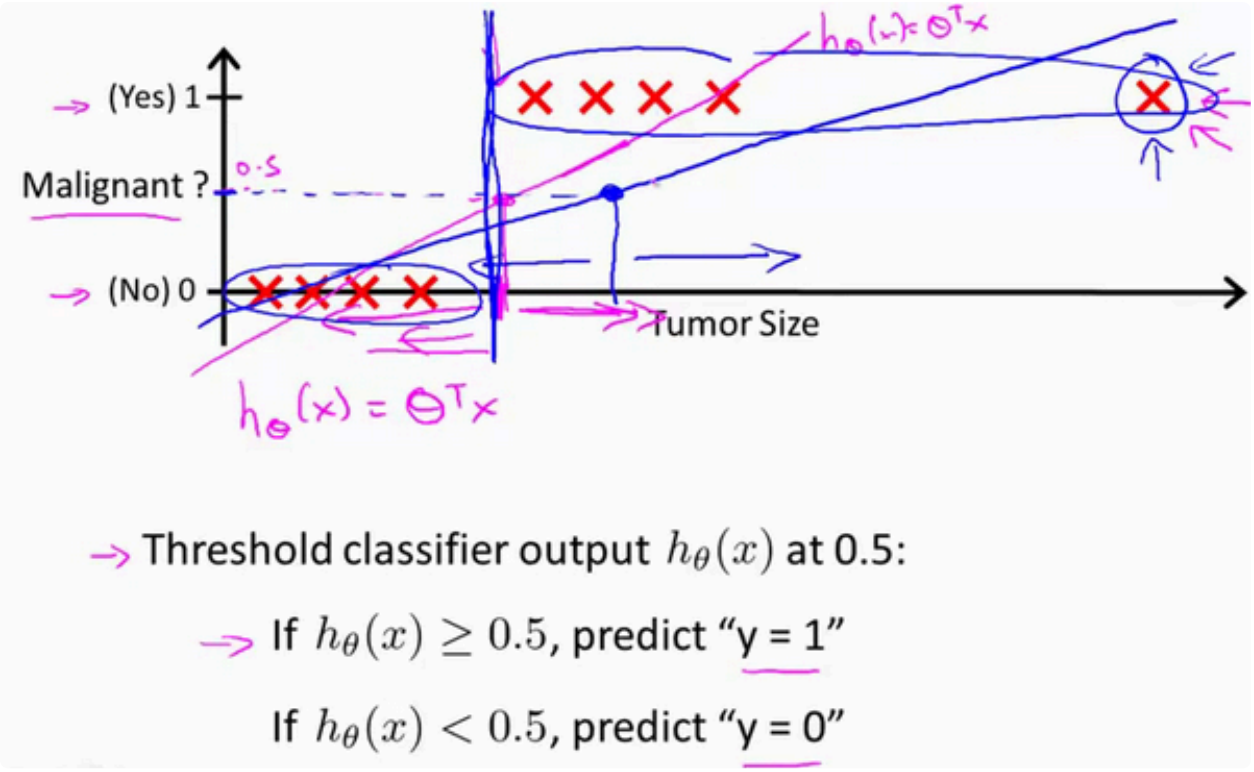
六、逻辑回归(Logistic Regression)

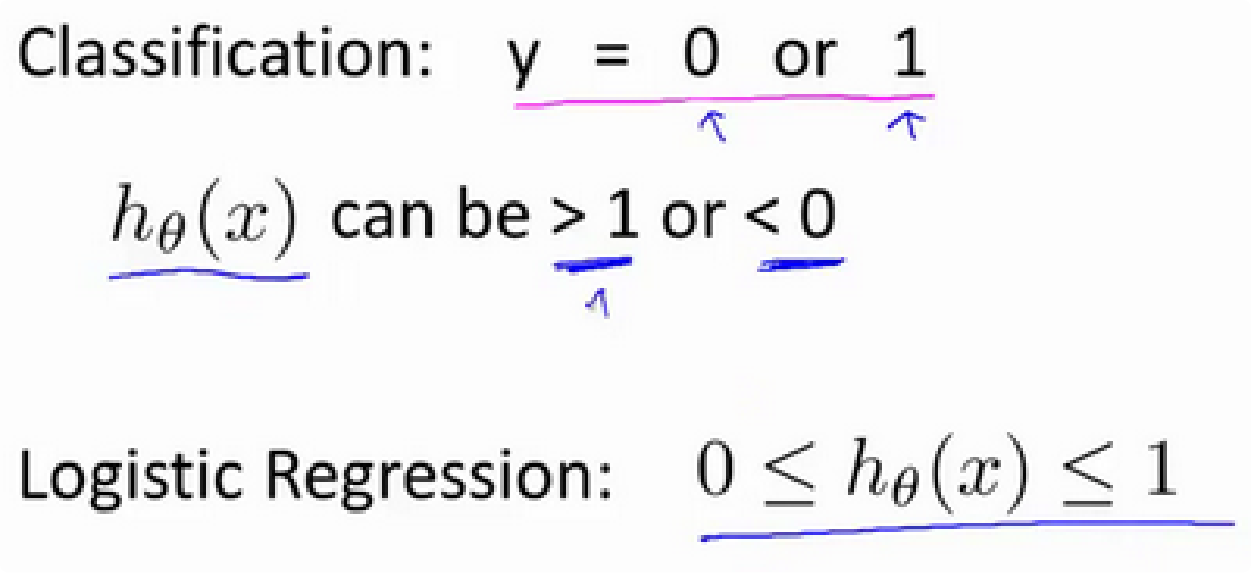
6.1 分类问题

在分类问题中，我们尝试预测的是结果是否属于某一个类（例如正确或错误）。分类问题的例子有：判断一封电子邮件是否是垃圾邮件；判断一次金融交易是否是欺诈；之前我们也谈到了肿瘤分类问题的例子，区别一个肿瘤是恶性的还是良性的。



