研究生算法课课堂笔记

上课日期：2016-12-05　　第(1)节课

组长学号及姓名：1601214472 马树铭

组员学号及姓名：1601111286 刘天宇　1601214495 张明华

内容概要：

本节课回顾了监督学习的重要概念，并讲授了 xgboost 的理论推导过程，具体如下：

1. 监督学习重要概念回顾
2. 回归树以及 Tree Ensemble
3. Tree Ensemble 方法三要素
4. Tree Ensemble 学习算法
5. 损失函数的泰勒展开
6. 模型正则项
7. 目标函数的最小化求解

详细内容：

1. 监督学习重要概念回顾

（1） 模型和参数

给定第 i 个训练样本 $x_i \in R^d$，模型(Model)的预测目标是根据给定的x_i得到对应的预测值$\hat{y}_i$。

我们以线性模型为例，线性模型(包含线性回归和 logistic 回归)：$\hat{y}_i = \sum_j w_j x_{ij}$。对于不同的模型，预测值$\hat{y}_i$有不同的含义：

对于线性回归任务，$\hat{y}_i$是对训练样本x_i的打分。

对于 logistic 回归，$1/(1+\exp(-\hat{y}_i))$反映了训练样本x_i成立的概率。

参数(Parameter)是模型从训练数据中学习得到的。对于线性模型，参数$\theta = \{w_j | j = 1,\cdots,d\}$，表示训练样本中每个特征的权重。

（2） 目标函数

目标函数(Objective)是我们模型训练的目标：

$$\mathrm{Obj}(\theta) = \mathrm{L}(\theta) + \varphi(\theta)$$

其中，$\mathrm{L}(\theta)$是模型的训练误差，它衡量了模型对训练数据的拟合程度；$\varphi(\theta)$是模型的正则项，它衡量了模型的复杂度，可以有效防止过拟合。

训练误差是训练集中所有样例的误差之和：

$$\mathrm{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

损失函数代表了训练样本的预测值与真实值的误差，我们介绍以下两种损失函数，他们一般分别适用于线性回归以及 logistic 回归任务：

平方损失函数：$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$

0-1 损失函数：$l(y_i, \hat{y}_i) = y_i \ln\left(1 + e^{-\hat{y}_i}\right) + (1 - y_i)\ln(1 + e^{\hat{y}_i})$

正则项则反映了模型的复杂程度，它是我们防止模型过拟合的重要手段。我们介绍以下两种正则项形式：

L2 正则项：$\varphi(\theta) = \lambda\|w\|^2$

L1 正则项：$\varphi(\theta) = \lambda\|w\|$

（3） 目标函数的偏差-方差均衡(bias-variance tradeoff)

我们的目标函数包含训练误差和正则项两个部分，这两个部分对我们的模型有着不同的贡献。优化训练误差让模型预测能力更强。如果模型能够很好的拟合训练数据，那么它有可能很好的预测样本分布，使模型的预测结果与样本真实分布更接近。考虑正则项让模型更简单。复杂度较低的模型 variance 较小，正则项可以让模型预测结果更加稳定，防止过拟合。

