Machine Learning - Week 3

赵燕

目录

1 Classification and Representation

1.1 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.

在分类问题中，需要预测的变量y是离散的值，引出要学习的逻辑回归算法（Logistic Regression），这是目前最流行使用的一种学习算法。

分类问题举例：

（1）判断一封电子邮件是否是垃圾邮件；

（2）判断一次金融交易是否是欺诈；

（3）判断肿瘤是 良性还是恶性；

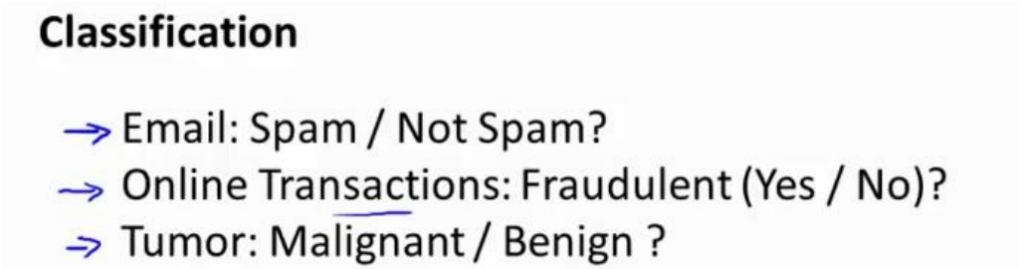

图 1: 分类问题举例

从二元的问题开始讨论：

将因变量（dependent variable）可能属于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。

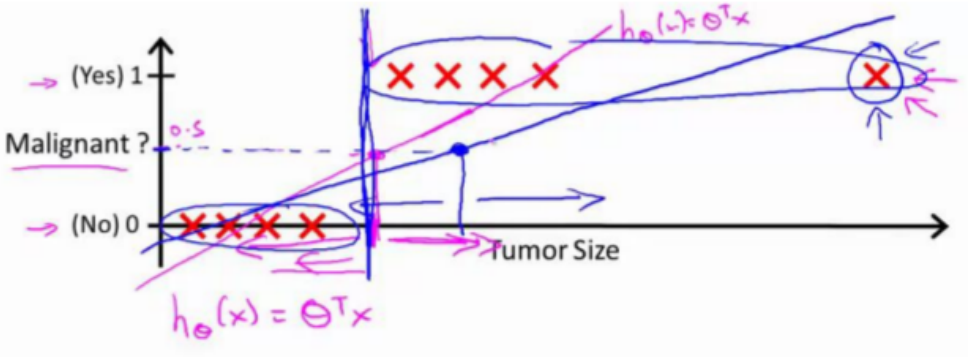

图 2: 图示

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.

如果我们要用线性回归算法来解决一个分类问题,对于分类,y 取值为 0 或者 1,但如果你使用的是线性回归,那么假设函数的输出值可能远大于 1,或者远小于 0,即使所有训练样本的标签 y 都等于 0 或 1。尽管我们知道标签应该取值 0 或者 1,但是如果算法得到的值远大于 1 或者远小于 0 的话,就会感觉很奇怪。所以我们在接下来的要研究的算法就叫做逻辑回归算法,这个算法的性质是:它的输出值永远在 0 到 1 之间。

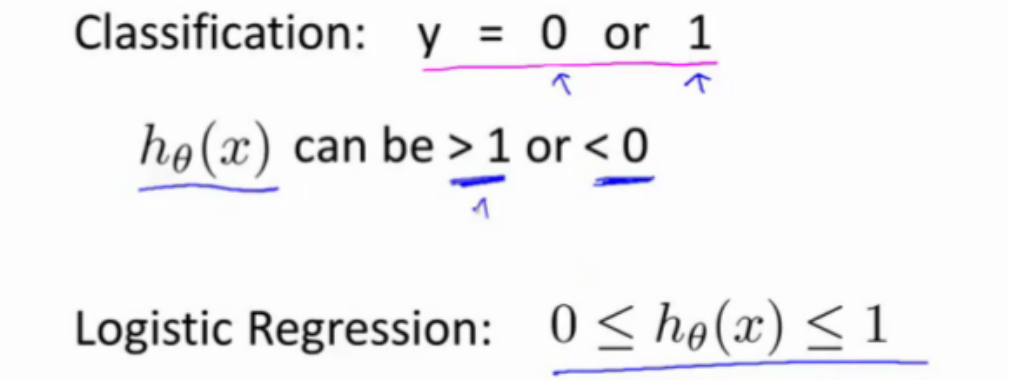

图 3: 逻辑回归算法

逻辑回归算法是分类算法，我们将它作为分类算法使用，有时候可能因为这个算法的名字中出现了"回归"使你感到困惑，但逻辑回归算法实际上是一种分类算法，它适用于标签y取值离散的情况下，如：1 ，0，0， 1 。

1.2 Hpothesis Representation

假设函数表达式:

我们希望分类器的输出在0和1之间，因此，要想出一个满足某个性质的假设函数，这个性质是它的预测值要在0和1之间。

回顾肿瘤分类问题：可以用线性回归的方法求出一条适合数据的一条直线：

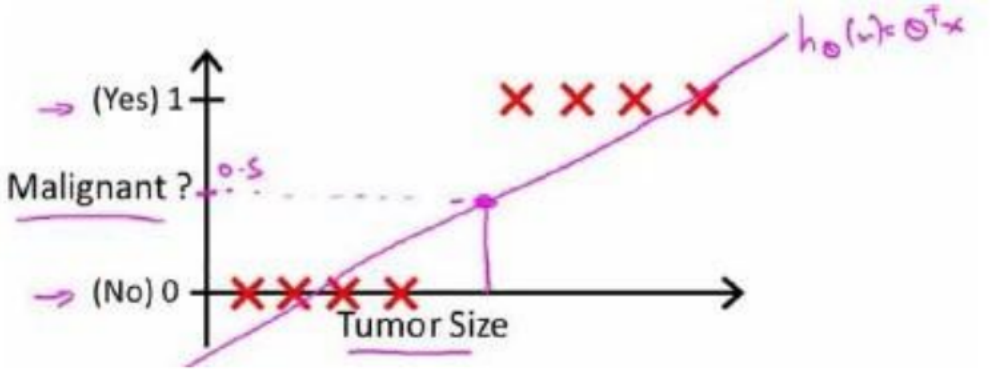

图 4: 肿瘤分类线性回归

根据线性回归我们只能预测连续的值，然而对于分类问题，我们要输出0或1，我们可以预测：（1）当h_θ大于等于 0.5 时,预测 y=1;（2）当 h_θ 小于 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。

