电子科技大学

University of Electronic Science and Technology of China

Ensemble Learning and

Two Popular Papers

计算机科学与工程学院

高崇铭（*201621060221*）

Prologue

本次研讨课除了XGBoost论文外，我们将分享一些集成学习的基础，再分享的过程中，我们将会提出大量问题供同学们思考、讨论。

由于我知识有限，有些概念可能不准确。对于我们分享过程中，有任何疑问，欢迎同学们打断一起讨论。

Framework

1. Background and Activation
   - 集成学习背景故事
   - 集成学习简介（假设、关注点、种类）
   - 方差与偏差、决策树模型
2. Introduction and discussion of Models
   - Bagging & 随机森林
   - Boosting(AdaBoost, Boosting tree, GBDT, XGBoost)
3. Two Popular Papers
   - XGBoost
   - Deep Forest

浅谈，概念建立 10~20min

深入探讨 40~50min

前沿 10~20min

1. 故事

故事……

南京大学 周志华教授

公认的国内机器学习、数据挖掘最强人。其领导的学术团队LAMDA组为国内机器学习最顶尖组。

翻译引进“集成学习”一词并发扬光大，被称为“集成学习集大成者”。

“深度学习里面有大量的Trick，所以今天来看就有点像老中医在治病一样，虽然能治病，但是什么东西是有用的，什么是没有用的，什么是起副作用的，都不太清楚，笼统地混到一起，有些浑水摸鱼的味道。”

竞赛与工程中的集成学习

在国际、国内各种数据竞赛中，获胜队伍几乎都用上了集成学习模型，保证准确度的同时也提升了防止过拟合的能力。

Python中的sklearn模块提供了集成学习的库，就算不懂集成学习的原理，也能轻易调用相关算法。

GBDT:

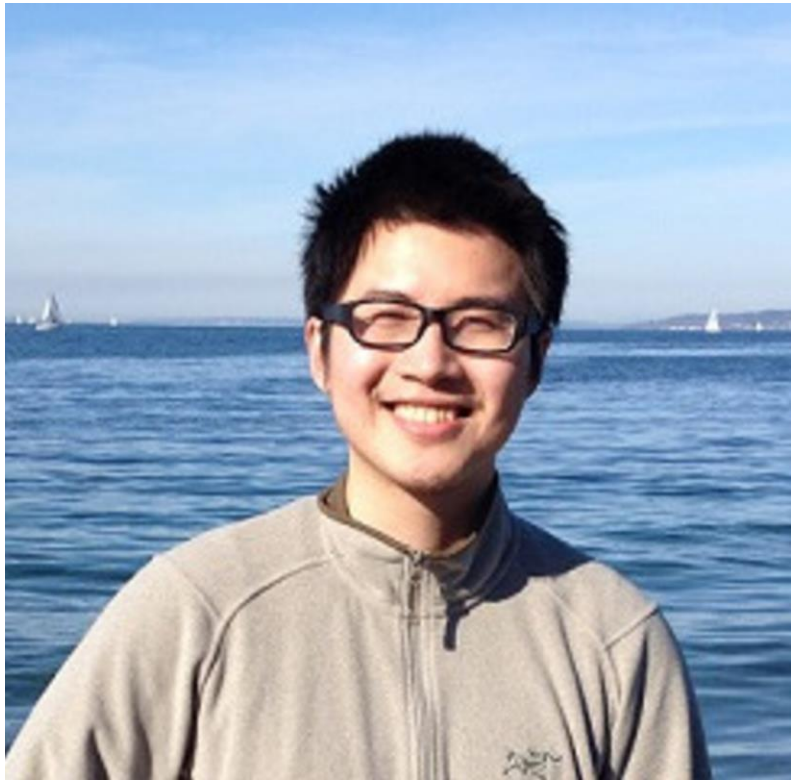
- 工业界如Yahoo、百度、阿里等作为推荐、检索算法
- 2016CFF数据竞赛初赛冠军所用算法

XGBoost: proposed by Tianqi Chen

- 2014.03 : Open-source software library release
- 2015 KDDCUP 前十名均采用该算法
- 2016 KDD 论文发表

故事……

XGBoost 作者

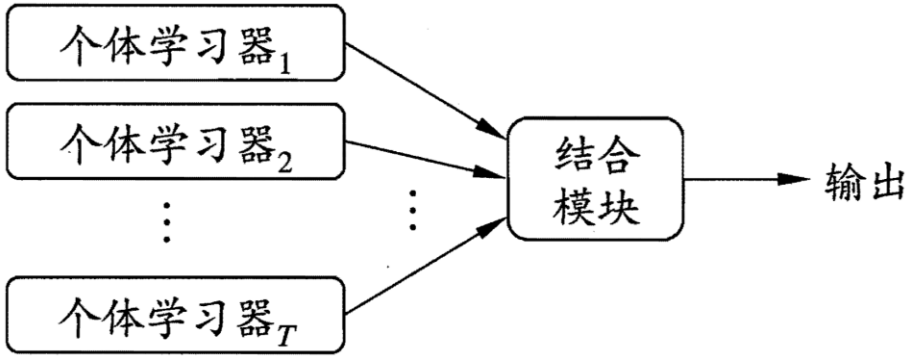


Ph.D. Student at University of Washington

Master and Bachelor student of Shanghai Jiao Tong University, China

思想

本身不是一个单独的机器学习算法，而是通过构建并结合多个机器学习器来完成学习任务。也就是我们常说的“博采众长”。



- AdaBoost 中的加法模型

$$H(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$$

- Bagging 思想
  - ✓ 分类：简单投票
  - ✓ 回归：简单平均

结合原则：“好而不同”

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集成	√	√	√

(a) 集成提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

(b) 集成不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

(c) 集成起负作用

- 准确性：学习器至少要比随机猜测结果好
- 差异性：学习器之间应该有差异

问题

准确性和差异性可以同时满足吗？

问题

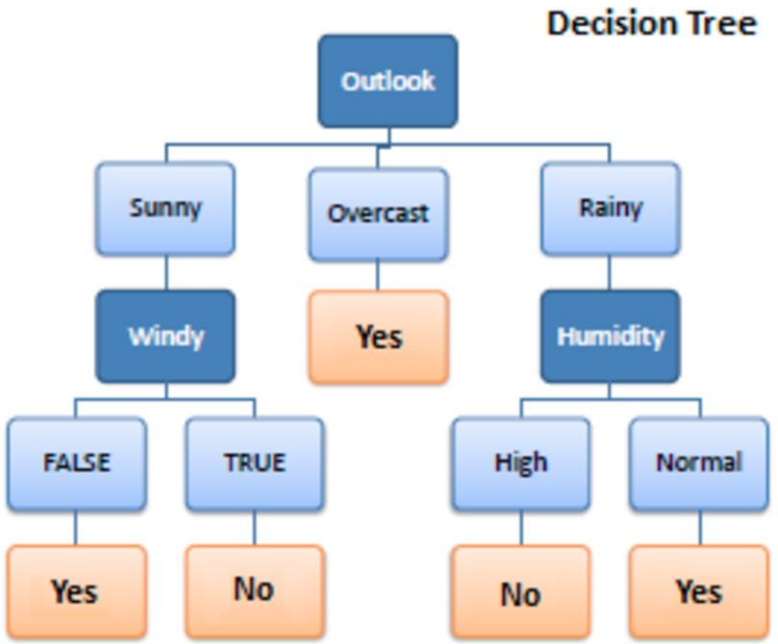
准确性和差异性可以同时满足吗？

事实上，个体学习器的“准确性”与“多样性”本身就存在冲突。一般的，准确性很高之后，要增加多样性就要牺牲准确性。事实上，如何产生并结合“好而不同”的个体学习器，恰是集成学习研究核心

Classification Tree

Predictors | Target

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



基本思想：通过对属性集中某一属性进行划分，使得样本集在划分后尽量被分类开。

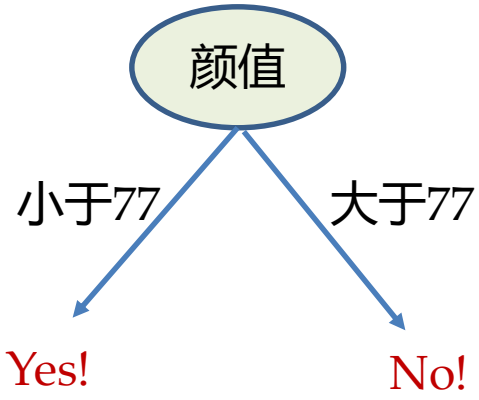
http://www.saedsayad.com/decision_tree.htm

Classification Tree

features　　labels

	颜值	成绩	钱	要不要
1号	43	87	富	no
2号	76	78	富	no
3号	78	90	富	yes
4号	90	68	穷	yes

instances



问题

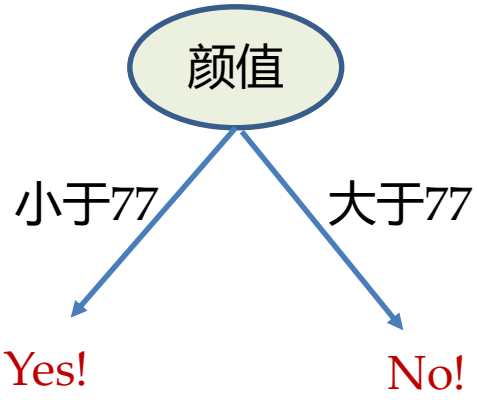
1. 如何衡量划分后的结果好坏，即分类结果的纯度？
2. 对于数值型feature，如何选择划分点，策略是什么？