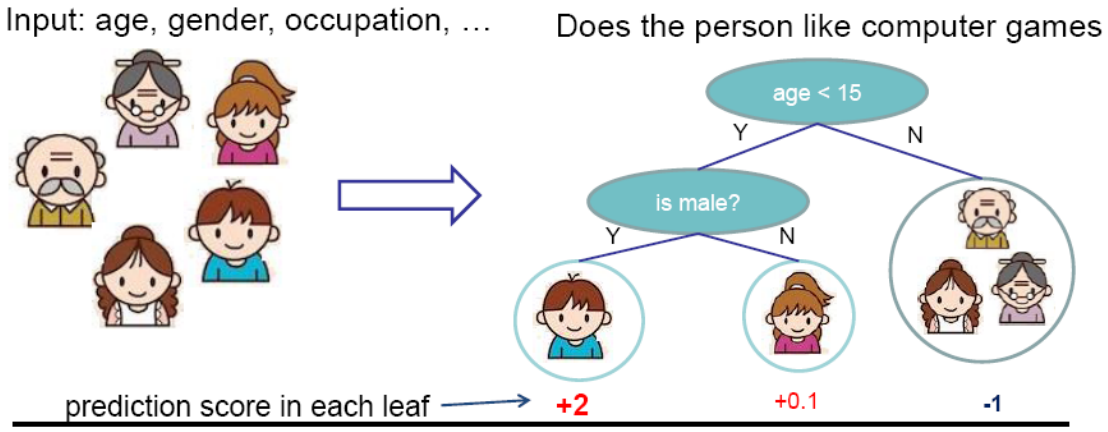
2. 回归树以及 Tree Ensemble

（1） 回归树定义

回归树(classification and regression tree)是一种应用很广泛的决策树模型，它既可以用于分类任务也可以用于回归任务。

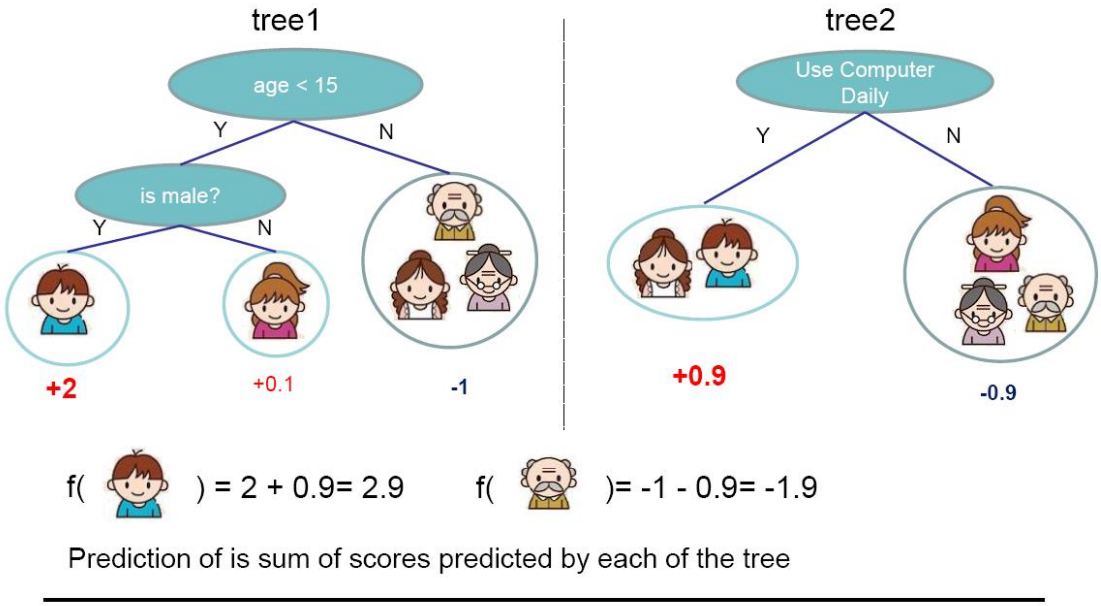
回归树的特征选择、生成方式和剪枝方式和决策树类似。回归树的每一个叶子节点会包含该叶子节点的打分。

如下图所示，我们的任务是预测某个人是否喜爱电脑游戏，决策的特征包括年龄、性别、职业等。对于年龄小于 15 岁的男性，预测得分最高，为+2；对于年龄大于 15 岁的人，预测得分最低，为-1。



（2） 回归树 Ensemble

Ensemble 的基本思想是用多棵树的预测结果增强最终的预测准确性。下图中，预测任务同样是一个人是否喜欢电脑游戏，但是使用了两棵回归树。我们可以看到年龄在 15 岁以下且每天都使用电脑的男性的预测得分由+2 增加到了+2.9；而年龄大于 15 岁且不是每天都使用电脑的人的得分更低了，从-1 变为-1.9。



（3） Tree Ensemble 的优势

1. 实用性很强，比如 Boosting 方法、随机森林等。很多数据挖掘的竞赛优胜者都会使用 Ensemble 的思想。
2. 可以不用特征归一化操作。
3. 能够学习到特征之间内在关联。
4. 扩展性很强，可以处理大规模数据，适用于工业界。