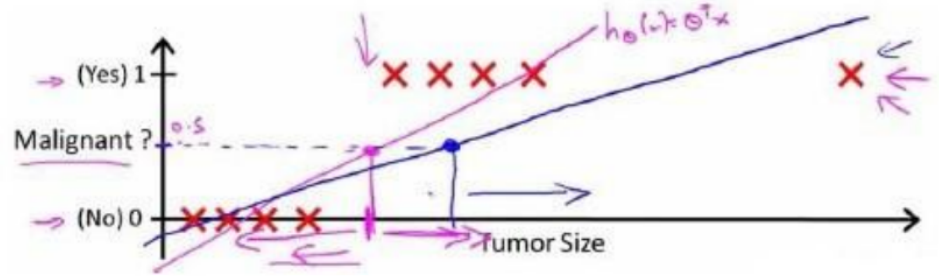

图 5: 肿瘤问题线性回归超范围

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。

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 θ (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 θ (x) to satisfy 0$\leqslant$h θ (x)$\leqslant$1. This is accomplished by plugging θ Tx into the Logistic Function.

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:

$$h_\theta(x) = g(\theta^T x) \tag{1}$$

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:

$$g(z) = \frac{1}{1+e^{-z}} \tag{2}$$

该函数的图像为：

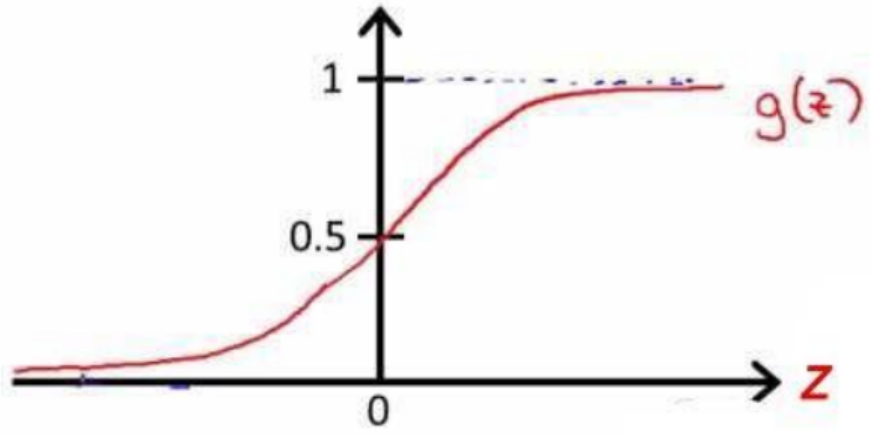

图 6: 逻辑函数（S函数）图像

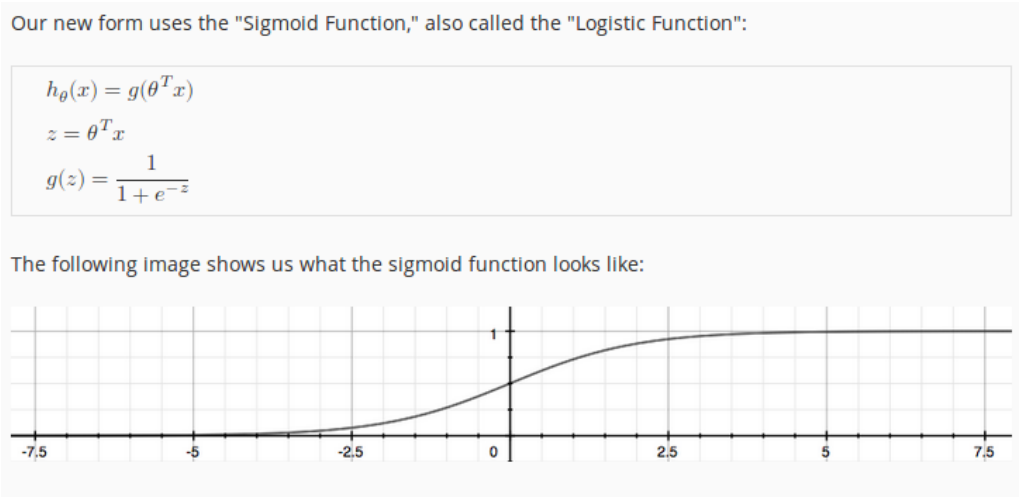

图 7: 逻辑函数表达式及图像

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.

逻辑回归模型的假设：

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \tag{3}$$

$h_\theta(x)$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

$$h_\theta(x) = P(y=1|x;\theta) \tag{4}$$

$h_\theta(x)$will give us the probability that our output is 1.

