概念

集成学习：聚合多个（异质/同制的）分类器的预测，目的是提高准确性。

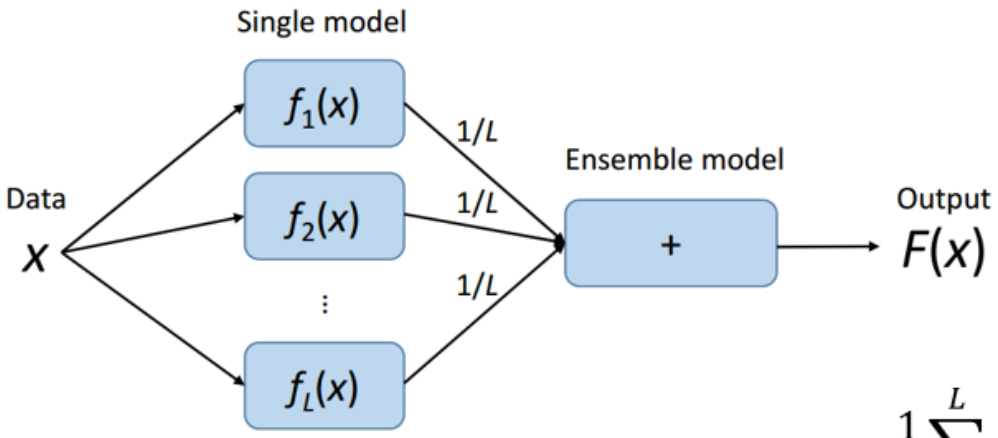
异制与同制

异制集成：是具有一组基本分类器的融合，这些基础分类器由使用不同算法创建的模型组成

同制集成：是指所有基础分类器都属于单一类型（例如，决策树），其模型参数、用于训练的数据或两者的组合不同。

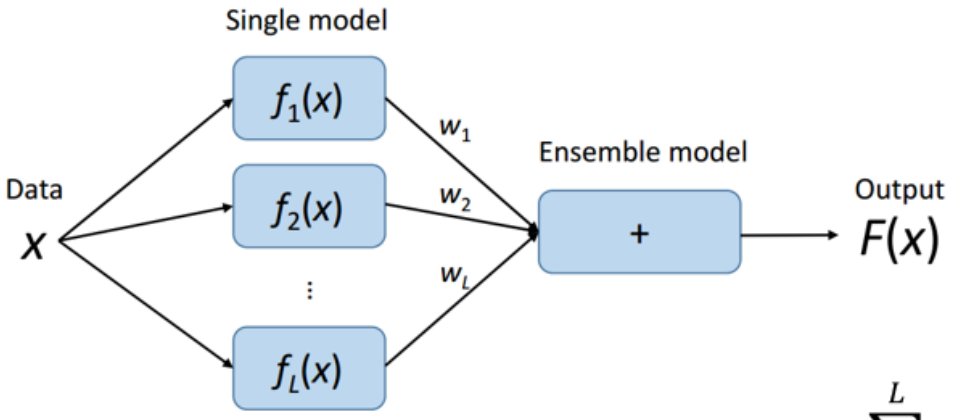
将不同分类器结果整合的方式

平均



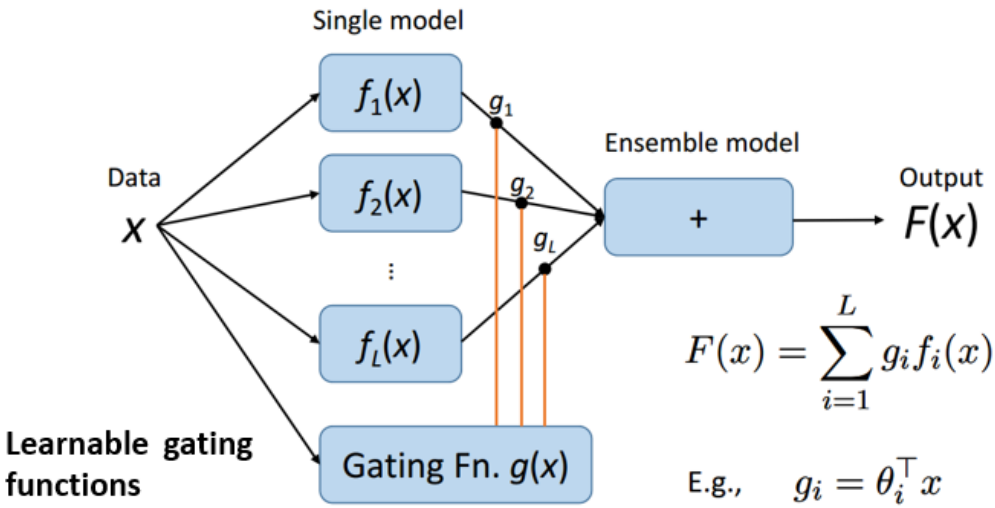
$$F(\boldsymbol{x}) = \frac{1}{L}\sum_{i=1}^{L} f_i(\boldsymbol{x})$$

