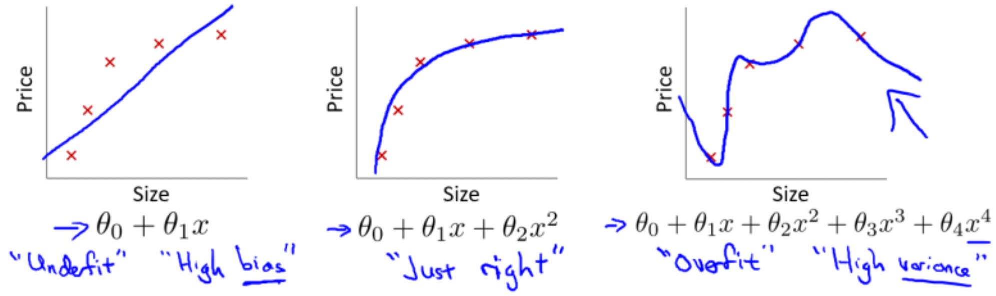
7 正则化

2022年10月26日　　20:16

7.1过拟合问题

在特定的机器学习应用中可能出现过拟合的问题导致其表现欠佳。

依然以房价预测为例：

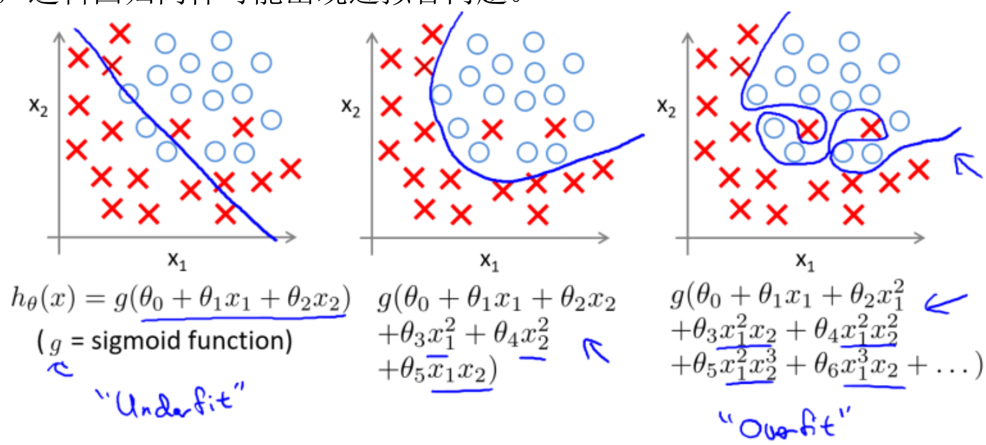


对第一个线性拟合的例子来说，我们可以获得一条拟合的直线，通过数据可以看出，随着房子面积的增大，住房价格逐渐稳定，越往右越平缓，它没有很好地拟合训练集。我们将其称为“欠拟合”(underfitting)，另一种说法是这个算法具有高偏差。
对于第二个二次函数拟合的例子来说，它的效果很好。
但是当再极端一点，用四次多项式来拟合时，似乎能够很好地拟合训练集，因为它通过了每一个数据点，但是它不停上下波动，我们不认为这是一个好模型。我们将其称为过度拟合(Overfitting)，或者说这个算法具有高方差。也即由于我们设置的变量太多，没有足够的数据对其进行约束来获得一个好的假设函数。

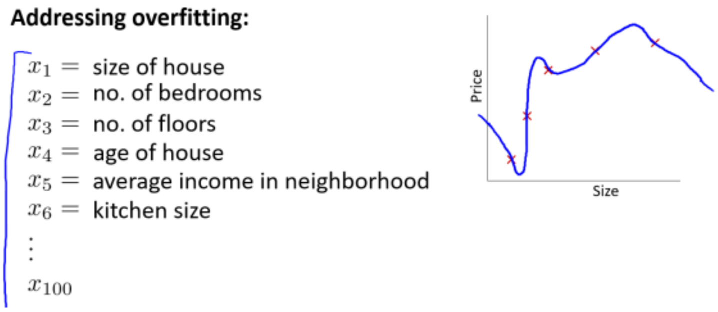
过拟合(Overfitting):如果有太多变量，假设函数可能很好地拟合数据集，但是它无法泛化到新样本中。

泛化(Generalize):一个假设模型应用到新样本的能力。

类似的，逻辑回归同样可能出现过拟合问题。



识别过拟合和欠拟合：

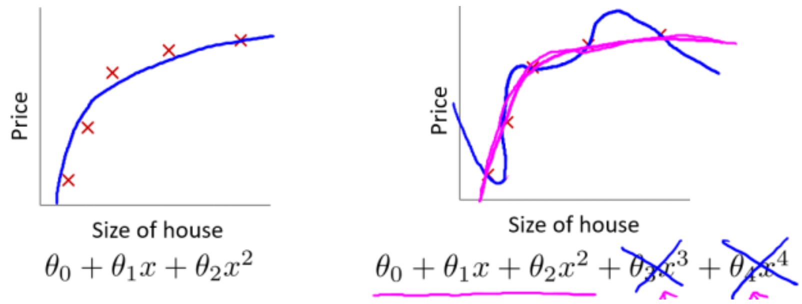


可以通过绘制图像来直观地判断是否过拟合，但是当特征量很多的时候，数据可视化就不是那么可行。

两个方法：
1. 尽量减少选取变量的数量
- 人工检查变量清单以此决定哪些变量更重要而应被保留
- 模型选择算法：自动选择特征变量的保留与否

然而虽然舍弃变量数目可以很好地缓解过拟合问题，但也同时意味着我们会丢失一些重要的问题信息。
2. 正则化
- 保留所有的特征变量，但是减少量级或者参数θ_j的大小。这种方法能适用于有很多特征量，而它们每一个对预测的y值都会产生一点影响。

