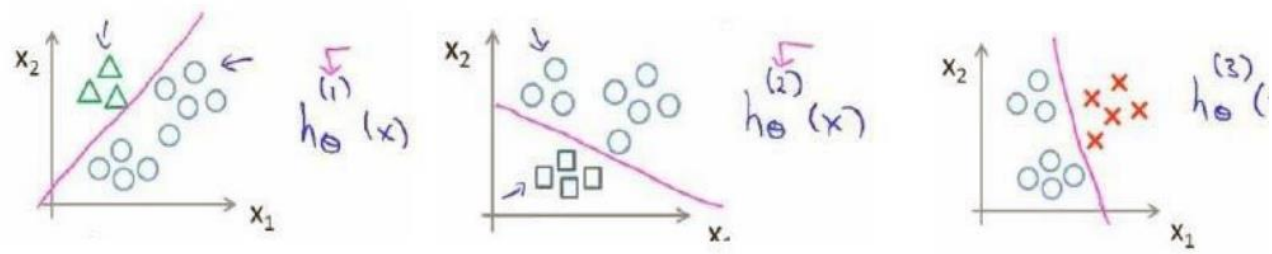
17.4.19（多分类问题与正则化）

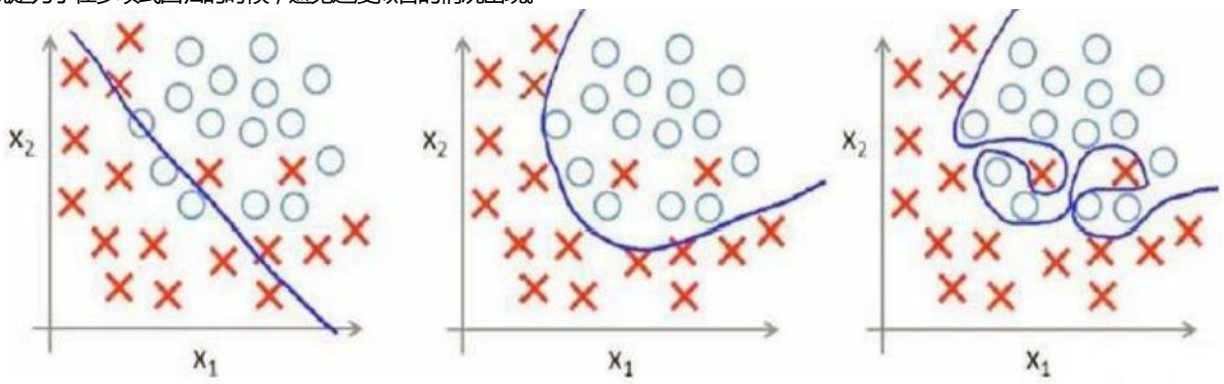
对与多类之间的分类，



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。

总之，我们已经把要做的做完了，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 $\max h_\theta^{(i)}(x)$。