加权平均



$$F(\boldsymbol{x}) = \sum_{i=1}^{L} w_i f_i(\boldsymbol{x})$$

函数集成



$$F(x) = \sum_{i=1}^{L} g_i f_i(x)$$

E.g., $g_i = \theta_i^{\top} x$

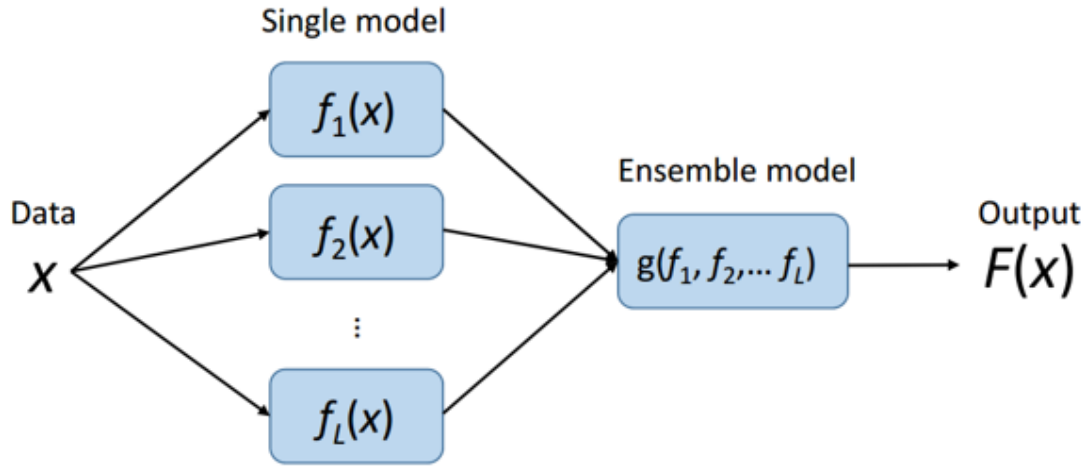
or $g_i = \frac{\exp(\boldsymbol{\theta_i}^{\boldsymbol{T}} \boldsymbol{x})}{\sum_{j=1}^{L} \exp(\boldsymbol{\theta_j}^{\boldsymbol{T}} \boldsymbol{x})}$