$$h_\theta(x) = P(y=1|x;\theta) = 1 - P(y=0|x;\theta) \tag{5}$$

$$P(y=0|x;\theta) + P(y=1|x;\theta) = 1 \tag{6}$$

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。

1.3 Decision Boundary

决策边界（Decision Boundary）可以更好的帮助我们理解假设函数在计算什么

首先回顾一下逻辑回归中假设函数的表达式：

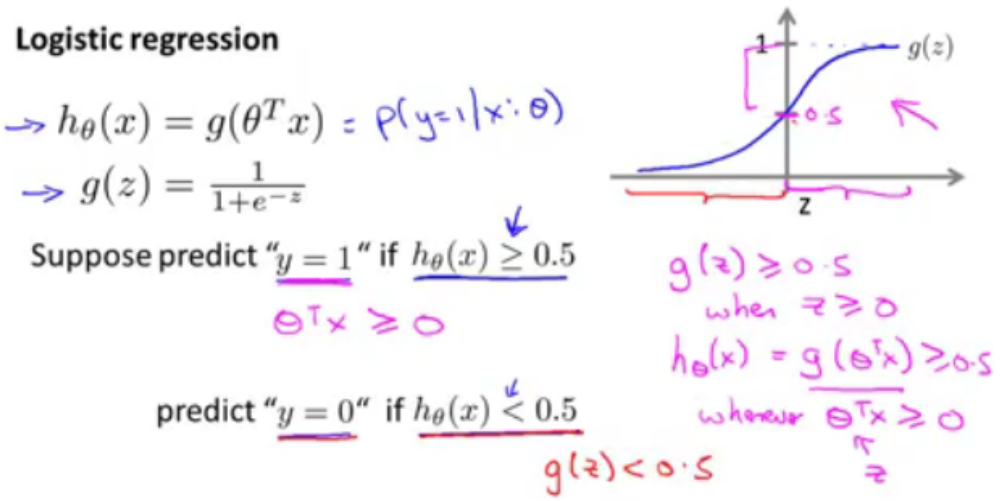

图 8: 逻辑函数回顾

根据上面绘制的S形函数的图像，我们可以知道：

当z=0时，g(z)=0.5;

当z>0时，g(z)>0.5;

当z<0时，g(z)<0.5;

又因为

$$z = \theta^T x \tag{7}$$

所以：

$\theta^T x \geq 0$时，预测y=1;

$\theta^T x < 0$时，预测y=0;

假设有一个模型：

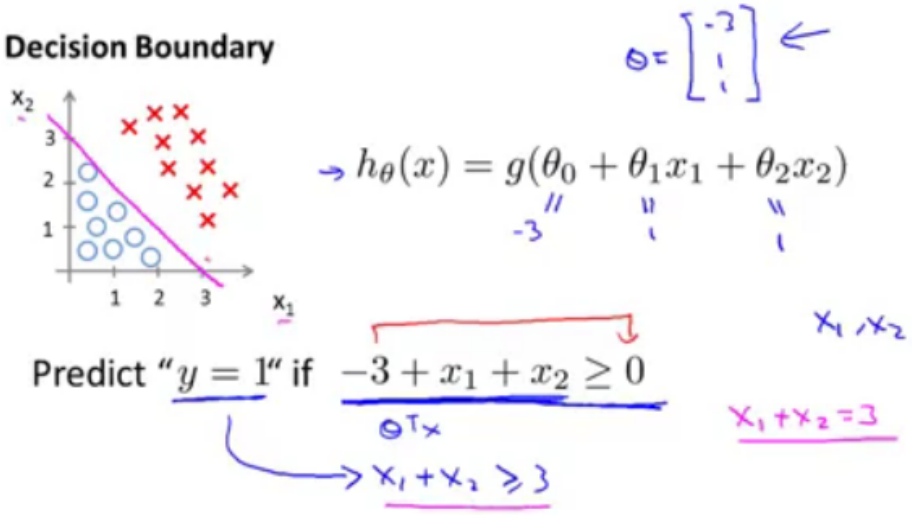

图 9: 线性决策边界

绘制直线$x_1 + x_2 = 3$,这条直线便是我们的模型的分界线，将预测y=1的区域和预测y=0的区分隔开，如图所示：

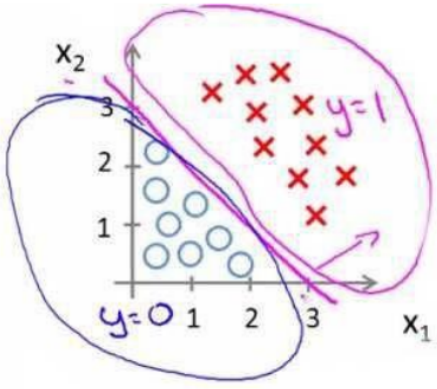

图 10: 线性决策边界图

另一种情况：

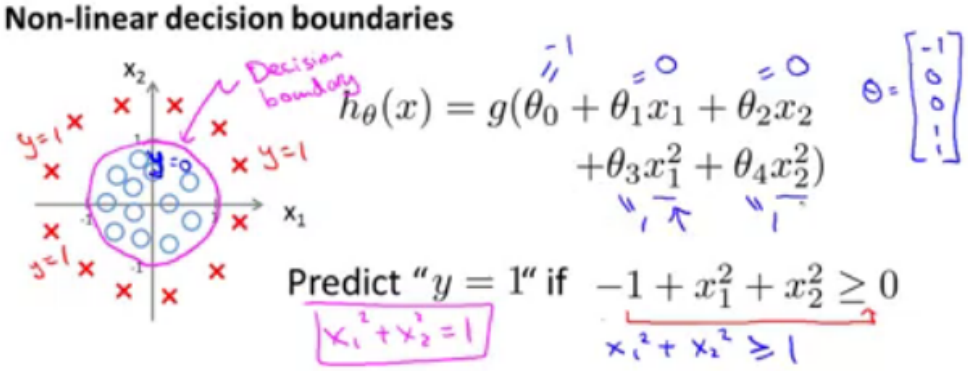

图 11: 非线性决策边界

需要二次方特征，得到的决策边界（判定边界）恰好是在原点且半径为1的圆形。

我们可以用非常复杂的模型来适应非常复杂的判定边界。 例如:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + ...) \tag{8}$$

我们可能会得到比椭圆等更复杂的判定边界:

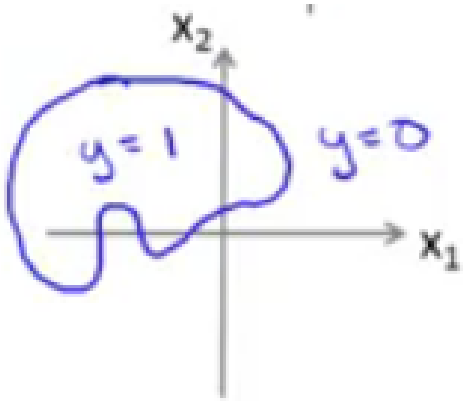

图 12: 更复杂的情况

2 Logistic Regression Model

2.1 Cost Function

介绍如何拟合逻辑回归模型的参数θ，具体来说，我们要定义来拟合参数的优化目标或者叫代价函数，这便是监督学习问题中的逻辑回归模型的拟合问题。

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

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

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

How to choose parameters θ ?