Classification Tree

features　　label

	颜值	成绩	钱	要不要
1号	43	87	富	no
2号	76	78	富	no
3号	78	90	富	yes
4号	90	68	穷	yes

instances



信息熵

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \ .$$

ID3 划分原则

$$\min \ \mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Ent}(D^v) \ .$$

基尼指数

$$Gini(D) = 1 - \sum_{i}^{n} p_i^2$$

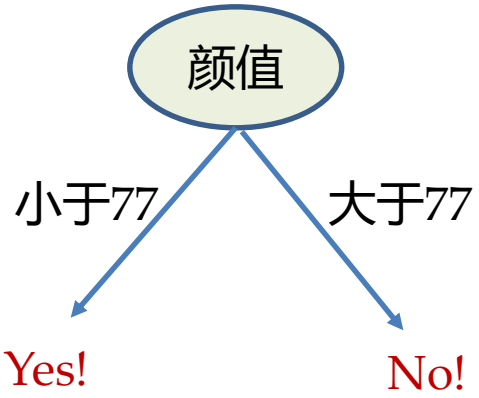
CART 划分原则

$$\min \ \mathrm{Gini\_index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Gini}(D^v) \ .$$

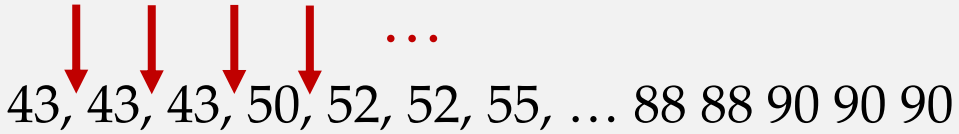
Boosting

小问题：属性**划分点查找**策略

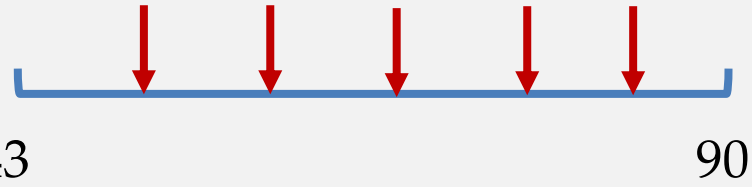
	颜值	成绩	钱	要不要
1号	43	87	富	no
2号	76	78	富	no
3号	78	90	富	yes
4号	90	68	穷	yes
…	…	…	…	…
n号	88	38	穷	yes



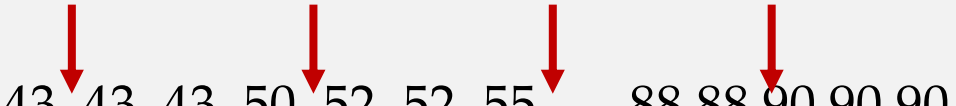
3.1 Basic Exact Greedy Algorithm



3.2 Approximate Algorithm



3.3 Weighted Quantile Sketch

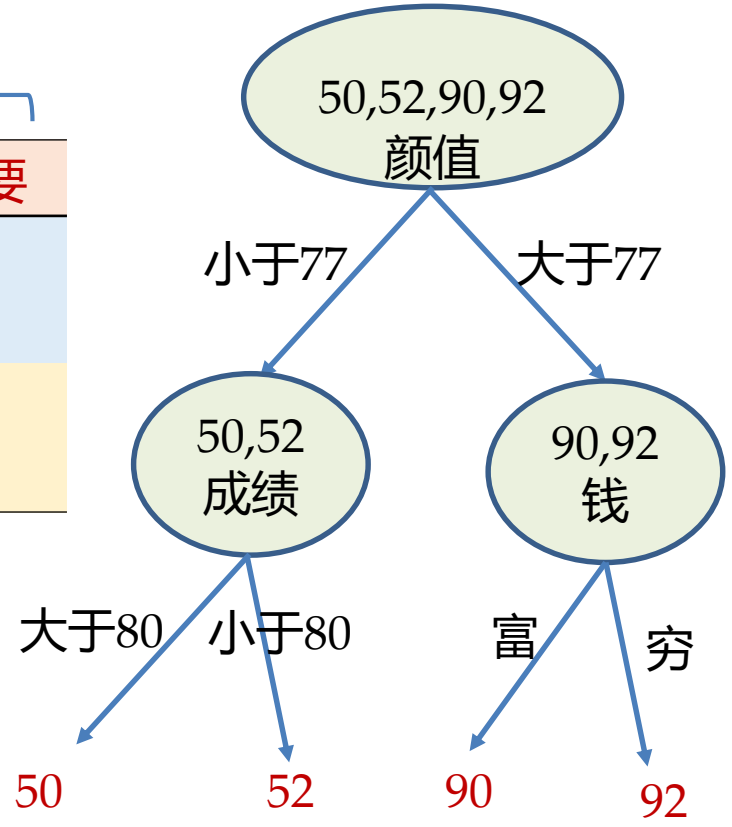


Regression Tree

features　　label

	颜值	成绩	钱	要不要
1号	43	87	富	50
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92

instances



划分原则：Least Square Deviation

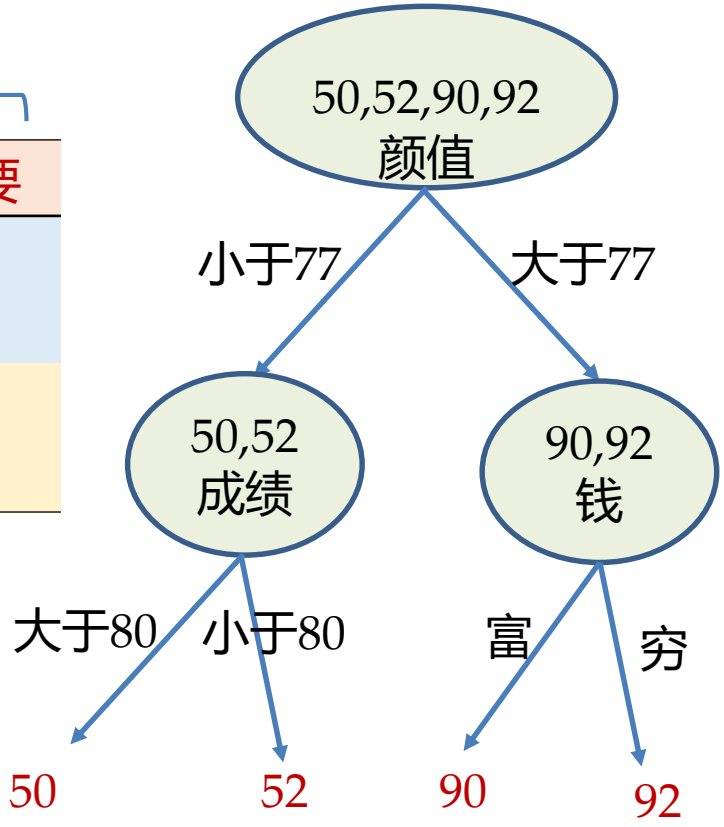
$$i(t)=\frac{\sum_{n\in\hbar(t)} w_n f_n (y_n-\bar{y}(t))^2}{\sum_{n\in\hbar(t)} w_n f_n}$$

Regression Tree

features　　label

	颜值	成绩	钱	要不要
1号	43	87	富	50
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92

instances



问题

为什么要剪枝？

如何剪枝？

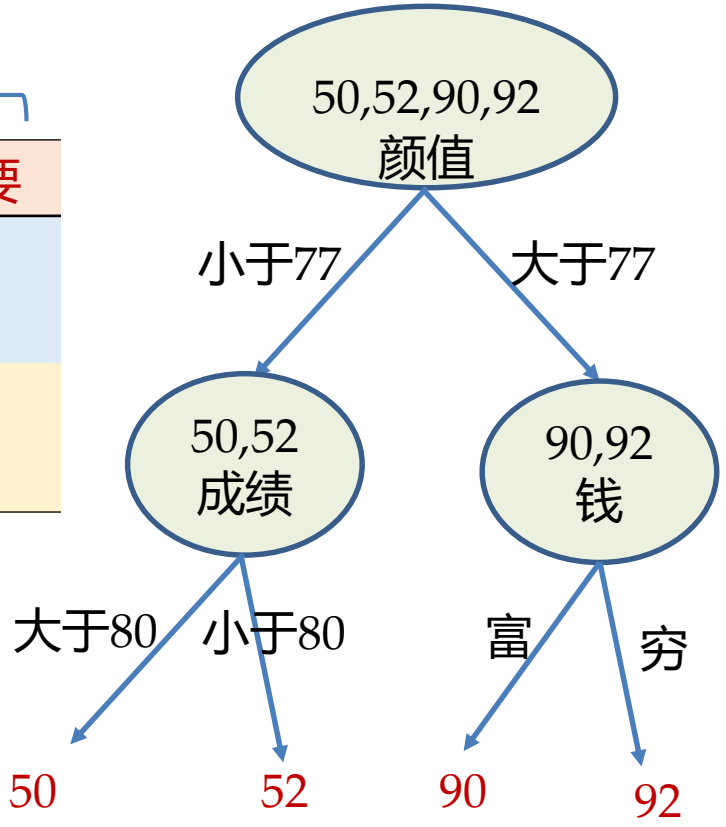
Decision Tree Learning

Regression Tree

features　　label

instances

	颜值	成绩	钱	要不要
1号	43	87	富	50
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92



