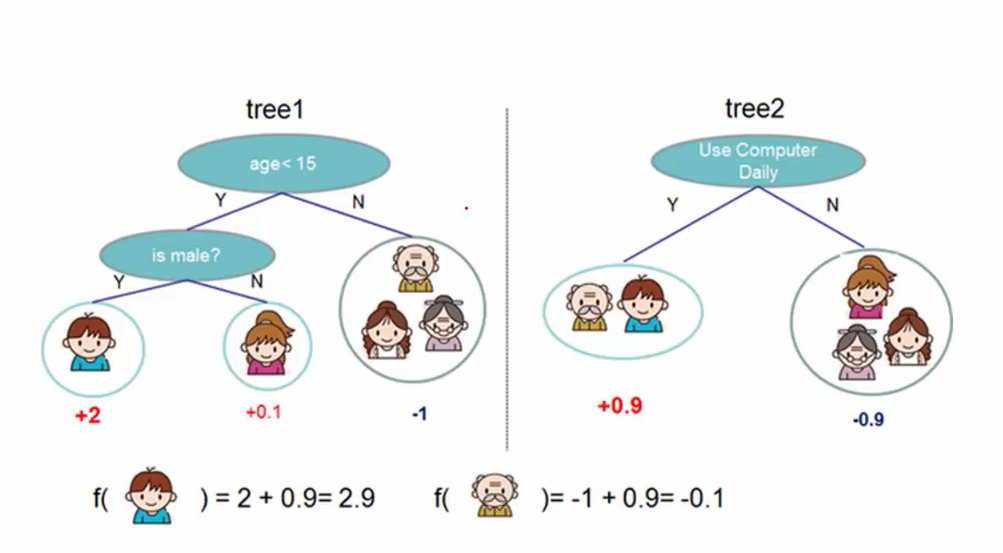
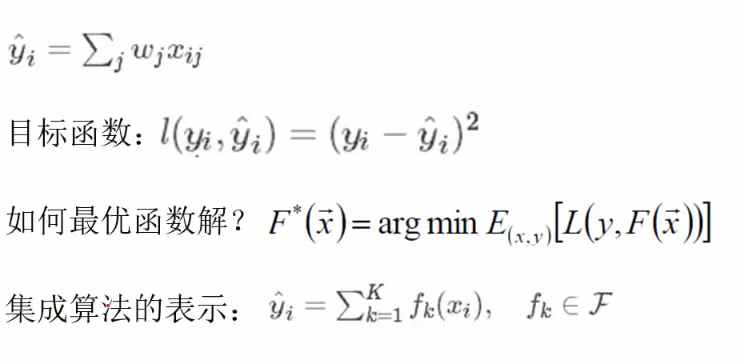
Xgb报告

1. Xgb的由多个弱分类器集成的一个分类算法，基础分类器就是决策树，我们看到一般单个决策树深度很小，不同的决策树可能用了不同的特征将同一个人分到了不同的叶子节点，然后每一个叶子节点都是一个权值的



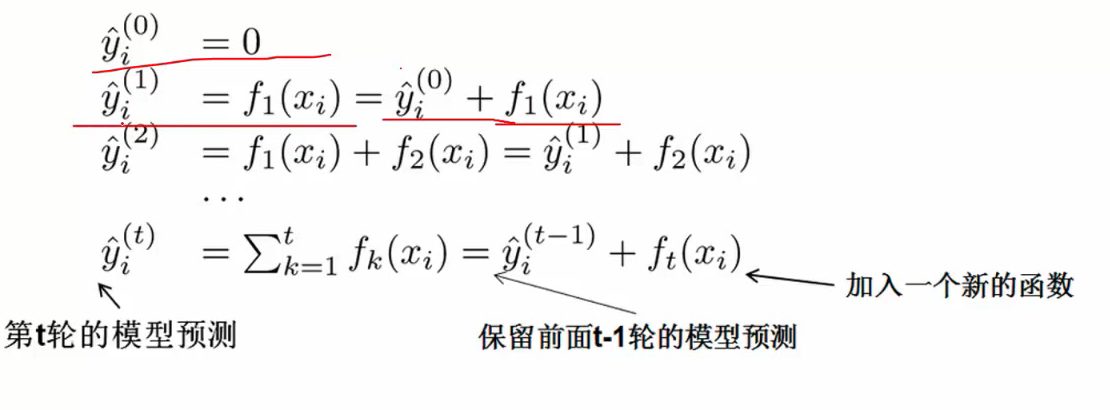
比如说图中这个分类任务是预测这个是不是喜欢玩游戏，我们有一系列特征，权值为正代表喜欢，比如第一个树这个小男孩得2分，第二个树得0.9分，综合得分2.9，