二元的分类问题
我们将因变量(**dependent variable**)可能属于的两个类分别称为负向类（**negative class**）和正向类（**positive class**），则因变量$y \in 0,1$ ，其中 0 表示负向类，1 表示正向类。

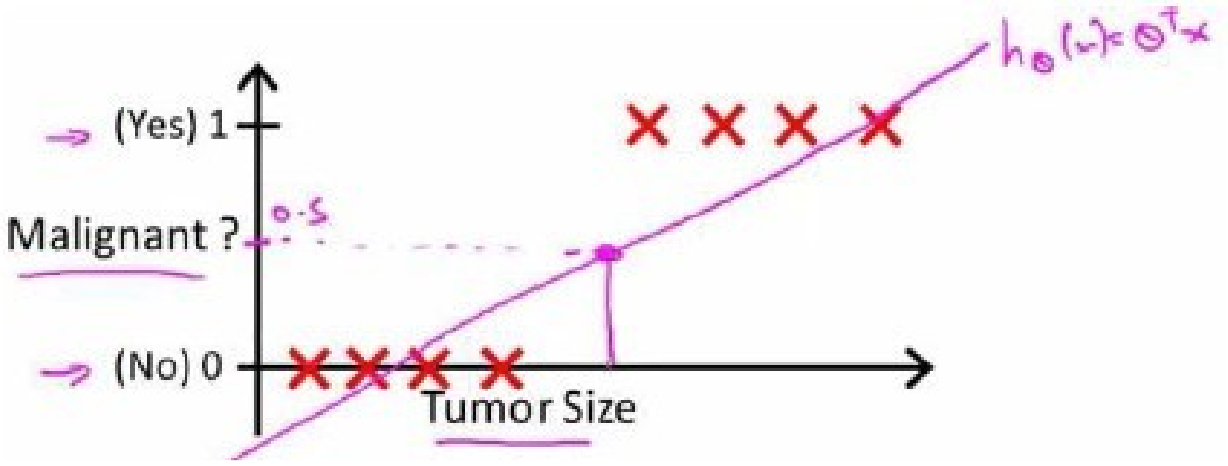




逻辑回归算法的性质是：它的输出值永远在0到 1 之间。
逻辑回归算法是分类算法，它适用于标签 y 取值离散的情况，如：1 0 0 1。

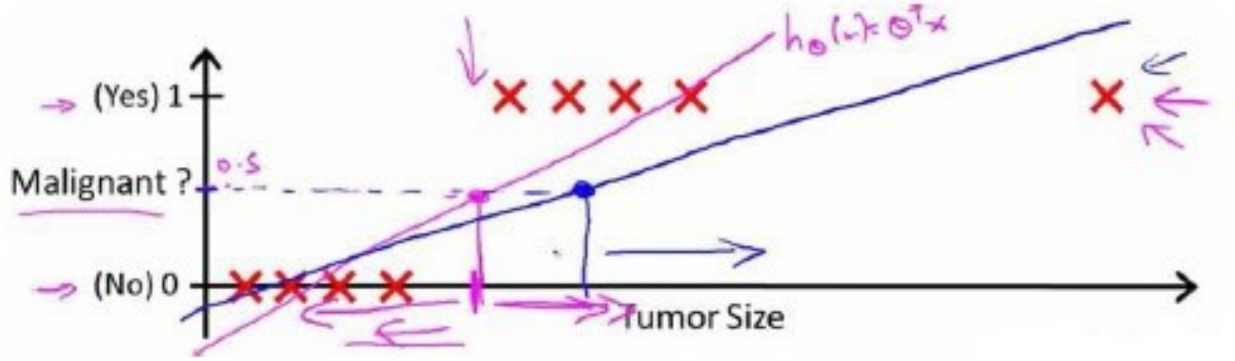
6.2 假说表示

回顾在一开始提到的乳腺癌分类问题，我们可以用线性回归的方法求出适合数据的一条直线：

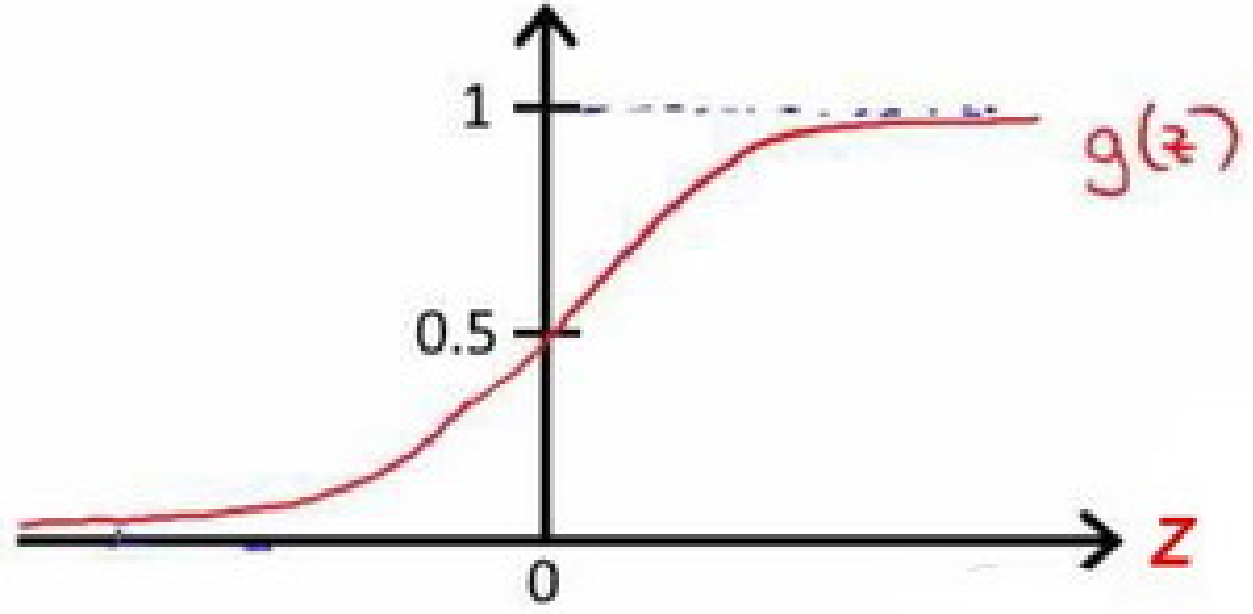


根据线性回归模型我们只能预测连续的值，然而对于分类问题，我们需要输出0或1，我们可以预测：
当$h_\theta(x) >= 0.5$时，预测 $y = 1$。
当$h_\theta(x) < 0.5$时，预测 $y = 0$ 。