（4） Tree Ensemble 的应用

回归树 Ensemble 定义了一个由输入到预测值的计算过程，依据具体目标函数的定义形式，回归树 Ensemble 可以用于分类，回归以及排序等实际任务。

使用平方损失函数，将得到一个通用的 gradient boosted machine：

$$l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

使用 Logistic 损失函数，将得到一个 LogitBoost：

$$l(y_i, \hat{y}_i) = y_i \ln\left(1 + e^{-\hat{y}_i}\right) + (1 - y_i) \ln\left(1 + e^{\hat{y}_i}\right)$$

3. Tree Ensemble 方法三要素

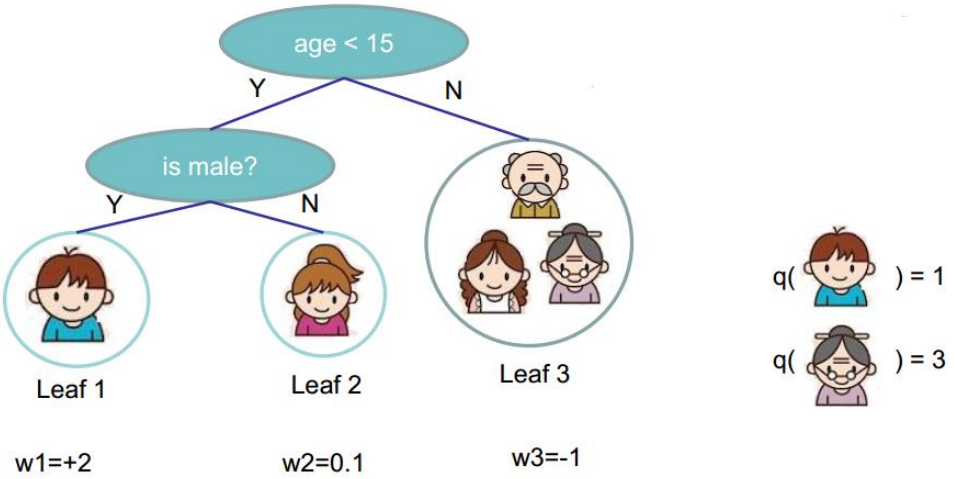
（1） 模型定义

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \in \mathcal{F}$$

以上模型总共包含 K 棵树，每棵树f_k是一个在函数空间$(\mathcal{F})$里面的函数，而$\mathcal{F}$对应了所有回归树的集合。我们将f_k形式化定义为如下结构：

$$f_k(x) = w_{q(x)}, \qquad \mathrm{w} \in R^T, \qquad q: R^d \to \{1,2,\cdots,T\}$$

其中w是T维向量，每一维w_t对应f_k的一个叶节点输出值，同时可以把q视为一棵数学化的树结构。对于给定的实例x，q函数将其映射到树的某一个节点，即可得到该实例的预测值。如下图包含 3 个叶节点的树结构示例：



（2） **模型参数**

从上面树结构的数学定义可知，模型参数主要包括两个部分：叶节点的输出值w以及每一棵树的映射结构q。

（3） **学习目标**

Bias-variance 权衡是机器学习中非常重要的研究内容，我们不仅希望模型能准确地做预测同时也可以尽量保持简单性。依据前面介绍的监督学习目标函数，这里也将运用 loss + regularization 的定义模式，把学习目标拆分为训练误差和模型复杂度两大部分，来实现在学习过程中较好的权衡预测的准确度和模型复杂度。

$$\text{Obj} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

对于复杂度Ω，可以将其定义为树的节点个数或者叶节点输出值w的L2正则，这部分可以参阅笔记后面的详细说明。

4. Tree Ensemble 学习算法

（1） **求解最优化问题**

构造出任务适配的目标函数之后，机器学习问题转化为求解最优化问题。对于约束最优化求解问题，通常借助拉格朗日对偶性将原始问题转换为对偶问题，以间接求解原始问题。当面对无约束最优化问题时，随机梯度下降（SGD）是常用的经典迭代求解算法。利用负梯度方向来决定每次迭代的搜索方向，逐步优化目标函数值。然而在 Tree Ensemble 学习过程中，我们的参数是回归树，相当于需要在一个函数空间里搜索最优参数，此时传统的 SGD 学习算法也变得无能为力。