总结出来就是，一个人的得分就可以表示为不同树的对应节点的特征值和权值的线性表示：WX。现在我们可以得到一个目标函数，然后可以进行优化了

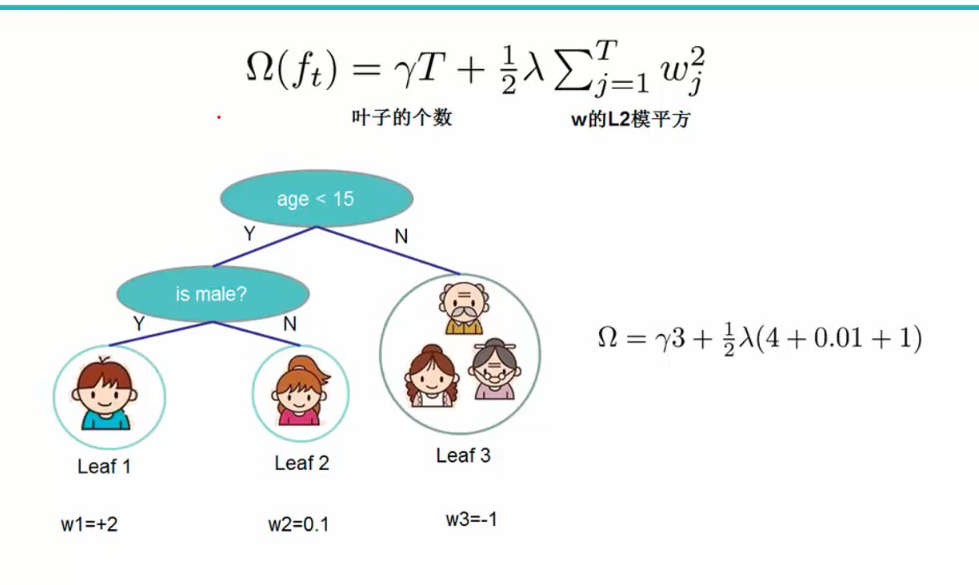


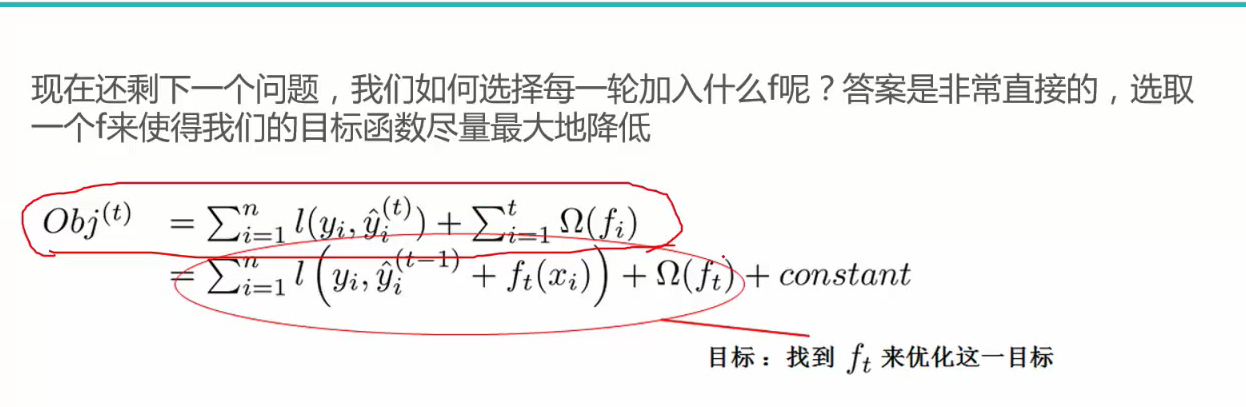
每一个样本都有一个损失值，最优解是所有样本的损失均值，相当于整体的期望求得一个最小解。

