机器学习
第7节

涉及知识点：
随机森林、广义加性模型、AdaBoost简介、提升算法简史、GBDT梯度提升决策树、深度森林、多粒度级联森林

集成学习

张伟楠 – 上海交通大学

随机森林

张伟楠 - 上海交通大学

为什么Bagging算法有效?

- 偏差-方差分解（Bias-Variance Decomposition）
  - 假设$Y = f(X) + \epsilon$，其中$\mathbb{E}[\epsilon] = 0 \;\; Var[\epsilon] = \sigma_\epsilon^2$
  - 在输入点x_0的期望预测误差为

$$
\begin{aligned}
Err(x_0) &= \mathbb{E}\left[\left(Y - \hat{f}(x_0)\right)^2 \middle| X = x_0\right] \\
&= \sigma_\epsilon^2 + \left[\mathbb{E}[\hat{f}(x_0)] - f(x_0)\right]^2 + \mathbb{E}\left[\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)]\right]^2 \\
&= \sigma_\epsilon^2 + Bias^2\left(\hat{f}(x_0)\right) + Var\left(\hat{f}(x_0)\right)
\end{aligned}
$$

- Bagging有效的原因是与原始模型相比偏差相同但是降低了方差(在整个数据集上进行训练)
  - 对低偏差和高方差的预测模型尤其有效果

Bagging算法

- Bagging有效的原因是与原始模型相比偏差相同但是降低了方差(在整个数据集上进行训练)
  - 对低偏差和高方差的预测模型尤其有效果

- 如果有一系列服从同一分布的变量（方差为σ^2），两两之间的关联性为ρ，对变量值取平均之后的方差为

$$\rho_{ij} = \frac{cov(i,j)}{\sigma_i \sigma_j}$$

$$\text{var} = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

$$\Sigma = \begin{bmatrix} \sigma^2 & \rho\sigma^2 & \cdots & \rho\sigma^2 \\ \rho\sigma^2 & \sigma^2 & & \\ \vdots & & \ddots & \\ \rho\sigma^2 & & & \sigma^2 \end{bmatrix} = \sigma^2 \begin{bmatrix} 1 & & \rho \\ & 1 & \\ \rho & & \ddots \; 1 \end{bmatrix}$$

如果采样样本无限大，该公式的值会降为$\rho\sigma^2$

$$r_0 = \frac{1}{B}\sum_{i=1}^{B} r_i \qquad Var(r_0) = \frac{1^T}{B}\Sigma\frac{1}{B} = \frac{1}{B^2}1^T\Sigma 1 = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Bagging算法

- Bagging有效的原因是与原始模型相比偏差相同但是降低了方差(在整个数据集上进行训练)
  - 对低偏差和高方差的预测模型尤其有效果

问题

- 用自举采样的数据进行训练得到的模型很可能是有一定相关性的

$$\begin{aligned}\mathrm{var} &= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \\ &= \rho\sigma^2\left(1-\frac{1}{B}\right) + \frac{1}{B}\sigma^2\end{aligned}$$

越小，最后集成的方差越小

随机森林

- 随机森林与bagging算法的主要区别在于构建了一个解耦合（de-correlated）的树，然后对结果取平均
- Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 532.