Just like linear regression or classification

Note: single model will not be updated when training ensemble model

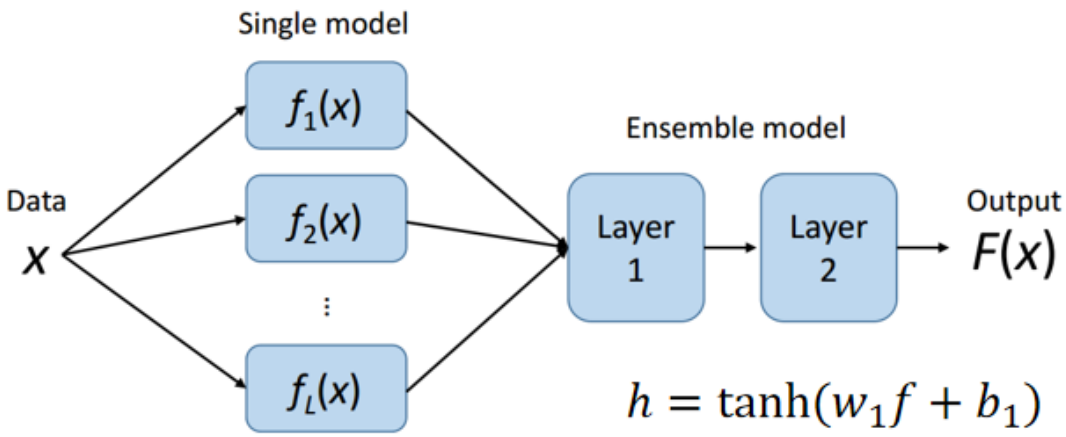
单个模型在集成训练过程中不会变化

叠加



$$F(x) = g(f_1(x), f_2(x), ..., f_L(x))$$

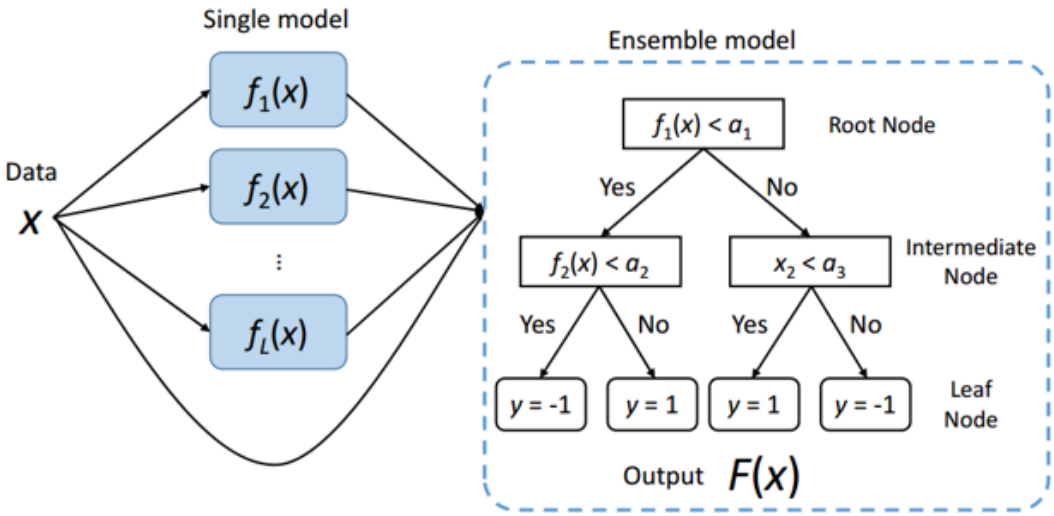
多层



$$h = \tanh(w_1 f + b_1)$$

$$F(x) = \sigma(w_2 h + b_2)$$

树状模型



集成原理

若结果采用多数投票

Suppose we have 5 completely independent classifiers

- If accuracy is 70% for each
  - ✓ $10 \times (0.7^3)\,(0.3^2) + 5 \times (0.7^4)(0.3) + (0.7^5)$
  - ✓ **83.7% majority vote accuracy**
- 101 such classifiers
  - ✓ **99.9% majority vote accuracy**

集成性能

机器学习竞赛通常通过使用数十个模型的模型平均值的方法获胜。

但不是集成学习就可以提高性能

	Test 1	Test 2	Test 3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
Ensemble	√	√	√

(a) Ensemble: Positive Effect.

	Test 1	Test 2	Test 3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
Ensemble	√	√	×

(b) Ensemble: No Effect.

	Test 1	Test 2	Test 3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
Ensemble	×	×	×

(c) Ensemble: Negative Effect.