1. 既然是集成算法，它其实就是不断的添加新的树上去然后和之前的所有树集成到一起看效果，但是新树的增加是有条件的，它是期望每一次新加一个树之后呢对之前的效果要有所提升的，

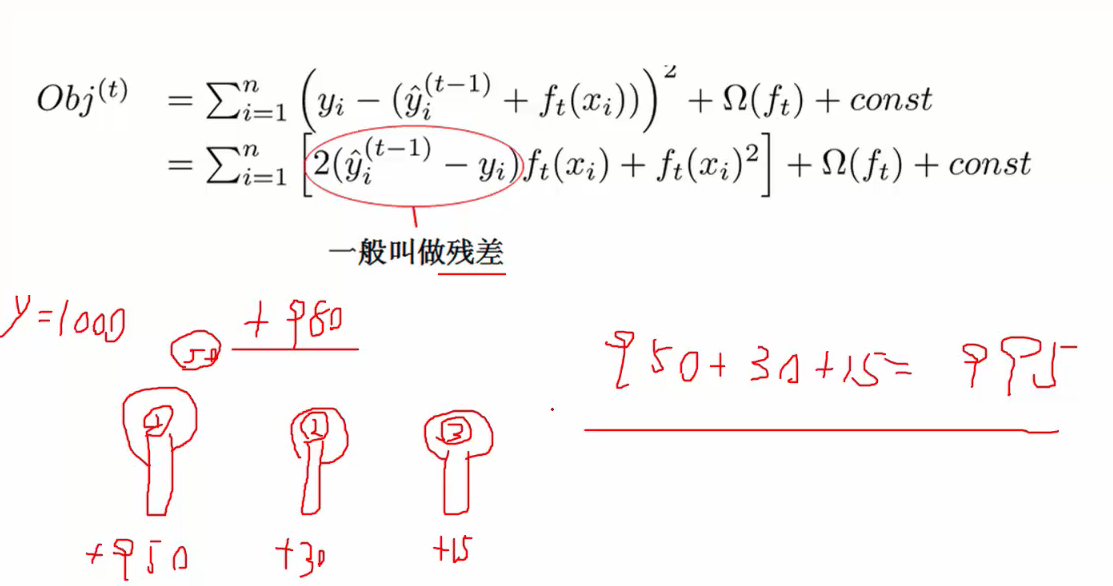


1. 由于决策树存在过拟合的风险，所有给最终的目标函数添加了一个惩罚项，这个惩罚项和叶子节点的个数有关。