问题

为什么要剪枝？

如何剪枝？

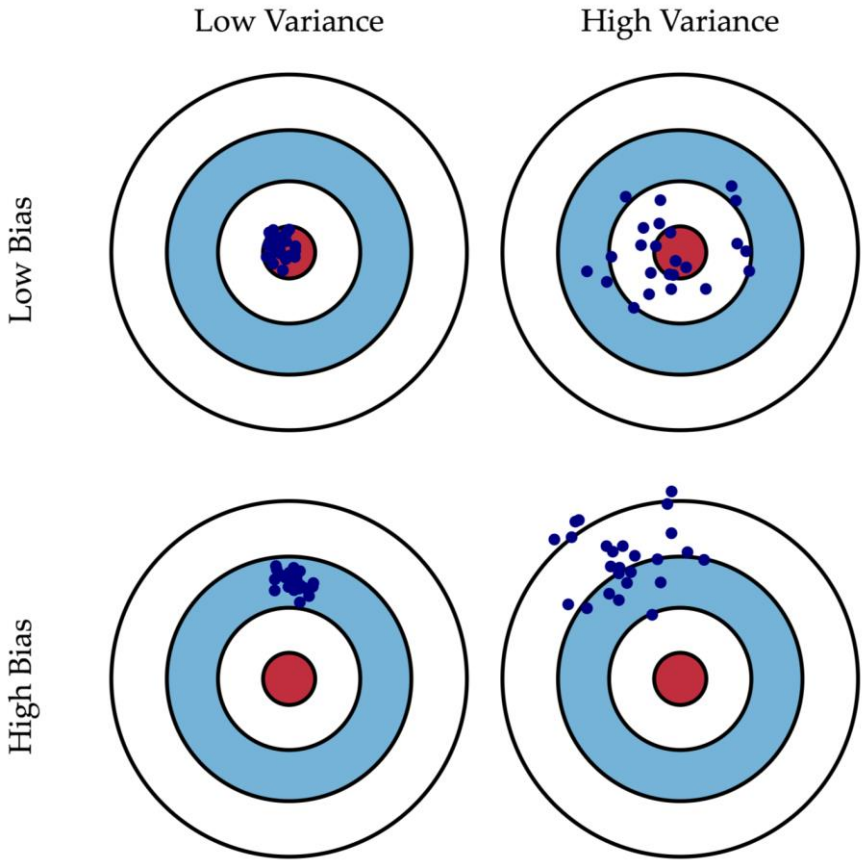
动机：减小模型复杂度，减小过拟合！

策略：1. 先剪枝、2. 后剪枝

Bias-Variance Decomposition

$$E\left[\left(y_i - \hat{f}(\mathbf{x}_i)\right)^2\right] = Var\{noise\} + bias^2\left\{\hat{f}(\mathbf{x}_i)\right\} + variance\left\{\hat{f}(\mathbf{x}_i)\right\}$$

- **bias** 描述的是算法依靠自身能力进行预测的平均准确程度
- **variance** 则度量了算法在不同训练集上表现出来的差异程度



Two Category of Ensemble models

- 个体models间存在强依赖关系，必须串行生成的序列化方法
- 目标在于减小模型间的偏差

✓ **Boosting**

1. AdaBoost
2. Boosting Tree(拟合残差)
3. GBDT(拟合残差的一阶导)
4. XGBoost(拟合残差带一阶导、二阶导)

问题

XGBoost在论文中是可以并行的啊，
为啥这里分到串行模型里？

- 个体models间不存在强依赖关系，可以同时生成的并行化方法
- 目标在于减小模型间方差

✓ **Bagging**

✓ **Random Forest**

（也减小偏差）

Framework

1. Background and Activation
   - 集成学习背景故事
   - 集成学习简介（假设、关注点、种类）
   - 方差与偏差、决策树模型

2. Introduction and discussion of Models
   - Bagging & 随机森林
   - Boosting(AdaBoost, Boosting tree, GBDT, XGBoost)

3. Two Popular Papers
   - XGBoost
   - Deep Forest

浅谈，概念建立 10~20min

深入探讨 20~30min

前沿 10~20min

2. 模型

Bagging & Random Forest 原理

Bagging 原理

不同的个体弱学习器的训练集是通过随机采样得到的。分别训练后，综合结果。

- ✓ 分类：简单投票
- ✓ 回归：简单平均

	颜值	成绩	钱	要不要
1号	43	87	富	50
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92

模型1数据集

	颜值	成绩	钱	要不要
1号	43	87	富	50
2号	76	78	富	52
4号	90	68	穷	92

模型2数据集

	颜值	成绩	钱	要不要
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92

……

Random Forest 原理

不同的个体弱学习器的训练集和特征集是通过随机采样得到的。分别训练后，综合结果。

其中：个体学习器为决策树

	颜值	成绩	钱	要不要
1号	43	87	富	50
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92

模型1数据集

	颜值	成绩	要不要
1号	43	87	50
2号	76	78	52
4号	90	68	92

模型2数据集

	颜值	钱	要不要
2号	76	富	52
3号	78	富	90
4号	90	穷	92

……

随机森林特点

问题

1. 随机森林的优缺点有哪些？

随机森林特点

问题

1. 随机森林的优缺点有哪些？

优点

1、 在当前的很多数据集上，相对其他算法有着很大的优势，表现良好

2、它能够处理很高维度（feature很多）的数据，并且不用做特征选择

PS：特征子集是随机选择的

3、在训练完后，它能够给出哪些feature比较重要

PS：http://blog.csdn.NET/keepreder/article/details/47277517

4、在创建随机森林的时候，对generalization error使用的是无偏估计，模型泛化能力强

5、训练速度快，容易做成并行化方法（训练时树与树之间是相互独立的）

6、 在训练过程中，能够检测到feature间的互相影响

7、 对于不平衡的数据集来说，它可以平衡误差。

8、如果有很大一部分的特征遗失，仍可以维持准确度。

缺点：

1、随机森林已经被证明在某些噪音较大的分类或回归问题上会过拟

2、对于有不同取值的属性的数据，取值划分较多的属性会对随机森林产生更大的影响，所以随机森林在这种数据上产出的属性权值是不可信的。

Ref：http://blog.csdn.net/keepreder/article/details/47273297

Two Category of Ensemble models

- 个体models间存在强依赖关系，必须串行生成的序列化方法
- 目标在于减小模型间的偏差

✓ **Boosting**

1. AdaBoost
2. Boosting Tree(拟合残差)
3. GBDT(拟合残差的一阶导)
4. XGBoost(拟合残差带一阶导、二阶导)

问题

XGBoost在论文中是可以并行的啊，
为啥这里分到串行模型里？

- 个体models间不存在强依赖关系，可以同时生成的并行化方法
- 目标在于减小模型间方差

✓ **Bagging**

✓ **Random Forest**

（也减小偏差）

Boosting

Adaboost 思想（数据挖掘 十大算法之一）

重复N次分类、预测，每次根据上一次的结果：

1. 提高被错分样本的权重
2. 提高之前分类正确模型的权重

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
7: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$
8: **end for**

输出： $H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

这个值怎么来的？

先凑、后有人用指数函数作为loss function推导而出

模型权值

样本权值

Ref：https://en.wikipedia.org/wiki/Gradient_boosting

Boosting

- Adaboost（数据挖掘 十大算法之一）

问题

1. Adaboost能分类，能做回归吗？
2. Adaboost作为Boosting的代表算法，其强大之处在哪里？

Adaboost（数据挖掘 十大算法之一）

问题

1. Adaboost能分类，能做回归吗？
2. Adaboost作为Boosting的代表算法，其强大之处在哪里？

1. Adaboost局限之处就在于只能做分类。
2. 在一轮一轮的迭代中，通过关注错误样本而改变数据分布，故引入模型间的“差异性”，同时提高效果好的模型而增大了“正确性”，好而不同！

小总结

问题

1. 解释Adaboost降低偏差而不降低方差
2. 解释Bagging降低方差而不降低偏差
3. 解释Random Forest同时减低偏差和方差

Boosting

小总结

问题

1. 解释Adaboost降低偏差而不降低方差
2. 解释Bagging降低方差而不降低偏差
3. 解释Random Forest同时减低偏差和方差

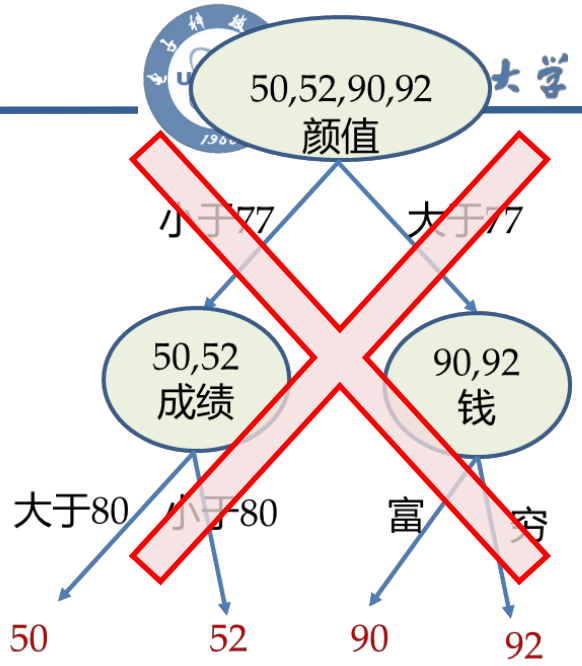
- 若各子模型独立，则有$Var\left(\frac{\sum X_i}{n}\right) = \frac{Var(\sum X_i)}{n}$，此时可以显著降低variance；
- 若其强线性相关，则有$Var\left(\frac{\sum X_i}{n}\right) = Var(\sum X_i)$，此时完全没有降低variance;