成功的集成需要有差异性

1. 选择不同的模型
2. 将训练数据分开，使用不同的数据进行训练
3. 模型选取部分特征。选的特征不同

尝试不同的模型，使用不同的数据集，使用不同的特征

Cause of the Mistake	Diversification Strategy
Pattern was difficult任务困难	Try different models
Overfitting过拟合	Vary the training sets
Some features are noisy有部分噪声特征	Vary the set of input features

并行算法

同时训练

Bagging

Booststrap Aggregating Booststrap聚合----**数据层面**

一种通过组合多个模型来减少泛化误差的技术

想法：分别训练几个不同的模型，然后让所有模型对测试示例的输出进行投票

基本原理：不同的模型通常不会在测试集上产生相同的错误。

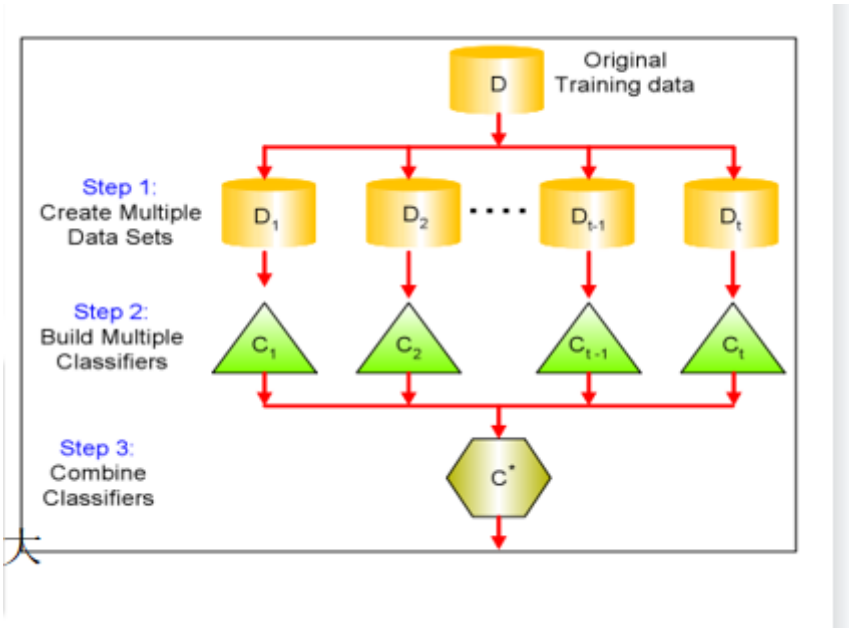
允许多次重复使用相同类型的模型/训练算法/目标函数。

从训练集D中获取重复的引导样本。

Bootstrap采样：给定包含N个训练示例的集合D，依托D随机绘制N个示例创建D'。，**生成Di时是进行的有放回的采样**

按多数票对新实例进行分类

- Bagging:
  - Create k bootstrap samples $D_1,\ldots,D_k$.
  - Train distinct classifier on each D_i.
  - Classify new instance by majority vote.



Di大概包含D的63.2%的数据

- A particular training data has a probability of $1 - 1/N$ not being picked
- Thus its probability of ending up in the test data (not selected) is:

$$(1 - 1/N)^N \approx e^{-1} = 0.368$$

- This means the training data will contain approximately 63.2% of the instances

- Bagging涉及构建k个不同的数据集。
- 每个数据集具有与原始数据集相同数量的示例
- 每个数据集都是通过对原始数据集进行采样和替换来构建的
  - 每个数据集都很有可能缺少原始数据集中的一些示例。
  - 每个数据集还包含几个重复的示例。
  - 平均而言，来自原始数据集的大约2/3的示例可以在生成的训练集中找到。

