研究生算法课课堂笔记

上课日期：2016-12-05　　第(2)节课

组长学号及姓名：1601214550 刘强强

组员学号及姓名：1601214738 谭楚婧 1601111322 陈帅

内容概要：

本节课回顾了监督学习的重要概念，并讲授了 xgboost 目标函数的求解的理论推导过程，第一次大作业存在的问题，第二次大作业的要求以及 DART，具体如下：

1.xgboost 目标函数的求解

2.第二次大作业要求

3.第一次大作业存在的问题

4.Dropouts in Mart(DART)

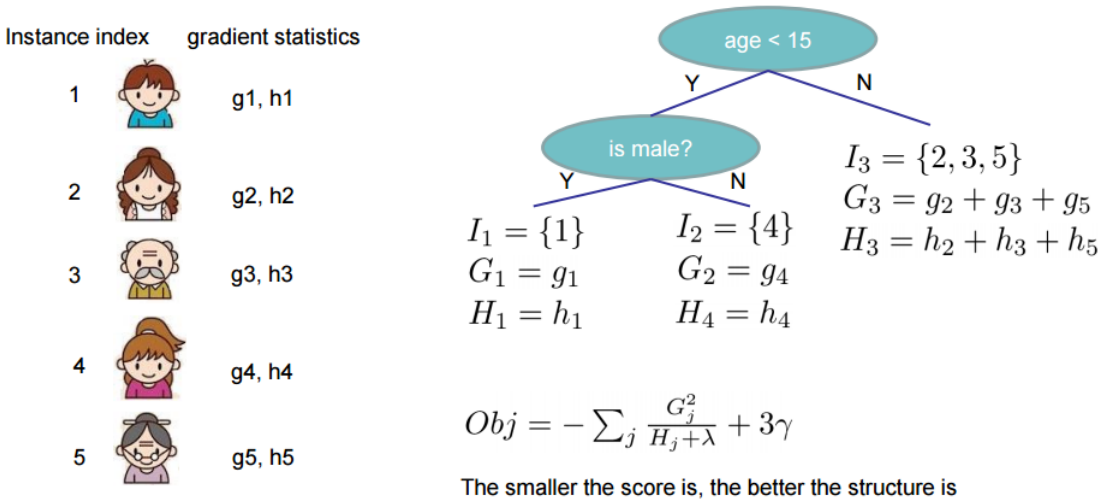
详细内容：

1. xgboost 目标函数的求解

（1） 分数计算

如下图所示目标函数为：

$$Obj = -\sum_j \frac{G_j^2}{H_j+\lambda} + 3\gamma$$



（2） 单棵树的搜索算法

① 枚举可能存在的树的结构 q

② 计算 q 中结构的分数

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

③ 使用最佳的叶子节点的权重计算最优树的结构

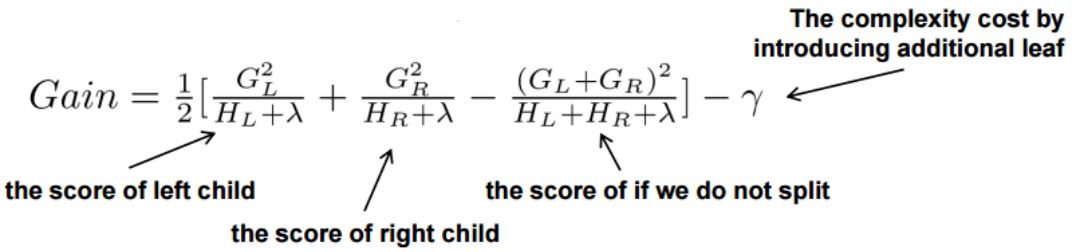
$$w_j^* = -\frac{G_j}{H_j+\lambda}$$

④ 但是可能存在无限种树的结构

（3） 使用贪心算法来生长树

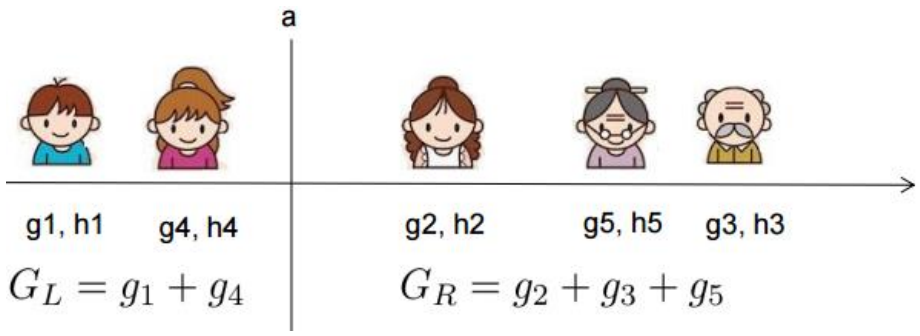
① 从树的深度为 0 的节点开始

② 对于树的每一个节点，计算分裂后的目标值的改变。判断得到使得 gain 最大的那个节点就为最佳分裂点。该分裂点为当前树生长的位置。



（4） 如何高效的寻找最佳分裂点

① 当 $x_j < a$ 时（其中 x_j 是年龄），Gain 的分裂规则如下：



② 计算 Gain 的总和

$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

③ 从左到右线性的扫描排序后的 instance，决定最优分裂点。

（5） 分裂节点寻找算法

① 对于每个节点，枚举所有的特征

对于每个特征，通过特征值对 instance 进行排序。使用线性扫描决定特征的最佳分裂点。

② 生长深度为 K 的树的时间复杂度

时间复杂度为 *O(ndKlogn)*：对于每层排序时间为 *O(nlogn)*, 有 d 个特征，我们需要做 K 层。这个还可以继续优化。可以处理大规模的数据。