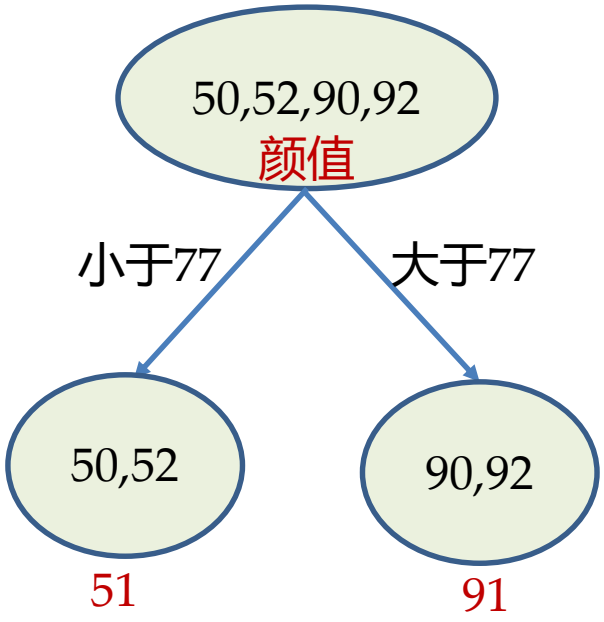
1. **Adaboost**是串行地最小化损失函数，其bias自然逐步下降。但由于是采取这种sequential、adaptive的策略，各子模型之间是强相关的，于是子模型之和并不能显著降低variance。
2. **Bagging** 的各模型有一些相关，一定程度降低variance.而Bagging各模型间用的是原数据集的抽样，都没有降低bias
3. **Random Forest**详见ESL p588公式15.1

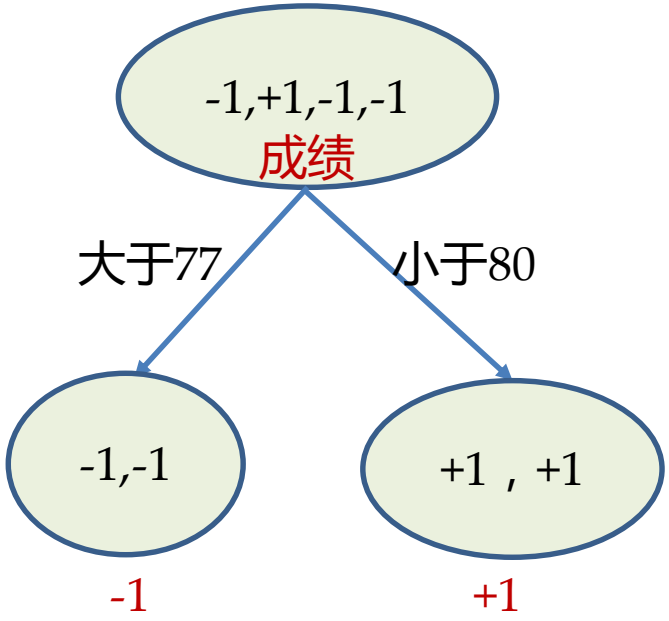
Boosting

Boosting Tree

	features			label
	颜值	成绩	钱	要不要
instances				
1号	43	87	富	50
2号	76	78	富	52
3号	78	90	富	90
4号	90	68	穷	92







核心思想：
后面的树拟合前面的树的残差！

1号：51 – 1 = 50
2号：51 + 1 = 52
3号：91 – 1 = 90
4号：92 + 1 = 92

Gradient boosting

核心思想：

后面的树拟合前面的树的**残差**！

第m步的预测值

当前树的输出值

第m+1步的预测值

真实值

$$F_{m+1}(x) = F_m(x) + h(x) = y$$

自选的Loss function

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma),$$

$$F_m(x) = F_{m-1}(x) + \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + f(x_i)),$$

解得$h(x) = f^*(x_i)$

Gradient Boosting Tree

问题

既然效果相同，为什么我们要用Boosting Tree这种多颗树的结构，而不是决策树那样一棵树？

Gradient Boosting Tree

问题

既然效果相同，为什么我们要用Boosting Tree这种多颗树的结构，而不是决策树那样一棵树？

减小模型复杂度，减小过拟合！

Gradient Boosting Tree

问题

树结构的Gradient boosting怎么体现Boosting？

Gradient Boosting Tree

问题

树结构的Gradient boosting怎么体现Boosting？

拟合残差，某种意义上也是对样本的 “欠处理程度” 的一种关注。这道理和Adaboost一样，也就是Boosting的思想核心。

比如：产品设计中，总是先解决60%用户的需求凑合着，再解决35%用户的需求，最后才关注那5%人的需求。

Gradient Boosting Tree

问题

树结构的算法，比起普通的强在哪里？

Gradient Boosting Tree

问题

树结构的算法，比起普通的强在哪里？

优势

- ✓ Learn higher order interaction between features.
- ✓ Can be scalable, used in Industry, e.g. BAT、Google、Yahoo…
- ✓ Invariant to scaling of inputs, so you do not need to do careful features normalization.

GBDT(Gradient Boosting Decision Tree)

核心思想：

后面的树拟合前面的树的**残差的梯度**（伪残差）！

$$F_m(x) = F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)),$$

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) - \gamma \frac{\partial L(y_i, F_{m-1}(x_i))}{\partial F_{m-1}(x_i)}\right)$$

GBDT (Gradient Boosting Decision Tree)

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Framework

1. Background and Activation
   - 集成学习背景故事
   - 集成学习简介（假设、关注点、种类）
   - 方差与偏差、决策树模型
2. Introduction and discussion of Models
   - Bagging & 随机森林
   - Boosting(AdaBoost, Boosting tree, GBDT, XGBoost)
3. Two Popular Papers
   - XGBoost
   - Deep Forest

浅谈，概念建立 10~20min

深入探讨 20~30min

前沿 10~20min

3. 前沿探讨

Chen T, Guestrin C. Xgboost，SIGKDD 2016

XGBoost: A Scalable Tree Boosting System

Tianqi Chen
University of Washington
tqchen@cs.washington.edu

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT

Tree boosting is a highly effective and widely used machine learning method. In this paper, we describe a scalable end-to-end tree boosting system called XGBoost, which is used widely by data scientists to achieve state-of-the-art results on many machine learning challenges. We propose a novel sparsity-aware algorithm for sparse data and weighted quantile sketch for approximate tree learning. More importantly, we provide insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. By combining these insights, XGBoost scales beyond billions of examples using far fewer resources than existing systems.

Keywords

Large-scale Machine Learning

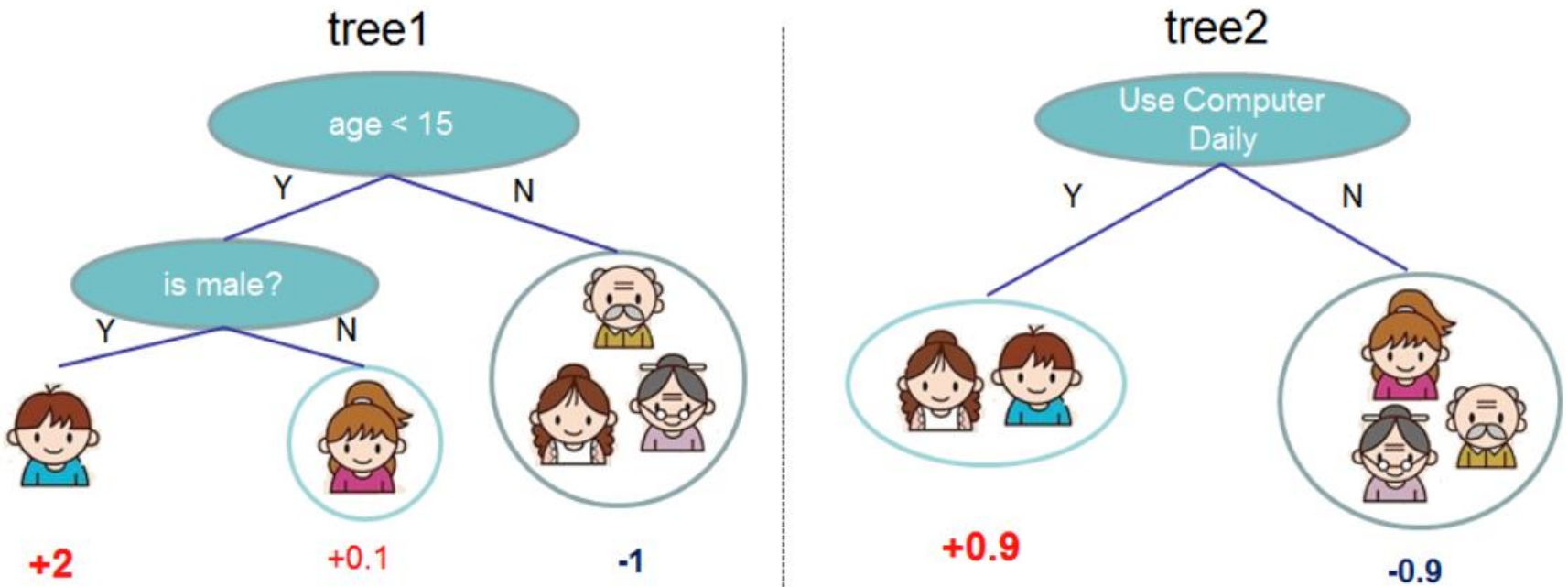
1. INTRODUCTION

Machine learning and data-driven approaches are becoming very important in many areas. Smart spam classifiers

problems. Besides being used as a stand-alone predictor, it is also incorporated into real-world production pipelines for ad click through rate prediction [15]. Finally, it is the de-facto choice of ensemble method and is used in challenges such as the Netflix prize [3].