（2） **Additive Training (Boosting)**

$$\hat{y}_i^{(0)} = 0$$

$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$

$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$

…

$$\hat{y}_i^{(r)} = \sum_{k=1}^{r} f_k(x_i) = \hat{y}_i^{(r-1)} + f_r(x_i)$$

在 additive training 学习中，模型初始化为常数值预测，之后的每一轮 boosting 保留原来的模型，并叠加一个新的函数f_r，获得的新模型用于当前轮的预测。那么如何选择新加入的f_r呢？我们应该在每一轮选取f_r实现目标函数的最优化。第 r 轮的目标函数定义如下：

$$Obj^{(r)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(r)}\right) + \sum_{k=1}^{r} \Omega(f_k)$$

$$= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(r-1)} + f_r(x_i)\right) + \Omega(f_r) + constant$$

考虑损失函数是平方损失的情形，此时可得到如下目标函数，其中$\left(\hat{y}_i^{(r-1)} - y_i\right)$是前 r-1 轮训练得到的模型预测值和实例真实值的残差。

$$Obj^{(r)} = \sum_{i=1}^{n} \left(y_i - \left(\hat{y}_i^{(r-1)} + f_r(x_i)\right)\right)^2 + \Omega(f_r) + constant$$

$$= \sum_{i=1}^{n} \left[2\left(\hat{y}_i^{(r-1)} - y_i\right) f_r(x_i) + f_r(x_i)^2\right] + \Omega(f_r) + constant$$

5. 损失函数的泰勒展开

（1） 问题：复杂的损失函数

考虑一般情形下的损失函数，第 r 轮（或者说第 r 棵树）目标函数的形式为

$$Obj^{(r)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(r-1)} + f_r(x_i)\right) + \Omega(f_r) + constant,$$

我们需要通过这个目标函数来找到使得目标函数最小的函数f_r，即

$$f_r = \underset{f'}{\operatorname{argmin}}\, Obj^{(r)}.$$

我们之前分析了损失函数$l\left(y_i, \hat{y}_i^{(r-1)} + f_r(x_i)\right)$为平方损失函数时$f_r$的求解方法，但是当损失函数不是平方损失函数时，求解就变得很复杂，因此我们需要将损失函数进行简化。

（2） 方法：泰勒展开

定义：泰勒展开是将一个在$x = x_0$处具有 n 阶导数的函数$f(x)$利用关于$(x - x_0)$的 n 次多项式来逼近函数的方法。

公式：函数$f(x + \Delta x)$处的泰勒展开公式为

$$f(x + \Delta x) \backsimeq f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2.$$

若我们把损失函数$l\left(y_i, \hat{y}_i^{(r-1)} + f_r(x_i)\right)$对$\hat{y}_i^{(r-1)}$的一阶导数和二阶导数标记为

$$g_i = \partial_{\hat{y}_i^{(r-1)}} l\left(y_i, \hat{y}_i^{(r-1)}\right),$$

$$h_i = \partial^2_{\hat{y}_i^{(r-1)}} l\left(y_i, \hat{y}_i^{(r-1)}\right),$$

则目标函数$Obj^{(r)}$可以根据泰勒公式展开为

$$Obj^{(r)} \backsimeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(r-1)}\right) + g_i f_r(x_i) + \frac{1}{2} h_i f_r^2(x_i)\right] + \Omega(f_r) + constant$$

（3） 例子：平方损失函数

以平方损失函数为例，一阶导数和二阶导数分别为

$$g_i = \partial_{\hat{y}_i^{(r-1)}}(\hat{y}_i^{(r-1)} - y_i)^2 = 2(\hat{y}_i^{(r-1)} - y_i)$$

$$h_i = \partial^2_{\hat{y}_i^{(r-1)}}(\hat{y}_i^{(r-1)} - y_i)^2 = 2$$

代入目标函数$Obj^{(r)}$的泰勒展开公式为

$$Obj^{(r)} = \sum_{i=1}^{n}\left(y_i - (\hat{y}_i^{(r-1)} + f_r(x_i))\right)^2 + \Omega(f_r) + constant$$

$$= \sum_{i=1}^{n}\left[2\left(\hat{y}_i^{(r-1)} - y_i\right)f_r(x_i) + f_r(x_i)^2\right] + \Omega(f_r) + constant$$

