Overfitting

Shen Hengheng

2017

该笔记是来自 Andrew Ng 的 Machine Learning 课程的第三周: **多分类和过拟合技术**的课堂记录，过拟合是机器学习优化的部分，主要讲解了以下几个内容:

- 多类别的分类问题
- 过/欠拟合问题
- 解决过拟合的方法
    1. 正则化方法

0.1 多类别的分类问题

在现实生活中多类别的问题更常见，比如:

- Email: 邮件的自动分类，e.g. work,friends,family,hobby
- 医疗诊断：Notil,Cold,Flu
- 天气预测：Sunny,Cloudy,Rain,Snow

在涉及到多分类问题，我们往往采取一种 “One VS All” 算法！在二分类问题上 $Y = \{0, 1\}$，在多分类上，我们将扩大 Y 的定义，即 $Y = \{0, 1, ..., n\}$。多分类的任务转化为 $n+1$ 个（**+ 1 因为索引从 0 开始**）二分类问题；在每一个二分类任务，预测的 “Y” 是一个类概率。

$$
\begin{aligned}
&y \in \{0, 1...n\} \\
&h_\theta^{(0)}(x) = P(y = 0|x; \theta) \\
&h_\theta^{(1)}(x) = P(y = 1|x; \theta) \\
&\cdots \\
&h_\theta^{(n)}(x) = P(y = n|x; \theta) \\
&\text{prediction} = \max_i (h_\theta^{(i)}(x))
\end{aligned}
$$

我们基本上是选择其中一个类别的数据作为 **positive**，然后将所有其他的类别数据为 **negative**，这样将问题转化为一个二分类问题，反复这样做，对每一种情况应用**二元 logistic 回归**，然后将所有情况中预测的最高的那个类作为我们的预测。下图1 显示了如何将 3 个类进行分类:

1. 对于每一个类训练一个二元逻辑回归分类器来预测 $y = 1$ 的概率.
2. 对于一个新的测试样本 x，选择 h_θ 最大的那一个类.

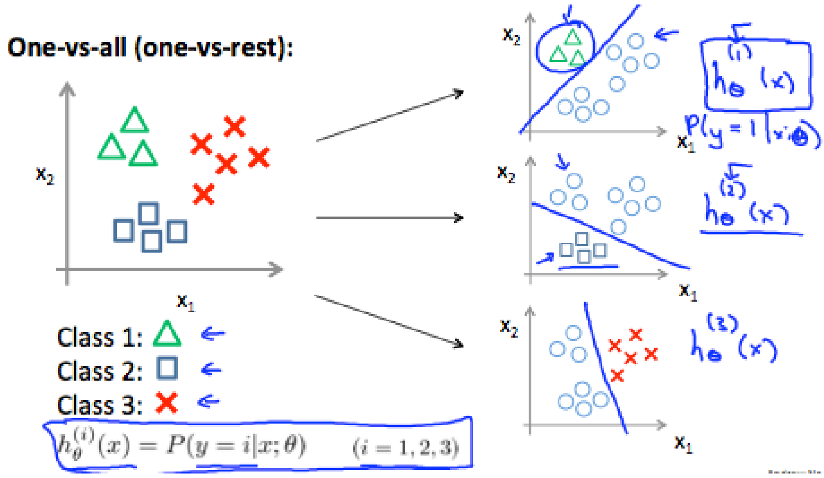

图 1: 将 3 个类进行分类

0.2 过/欠拟合问题

机器学习的目的不仅仅对训练集的拟合效果好，在测试集上我也要达到一样好！所以往往在机器学习建模时往往出现过拟合问题，就是对训练数据的预测达到 100%，但是在测试集上的效果很差！还有一种情况就是在训练集上的效果低，这叫欠拟合问题，如图所示

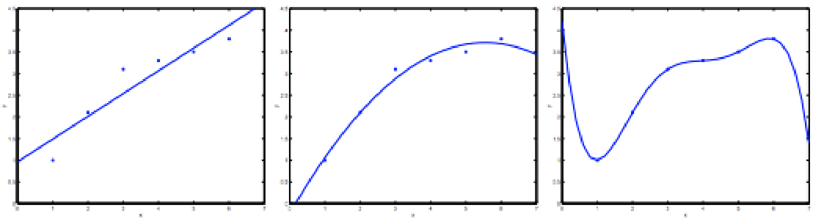

图 2: 欠拟合/正确/过拟合

考虑 $x \in \mathbb{R}$ 预测 y 的问题。上图2的最左边的图显示了 $y = \theta_0 + \theta_1 x$ 拟合数据集 X 的结果。我们看到数据不是真的在直线上，所以拟合程度不是很好。相反，如果我们添加了一个额外的特征 x^2，用 $y = \theta_0 + \theta_1 x + \theta_2 x^2$ 来拟合 y, 那么我们获得一个稍微更好的数据拟合（见中图）。看起来添加的特征越多越好，然而，增加太多特征也有一个危险：最右边的图是拟合 5 阶多项式 $y = \sum_{j=0}^{5} \theta_j x^j$ 的结果，可以看到即使曲线完美地拟合了所有数据，但是也不会是个很好的模型。左边的图片是一个**欠拟合**的例子，而右侧的图就是过拟合的例子。