对于上图所示的数据，这样的一个线性模型似乎能很好地完成分类任务。假使我们又观测到一个非常大尺寸的恶性肿瘤，将其作为实例加入到我们的训练集中来，这将使得我们获得一条新的直线。



这时，再使用0.5作为阀值来预测肿瘤是良性还是恶性便不合适了。可以看出，线性回归模型，因为其预测的值可以超越[0,1]的范围，并不适合解决这样的问题。
我们引入一个新的模型，逻辑回归，该模型的输出变量范围始终在0和1之间。
逻辑回归模型的假设是： $h_\theta(x) = g(\theta^T X)$
其中：
X 代表特征向量
g 代表逻辑函数（**logistic function**)是一个常用的逻辑函数为**S**形函数（**Sigmoid function**），公式为： $g(z) = \frac{1}{1+e^{-z}}$。
该函数的图像为：



合起来，我们得到逻辑回归模型的假设：

对模型的理解： $g(z) = \frac{1}{1+e^{-z}}$。

$h_\theta(x)$的作用是，对于给定的输入变量，根据选择的参数计算输出变量=1的可能性（**estimated probablity**）即$h_\theta(x) = P(y=1|x;\theta)$
例如，如果对于给定的x，通过已经确定的参数计算得出$h_\theta(x) = 0.7$，则表示有70%的几率y为正向类，相应地y为负向类的几率为1-0.7=0.3。