In this paper, we describe XGBoost, a scalable machine learning system for tree boosting. The system is available as an open source package[2]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges. Take the challenges hosted by the machine learning competition site Kaggle for example. Among the 29 challenge winning solutions [3] published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top-10. Moreover, the winning teams reported that ensemble methods outperform a well-configured XGBoost by only a small

Question: Does the person like computer games?



f( ) = 2 + 0.9= 2.9 f( )= -1 - 0.9 = -1.9

Model Overview

损失函数 $\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$

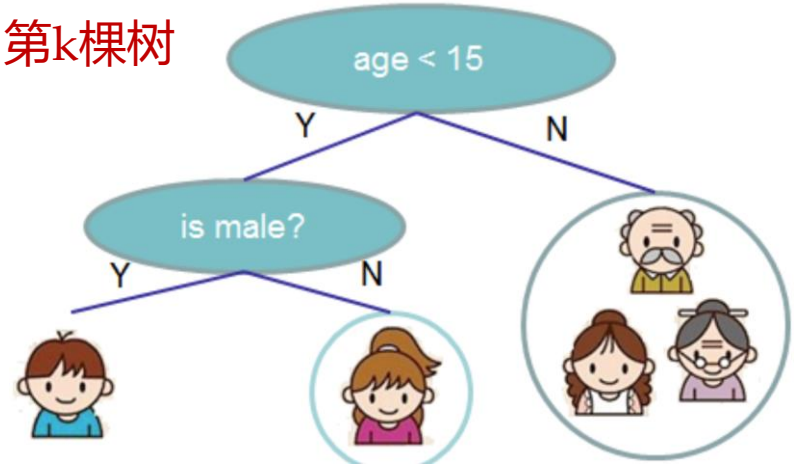
Additive function(K棵树)

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F},$$

$$\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$$

第i个样本的输出

第k棵树



$f_k(x_1) = \omega_1$ $f_k(x_2) = \omega_2$ $f_k(x_3) = f_k(x_4) = f_k\ (x_5) = \omega_3$

细化：

1. 原始Loss function

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

2. t 时刻的Loss function(第t棵树)

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(\mathbf{x}_i)) + \Omega(f_t)$$

3. 二阶泰勒展开

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ are first and second order gradient statistics on the loss func-

4. 去除常数项

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

细化：

4. 去除常数项

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \Omega(f_t)$$

5. 将每个样本的$f_t(x_i)$与正则项Ω关于树的参数ω展开，得到最终Loss Function

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

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

1. 关于参数ω求最优值：

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda},$$

2. 带入最终Loss Function得树的最优值：

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T.$$

1. 关于参数ω求最优值：

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

2. 带入最终Loss Function得树的最优值：

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

问题

这两个等式分别是什么？

1. 关于参数ω求最优值：

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

2. 带入最终Loss Function得树的最优值：

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

问题

这两个等式分别是什么？

1. 回归树叶节点的输出

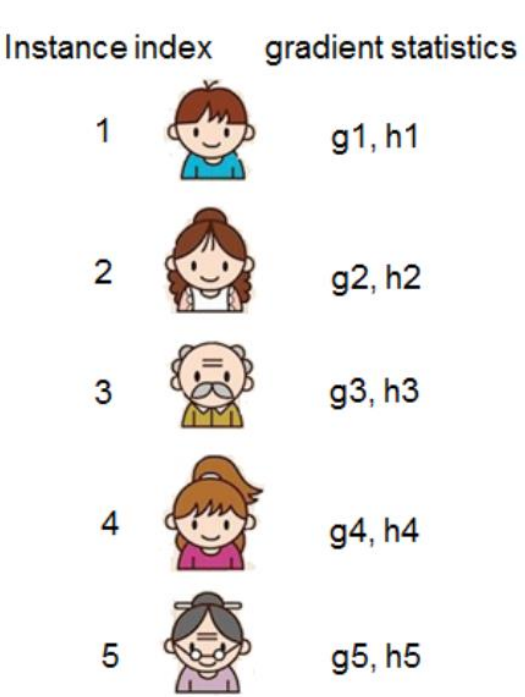
2. 回归树的最优划分准则(类比信息熵、Gini指数)

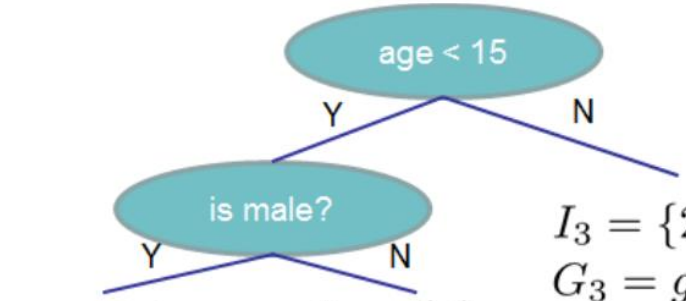
1. 关于参数ω求最优值：

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda},$$

2. 带入最终Loss Function得树的最优值：

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$





$I_1 = \{1\}$
$G_1 = g_1$
$H_1 = h_1$
$f_k(x_1) = \omega_1$

$I_2 = \{4\}$
$G_2 = g_4$
$H_4 = h_4$
$f_k(x_4) = \omega_2$

$I_3 = \{2, 3, 5\}$
$G_3 = g_2 + g_3 + g_5$
$H_3 = h_2 + h_3 + h_5$
$f_k(x_2) = f_k(x_3) = f_k\ (x_5) = \omega_3$

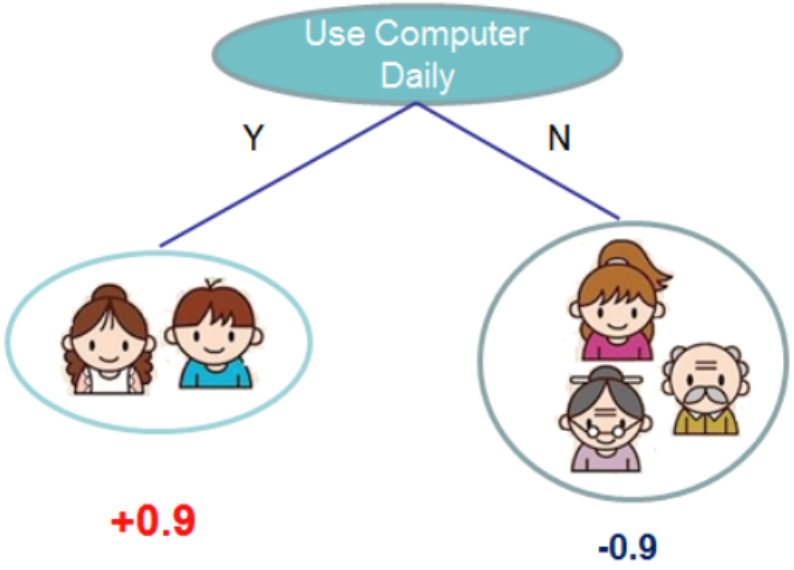
$$Obj = -\sum_j \frac{G_j^2}{H_j+\lambda} + 3\gamma$$

The smaller the score is, the better the structure is

➢ 寻找属性并及属性上的划分点

寻找划分点准则(类比信息增益)

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$



Tree Dividing Principles

信息熵

$$\mathrm{Ent}(D) = -\sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k \ .$$

ID3 划分原则

$$\min \ \mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Ent}(D^v) \ .$$

基尼指数

$$Gini(D) = 1 - \sum_{i}^{n} p_i^2$$

CART 划分原则

$$\min \ \mathrm{Gini\_index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Gini}(D^v) \ .$$

Least Square Deviation

$$i(t) = \frac{\sum_{n\in\hbar(t)} w_n f_n (y_n - \bar{y}(t))^2}{\sum_{n\in\hbar(t)} w_n f_n}$$

最小方差划分原则

$$\sum_{i:x_j\le s} (y_i - \hat{c}_1)^2 + \sum_{i:x_j>s} (y_i - \hat{c}_2)^2$$

XGBoost 的优化项：

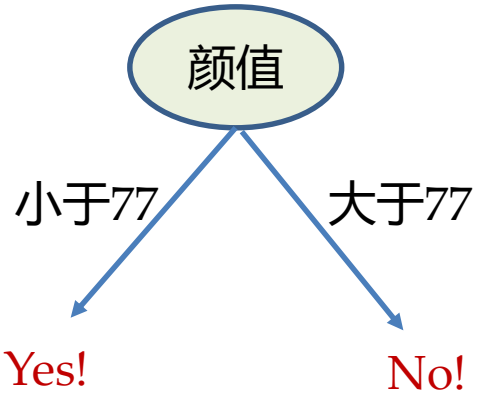
$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T.$$