7.2代价函数



如前所述，过多的变量值会导致过拟合问题。我们不妨在函数中加入惩罚项，使得参数θ_3和θ_4的值很小。
现在对原有的代价函数进行改写，如下：

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

$$\theta_3 \approx 0 \qquad \theta_4 \approx 0$$

那么为了使改写后的代价函数尽可能的小，就需要θ_3和θ_4尽可能小。最后可以发现模型拟合得更好了，它近似于二次函数，但是优于二次函数。
加入惩罚项使正则化的核心思想。

正则化(Regularization):
减小参数:
- 更简单的假设函数：加入惩罚项，更不容易出现过拟合

依然以房价预测为例，

Housing:

— Features: $x_1, x_2, \ldots, x_{100}$

— Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

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

$\cancel{\theta_0}$, $\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$

更多的特征量也意味着更多的参数，对假设函数进行修改，通过加上一个正则项来缩小每一个参数的值。

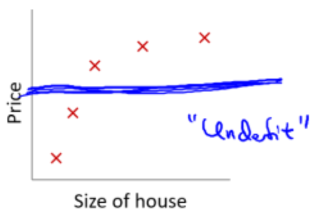
$$J(\theta) = \frac{1}{2m}\left[\underbrace{\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2}_{} + \underbrace{\lambda \sum_{j=1}^{n}\theta_j^2}_{}\right]$$

regularization parameter

$$\min_\theta J(\theta)$$

通过λ控制两个目标间的取舍：第一个目标，也即函数的第一项，更好地拟合训练集；第二个目标，也即函数的第二项，保持参数尽量地小。

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$\theta_1, \theta_2, \theta_3, \theta_4$

$\theta_1 \approx 0$, $\theta_2 \approx 0$

$\theta_3 \approx 0$, $\theta_4 \approx 0$

$\boxed{h_\theta(x) = \theta_0}$

$h_\theta(x)$

$$\theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

λ大小的设置非常重要，如果设置的过大，也就意味着对参数的惩罚力度过大，造成参数过小，体现在假设函数中相当于把全部项都忽略了，它就变成欠拟合了。这对于数据集的拟合是失败的，它的偏见性太高了。

7. 3线性回归的正则化

如前所述，通过梯度下降法更新参数值来使代价函数变小。而将正则化方法应用其中，显然需要将第0个参数分离出来，其他的更新规则不变。

Repeat {

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

$\frac{\partial}{\partial \theta_0} J(\theta)$

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

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

}

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

$\to J(\theta)$

$1 - \alpha\frac{\lambda}{m} < 1$ 0.99 $\theta_j \times 0.99$ θ_j^2

由此得到加入正则化后的更新函数。可以关注合并化简后的第一项，由于一般学习率很小、样本数量很大，那么$(1 - \alpha\frac{\lambda}{m})$这一项也就是一个略小于1的数，我们所做的工作也就是把θ缩小了一点点，也即θ_j的平方范数变小了。而第二项与添加正则项之前一模一样。

第二种更新参数的方法就是正规方程。引入正则项后的最小参数值如下：

$$\Theta = \Big(X^TX + \lambda \underbrace{\begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix}}_{(n+1)\times(n+1)}\Big)^{-1} X^Ty$$

E.g. $n=2$ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

关于不可逆的另一个补充(选学):
当样本数量小于特征量的时候，X^TX矩阵不可逆，也说它是奇异的或者退化的。而正则化则考虑到这个问题，只要正则化参数 λ 是严格大于0的，就可以保证需要求逆的那个矩阵是可逆的。

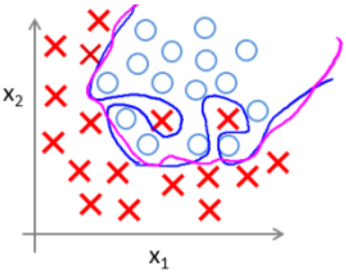
If $\lambda > 0$,

$$\theta = \underbrace{\left(X^TX + \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \right)^{-1}}_{\text{invertible.}} X^Ty$$

所以正则化处理还能解决原先的不可逆问题。

7.4逻辑回归的正则化

同前所述，当逻辑回归的特征变量非常多的时候，也可能出现过拟合问题，



$$h_\theta(x) = g(\theta_0 + \theta_1x_1 + \theta_2x_1^2 + \theta_3x_1^2x_2 + \theta_4x_1^2x_2^2 + \theta_5x_1^2x_2^3 + \dots)$$

类似地，对其使用正则化：

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

得到添加正则化项的损失函数，从而避免过拟合。
那么相应的，梯度下降的更新算法表示如下：

Repeat {

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

$$\theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} - \frac{\lambda}{m}\theta_j\right]}_{\frac{\partial}{\partial\theta_j}J(\theta)}$$

$(j = \cancel{0}, 1, 2, 3, \dots, n)$ $\theta_1, \dots, \theta_n$

}

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

具体的高阶算法实现：

fminunc(@costFunction...) $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ theta(1) theta(2) theta(n+1)

Advanced optimization

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

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

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

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

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

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

$J(\theta)$

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

gradient(3) = [code to compute $\boxed{\frac{\partial}{\partial\theta_2}J(\theta)}$];

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

$\vdots$

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