正则化就是为了在多项式回归的时候，避免过度拟合的情况出现。



就以多项式理解，x 的次数越高，拟合的越好，但相应的预测的能力就可能变差。

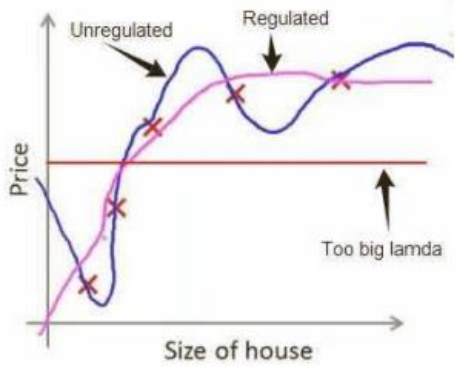
问题是，如果我们发现了过拟合问题，应该如何处理？

1. 丢弃一些不能帮助我们正确预测的特征。可以是手工选择保留哪些特征，或者使用一些模型选择的算法来帮忙（例如 PCA）

2. 正则化。 保留所有的特征，但是减少参数的大小（magnitude）。

我们可以从之前的事例中看出，正是那些高次项导致了过拟合的产生，所以如果我们能让这些高次项的系数接近于 0 的话，我们就能很好的拟合了。

所以我们要做的就是在一定程度上减小这些参数 θ 的值，这就是正则化的基本方法。



如果选择的正则化参数 λ 过大，则会把所有的参数都最小化了，导致模型变成 $h_\theta(x)=\theta_0$

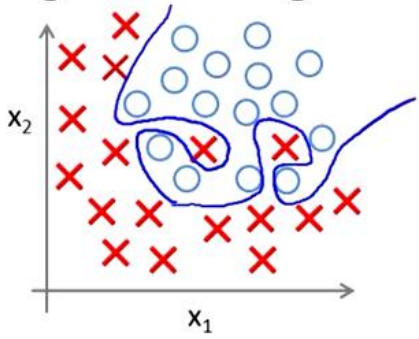
也就是上图中红色直线所示的情况，造成欠拟合。

因为如果我们令λ的值很大的话，为了使 Cost Function 尽可能的小，所有的 θ 的值（不包括 θ 0 ）都会在一定程度上减小。

但若λ的值太大了，那么 θ（不包括 θ 0 ）都会趋近于 0，这样我们所得到的只能是一条平行于 x 轴的直线。

所以对于正则化，我们要取一个合理的λ的值，这样才能更好的应用正则化。

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:

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log (1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2 \quad | \quad \theta_1, \theta_2, \dots \theta_n$$

Andrew N

正则化（regularized）cost function

Gradient descent

Repeat {

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

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \quad (j = 1, 2, 3, \ldots, n)$$

}

$\theta_1 \ldots \theta_n$

$\frac{\partial}{\partial \theta_j} J(\theta)$

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

regularized gradient descent（正则化梯度下降）

Advanced optimization

fminunc (@costFunction, θ, options)

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ theta(1), theta(2), theta(n+1)

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)];
```

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

```
    gradient(1) = [code to compute ∂/∂θ0 J(θ)];
```

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

```
    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
```

$$\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)} \right) + \frac{\lambda}{m} \theta_1$$

```
    gradient(3) = [code to compute ∂/∂θ2 J(θ)];
    ⋮
```

$$\left(\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)} \right) + \frac{\lambda}{m} \theta_2$$

```
    gradient(n+1) = [code to compute ∂/∂θn J(θ)];
```

J(θ)

regularized advanced optimization（正则化高级算法）

注：看上去同线性回归一样，但是知道 $h_\theta(x)=g(\theta^T X)$，所以与线性回归不同。

Octave 中，我们依旧可以用 fminuc 函数来求解代价函数最小化的参数，值得注意的是参数 θ_0 的更新规则与其他情况不同。

注意：

1.虽然正则化的逻辑回归中的梯度下降和正则化的线性回归中的表达式看起来一样，但由于两者的 h(x)不同所以还是有很大差别。

2. θ_0 不参与其中的任何一个正则化。

当 h_θ 大于等于 0.5 时，预测 y=1

当 h_θ 小于 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：

$\theta^T X$ 大于等于 0 时，预测 y=1

$\theta^T X$ 小于 0 时，预测 y=0

```
h = sigmoid(X*theta);
J = (-sum(y.*log(h)+(1-y).*log(1-h))/m) + ((lambda/(2*m))*sum(theta(2:size(X,2),1).^2));

grad_0 = (X(:,1)'*(h-y))/m;
% for i=2:size(theta)
%     grad(i) = sum(X(:,i)'*(h-y)+(lambda)*theta(i))/m;
% end
grad = grad + (X'*(h-y)+(lambda)*theta)/m;
grad(1) = grad_0;
```

J就是cost function 的正则化以后的结果。

grad就是正则化的梯度，即cost function 的偏导数。注：是求完偏导数以后再统一除去的m.