6.3 判定边界

在逻辑回归中，我们预测：

当$h_\theta(x) >= 0.5$时，预测 $y = 1$。

当$h_\theta(x) < 0.5$时，预测 $y = 0$ 。

根据上面绘制出的 **S** 形函数图像，我们知道当

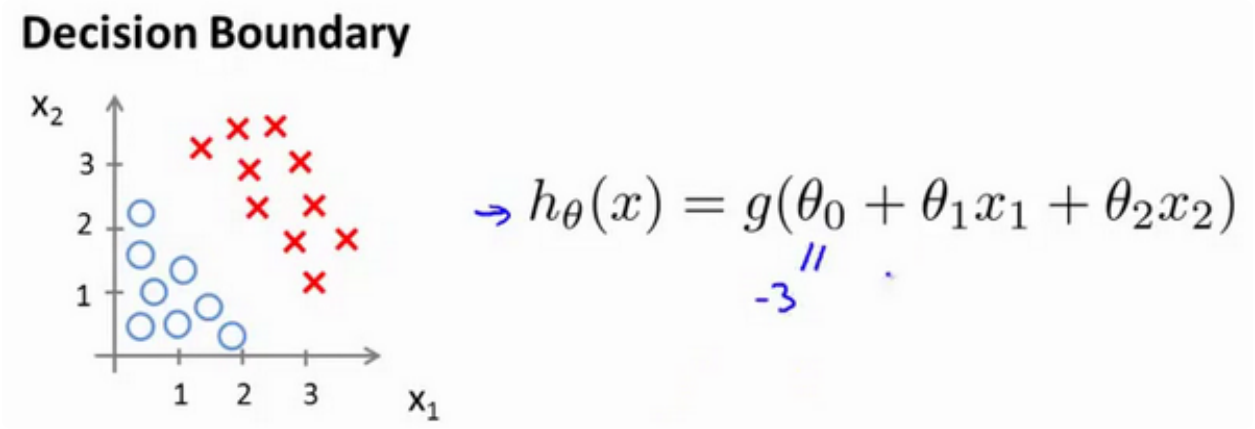
$z = 0$ 时 $g(z) = 0.5$

$z > 0$ 时 $g(z) > 0.5$

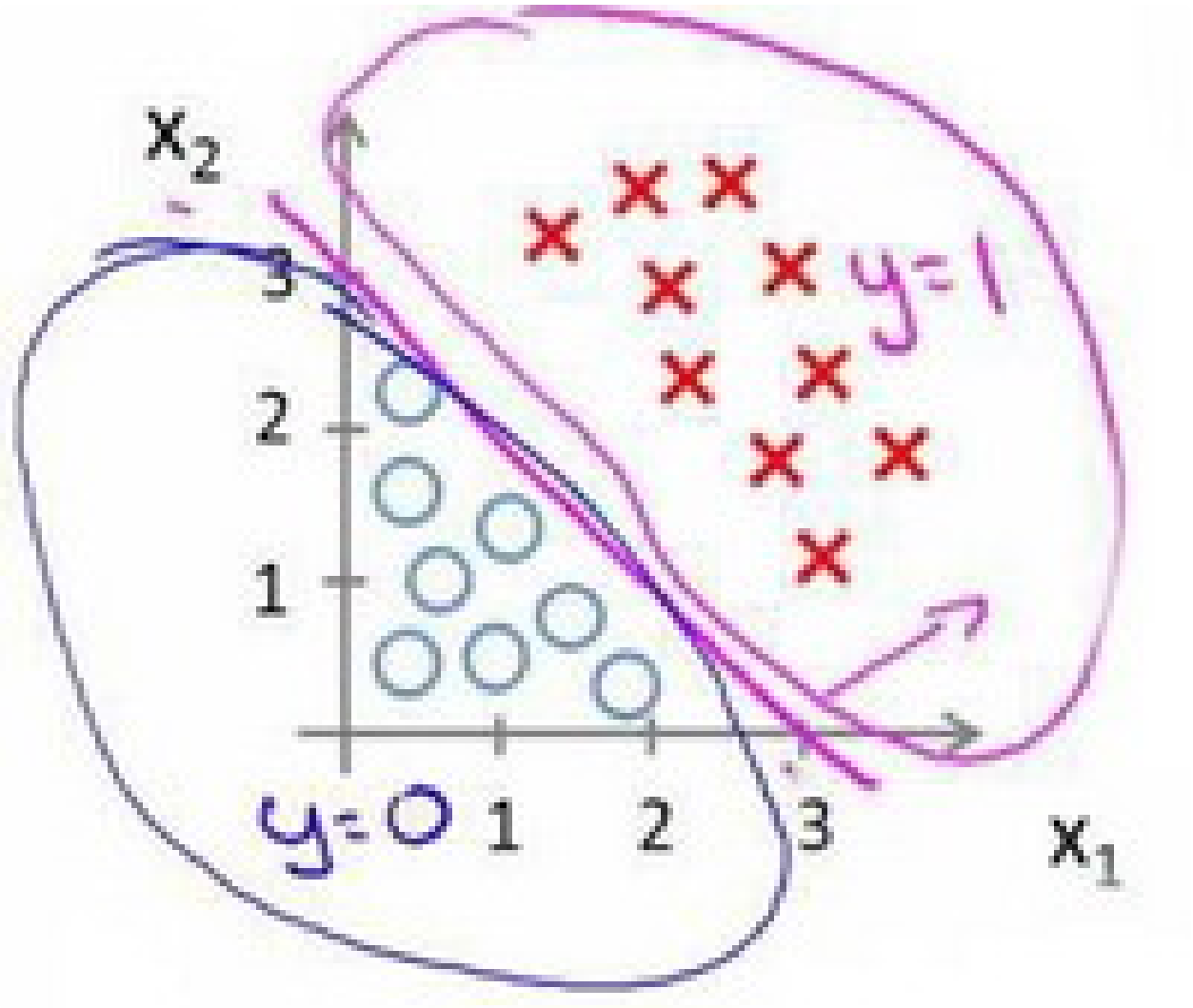
$z < 0$ 时 $g(z) < 0.5$

又 $z = \theta^T x$ ，即：
$\theta^T x >= 0$ 时，预测 $y = 1$
$\theta^T x < 0$ 时，预测 $y = 0$

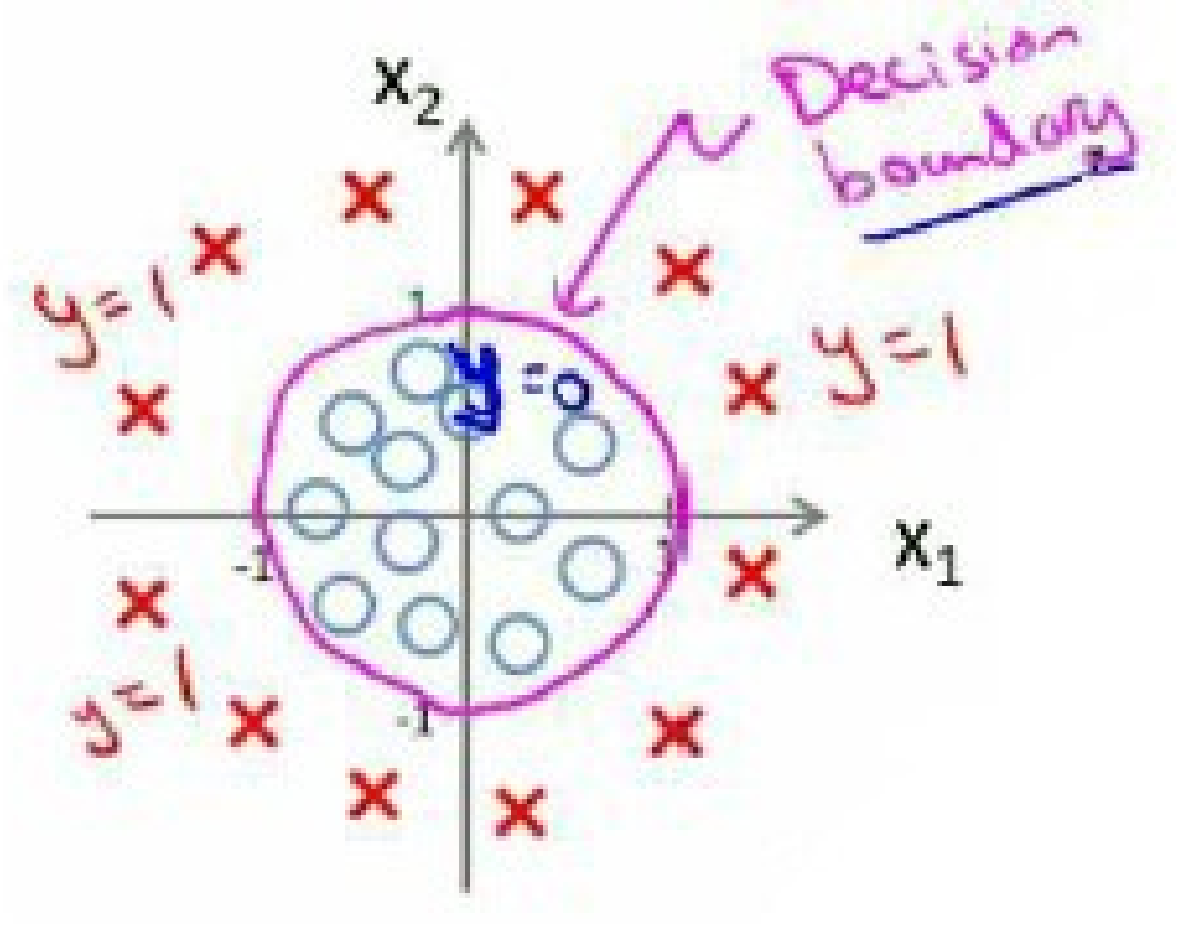
现在假设我们有一个模型：



并且参数θ 是向量[-3 1 1]。 则当$-3 + x_1 + x_2 \geq 0$，即$x_1 + x_2 \geq 3$时，模型将预测 $y = 1$。
我们可以绘制直线$x_1 + x_2 = 3$，这条线便是我们模型的分界线，将预测为1的区域和预测为 0的区域分隔开。