XGBoost 的划分原则：

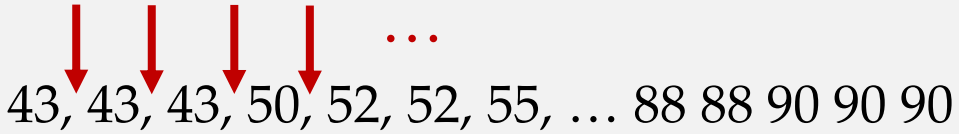
$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

小问题：属性**划分点查找**策略

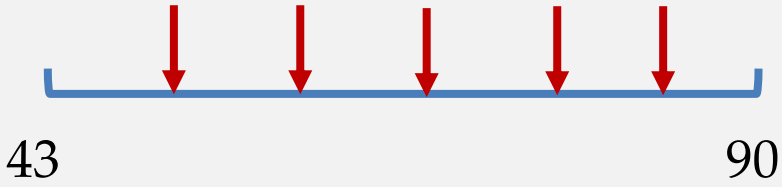
	颜值	成绩	钱	要不要
1号	43	87	富	no
2号	76	78	富	no
3号	78	90	富	yes
4号	90	68	穷	yes
…	…	…	…	…
n号	88	38	穷	yes



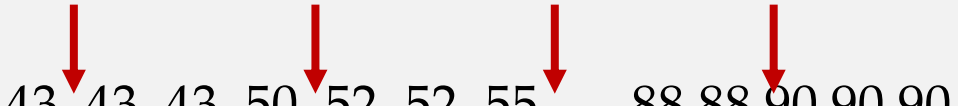
3.1 Basic Exact Greedy Algorithm



3.2 Approximate Algorithm



3.3 Weighted Quantile Sketch



3.3 Weighted Quantile Sketch

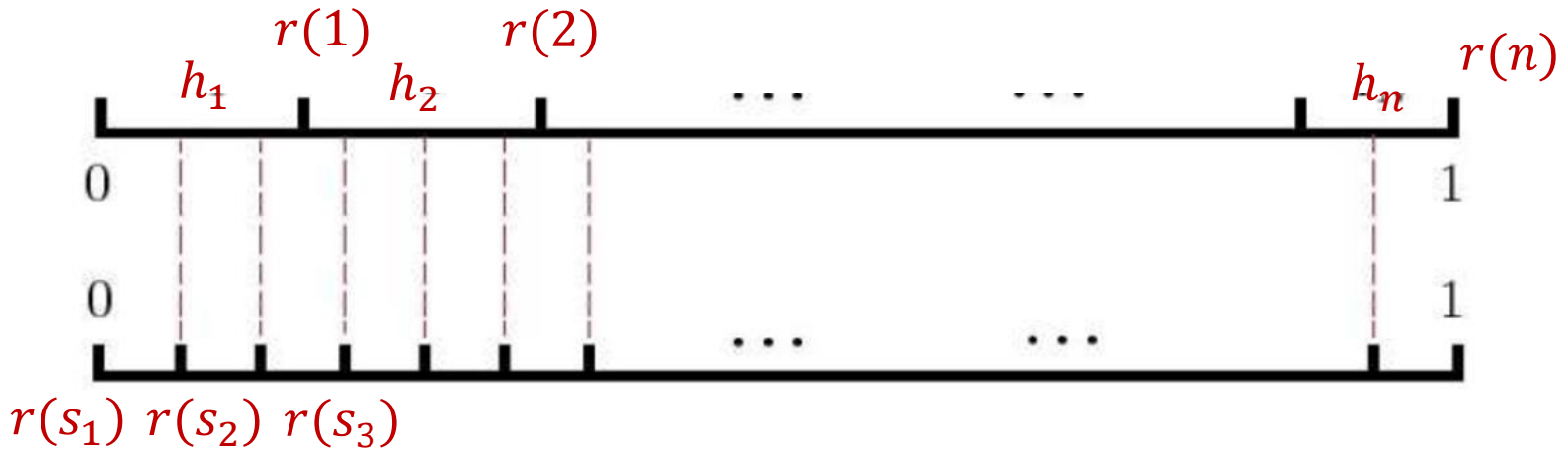
statistics of each training instances. We can define a rank functions $r_k : \mathbb{R} \to [0, +\infty)$ as

$$r_k(z) = \frac{1}{\sum_{(x,h)\in\mathcal{D}_k} h} \sum_{(x,h)\in\mathcal{D}_k, x<z} h, \tag{8}$$

which represents the proportion of instances whose feature value k is smaller than z. The goal is to find candidate split points $\{s_{k1}, s_{k2}, \cdots s_{kl}\}$, such that

$$|r_k(s_{k,j}) - r_k(s_{k,j+1})| < \epsilon, \quad s_{k1} = \min_i \mathbf{x}_{ik}, s_{kl} = \max_i \mathbf{x}_{ik}. \tag{9}$$

Here ϵ is an approximation factor. Intuitively, this means that there is roughly $1/\epsilon$ candidate points. Here each data



XGBoost

小问题：理解XGBoost中的**并行**

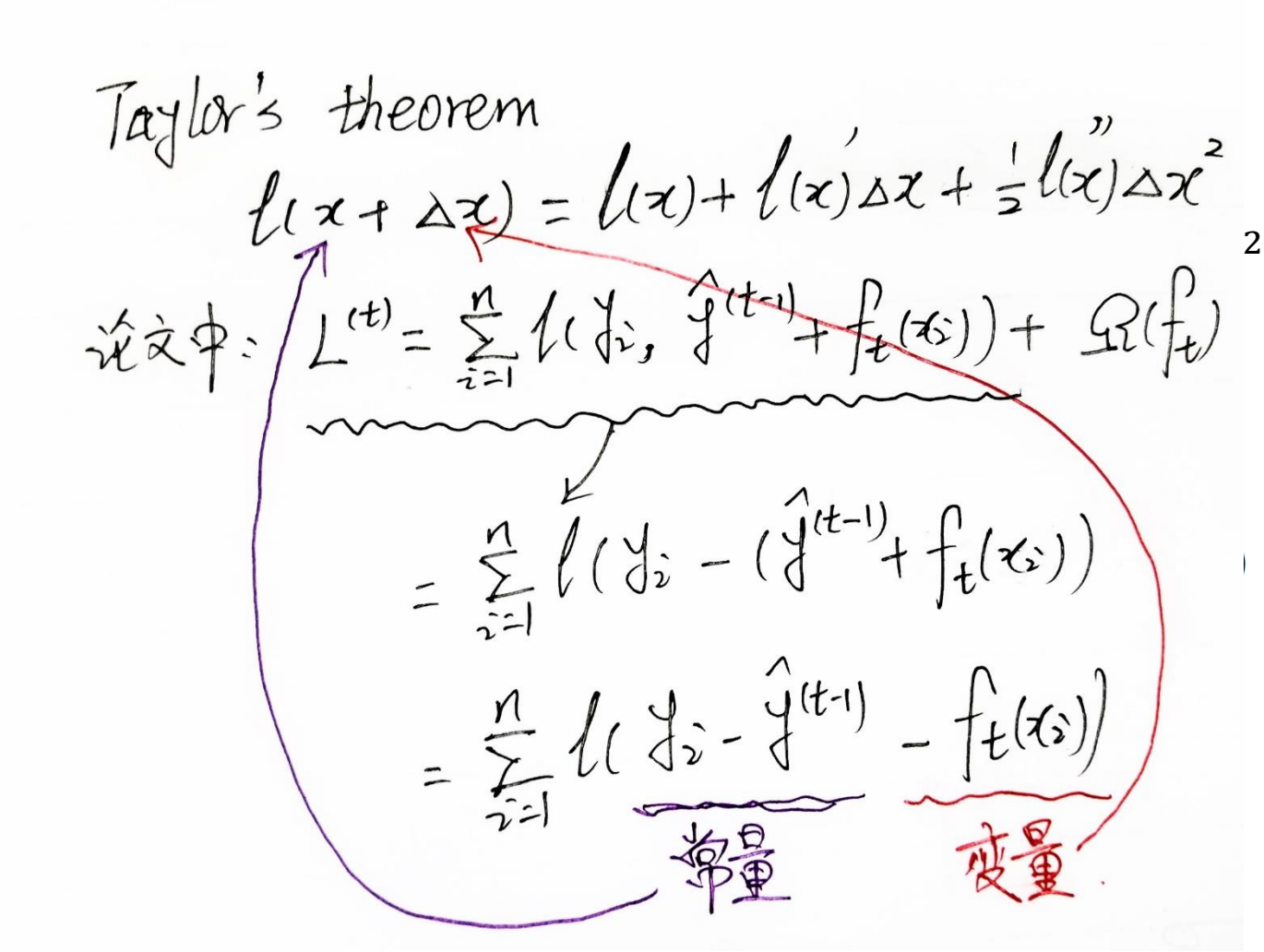
是属性集粒度的并行，将属性集划分到多机上排序、计算式子：

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

模型迭代过程还是串行的！

XGBoost

小问题：论文中泰勒展开的来源



XGBoost

小问题：**Shrinkage**理解

$$F_{m+1}(x) = F_m(x) + h(x)$$

$$0 < \alpha < 1$$

$$F_{m+1}(x) = F_m(x) + \alpha h(x)$$

人生哲理：一个对XGBoost形象化的理解

1. 目标明确

XGBoost的代价函数就是人生的奋斗目标，所有努力都是为了降低XGBoost。有时候我们的目的不一样，所以我们需要自己定义自己的目标函数。有时候为了查全率，有时候是为了查准率，有时候为了F1，F0.5或者F2值。

