斯坦福大学公开课 Machine Learning 学习笔记

第六章 机器学习算法的使用建议

之前我们学习了一些算法，但在实际训练中还会有一些工程方面的问题，下面的学习为我们在选择、训练和验证算法准确性方面提供了宝贵的建议。

评估一个算法

确定下一步需要做什么

当我们在一个新的房价数据集上测试我们的房价估计函数时，发现我们的算法产生了一个难以接受的误差，那么接下来我们应该做什么？下面是一些建议：

- 获取更多的训练数据集（针对高方差）
- 尝试采用更少的特征集（针对高方差）
- 尝试采用更多的特征集（针对高偏差）
- 尝试采用更多的多项式项（如x_1^2, x_2^2, x_1x_2 等，针对高偏差）
- 尝试减小正则化参数λ（针对高偏差）
- 尝试增大正则化参数λ（针对高方差）。

评估假设函数

在训练样本上得到一个很小的误差不一定是一件好事，如果不能很好的泛化到新样例，那么就是过拟合。那么，如何评价一个假设函数是否过拟合？

将原本训练数据集的一部分拿出来作为测试假设函数的数据集，即数据集的一部分作为训练集，剩下的作为测试集。一般情况下，会将数据集的 70%作为训练集（train 集），30%作为测试集（test 集）。然后使用下面的方式进行测试：

1. 使用训练集训练权重Θ并最小化$J_{train}(\Theta)$；
2. 计算测试集的误差值$J_{test}(\Theta)$。

其中，对于线性回归，$J_{test}(\Theta)$的计算方式：

$$J_{test}(\Theta) = \frac{1}{2m_{test}}\sum_{i=1}^{m_{test}}(h_\Theta\left(x_{test}^{(i)}\right) - y_{test}^{(i)})^2 \quad \text{(公式 1)}$$

对于逻辑回归或者神经网络等分类算法，测试集误差：

$$\text{Test Error} = \frac{1}{m_{test}}\sum_{i=1}^{m_{test}} err(h_\Theta\left(x_{test}^{(i)}\right), y_{test}^{(i)}) \quad \text{(公式 2)}$$

其中函数err为误分类错误函数，公式：

$$err(h_\Theta(x), y) = \begin{cases} 1, \text{ 如果} h_\Theta\left(x_{test}^{(i)}\right) \geq 0.5 \text{ 并且} y_{test}^{(i)} = 0, \text{ 或} h_\Theta\left(x_{test}^{(i)}\right) < 0.5 \text{ 并且} y_{test}^{(i)} = 1 \\ 0, \text{ 其他} \end{cases}$$

(公式 3)

由于误分类函数将每个样本的误差值转化为 1，所以逻辑回归的测试集误差就代表了在测试集中有多少比例的样例被错误分类了。

模型选择，训练集、验证集和测试集

由于不同的多项式项的假设函数最终的拟合程度不同，那么我们怎样挑选一个比较好的多项式呢？简单的方法是通过上节的验证方法选择，假设我们选择了十组多项式项，经过训练得出了不同的权重向量，如：

d=1 $h_\theta(x) = \theta_0 + \theta_1 x_1$——————————>min$J_{train}(\Theta^{(1)})$ ————>$\Theta^{(1)}$

d=2 $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$——————————>min$J_{train}(\Theta^{(2)})$ ————>$\Theta^{(2)}$

d=3 $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$——————————>min$J_{train}(\Theta^{(3)})$ ————>$\Theta^{(3)}$

…

d=10 $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_{10} x_{10}$———>min$J_{train}(\Theta^{(10)})$ ————>$\Theta^{(10)}$

然后假设由于$\Theta^{(5)}$在测试集上泛化的最好，故选择了$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_5 x_5$。

以上的测试还是有问题的，因为通过$J_{test}(\Theta^{(5)})$选择的$\Theta^{(5)}$可能过度拟合了测试集样本，但不一定也能很好地拟合新的样本，因为$\Theta^{(5)}$也是通过测试集选择的。

这样就需要一个新的解决办法：将原有数据集分成三份，增加一个交叉验证集，比较经典的分发为：

60%的训练集，用于训练计算假设函数的权值；

20%的交叉验证集（Cross Validation Set），用于挑选非过拟合的权值集合；

20%的测试集，用于评估算法挑选的假设函数的误差，即评估泛化误差。

这样的方法，选择的Θ就不是通过测试集训练得出的，过拟合的可能性就比较小了。

偏差与方差（Bias VS Variance）

一个算法不理想要么是因为方差太大（过拟合-Underfit），要么是因为偏差太大（欠拟合-Overfit），确定是哪一种，是改进算法的最有效方法。下面介绍多项式维度、正则化参数和数据集规模与偏差和方差的关系，从而解释了”确定下一步需要做什么“小节中给出的解决偏差或者方差的建议是可行的。