假使我们的数据呈现这样的分布情况，怎样的模型才能适合呢？



因为需要用曲线才能分隔 $y=0$ 的区域和 $y=1$ 的区域，我们需要二次方特征：
$h_\theta(x)=g\left(\theta_0+\theta_1x_1+\theta_2x_2+\theta_3x_1^2+\theta_4x_2^2\right)$是[-1 0 0 1 1]，则我们得到的判定边界恰好是圆点在原点且半径为1的圆形。

我们可以用非常复杂的模型来适应非常复杂形状的判定边界。

6.4 代价函数

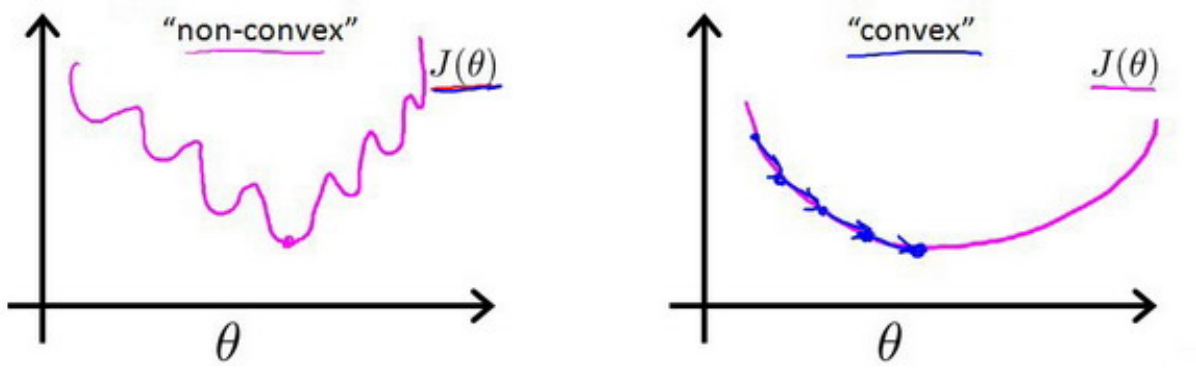
Training set: $\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$

m examples $\quad x\in\begin{bmatrix}x_0\\x_1\\\cdots\\x_n\end{bmatrix}\quad x_0=1,y\in\{0,1\}$

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

How to choose parameters θ ?

对于线性回归模型，我们定义的代价函数是所有模型误差的平方和。理论上来说，我们也可以对逻辑回归模型沿用这个定义，但是问题在于，当我们将$h_\theta(x)=\frac{1}{1+e^{-\theta^T x}}$带入到这样定义了的代价函数中时，我们得到的代价函数将是一个非凸函数（**non-convexfunction**）。



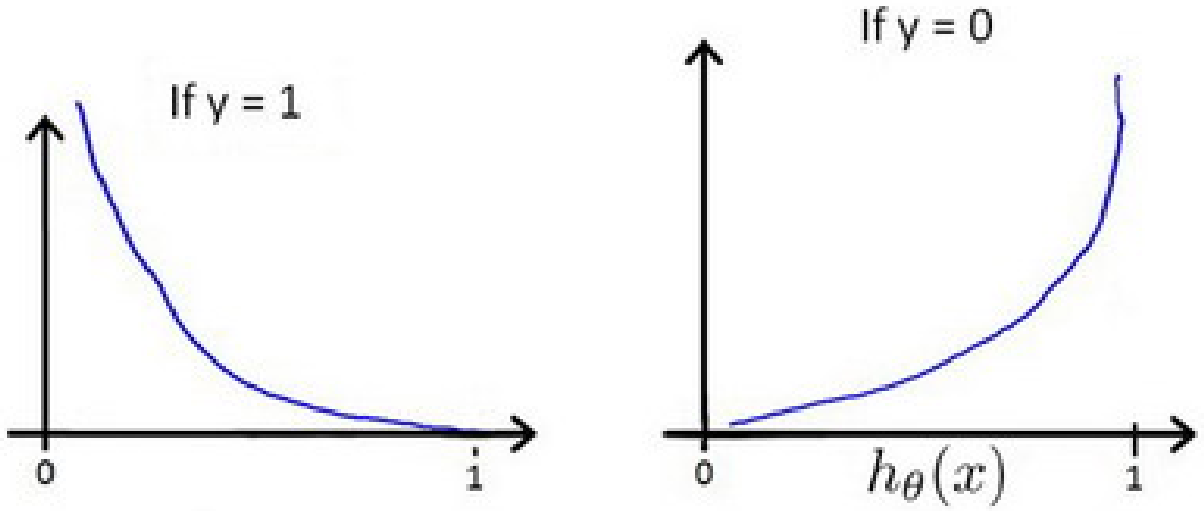
这意味着我们的代价函数有许多局部最小值，这将影响梯度下降算法寻找全局最小值。

线性回归的代价函数为：$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2$ 。

我们重新定义逻辑回归的代价函数为：$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}Cost\left(h_\theta\left(x^{(i)}\right),y^{(i)}\right)$，其中