图 13: 如何选择参数θ

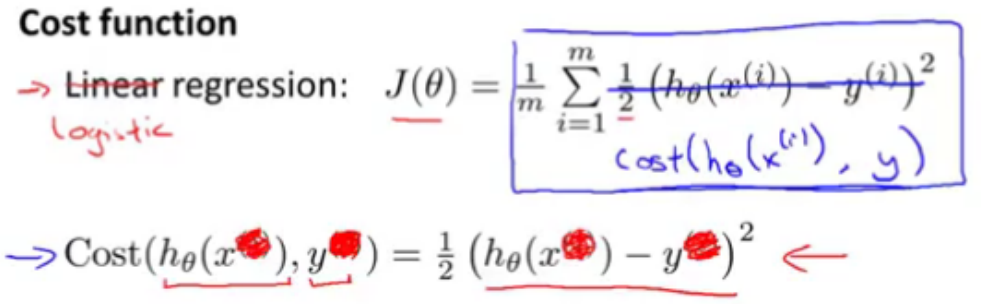

图 14: 线性代价函数讨论

Cost Function:

$$J(\theta) = \frac{1}{m}\sum_{i+1}^{m}\frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{9}$$

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$h_\theta = \frac{1}{1+e^{-\theta^T x}}$带入到这样定义的代价函数中去，我们得到的代价函数将是一个非凸函数（non-convex function）。

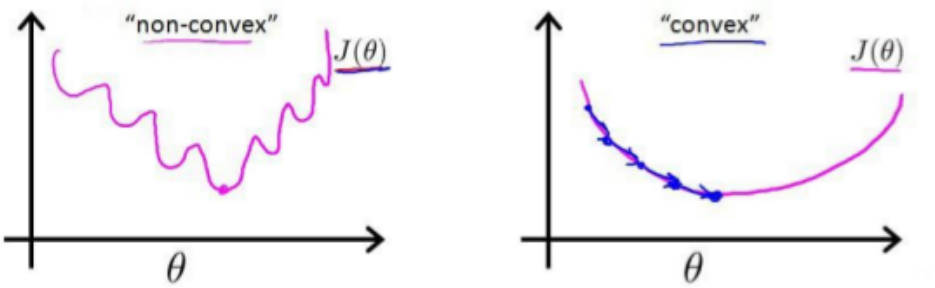

图 15: 非凸函数与凸函数

这就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。

线性回归代价函数:

$$J(\theta) = \frac{1}{m}\sum_{i+1}^{m}\frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{10}$$

重新定义逻辑回归的代价函数:（这种方式可以保证$J(\theta)$对于逻辑回归是凸函数）

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}Cost(h_\theta(x^{(i)}), y^{(i)}) \tag{11}$$

其中:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & if y = 1 \\ -log(1 - h_\theta(x)) & if y = 0 \end{cases} \tag{12}$$