在每次分裂的时候都需要遍历每个特征。对于每个特征都需要把所有的 instance 做一个排序。这是对连续型特征的处理。Gain 可能为负，因为我们减去了叶子节点的惩罚项，当我们发现最好的切分都为负的时候，就不做切分了。将整棵树切分到不能再分时，再回退，看哪一步 gain 为负然后再做调节。

（6） 类别型数据目标函数求解

做完 One-Hot Encoding 之后做类似的处理。

（7） xgboost 算法回顾

① 每次循环添加一棵新的树

③ 在每个循环的开始计算

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

③ 使用贪心算法生成树 $f_t(x)$

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

④ 将 $f_t(x)$ 加入模型

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$$

通常情况下，我们做

$$y^{(t)} = y^{(t-1)} + \epsilon f_t(x_i)$$

$\in$叫做步长，通常我们设置为 0.1。我们设置为 0.1 意味着我们并没有在开始就做充分的优化，为了让一次不要学习太多，给以后的优化留点空间减少 overfitting。

（8） 总结

有监督的学习最重要的三个部分，模型，优化目标，对应的参数。这些可以帮组我们得到一个一般的优化的形式。

2. 第二次大作业要求

（1） 任务描述

在这个挑战中，你需要根据某网站上用户的注册信息以及历史记录，预测其是否会购买网站提供的产品。所有在此数据集的用户都来自美国。网站提供的待购买商品是比较贵重的商品。

（2） 作业要点

我们提供四个文件：

1. train_users.csv ，训练集用户的基本信息
2. test_users.csv，测试集用户的基本信息
3. sessions.csv，所有用户的 session 历史
4. sample_submission.csv，提交样例

相比较第一次作业，这次作业提供的数据更加原始，需要同学们基于这些原始数据自己尝试着构造 feature。比如在 train_users.csv 中可能存着用户账户创建的日期信息。此外，sessions.csv 文件包含更多可以利用的信息，我们需要从这些原始的数据中抽取出更加丰富的信息。比如用户在网站上的操作的次数，搜索的次数，时间间隔等。

这次作业是一个典型的分类任务，评价指标采用 log loss。最后的提交格式参考 sample_submission.csv，为每一条预测数据提交一个 0 到 1 之间的实数，把 log loss 优化到最低。

