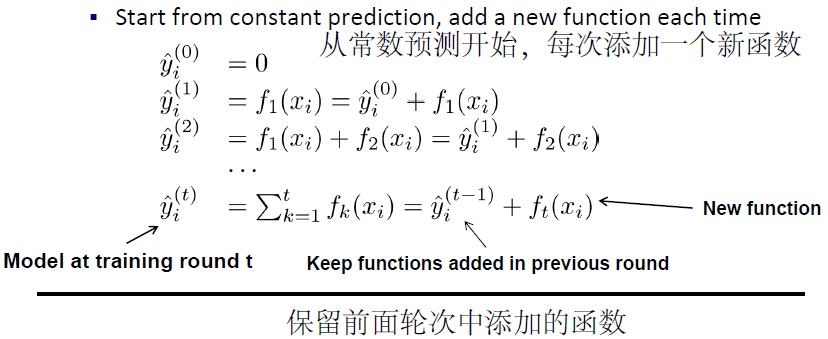
# 我们的目标函数是

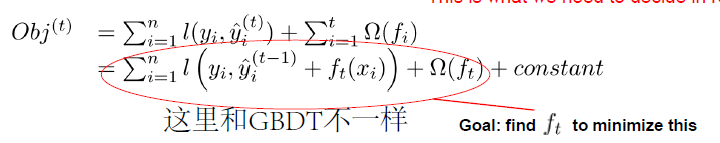


代价函数（每个损失的和）+正则化项（k棵树的复杂度的和）

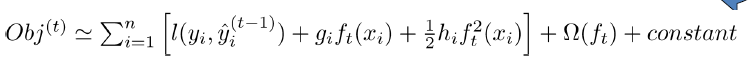
我们想让目标函数最小，但这是一个复杂的优化问题，要使用前向分布学习算法来求解，求解每一棵树不能使用SGD之类的算法，因为我们的模型是树（他不像线性模型），因此要使用加法模型



那么每一步的加法模型的求解就是最小化下图红框里的东西（代价+模型复杂度）

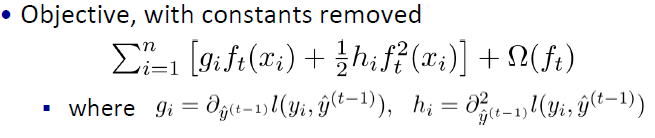


这里作者并没有直接进行最优化，而是先使用泰勒公式对目标函数变形，即：

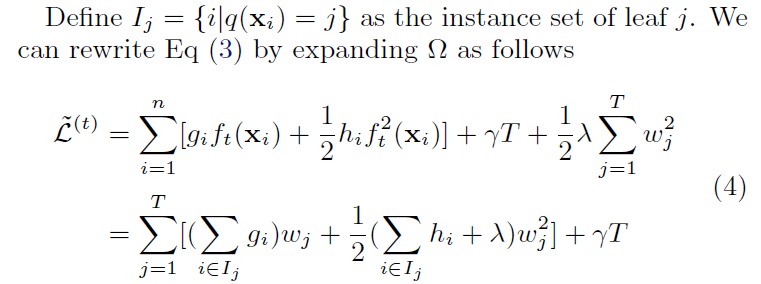


好处是使用二阶近似加快求解，另外也将损失函数抽离出来，以便日后你自定义损失函数，而内部代码无需修改，这是工程上一贯的作风（Second-order approximation can be used to quickly optimize the objective in the **general** setting）

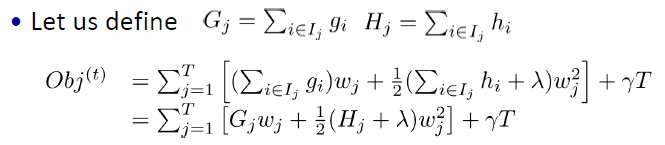
移除目标函数中的常量，变为：



作者将一棵树拆解为权值向量w和树的结构q(x)，并对新树ft(x)做了替代，同时将目标函数由单个样本的层面转到了叶子节点的层面，即n已经变为T了。



接着，作者这里又做了一些定义【注意：论文中的就是ppt中的】

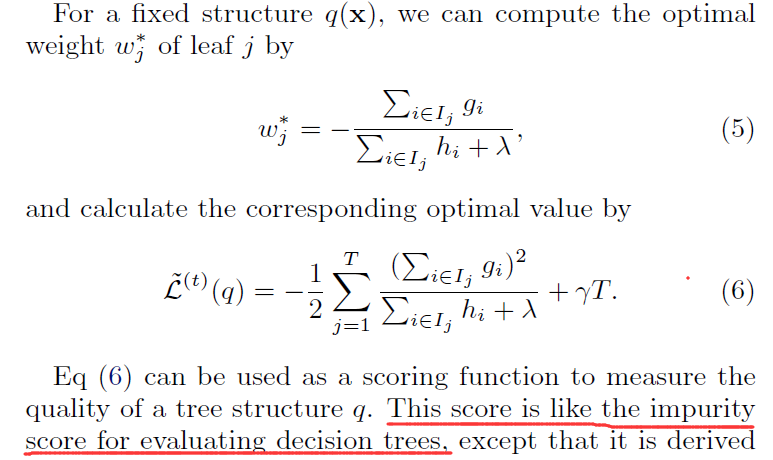


从上图可以看出，只要树的结构被确定，即知道了q(x)【给我一个样本，我可以返回他所在的节点号】，那我的和就是常量了，那么Obj中就只剩一个变量w了，根据初中二次函数的知识，就能得到w\*，还能得到Obj的极小值。

那么我们问题的就归结到确定树结构q(x)的问题上了。

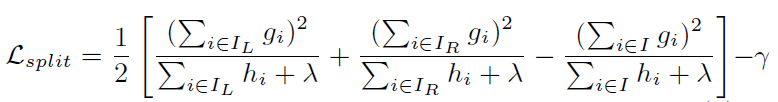
我们的思路是这样的：既然q(x)一经确定就有一个Obj，又希望Obj最小。大家有没有想到决策树构造的过程，根节点不纯度很大，随着节点的划分，都会让不纯度降低。那么在xgboost中，这个Obj就可以看做是不纯度，然后一层层地构建树（即，确定q(x)），从而让Obj变小

其实作者就是这个思路，如下图：



如果这个理解了，下面不就是像决策树一样使用信息增益构建树一样了吗？

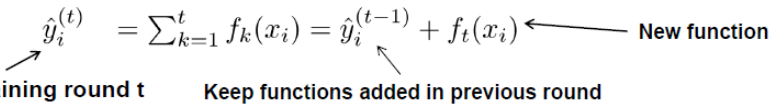
贪心策略为：



最后的r表示：每次构建会多一个叶节点，这个是xgboost独有的，把模型复杂度考虑到贪心指标里了

# 候选解集合不加赘述

以上过程完成后，也就完成了加法模型中一棵树的构建



加入加法模型，继续加入新的树，重复以上操作，直至达到收敛条件或达到k次循环。