欠拟合（高偏差，低方差），它通常是由于假说函数过于简单或使用的特征太少而造成的。另一个极端，**过拟合（低偏差，高方差）**，它通常是由一个复杂的假说函数或者特征太太多引起的，而导致模型不能很好的泛化[1]。

[1]泛化指一个假设模型能够应用到新的样本数据的能力

解决过拟合问题两个主要策略:

1. 减少特征数量
   - 人工检查变量的数目，决定哪些变量重要，哪些变量不重要.
   - 模型选择算法
2. 正则化
   - 当我们有很多特征变量时，其中每一个变量都能对预测产生一点影响.
   - 保留所有特性，但减少参数 θ_j 的大小

0.3 修改代价函数

如果模型（假设/说函数）出现过拟合，我们可以减轻部分权重 θ_j，进而增加他们的代价。比如说，我们想让下面的函数看起来更加像二次方程

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

所以我们减弱 $\theta_3 x^3$ 和 $\theta_4 x^4$ 的影响. 实际上, 如果不去掉这些特征或者改变假设函数 $h_\theta(x)$ 的形式, 我们可以改变我们的代价函数:

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

在公式后面增加了两项用来增加 θ_3 和 θ_4 的代价，为了使得我们的代价函数接近 0，所以不得不将 θ_3 和 θ_4 的值接近于 0。在假说函数中，将大大降低 $\theta_3 x^3$ 和 $\theta_4 x^4$ 的值。因此, 可以看到新的假设 (**由粉红色曲线描述**) 看起来更像一个二次函数, 并且也更好的拟合我们训练数据。

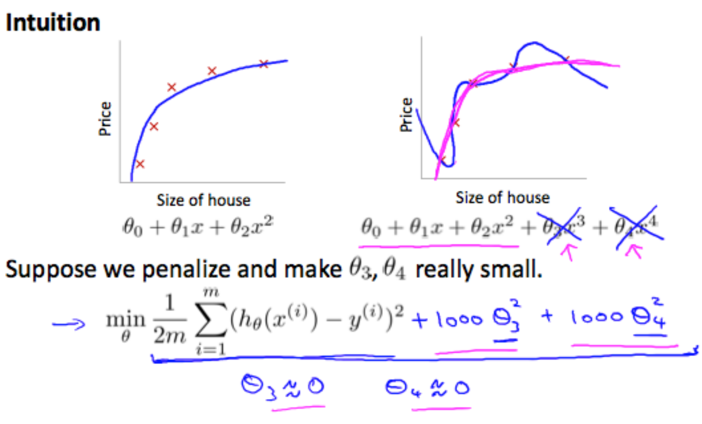

图 3: 原有的假说 vs 增加了对 θ_3 和 θ_4 的惩罚

我们还可以将所有的 θ 参数求和来进行正则化处理, 在代价函数最后增加一个**正则项** $\lambda\sum_{j=1}^{n}\theta_j^2$，其中 λ 是一个正则化参数（**超参数**）. 如下所示.

$$min_\theta\ \frac{1}{2m}\ \sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2+\lambda\ \sum_{j=1}^{n}\theta_j^2$$

0.4 给线性回归模型增加正则项

0.4.1 Gradient Descent For Fixed Linear Regression

$$
\begin{aligned}
&\text{Repeat } \{ \\
&\quad \theta_0 := \theta_0 - \alpha\ \frac{1}{m}\ \sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_0^{(i)} \\
&\quad \theta_j := \theta_j - \alpha\ \left[\left(\frac{1}{m}\ \sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}\right)+\frac{\lambda}{m}\theta_j\right] \qquad\qquad j\in\{1,2...n\} \\
&\}
\end{aligned}
$$

其中上面的 θ_j 的更新公式也可以这样写:

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

.

0.4.2 Normal Equation

$$\theta=\left(X^TX+\lambda\cdot L\right)^{-1}X^Ty,\ \text{where}\ \ L=\begin{bmatrix}0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1\end{bmatrix}_{(n+1)\times(n+1)}$$

前面提到过, 如果 $m<n$, 那么 X^TX 是不可逆矩阵。然而, 当我们添加 λL 后 X^TX 变得可逆了。

0.5 给逻辑回归模型增加正则项

0.5.1 Cost Function

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

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

公式 (2) 的第二个 sum 项 $\sum_{j=1}^{n}\theta_j^2$ 中的 θ_j 不包括 θ_0!

0.5.2 Gradient Descent For Fixed Logistic Regression

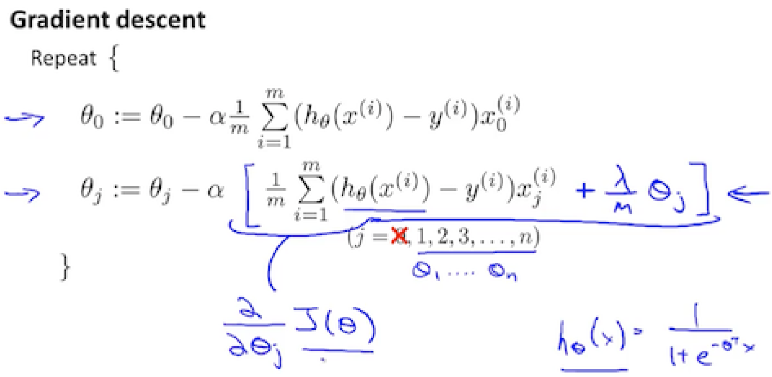

图 4: 逻辑回归改进后的梯度下降算法