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

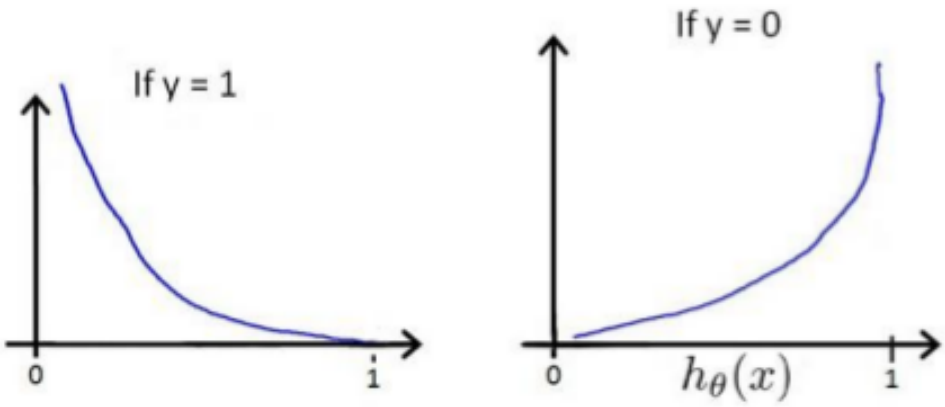

图 16: $h_\theta(x)$与$Cost(h_\theta(x), y)$的关系

这样构建的函数的特点是：

（1）当实际的y=1且h_θ也为1时的误差为0；

（2）当y=1但不为1时误差随着的变小而变大；

（3）当实际的y=0且h_θ也为0时代价为0；

（4）当y=0但h_θ不为0时误差随着h_θ的变大而变大；

2.2 Simplified Cost Function and Gradient Descent

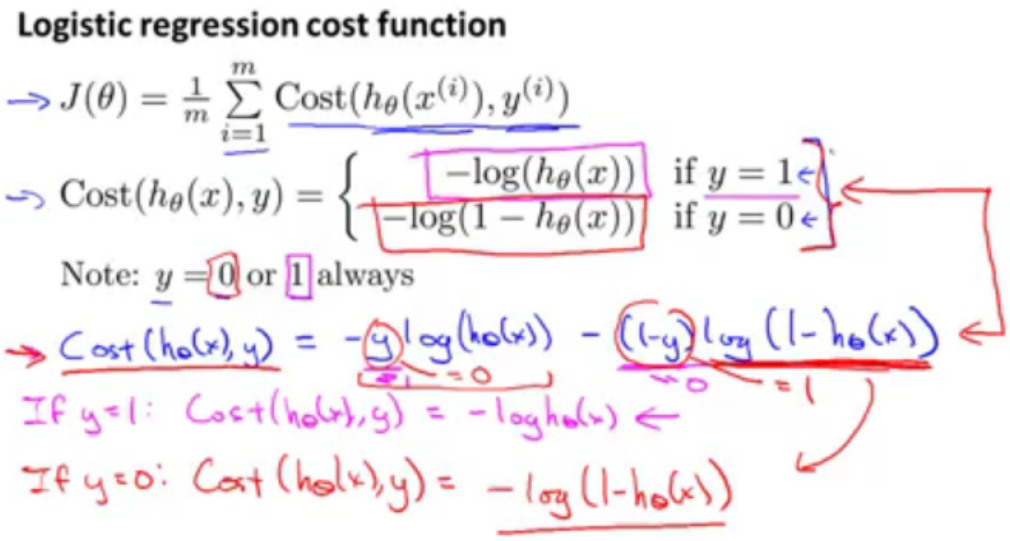

图 17: $h_\theta(x)$与$Cost(h_\theta(x), y)$的关系

将构建的构造函数简化如下:

$$Cost(h_\theta(x)) = -ylog(h_\theta(x)) - (1-y)log(1 - log(h_\theta(x)) \tag{13}$$

带入到代价函数中得到:

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} log h_\theta(x^{(i)}) + (1-y^{(i)})log(1 - h_\theta(x^{x(i)}))] \tag{14}$$

A vectorized implementation is:

$$h = g(X\theta) \tag{15}$$

$$J(\theta) = -\frac{1}{m} \cdot (-y^T log(h) - (1-y)^T log(1-h)) \tag{16}$$

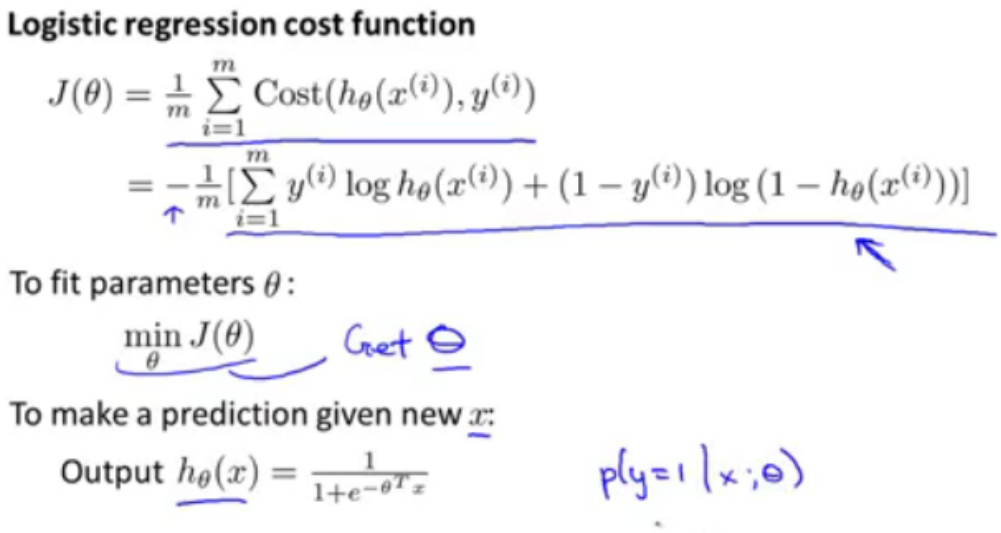

图 18: 梯度下降算法的导出

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θ，算法为：

repeat{

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{17}$$

$$(simultaneouslyupdateall) \tag{18}$$

}

对θ_j求导后得出：

repeat{

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{19}$$

$$(simultaneouslyupdateall) \tag{20}$$

}

A vectorized implementation is:

$$\theta_j := \theta_j - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y}) \tag{21}$$

在这个视频中，我们定义了单训练样本的代价函数，凸性分析的内容是超出我们这门课的范围的，但是可以证明我们所选的代价值函数会给我们一个凸优化问题。代价函数$J(\theta)$会是一个凸函数，并没有局部最优值。

注意：

线性回归假设函数：

$$h_\theta(x) = \theta^T x \tag{22}$$

逻辑回归假设函数：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{23}$$

虽然得到的梯度下降算法表面看上去与线性回归的梯度下降算法一样，但是这里的逻辑回归假设函数与线性回归$h_\theta = \theta^T x$的不同，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有必要的，特征缩放也适用与逻辑回归。

逻辑回归是一种非常强大的，甚至是世界上使用最广泛的一种分类算法。

2.3 Advanced Optimization

利用高级优化算法，使通过梯度下降进行逻辑回归的速度大大提高，而且这也将使算法更加适合解决大型的机器学习问题，比如我们有数目庞大的特征量。

换个角度来看什么是梯度下降，我们有一个代价函数$J(\theta)$，我们想要使其最小化，我们需要做的是编写代码，当输入参数θ时，它们会计算出两种东西：$J(\theta)$以及J等于0、1直到n时的偏导数项。