处理过拟合效果比较好

Random Forest随机森林

CART

建造一个二叉树，在每个节点将数据分到两个孩子节点中

对于回归任务，节点处的预测值是该节点中所有观测值的平均响应变量。

对于分类，预测类是节点中最常见的类（多数投票）。

优点

可以进行分类和回归

自然地处理分类预测因子

计算简单且快速，即使对于大型问题也是如此。

非参数化

可以处理高度非线性的交互和分类边界

如果树非常小的话，容易解释

缺点

准确率：当前的方法，如SVM和集成分类器，通常具有比CART低30%的错误率。------准确率低

不稳定性：如果我们稍微改变一下数据，树的图片就会改变很多。因此，解释并不像看上去那么简单。-----树的形状非常依赖数据

随机森林

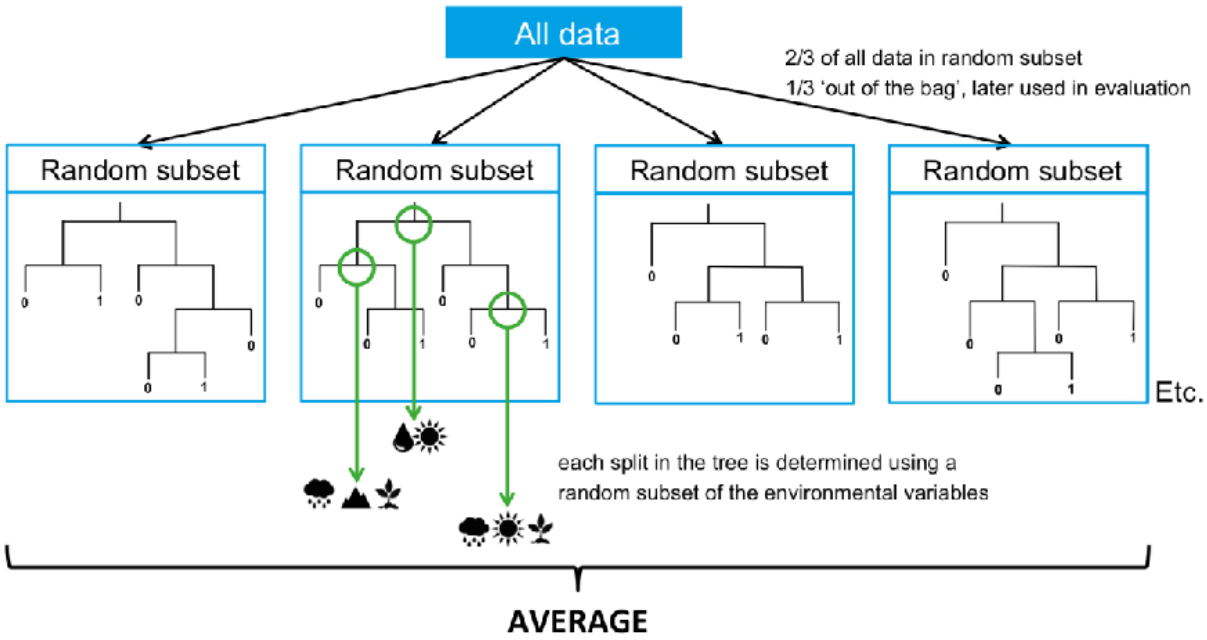
专门为决策树分类器设计的集成方法。

随机森林是对bagging的实质性修改，它建立了**大量不相关**的决策树集合，然后对它们进行平均。

随机性的两个来源：

bagging方法：每棵树，使用bagging获取数据

随机向量方法：在每个节点，从m个属性的随机样本中选择最佳分配，而不是从所有属性中选择



Random Forest Algorithm

1. For b=1 to B:随机森林的数个数
   - Draw a bootstrap sample $\boldsymbol{D}^*$ of size N from the training data
   - Grow a tree T_b from $\boldsymbol{D}^*$, by recursively repeating the following steps
     For each terminal node of the tree, until the minimum node size n_{min} is reached.
     - Select m variables at random from the d variables.
       - Typically $m = log_2 d$
     - Pick the best variable/split-point among the m.
     - Split the node into two daughter nodes
2. Output the ensemble of trees $\{T_b\}_1^B$.