（3） 作业提交方式

提交到教学网上。将这次作业按照第一次的提交要求提交，提交作业报告（包括代码），最好在作业报告中提到作业中存在的问题，思路以及如何解决。每组只需要交一份作业。

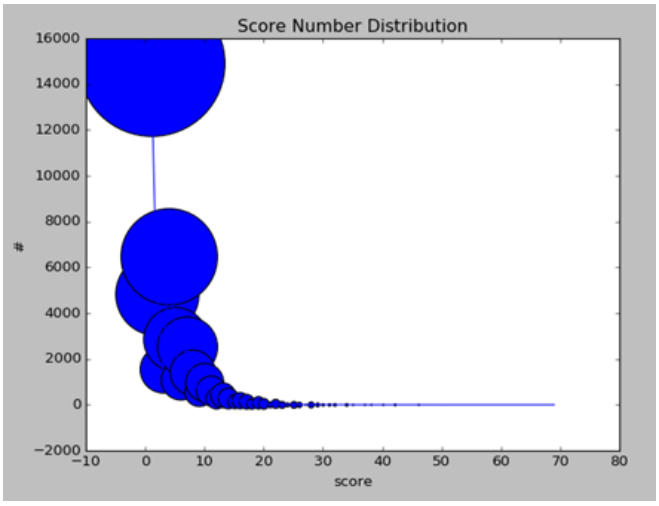
2. 针对第一次大作业提出的疑问

（1） RMSE 达到多少才算好？

对于使用单个 XGBoost 模型来做 Cross Validation 的同学来说，做出 3.7 的结果已经不错了。需要注意的是，在最终的 Test Set 上的结果通常会略差于 CV 的结果。所以，同学们应该先从 Ground Truth 中取出 20%做为你们测试用的 Test Set。用余下的 80%做 CV 来调整参数。

（2） 风险指数分布情况是怎样？

这次作业是有关风险指数的数据，低风险的数目较多，高风险的数目较少，如下图所示。对于预测数据，如果我们只能预测一个值的话，那么这个值应该就是训练数据的所有 y 值的平均数，这个数约等于 4。大家可以算一下，此时 Test Set 的 square loss 也差不多是 4（这两者相等只是巧合）。



圆的大小代表该指数的数据的数量，可以看出数据主要分布在 0~10 的范围内，大部分的数据的风险指数小于 10，所有数据的风险指数的平均值为 4。

（3） 如何说明我们的模型 work 了？

虽然 3.7 不是很低，但是如果我们随机的话一般是大于 4 的，虽然初始化的 square loss 可能比较高，当 training loss 和 validating loss 在逐渐降低，就说明了我们的模型 work 了。

（4） 为什么 validating loss 下降速度很慢？

training loss 和 validating loss 的下降速度不一致，training loss 下降速度很快，而 validating loss 下降速度很慢，一般情况下是出现了 overfitting。

（5） 分而治之

同学们可以尝试首先把问题转化为一个二分类问题：是否可以找到一个分界点 A，可以训练一个分类器很好的预测目标值 Y 值是否大于这个分界点。如果可以找到这样的一个分界点 A 并训练一个准确率高的分类器的话，我们就可以分别针对大于分界点 A 的训练样本训练一个回归模型，然后再针对小于分界点 A 的训练样本训练一个回归模型。

（6） 其他评价指标（这是写课堂笔记的同学自己找的材料，不一定正确）

准确率(Accuracy)

准确率是指在分类中，使用测试集对模型进行分类，分类正确的记录个数占总记录个数的比例：

$$accuracy = \frac{n_{correct}}{n_{total}}$$

准确率看起来非常简单。然而，准确率评价指标没有对不同类别进行区分，即其平等对待每个类别。例如在病患诊断中，诊断患有癌症实际上却未患癌症（False Positive）与诊断未患有癌症的实际上却患有癌症（False Negative）的这两种情况的重要性不一样。另一个原因是，可能数据分布不平衡，即有的类别下的样本过多，有的类别下的样本个数过少，两类个数相差较大。这样样本占大部分的类别主导了准确率的计算，为了解决这个问题，对准确率进行改进，得到平均准确率。

平均准确率(Average Per-class Accuracy)