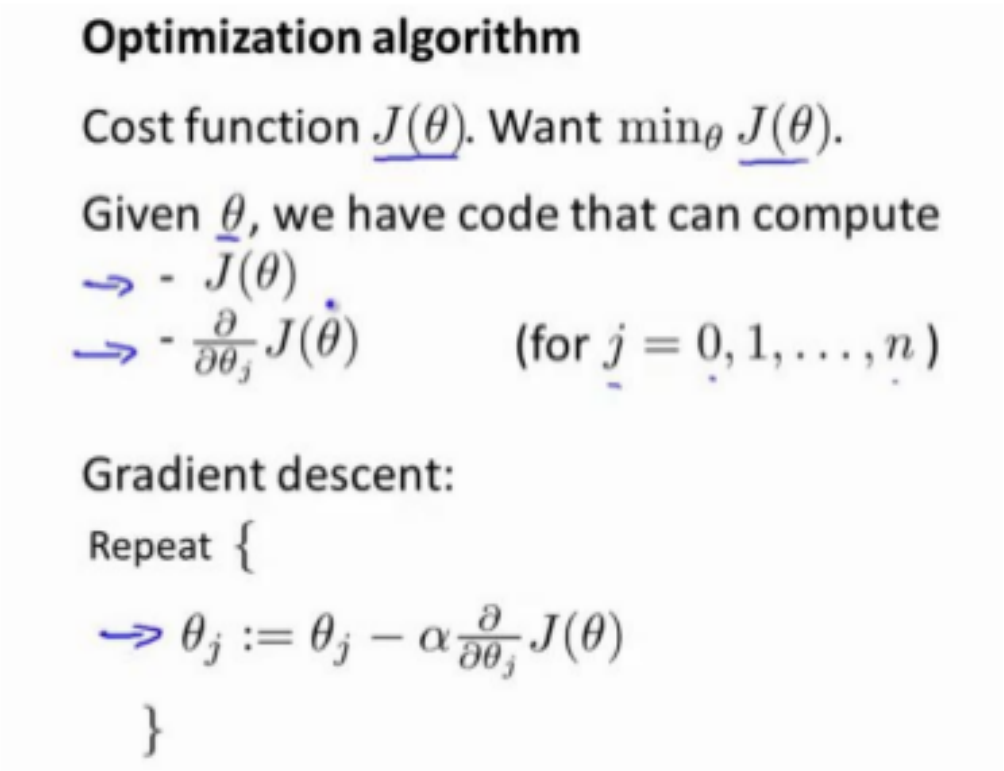

图 19: 梯度下降执行过程

假设我们已经完成了可以实现这两件事的代码，那么梯度下降所做的就是反复执行这些更新。

另一种考虑梯度下降的思路是：我们需要写出代码来计算$J(\theta)$和这些偏导数，然后把这些插入到梯度下降中，然后它就可以为我们最小化这个函数。

对于梯度下降，不需要编写代码去计算代价函$J(\theta)$，只需要计算导数项，但是如果你希望代码还要能够监控这些$J(\theta)$的收敛性，那就需要我们自己去编写代码来计算代价函数$J(\theta)$和偏导数项$\frac{\partial}{\partial\theta_j}J(\theta)$，所以在写完能够计算这两者的代码之后，我们就可以使用梯度下降。

共轭梯度法，BFGS（变尺度法）和L-BFGS（限制变尺度法）就是其中一些更高级的优化算法，她们需要有一种方法来最小化代价函数$J(\theta)$。

三种方法的优缺点：

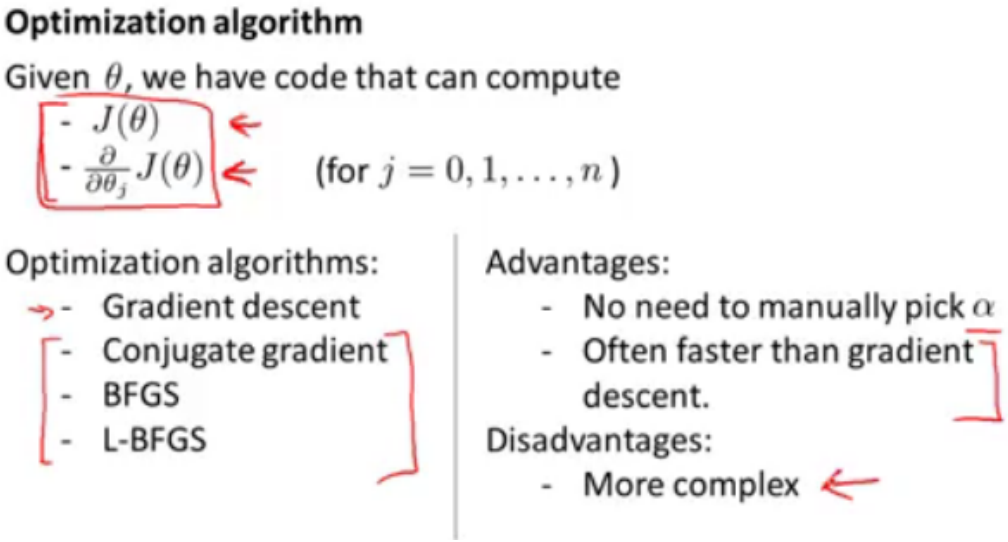

图 20: 高级优化算法优缺点

实际上,我不会建议你们编写自己的代码来计算数据的平方根,或者计算逆矩阵,因为对于这些算法,我还是会建议你直接使用一个软件库,比如说,要求一个平方根,我们所能做的就是调用一些别人已经写好用来计算数字平方根的函数。幸运的是现在我们有 Octave 和与它密切相关的 MATLAB 语言可以使用。

Octave 有一个非常理想的库用于实现这些先进的优化算法,所以,如果你直接调用它 自带的库,你就能得到不错的结果。我必须指出这些算法实现得好或不好是有区别的,因此, 如果你正在你的机器学习程序中使用一种不同的语言,比如如果你正在使用 C、C + + 、Java 等等,你可能会想尝试一些不同的库,以确保你找到一个能很好实现这些算法的库。因为在 L-BFGS 或者等高线梯度的实现上,表现得好与不太好是有差别的,因此现在让我们来说明:

如何使用这些算法:

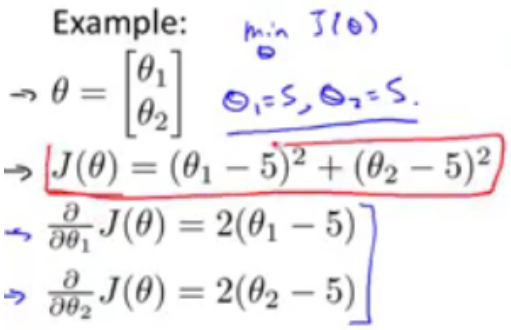

图 21: 举例

运行一个像这样的Octave函数:

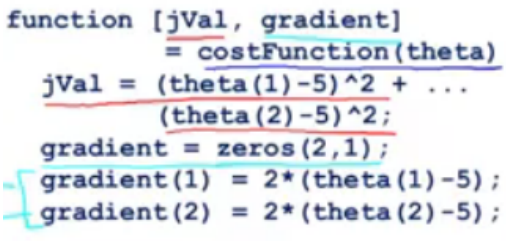

图 22: Octave函数

这样就计算出这个代价函数,函数返回的第二个值是梯度值,梯度值应该是一个 2×1的向量,梯度向量的两个元素对应这里的两个偏导数项,运行这个 costFunction 函数后,你 就可以调用高级的优化函数,这个函数叫 fminunc,它表示 Octave 里无约束最小化函数。调用它的方式如下:

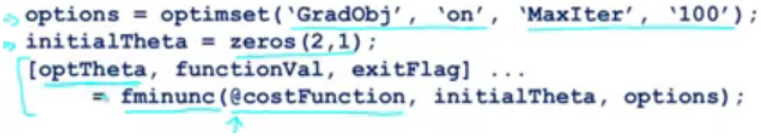

图 23: 调用高级优化函数

你要设置几个options,这个 options 变量作为一个数据结构可以存储你想要的 options,所以 GradObj 和 On,这里设置梯度目标参数为打开(on),这意味着你现在确实要给这个算法提供一个梯度,然后设置最大迭代次数,比方说 100,我们给出一个θ的猜测初始值,它是一个 2×1 的向量,那么这个命令就调用 fminunc,这个@符号表示指向我们刚刚定义的costFunction 函数的指针。如果你调用它,它就会使用众多高级优化算法中的一个,当然你也可以把它当成梯度下降,只不过它能自动选择学习速率 α,你不需要自己来做。然后它会尝试使用这些高级的优化算法,就像加强版的梯度下降法,为你找到最佳的θ值。

总结：需要学到的主要内容是：写一个函数，它能返回代价函数值，梯度值，因此要把这个应用到逻辑回归中，甚至线性回归中，你可以把这些优化算法用于线性回归，你需要做的就是输入合适的代码来计算这里的这些东西。

3 Multiclass Classification

3.1 Multiclass Classification:One-vs-all

利用逻辑回归解决多类别分类问题，具体来说，就是想通过一个叫做“一对多（one-vs-all）”的分类方法。

例1：需要一个算法能够自动的将邮件归类到不同的文件夹里，或者说是自动地加上标签。

例2：关于药物诊断，如果一个人因为鼻塞来到你的诊所，他可能并没有生病,用 y=1 这个类别来代表;或者患了感冒,用 y=2 来代表;或者得了流感用 y=3 来代表。

例3：如果你正在做有关天气的机器学习分类问题,那么你可能想要区分哪些天:是晴天、多云、雨天、或者下雪天,对上述所有的例子,y 可以取一个很小的数值,一个相 对”谨慎”的数值,比如 1 到 3、1 到 4 或者其它数值,以上说的都是多类分类问题,顺便一提的是,对于下标是 0 1 2 3,还是 1 2 3 4 都不重要,我更喜欢将分类从 1 开始标而不是0,其实怎样标注都不会影响最后的结果。

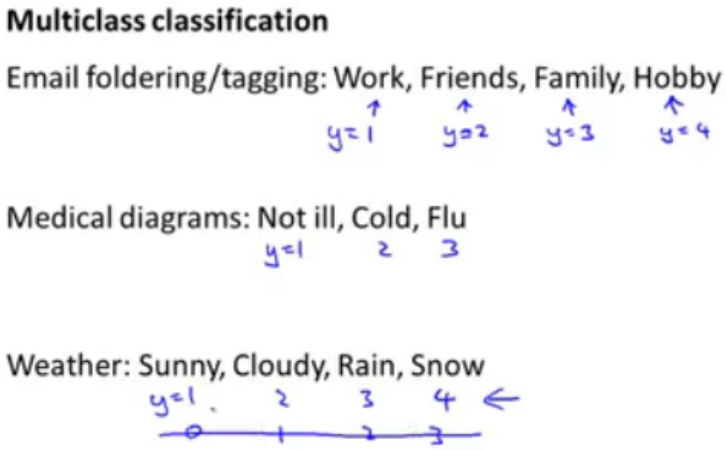

图 24: 分类算法举例

然而对于之前的一个二元分类问题，我们的数据看起来可能是像这样：

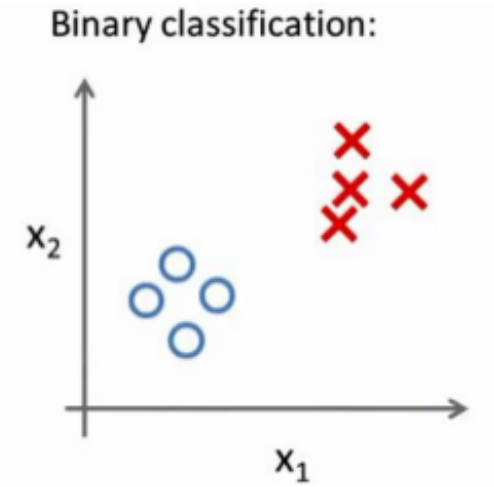

图 25: 二元分类问题

对于一个多类的分类问题，我们的数据或许看起来像这样：

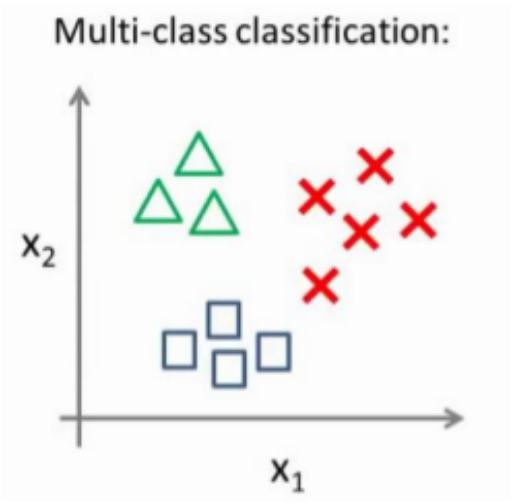

图 26: 多类分类问题

用三种不同的符号代表三个类别，问题就是给出三个类型的数据集，如何得到一个学习算法来进行分类呢?