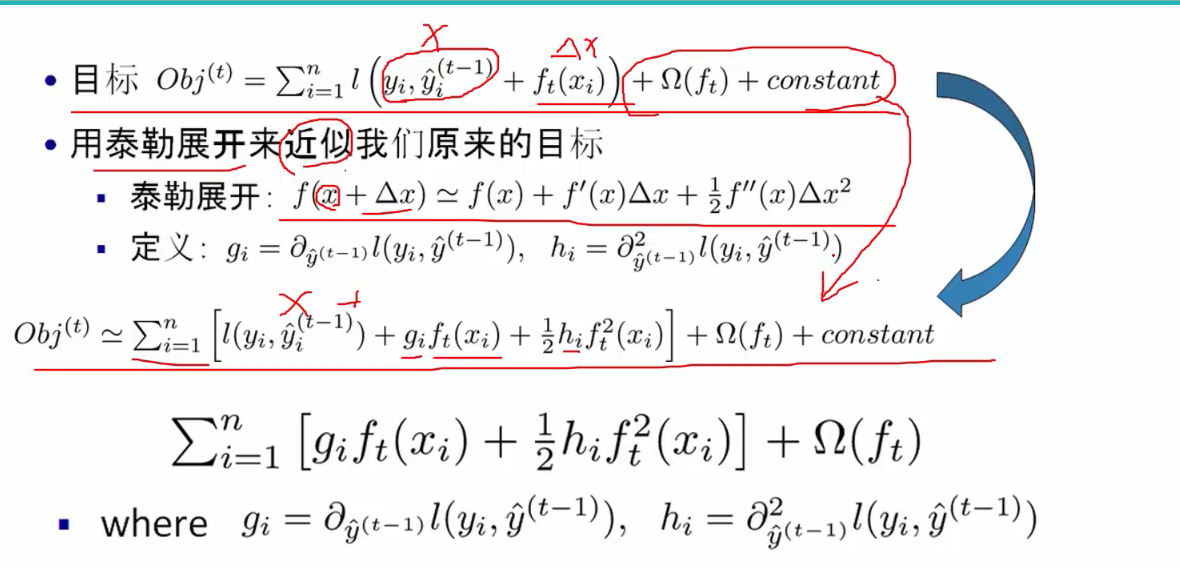


Xgb就是梯度提升的，



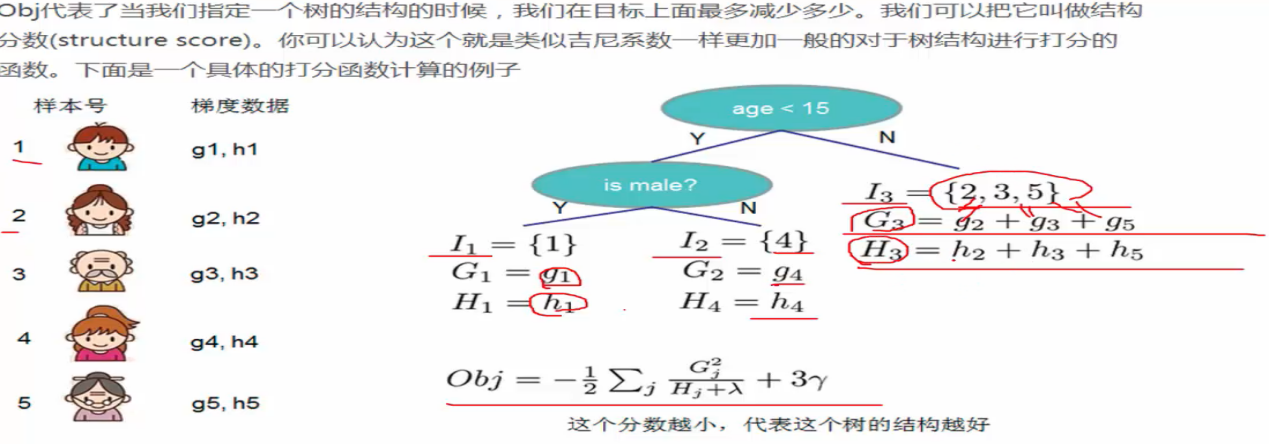
比如真实值是1000，t-1次预测结果是950，那这50便是预测残差，那新加一个t轮的树，需要更接近预测结果，它在提升30，这样以此类推，所以核心就在新增树很根据之前的残差来构建新的决策树