为了应对每个类别下样本的个数不一样的情况，对准确率进行变种，计算每个类别下的准确率，然后再计算它们的平均值。因为每个类别下类别的样本个数不一样，即计算每个类别的准确率时，分母不一样，则平均准确率不等于准确率，如果每个类别下的样本个数一样，则平均准确率与准确率相等。

平均准确率也有自己的缺点，比如，如果存在某个类别，类别的样本个数很少，那么使用测试集进行测试时（如 k-fold cross validation），可能造成该类别准确率的方差过大，意味着该类别的准确率可靠性不强。

平方根误差（RMSE）

回归模型中最常用的评价模型便是 RMSE（root mean square error，平方根误差），其又被称为 RMSD（root mean square deviation）。 RMSE 虽然广为使用，但是其存在一些缺点，因为它是使用平均误差，而平均值对异常点（outliers）较敏感，如果回归器对某个点的回归值很不理性，那么它的误差则较大，从而会对 RMSE 的值有较大影响，即平均值是非鲁棒的。

4. Dropouts in Mart(DART)

同学们在使用 xgboost 做回归任务的时候，参数 booster 有三个选项：gbtree、gblinear 和 Dart。其中：gbtree 对应着子模型采用树型结构；gblinear 对应着子模型采用线型结构；Dart 则不是很容易理解，所以这节课详细介绍一下 Dart。

Dart 就是 Dropouts in Mart，我们分别介绍 Dropouts 和 Mart。

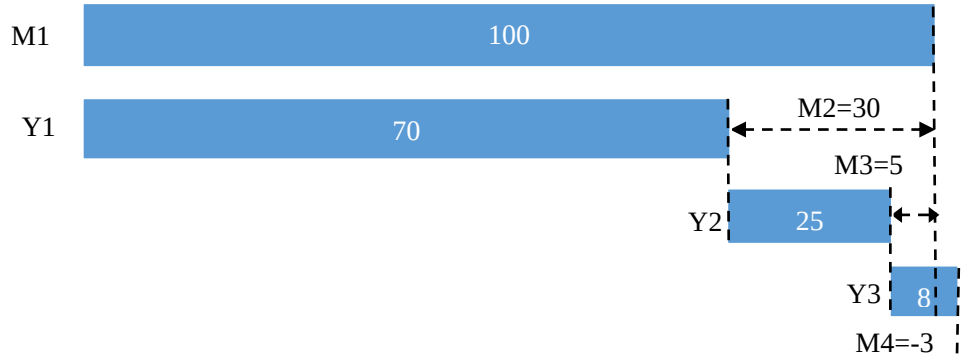
Dropouts：Dropouts 是近两年神经网络研究的一个热点，是指在训练神经网络时，随机的按照一定概率让一部分神经元不参与训练。以随机梯度下降为例，每次 mini-batch 训练的都是不同的更“瘦”的网络，不仅减弱了神经元之间的联合适应性理解，增强了泛化能力，还缩短了训练时间。

MART：理解 MART 首先要提到 Decision Tree，基于 Decision Tree 又要引伸出 4 个概念：Random Forest、AdaBoost Decision Tree、GBDT 和 Xgboost。

- Random Forest ：即把多个 Decision Tree 放在一起做一个 uniform 的 blending。由于每个 Decision Tree 是平均投票，为了降低模型的 bias，该算法希望每个 Decision Tree 尽可能不同，并且各自有比较高的准确率。
- AdaBoost DT：同 Random Forst 不同，AdaBoost DT 不是大家平均投票，而是每个 Decision Tree 都有自己的权重。Adaboost DT 提出时是专门用来做分类任务的，后一个 DT 学习前一个 DT 没有学习好的部分（调整数据的权重）。加大分类误差率小的弱分类器的权值，使其在表决中起到较大的作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。
- GBDT：是 AdaBoost DT 的泛化，能做分类、回归、排序等各种任务。而 MART 其实就是 GBDT 的另一种叫法，所以 Dropouts in Mart 就是 Dropouts in GBDT。
- XGBoost：是 GBDT 的一种 C++的高效实现。

我们为什么要把 Dropouts 引入 MART 呢？

首先举一个 MART 做回归任务的例子：



假设 GBDT 训练的目标值 M1 是 100，但第一棵树能力有限只预测出了 70，则第二棵树的优化目标 M2 应该是 M1 与 Y1 的残差 M1-Y1=30；同样的，第二棵树可能也没有准确预测出 30，而是预测了 Y2=25，还差 M3=M1-Y1-Y2=5，那么 M3 就是第三棵树要优化的目标，以此类推。

即 GBDT 的思想可表述如下：

- 逐个训练子模型
- 子模型 Ti+1 的优化目标是“当前所有子模型的输出值之和”与“目标值”的残差 M.