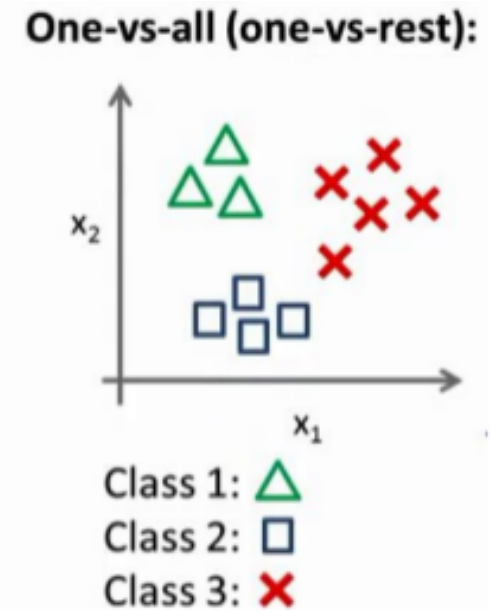

图 27: One-vs-all

如何进行二元分类：使用逻辑回归，可以用一条直线将数据集一分为二，分为正类和负类。用一对多的分类思想，我们可以将其用在多类分类问题上。

下面开始介绍如何进行一对多的分类工作，有时这个方法也叫作“一对余”方法:

Now we will approach the classification of data when we have more than two categories. Instead of y = 0,1 we will expand our definition so that y = 0,1...n.

Since y = 0,1...n, we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$$y \in \{0, 1, 2, ..., n\} \tag{24}$$
$$h_\theta^{(0)}(x) = P(y = 0|x; \theta) \tag{25}$$
$$h_\theta^{(1)}(x) = P(y = 1|x; \theta) \tag{26}$$
$$..... \tag{27}$$
$$h_\theta^{(n)}(x) = P(y = n|x; \theta) \tag{28}$$
$$prediction = \max_i(h_\theta^{(i)}(x)) \tag{29}$$

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.

The following image shows how one could classify 3 classes:

有三个类别，使用一个训练集，将其分为三个二元分类问题:

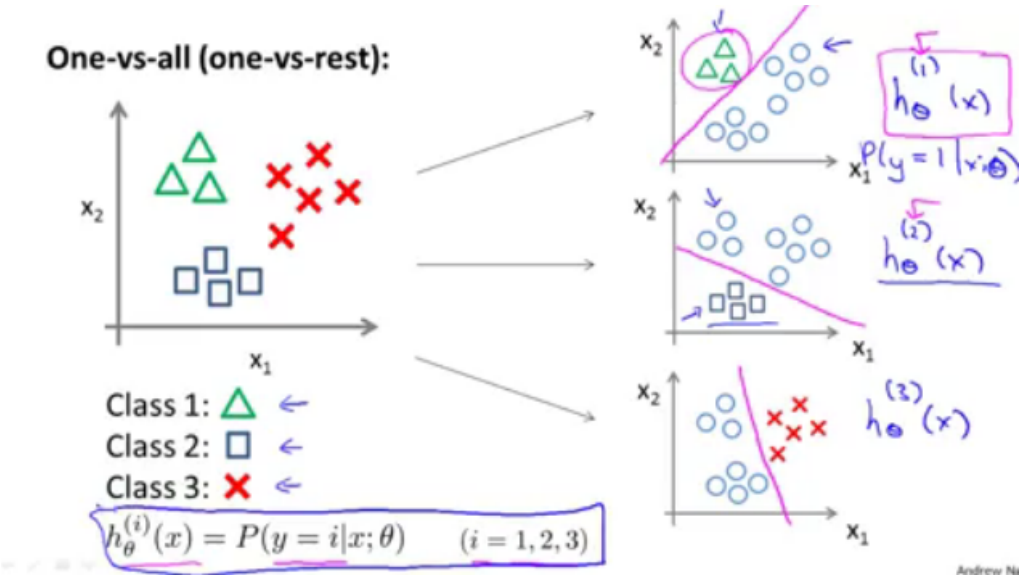

图 28: One-vs-all(one-vs-rest)

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。

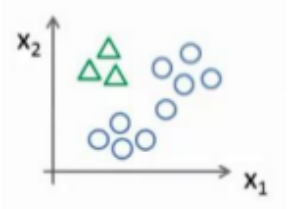

图 29: One-vs-all

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。

最后得到一系列的模型，记作：$h_\theta^{(i)} = P(y = i|x;\theta)$，其中i=(1，2，3，...，k)

接下来要训练这个逻辑回归分类器：$h_\theta^{(i)}(x)$，其中i对应的每一个可能的y=i，最后为了做出预测，我们给出一个输入新的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$\max_i(h_\theta^{(i)}(x))$。

了解了基本挑选分类器的方法，选择出哪一个分类器是可信度最高效果最好的，那么就可以认为得到了一个正确的分类，无论i的值是多少，都有最高的概率值，我们要预测的y就是那个值。这就是多类别分类问题，以及一对多的方法，通过这个小方法，就可以将逻辑回归分类器用在多类分类问题上。

4 Solving the Problem of Overfitting

4.1 The Problem of Overfitting

4.2 Cost Function

4.3 Regularized Linear Regression

4.4 Regularized Logistic Regression