1. 数学原理

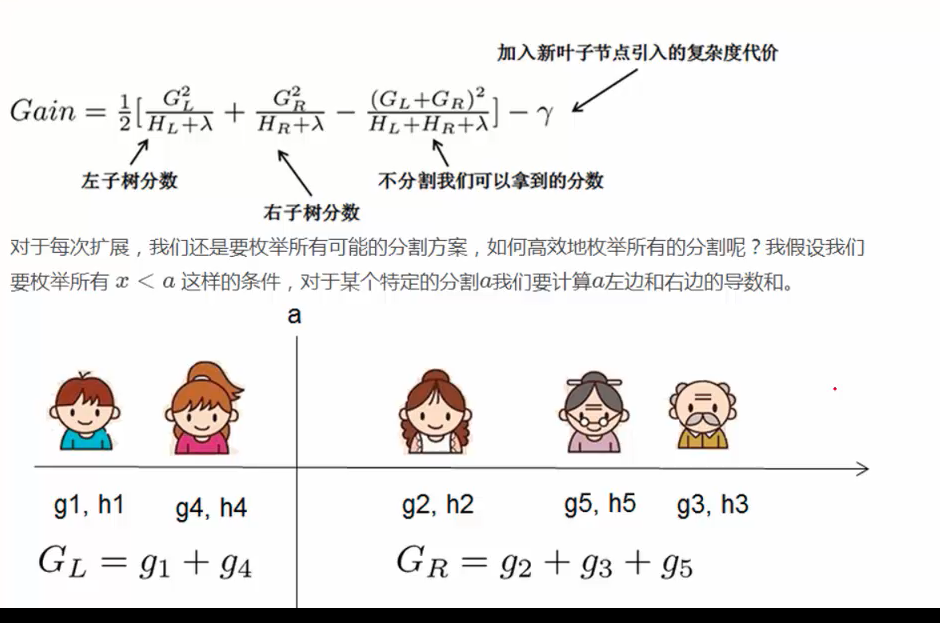


我们来看目标函数其实可以抽象为f（x+dx）微分的表现形式，又可以进行泰勒展开，对于dy=Adx我们近似的用函数的一阶导g和二阶导h的和来表示，

最终目标函数可以转化为这样的形式



在xgb中决策树的生成策略就不是信息增益或是基尼系数了，而是用这个优化函数因为最终的优化函数与叶子节点的个数有关



分割前后做一个差值，评估这一次分割的效果，只不过评价方法变了

2 xgb在金盘特征上的使用：

样本量：4.6万， 逾期样本：4.5k 历史特征

不经过调参压根没有效果；进行参数优化以后，在训练集上有较好的表现，在测试集上扑街，

|  |  |
| --- | --- |
| Train Accuracy: 96.32% | Test Accuracy: 89.91% |
| AUC Score (Train): 0.815767 | AUC Score (test): 0.545441 |
| Train KS: 0.6315335122825999 | Test KS: 0.09088159983612434 |

接下来我做了这样的事情

1. scale\_pos\_weight=7, # 样本不均衡调节 在测试集上达到了最优效果，也就是说逾期样本大概占比一半

|  |  |
| --- | --- |
| Train Accuracy: 90.57% | Test Accuracy: 82.39% |
| AUC Score (Train): 0.857201 | AUC Score (test): 0.589932 |
| Train KS: 0.7144011409917838 | Test KS: 0.17986480258104165 |

1. 然后我感觉可能是样本不均衡造成的过拟合，所以我进行了下采样，我又把原数据直接删除了一部分好用户最终样本量13000，坏样本三分之一

|  |  |
| --- | --- |
| AUC Score (Train): 0.871766 | AUC Score (test):0.603698 |
| Train KS: 0.7435324629942068 | TestKS: 0.20739537884874643 |

样本量：4.6万， 逾期样本：8.5k 地址特征

1 参数寻优后直接跑：

|  |  |
| --- | --- |
| Train Accuracy: 86.19% | Test Accuracy: 82.23% |
| AUC Score (Train): 0.631231 | AUC Score (test): 0.549756 |
| Train KS: 0.262461713198759 | Test KS: 0.0995113473479419 |

2 调整样本比例 scale\_pos\_weight=5

|  |  |
| --- | --- |
| Train Accuracy: 69.69% | Test Accuracy: 63.14% |
| AUC Score (Train): 0.722641 | AUC Score (test): 0.602422 |
| Train KS: 0.4452821152343893 | Test KS: 0.204843264797053 |