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

$h_\theta(x)$与 $Cost\left(h_\theta(x),y\right)$之间的关系如下图所示：



这样构建的$Cost\left(h_\theta(x),y\right)$函数的特点是：

- 当实际的 $y=1$ 且$h_\theta(x)$也为 1 时误差为 0，当 $y=1$ 但$h_\theta(x)$不为1时误差随着$h_\theta(x)$变小而变大；
- 当实际的 $y=0$ 且$h_\theta(x)$也为 0 时代价为 0，当$y=0$ 但$h_\theta(x)$不为 0时误差随着 $h_\theta(x)$的变大而变大。

  将构建的 $Cost\left(h_\theta(x),y\right)$简化如下：

$Cost\left(h_\theta\left(x\right),y\right)=-y\times log\left(h_\theta\left(x\right)\right)-\left(1-y\right)\times log\left(1-h_\theta\left(x\right)\right)$
带入代价函数得到：
$J\left(\theta\right)=\frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)-\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)]$
即：$J\left(\theta\right)=-\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)+\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)]$
在得到这样一个代价函数以后，我们便可以用梯度下降算法来求得能使代价函数最小的参数了。
算法为：
Repeat {
$\theta_j:=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta)$
(**simultaneously update all**)
}

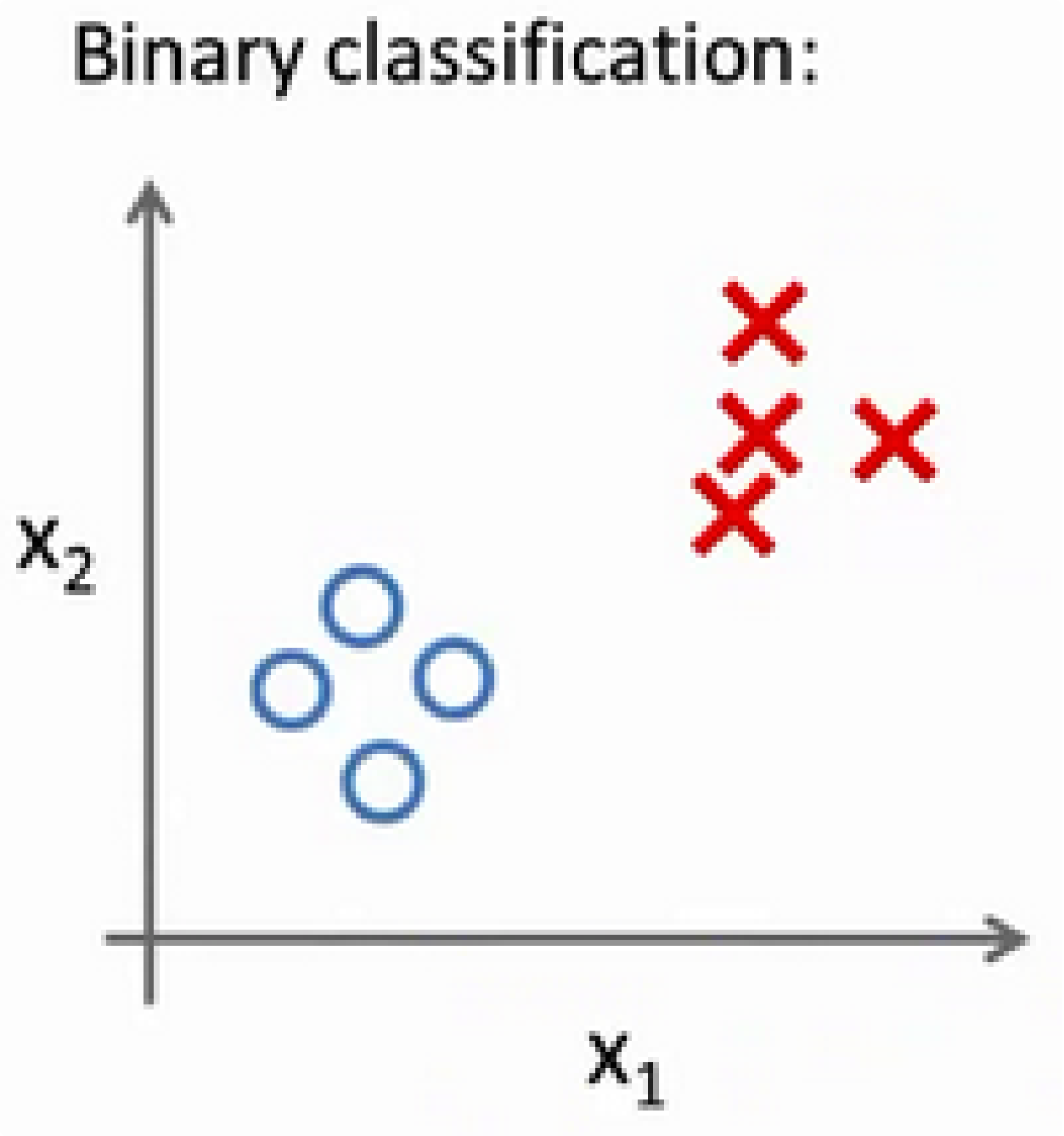
求导后得到：
Repeat {
$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)x_j^{(i)}$
(simultaneously update all)
}
注：

- 虽然得到的梯度下降算法表面上看上去与线性回归的梯度下降算法一样，但是这里的$h_\theta\left(x\right)=g\left(\theta^TX\right)$与线性回归中不同，所以实际上是不一样的。
- 另外，在运行梯度下降算法之前，进行特征缩放依旧是非常必要的。