多项式维度与偏差和方差的关系

训练误差$J_{train}(\theta)$会随着多项式维度的增加而减少。同时，交叉验证误差$J_{cv}(\theta)$会在维度增加到一定阶段减小到最低然后随着维度增加而增加，形式如图 1。

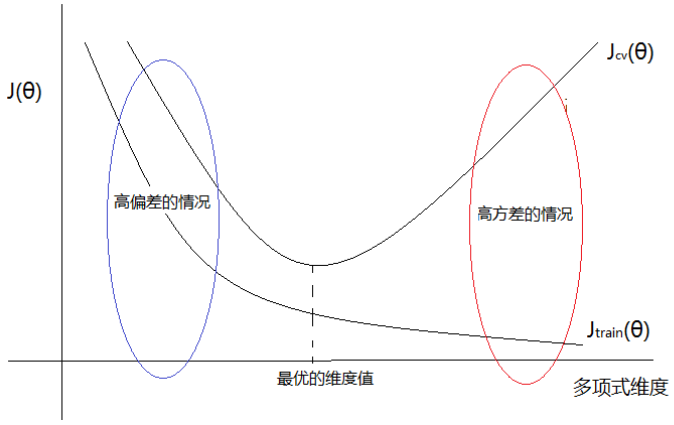

图 1

从图中可以看出：
对于高偏差的情况，$J_{train}(\theta)$和$J_{cv}(\theta)$都很高，而且两值差不多相等；
对于高方差的情况，$J_{train}(\theta)$比较低，$J_{cv}(\theta)$比较高，而且$J_{cv}(\theta)$远远大于$J_{train}(\theta)$；
理想的维度值是$J_{train}(\theta)$和$J_{cv}(\theta)$都很低，而且两值相差不多。

正则化参数与偏差和方差的关系

对于高方差的问题，如之前所述可通过适量增大正则化参数λ来解决。但是，选择怎样的λ比较合适，用怎样的方式来衡量合适呢？我们可以通过以下方式选择合适的正则化参数λ：

1. 预先选择一组λ，如{0,0.01,0.02,0.04,0.08,0.16,0.32,0.64,1.28,2.56,5.12,10.24}；
2. 选择一组假设函数模型，每个拥有不同的多项式或者其他不同的变量；
3. 迭代每个预先选择的正则化参数λ，在λ确定的情况下训练每个模型，得到权值；
4. 计算每个已经训练出权值的不含正则化项的模型的交叉验证集误差；
5. 选择交叉验证集误差最小的模型和λ，并在测试集上验证，看它们是否能很好地泛化到测试集。

数据集规模与偏差和方差的关系

对于极少的数据集（如 1/2/3）我们能使训练误差为 0，因为我们总能找到一个二次方程完全拟合训练集，但是随着数据集规模的增加，误差就开始增加。
对于高偏差的训练模型来说：
少量的数据集会使得$J_{train}(\Theta)$很低但是$J_{cv}(\Theta)$很高；
大量的数据集会使得$J_{train}(\Theta)$和$J_{cv}(\Theta)$都很高，而且两值相当，如下图 2 所示：

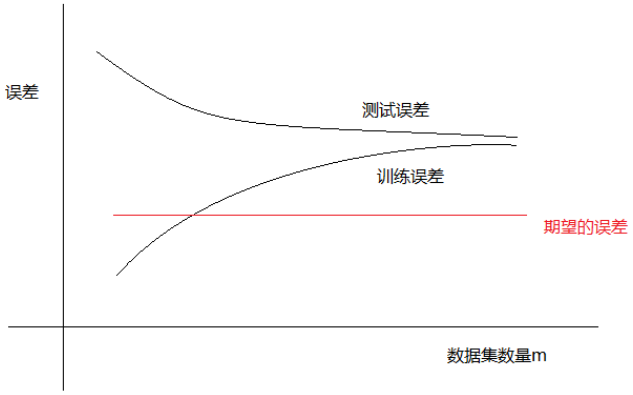

图 2

所以对于高偏差的模型，用增加训练数据集规模的方式降低偏差是没有帮助的。

对于高方差的训练模型来说：

少量的数据集会使得$J_{train}(\Theta)$很低但是$J_{cv}(\Theta)$很高；

大量的数据集会使得$J_{train}(\Theta)$增加但是$J_{cv}(\Theta)$也会持续降低且不会趋于平稳，虽然$J_{train}(\Theta) < J_{cv}(\Theta)$但是两者相差仍比较大，如下图 3。