2. 高瞻远瞩

有时候我们不仅仅看到，下一步怎么走，还有看到下一步的下一步。

3. 学会多任务能力

同一时间，要学会处理多件事情。

4. 学会快人一步，积累经验

需要预学习，利用既往的经验，达到快速处理任务的能力。

Ref：https://www.zhihu.com/question/41354392/answer/150479474

XGBoost总结

问题

XGBoost有什么特点？与Boosting tree或者GBDT有什么异同？

XGBoost总结

问题

XGBoost有什么特点？与Boosting tree或者GBDT有什么异同？

1. 这几个模型，不是简单的算法，都是框架！！其Loss function和求导实现、并行计算都可以自己定制。可以说，XGBoost是Boosting Tree的高校实现。
2. XGBoost公式推导里用到了二阶导数信息，而普通的GBDT只用到一阶，或0阶
3. XGBoost允许使用column(feature) sampling来防止过拟合，借鉴了Random Forest的思想。
4. XGBoost在代价函数里加入了正则项，用于控制模型的复杂度。降低了模型的variance，使学习出来的模型更加简单，防止过拟合，这也是XGBoost优于传统GBDT的一个特性。

Ref:https://www.zhihu.com/question/41354392/answer/98658997

XGBoost总结

问题

XGBoost有什么特点？与Boosting tree或者GBDT有什么异同？

5. 学习速率控制——Shrinkage（缩减）。XGBoost和在进行完一次迭代后，会将叶子节点的权重乘上该系数，主要是为了削弱每棵树的影响，让后面有更大的学习空间。
6. 对缺失值的处理。对于特征的值有缺失的样本，XGBoost可以自动学习出它的分裂方向。
7. 候选的分割点近似算法
8. XGBoost工具在属性层面上支持并行。
9. 为避免当以行计算梯度数据时会导致内存的不连续访问与cache miss，降低算法效率。paper中提到，可先将数据收集到线程内部的buffer，然后再计算，提高算法的效率。
10. XGBoost 还考虑了当数据量比较大，内存不够时怎么有效的使用磁盘，主要是结合多线程、数据压缩、分片的方法，尽可能的提高算法的效率。

Ref:http://blog.csdn.net/shenxiaoming77/article/details/51542982

Ref:https://www.zhihu.com/question/41354392/answer/98658997

Zhou Z H, Feng J. Deep Forest. arXiv 2017

Deep Forest: Towards An Alternative to Deep Neural Networks

Zhi-Hua Zhou and **Ji Feng**
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
{zhouzh, fengj}@lamda.nju.edu.cn

Abstract

In this paper, we propose gcForest, a decision tree ensemble approach with performance highly competitive to deep neural networks. In contrast to deep neural networks which require great effort in hyper-parameter tuning, gcForest is much easier to train. Actually, even when gcForest is applied to different data from different domains, excellent performance can be achieved by almost same settings of hyper-parameters. The training process of gcForest is efficient and scalable. In our experiments its training time running on a PC is comparable to that of deep neural networks running with GPU facilities, and the efficiency advantage may be more apparent because gcForest is naturally apt to parallel implementation. Furthermore, in contrast to deep neural networks which require large-scale training data, gcForest can work well even when there are only small-scale training data. Moreover, as a tree-based approach, gcForest should be easier for theoretical analysis than deep neural networks.

1 Introduction

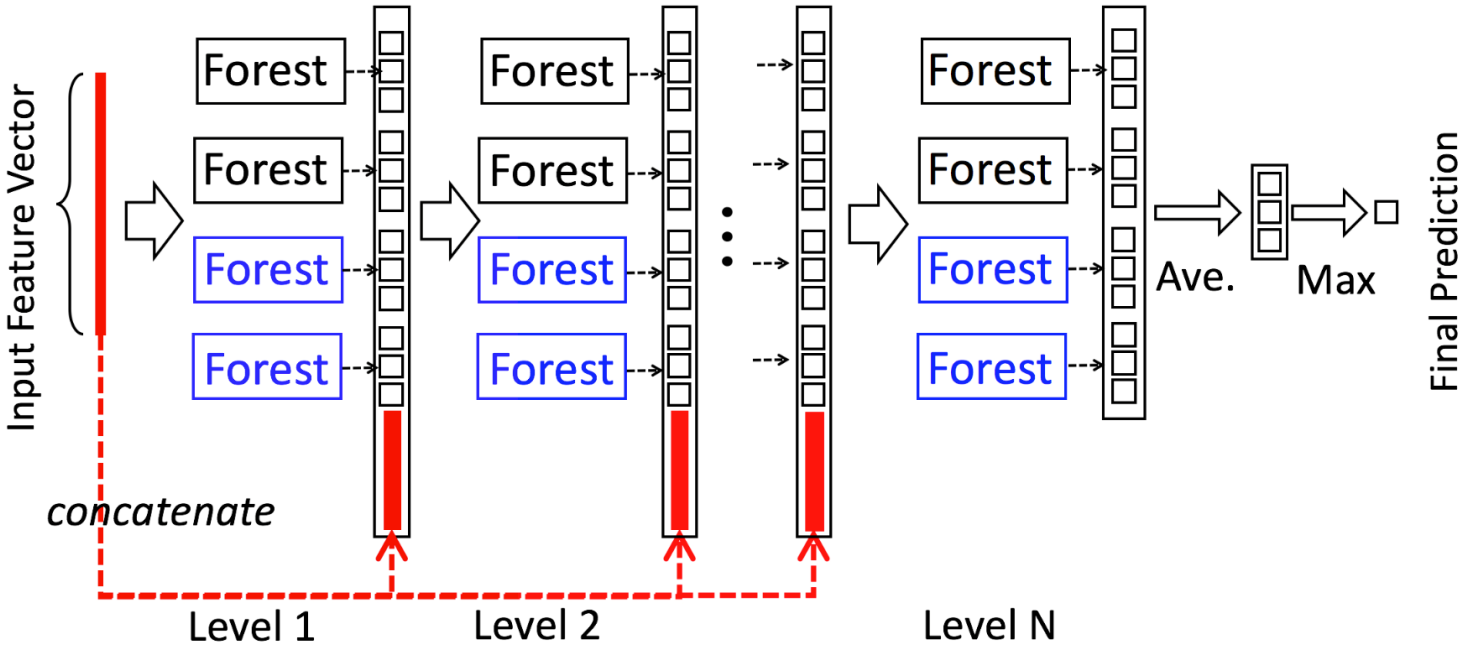
In recent years, deep neural networks have achieved great ral networks [LeCun *et al.*, 1998; Krizhenvsky *et al.*, 2012; Simonyan and Zisserman, 2014], they are actually using different learning models due to the many different options such as the convolutional layer structures. This fact not only makes the training of deep neural networks very tricky, like an art rather than science/engineering, but also makes theoretical analysis of deep neural networks extremely difficult because of too many interfering factors with almost infinite configurational combinations.

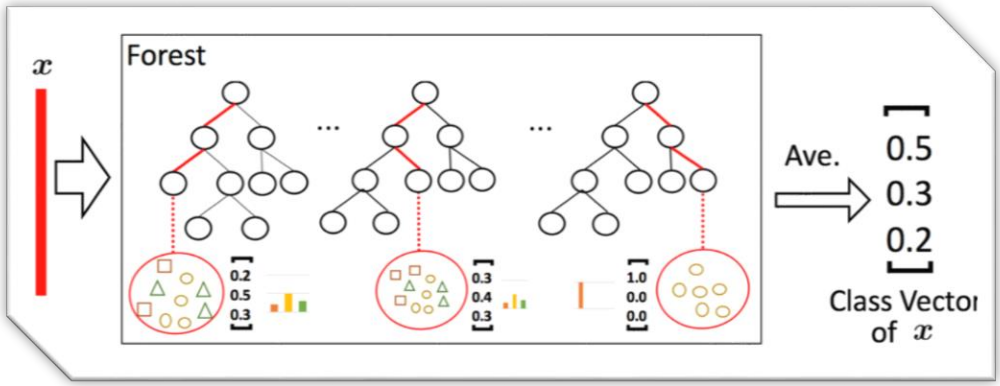
It is widely recognized that the *representation learning* ability is crucial for deep neural networks. It is also noteworthy that, to exploit large training data, the capacity of learning models should be large; this partially explains why the deep neural networks are very complicated, much more complex than ordinary learning models such as support vector machines. We conjecture that if we can endow these properties to some other suitable form of learning models, we may be able to achieve performance competitive to deep neural networks but with less aforementioned deficiencies.