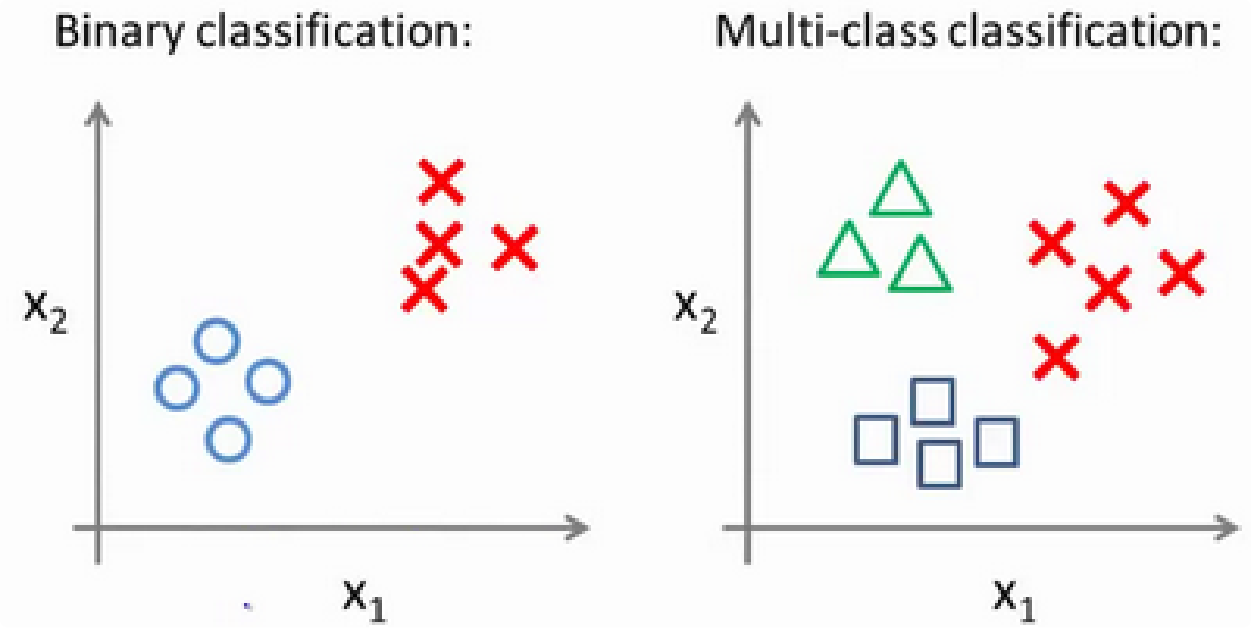
一些梯度下降算法之外的选择
除了梯度下降算法以外，还有一些常被用来令代价函数最小的算法，这些算法更加复杂和优越，而且通常不需要人工选择学习率，通常比梯度下降算法要更加快速。这些算法有：**共轭梯度**（**Conjugate Gradient**），**局部优化法**(**Broyden fletcher goldfarb shann,BFGS**)和**有限内存局部优化法**(**LBFGS**)，**fminunc**是 **matlab**和**octave** 中都带的一个最小值优化函数，使用时我们需要提供代价函数和每个参数的求导。

6.5 多类别分类：一对多

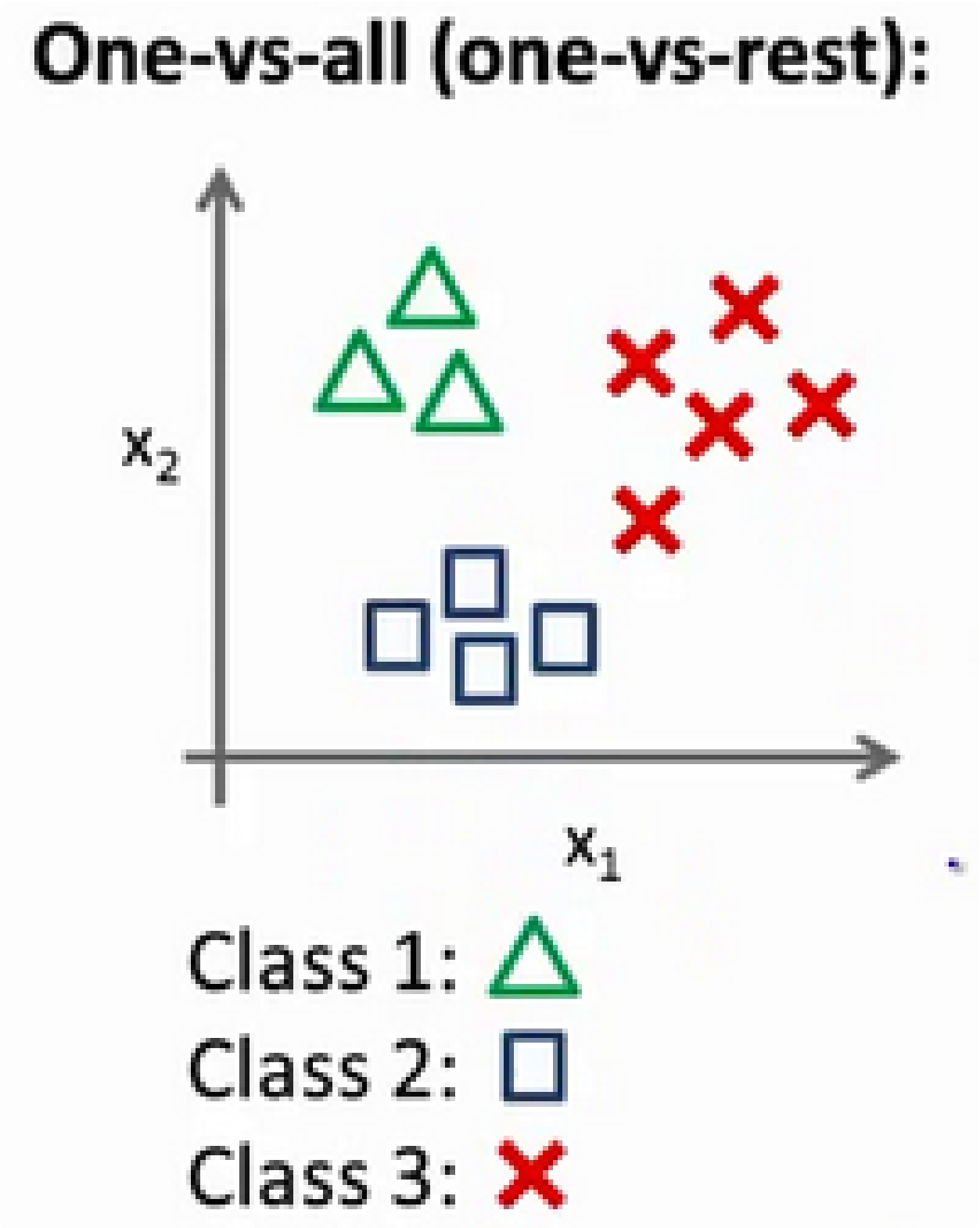
在本节视频中，我们将谈到如何使用逻辑回归 (**logistic regression**)来解决多类别分类问题，具体来说，我想通过一个叫做"一对多" (**one-vs-all**) 的分类算法。
二元分类问题，我们的数据看起来可能是像这样：