这个公式与之前直接代入平方损失函数的推导是一致的。

（4） 新的目标函数

在去掉常数项后，目标函数转化为

$$Obj^{(r)} \backsimeq \sum_{i=1}^{n}\left[g_i f_r(x_i) + \frac{1}{2}h_i f_r^2(x_i)\right] + \Omega(f_r)$$

6. 模型正则项

（1） 定义

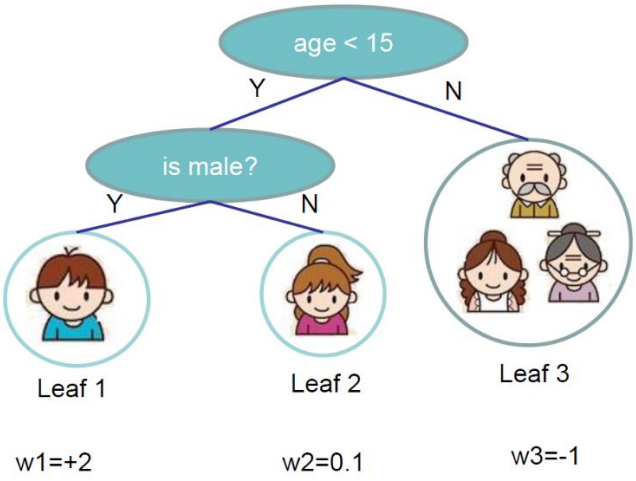
正则项描述了模型的复杂度，boosted tree 的模型复杂度取决于叶子的个数以及叶子的分数，因此可以定义一棵树的复杂度为

$$\Omega(f_r) = \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

（2） 例子

如下图所示，叶子个数为 3 个，每个叶子的分数分别为 2,0.1 和-1，因此复杂度为

$$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$



7. 目标函数的最小化求解

（1） 目标函数的形式

通过对损失函数的简化和对正则项的定义，我们可以得到目标函数的形式。我们先定义I_j为落在第 j 个叶子上的训练样本的集合：

$$I_j = \{i|q(x_i) = j\},$$

那么目标函数可以写成：

$$Obj^{(r)} \backsimeq \sum_{i=1}^{n}\left[g_i f_r(x_i) + \frac{1}{2}h_i f_r^2(x_i)\right] + \Omega(f_r)$$

$$= \sum_{i=1}^{n}\left[g_i w_q(x_i) + \frac{1}{2}h_i w_q^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}\left[\left(\sum_{i\epsilon I_j} g_i\right)w_j + \frac{1}{2}\left(\sum_{i\epsilon I_j} h_i + \lambda\right)w_j^2\right] + \gamma T$$

也就是 T 个独立的二次函数的求和，因此可以求解二次函数得到使目标函数最大的函数f_r。

（2） 目标函数的求解

对于一般形式的二次函数$Gx + \frac{1}{2}Hx^2$，它的最小值为

$$\min_x Gx + \frac{1}{2}Hx^2 = -\frac{1}{2}\frac{G^2}{H}$$

$$\operatorname{argmin}_x Gx + \frac{1}{2}Hx^2 = -\frac{G}{H}, H > 0$$

若我们定义$G_j = \sum_{i\epsilon I_j} g_i, H_j = \sum_{i\epsilon I_j} h_i$，那么目标函数可以改写为

$$Obj^{(r)} = \sum_{j=1}^{T}\left[\left(\sum_{i\epsilon I_j} g_i\right)w_j + \frac{1}{2}\left(\sum_{i\epsilon I_j} h_i + \lambda\right)w_j^2\right] + \gamma T$$

$$= \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2\right] + \gamma T$$

假设树的结构是确定的，那么使得目标函数$Obj^{(t)}$最小的叶子的权重为

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

此时目标函数为

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

其中$\frac{G_j^2}{H_j+\lambda}$这一项衡量了树结构的好坏。