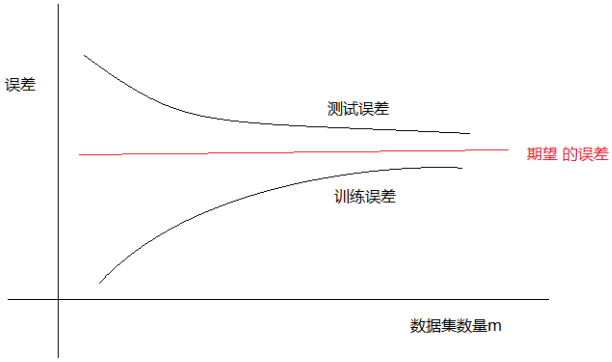

图 3

所以对于高方差的模型，用增加训练数据集规模的方式降低过度拟合是可能的。

机器学习系统设计举例——建立一个垃圾邮件分类器

优先应该做什么？

如何分配时间使得分类器有一个低误差？

- 收集大量的数据
- 根据邮件的路由信息收集垃圾邮件的特征值
- 根据邮件的消息体收集垃圾邮件的特征值
- 设计精细的算法从而检测错误拼写

错误分析

推荐的方法：

- 一开始使用一个简单的算法快速地实现分类器并且使用交叉验证集进行测试。
- 画出$J_{train}(\theta)$和$J_{cv}(\theta)$对数据集和特征值数量等的学习曲线，以确定更多的数据和特征值等是否有帮助。
- 手动检查算法在交叉验证集中的错误分类样例，分析算法错误分类的原因，从而有针对地调整算法。手动检测能够发现算法可能会犯的错误，这经常能够帮助我们找到更加有效的手段；这也解释了第一条推荐的方法，即先实践一种快速但是不完美的算法，因为对于不同的算法，它们所遇到的问题一般总是相同的，通过实践一些快速即便不完美的算法，能够快速地找到错误所在，并快速找出算法难以处理的例子，这样就能够集中精力在这些真正的问题上。

误差分析可能并不能用来测试某种算法可以提高性能，唯一的方法就是尝试。

处理偏斜（Skewed）数据集

偏斜类的误差度量

先举一个例子：癌症预测。
假如训练了一个逻辑回归模型$h_\theta(x)$（如果是癌症，则 y=1，否则 y=0），而且发现模型在测试数据集上有 1%的错误（99%的诊断正确率）。但是，调查发现只有 0.5%的病人患有癌症，也就是说如果人工随意选择一个样本，猜测它不是癌症的准确率高达 99.5%，竟然比训练的模型的准确率还高。那么，训练的模型还有没有预测的意义？这便是偏斜类导致的问题。
偏斜类：在数据集中，一个类的样本数比另一类多很多的情况。
一种对偏斜类有用的评估度量方法：准确率（Precision）和召回率（Recall），在计算这两个数值之前需要用到下面的参数：

	真实类（Actual class）		
预测类		1	0
(Predict	1	True Positive	False positive
class)	0	False negtive	True negtive

上表将所有样本按模型预测出来数据的正确分为四个部分。其中，真实类表示数据样本的 y 值，而预测类表示模型预测的$\bar{y}$值。True Positive 表示当$\bar{y}=1$ 且 y = 1时，即本来是癌症，且预测结果也是癌症的所有样本数；False positive 表示当$\bar{y}=1$ 且 y = 0时，即本来不是癌症，但预测结果是癌症，预测是错误的所有样本数；False negtive 表示当$\bar{y}=0$ 且 y = 1时，本来是癌症，但预测结果不是癌症的所有样本数；True negtive 表示当$\bar{y}=0$ 且 y = 0时，即本来不是癌症，且预测结果也不是癌症的所有样本数。
则准确率的计算方式为：

$$\text{Precision} = \frac{True\ positive}{\#Predict\ positive} = \frac{True\ positive}{True\ positive + False\ positive} \quad \text{(公式 1)}$$

从上面的公式可以看出准确率衡量的是模型的“预测结果是癌症，且本来就是癌症的情况”的精度。

召回率的计算公式如下：

$$\text{Recall} = \frac{True\ positive}{\#Actual\ positive} = \frac{True\ positive}{True\ positive + False\ negtive} \quad \text{(公式 2)}$$

从上面的公式可以看出召回率衡量的是模型的“本来是癌症，且预测结果也是癌症的情况”的精度。

权衡精度和召回率

有了精度和召回率，我们如何将它们应用在衡量算法上呢？

- 如果我们希望只有在有十足的把握的情况下，才会将预测值置为 1，即当$h_\theta(x) \geq 0.9$（阈值为 0.9）时，认为是癌症。我们的算法应该满足：高准确率和低召回率。
- 如果我们不希望漏掉任何可能是癌症的预测，即$h_\theta(x) \geq 0.3$（阈值为 0.3）时，便认为是癌症。我们的算法应该满足：高召回率和低准确率。

如何通过准确率和召回率来比较几种算法呢？使用以下公式：

$$\text{F} - \text{score} = 2\frac{Precision * Recall}{(Precision + Recall)} \quad \text{(公式 3)}$$