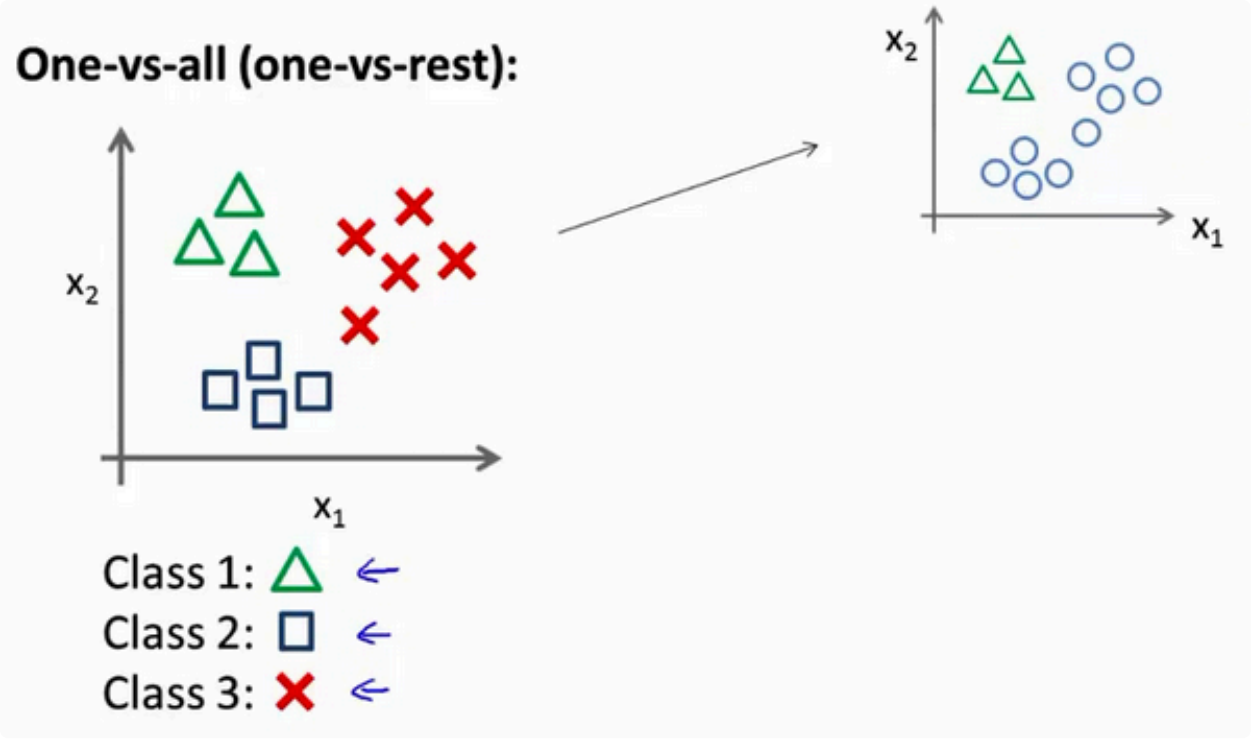
对于一个多类分类问题，我们的数据集或许看起来像这样：



我用3种不同的符号来代表3个类别，问题就是给出3个类型的数据集，我们如何得到一个学习算法来进行分类呢？



现在我们有一个训练集，好比上图表示的有3个类别，我们用三角形表示 $y=1$，方框表示$y=2$，叉叉表示 $y=3$。我们下面要做的就是使用一个训练集，将其分成3个二元分类问题。
我们先从用三角形代表的类别1开始，实际上我们可以创建一个，新的"伪"训练集，类型2和类型3定为负类，类型1设定为正类，我们创建一个新的训练集，如下图所示的那样，我们要拟合出一个合适的分类器。



这里的三角形是正样本，而圆形代表负样本。可以这样想，设置三角形的值为1，圆形的值为0，下面我们来训练一个标准的逻辑回归分类器，这样我们就得到一个正边界。

为了能实现这样的转变，我们将多个类中的一个类标记为正向类（$y=1$），然后将其他所有类都标记为负向类，这个模型记作$h_\theta^{(1)}(x)$。接着，类似地第我们选择另一个类标记为正向类（$y=2$），再将其它类都标记为负向类，将这个模型记作 $h_\theta^{(2)}(x)$,依此类推。
最后我们得到一系列的模型简记为： $h_\theta^{(i)}(x)=p(y=i|x;\theta)$其中：$i=(1,2,3....k)$

最后，在我们需要做预测时，我们将所有的分类机都运行一遍，然后对每一个输入变量，都选择最高可能性的输出变量。
现在要做的就是训练这个逻辑回归分类器：$h_\theta^{(i)}(x)$， 其中 i 对应每一个可能的 $y=i$，最后，为了做出预测，我们给出输入一个新的 x 值，用这个做预测。我们要做的就是在我们三个分类器里面输入 x，然后我们选择一个让 $h_\theta^{(i)}(x)$ 最大的i，即$\max\limits_i h_\theta^{(i)}(x)$。