In this paper, we propose gcForest(multi-Grained Cascade forest), a novel decision tree ensemble method. This method generates a deep forest ensemble, with a cascade structure which enables gcForest to do representation learning. Its representational learning ability can be further enhanced by multi-grained scanning when the inputs are with high dimen-

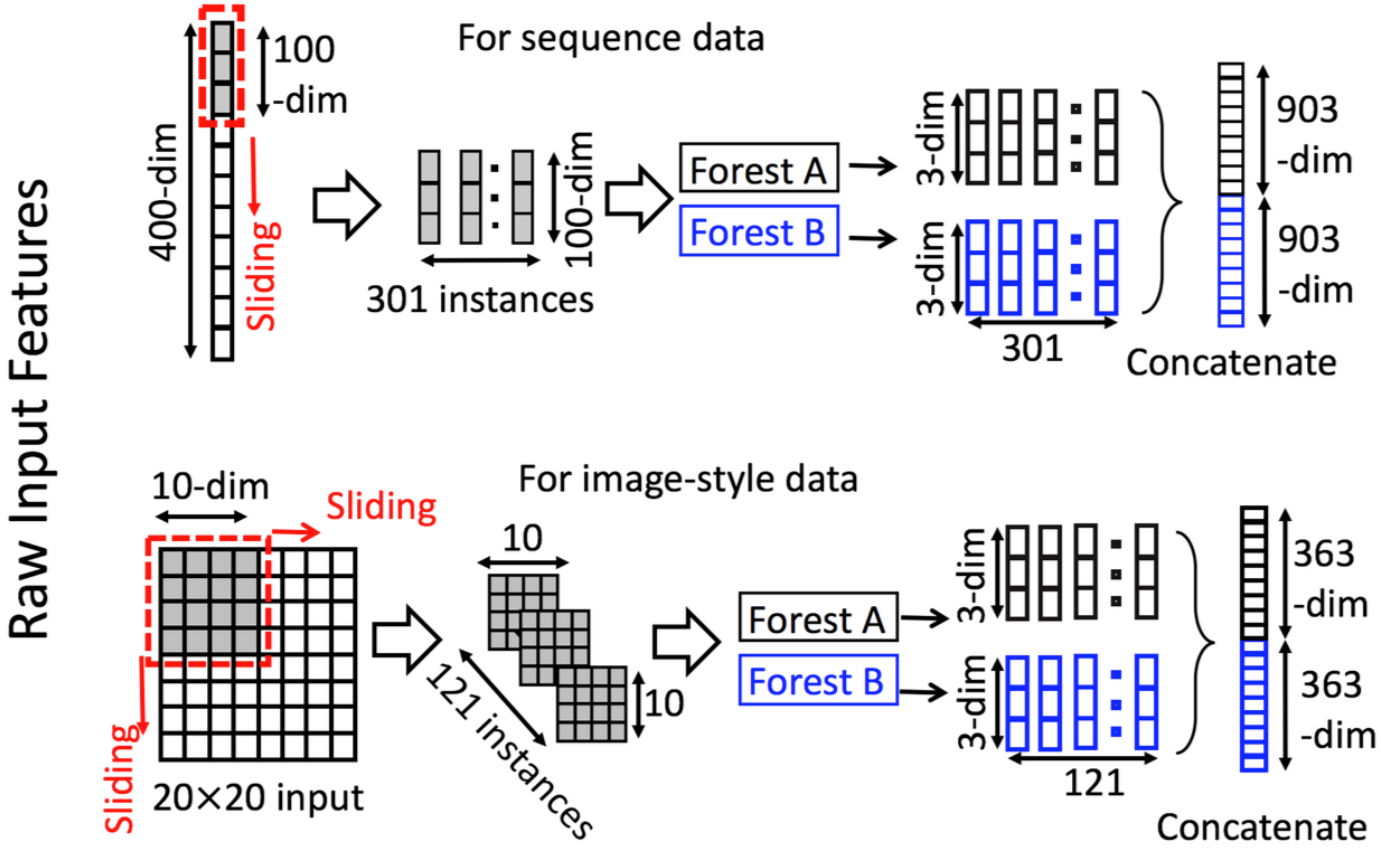
Cascade Forest

two random forests (blue) and two complete-random tree forests (black).





Multi-Grained Scanning



Zhou Z H, Feng J. Deep Forest. arXiv 2017

gcForest

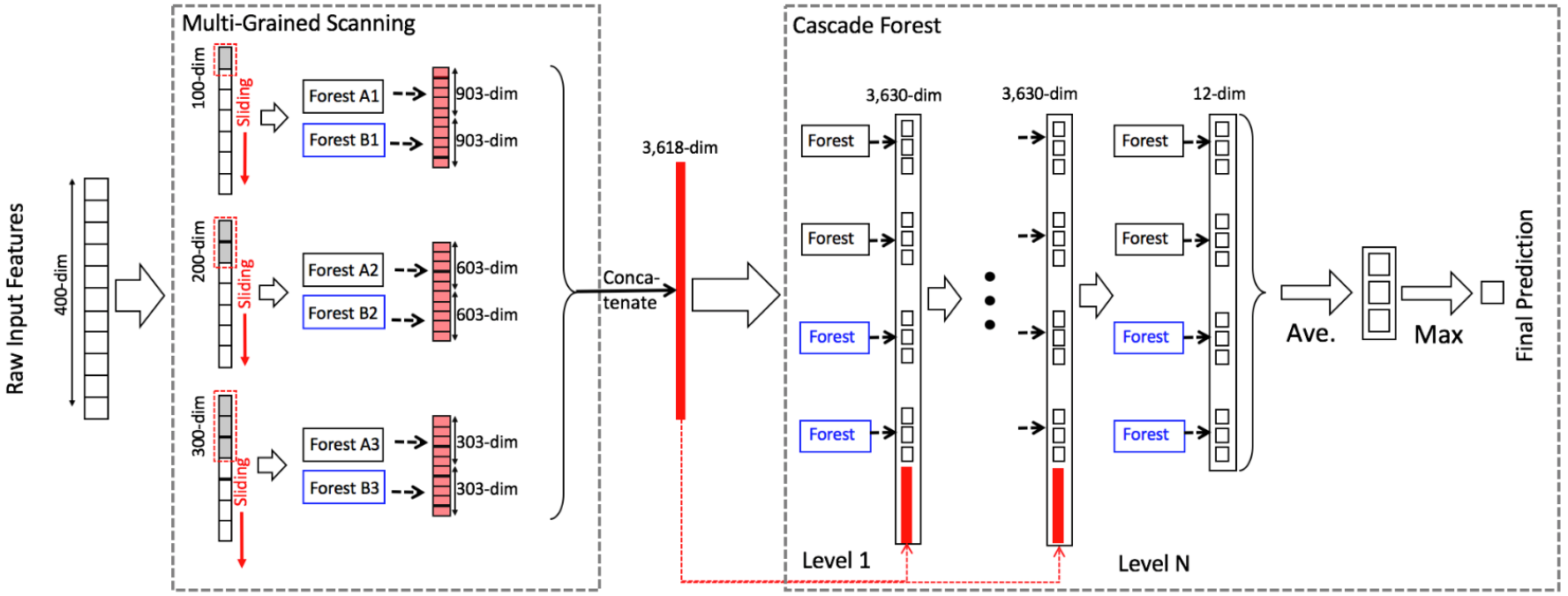


Figure 4: The overall procedure of gcForest. Suppose there are three classes to predict, raw features are 400-dim, and three sizes of sliding windows are used.

gcForest Performance

Table 2: Comparison of test accuracy on MNIST

DNN (LeNet-5)	99.05%
gcForest	**98.96%**
DNN (Deep Belief Net)	98.75% [Hinton *et al.*, 2006]
SVM (rbf kernel)	98.60%
Random Forest	96.80%

Table 3: Comparison of test accuracy on ORL

	1 image	5 images	9 images
gcForest	**63.06%**	**94.25%**	**98.30%**
Random Forest	61.70%	91.20%	97.00%
DNN (CNN)	3.30%	86.50%	92.50%
SVM (rbf kernel)	57.90%	78.95%	82.50%
kNN	19.40%	77.60%	90.50%

Table 4: Comparison of test accuracy on GTZAN

gcForest	**65.67%**
CNN	59.20%
MLP	58.00%
Random Forest	50.33%
Logistic Regression	50.00%
SVM (rbf kernel)	18.33%

Table 5: Comparison of test accuracy on sEMG data

gcForest	**55.93%**
LSTM	45.37%
MLP	38.52%
Random Forest	29.62%
SVM (rbf kernel)	29.62%
Logistic Regression	23.33%

Table 6: Comparison of test accuracy on IMDB

gcForest	**89.32%**
DNN (CNN)	89.02% [Kim, 2014]
DNN (MLP)	88.04%
Logistic Regression	88.62%
SVM (linear kernel)	87.56%
Random Forest	85.32%

Framework

1. Background and Activation
   - 集成学习背景故事
   - 集成学习简介（假设、关注点、种类）
   - 方差与偏差、决策树模型
2. Introduction and discussion of Models
   - Bagging & 随机森林
   - Boosting(AdaBoost, Boosting tree, GBDT, XGBoost)
3. Two Popular Papers
   - XGBoost
   - Deep Forest

浅谈，概念建立 10~20min

深入探讨 20~30min

前沿 10~20min

4. 提问总结

- 两篇文章是近期非常火热的文章, 难度都适中，能被我们普通人理解。
- 好工作不是只属于数学家，只要沉下心，普通人也能做出好工作。

Thanks

计算机科学与工程学院

高崇铭