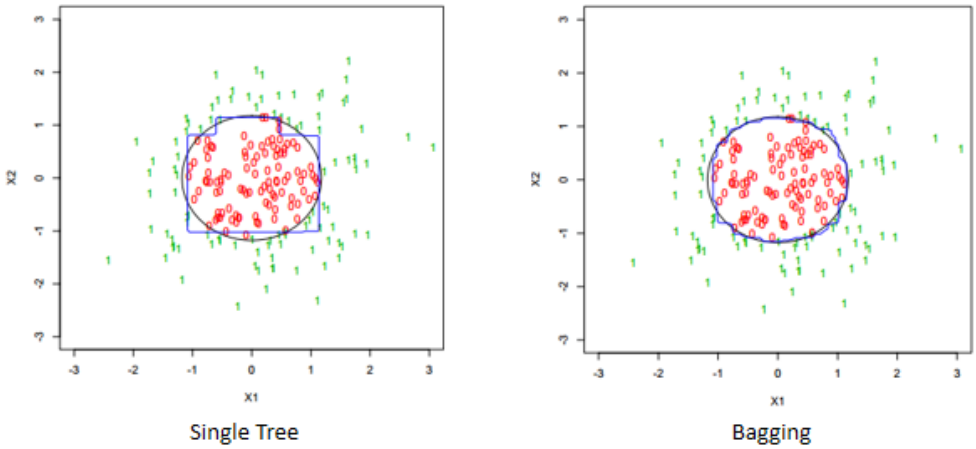
To make a prediction at a new points,

- *Regression*: prediction average $\quad \hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$
- *Classification*: majority voting $\quad \hat{C}_{rf}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B.$

随机森林将不再有可解释性，其他的集成CART

Decision Boundary: Bagging

- Bagging averages many trees, and produces smoother decision boundaries.



分离边界平滑

序列算法

训练后面的模型，用到了之前模型训练的结果

Boosting（Adaboost)提升方法

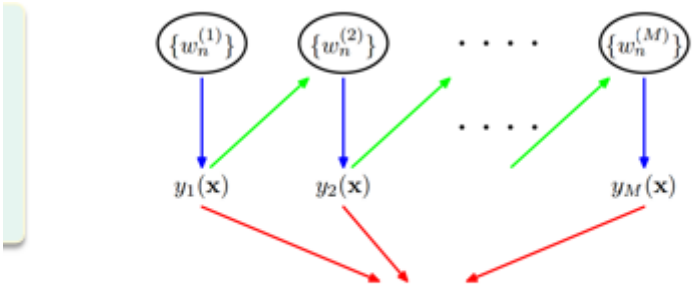
弱可学习：学习的正确率仅仅比随机猜测好

Boosting是一种强大的技术，用于组合多个“基本”分类器，以产生一种委员会形式，其性能可以明显优于任何基本分类器。

即使基本分类器的性能仅略好于随机分类器，Boosting也能给出良好的结果，因此有时基本分类器被称为弱学习器

- 基本分类器**按顺序进**行训练，
- 使用数据集的加权形式来训练每个基本分类器
- 与每个数据点相关联的加权系数取决于先前分类器的性能。---分错的点权重变大