从上述流程中我们可以发现，GBDT 存在一个问题：第一棵树在整个决策过程中的作用较大，一旦预测出现偏差，对之后预测结果产生很大影响。为了减少第一棵树的在整个训练过程中起到的作用，我们引进了 Dart。GBDT 在训练第 i+1 个子树的时候，其优化目标是前面所有子模型（从 T1 到 Ti）的输出值之和和目标值之间的残差，如公式（1）所示：

$$M = \sum_{i=1}^{n} T_i$$

Dart 的优化目标则是 GBDT 优化目标的一个子集，即随机丢弃在 T1 到 Ti 间的一部分子模型，如公式（2）所示：

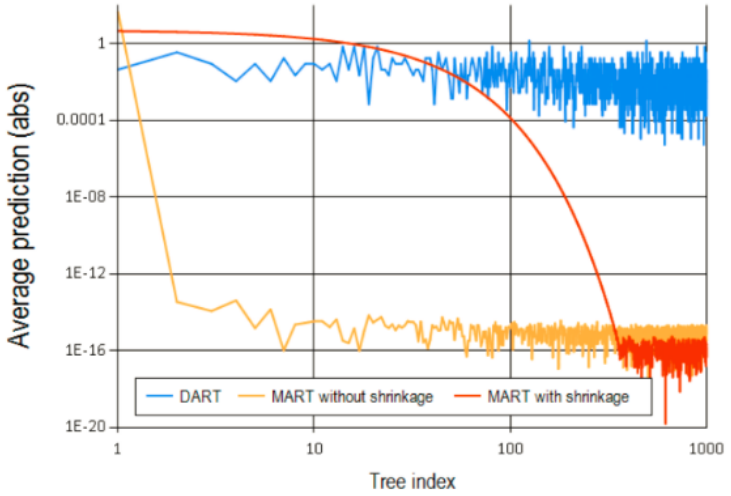
$$\hat{M} = \sum_{i \in I} T_i$$

下面考虑两个极端的情况：

- 每次都把前 i 棵树全扔掉：那么，每棵树的优化目标直接就是 y，Dart 退化成了 Random Forest，每棵树都优化同样的目标值，互不影响。
- 每次一棵树都不扔：Dart 退化成了 GBDT，每棵树的优化目标都是前 i 棵树的所有残差。

因此，Dart 是一个折中的方案，即按照一定比例扔掉一部分树，如 20%。在这种情况下，当前训练的这棵树既需要干 GBDT 中本来这棵树该干的活，还需要干扔掉的那部分树干的活，这样每棵树都是我中有你，你中有我。所以会造成即使其中的第一棵树出现了一些偏差，别的树还能把这个偏差拉回来。

用一张图理解 Dart



X 轴：表示在 Tree blend 模型学习过程中，Tree 的 id。

Y 轴：每棵树叶子节点上的数值的绝对值的平均值。

黄色线表示 Mart：虽然每棵树的权重一样，但由于第一棵树的 y 值很大，导致第一棵树拟合的占比很大。第二棵树拟合第一棵树的残差，值就小了很多，于是产生一个陡变，之后都是拟合前面 i 棵树的残差，叶子节点上数值的大小变动也会趋于一个较小值。

红色线表示 Mart with shrinkage：Shrinkage 对于残差学习出来的结果，只累加一小部分（step*残差）以逐步逼近目标，step 一般都比较小，如 0.01~0.001，导致各个树的残差是渐变的而不是像没用 shrinkage 时的陡变。

蓝色线表示 Dart：后面每棵树的叶子节点大小和树大小的数值非常接近，蓝线的具体形状与 Dart 中扔掉子树的比例有关。