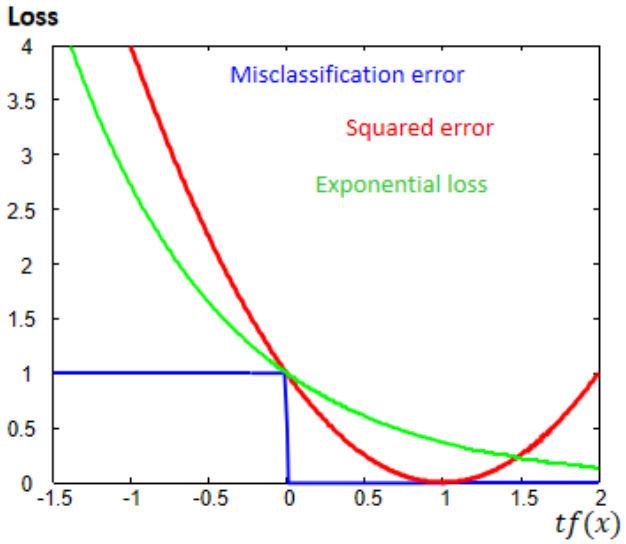
一旦所有分类器都经过训练，它们的**预测就通过加权多数投票方案进行组合。**



Additive model $Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$

AdaBoost

指数损失函数



Squared error

$$J = \sum_{n=1}^{N} (t_n - f(x_n))^2$$

Exponential loss

$$J = \sum_{n=1}^{N} \exp\{-t_n f(x_n)\}$$

指数损失是错误分类损失的单调、光滑的上界。
导致简单的重新加权方案。

Adaboost训练误差是以指数速率下降的

Graident Boosting Decision Trees

梯度提升决策树

Gradient Boosting = Gradient Descent + Boosting

在Adaboost中，shortcomings被高权重的数据点表示

在Gradient中，shortcomings被梯度表示

权重和梯度都是告诉我们如何提升模型性能的

yi-F(xi)叫做残差，平方损失函数中残差等于负梯度

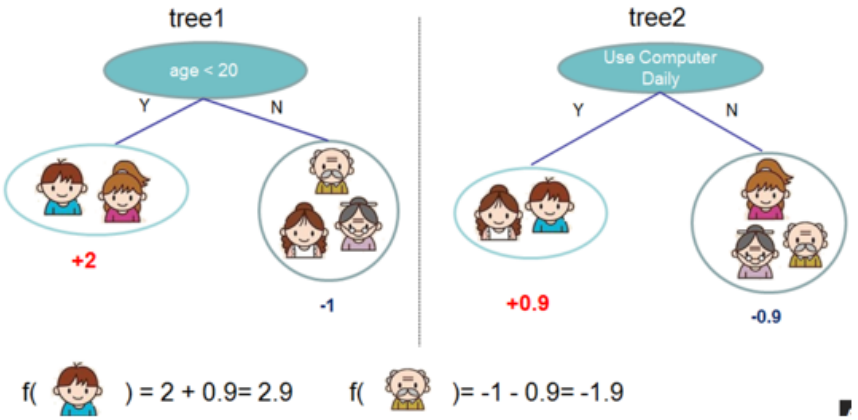
平方损失函数，**太注意噪声点了，过分关注异常值**。尽量在模型中加入异常值。降低整体性能。

树的重定义

我们通过叶中的分数向量和将实例映射到叶的叶索引映射函数来定义树

树的集成

- Usually, a single tree is not strong enough to be used in practice.
- The ensemble model sums the prediction of multiple trees together.
- Look at the example, an important fact is that the two trees try to *complement* each other.两棵树互相补充，集成模型，将w加起来

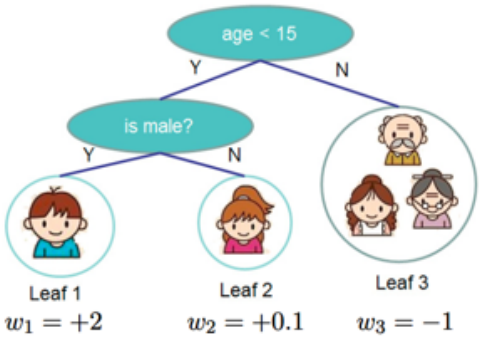


树的复杂度

- We could define the tree complexity as,树的复杂度

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

Number of leaves **L2 norm of leaf scores**



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

对于每个节点，枚举所有特征。

对于每个特征，按特征值对实例进行排序。使用线性扫描来决定沿该功能的最佳拆分。

沿所有特征采用最佳拆分解决方案。

构建树的过程

每次迭代构建一个新树

在每次迭代开始前，计算

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

使用贪心策略，生成树

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

将新建树加入模型中

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

实际上，采用

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

学习率，这意味着我们没有在每一步都进行充分的优化，并为未来的回合保留机会，这有助于防止过度拟合