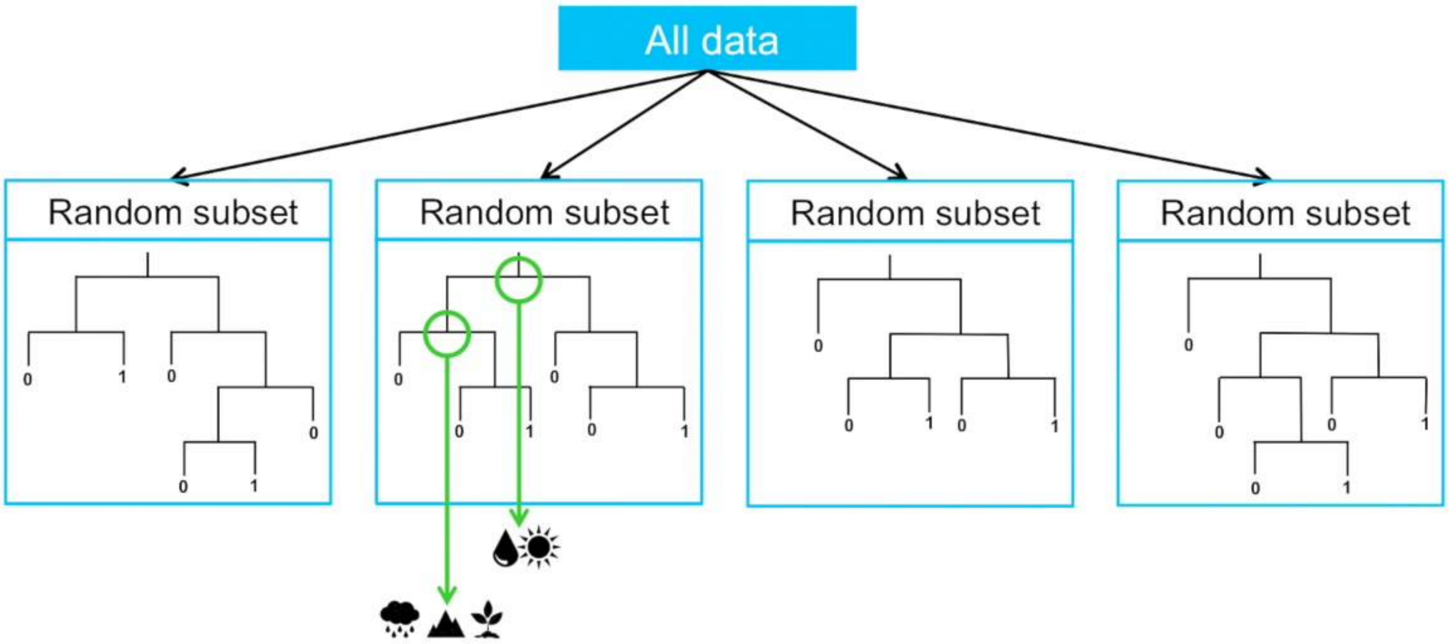
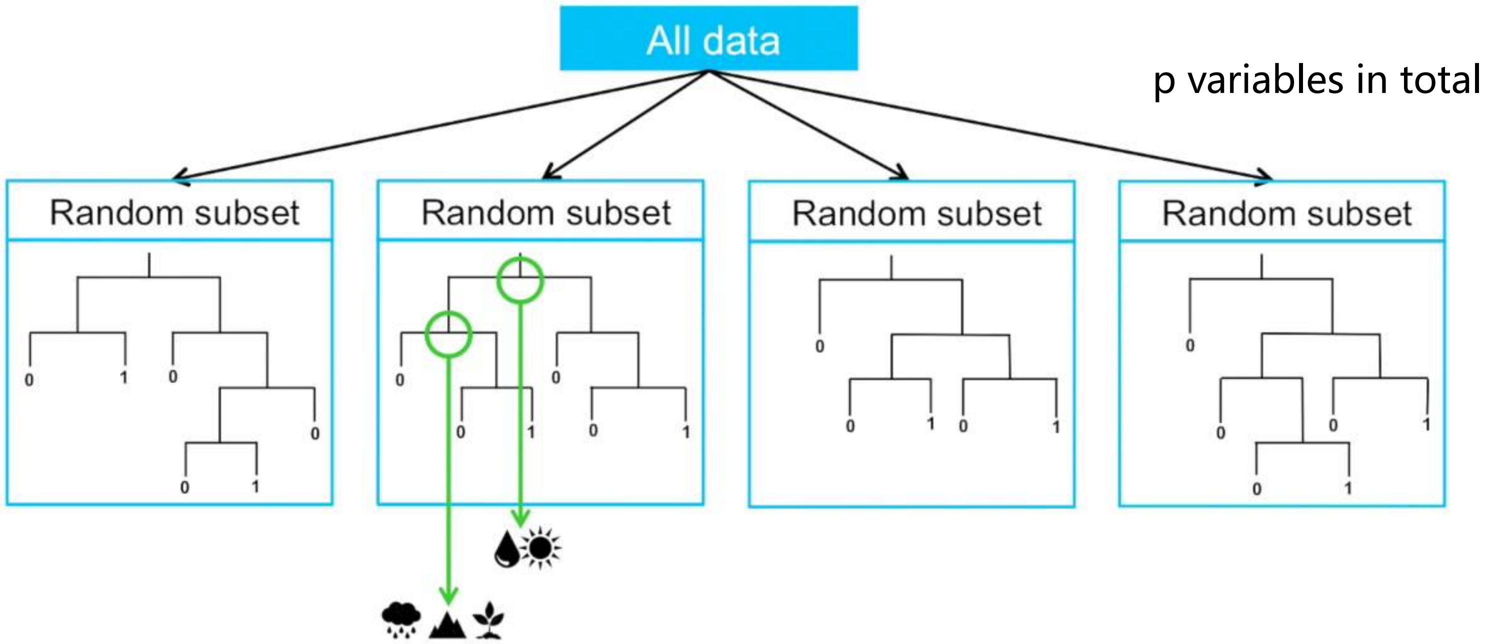


Image credit: https://i.ytimg.com/vi/-bYrLRMT3vY/maxresdefault.jpg

随机森林中的树解耦合



- 在每个树节点分裂前，随机选择$m \le p$个变量作为分裂的候选变量
  - 一般$m = \sqrt{p}$甚至取值为1

随机森林算法

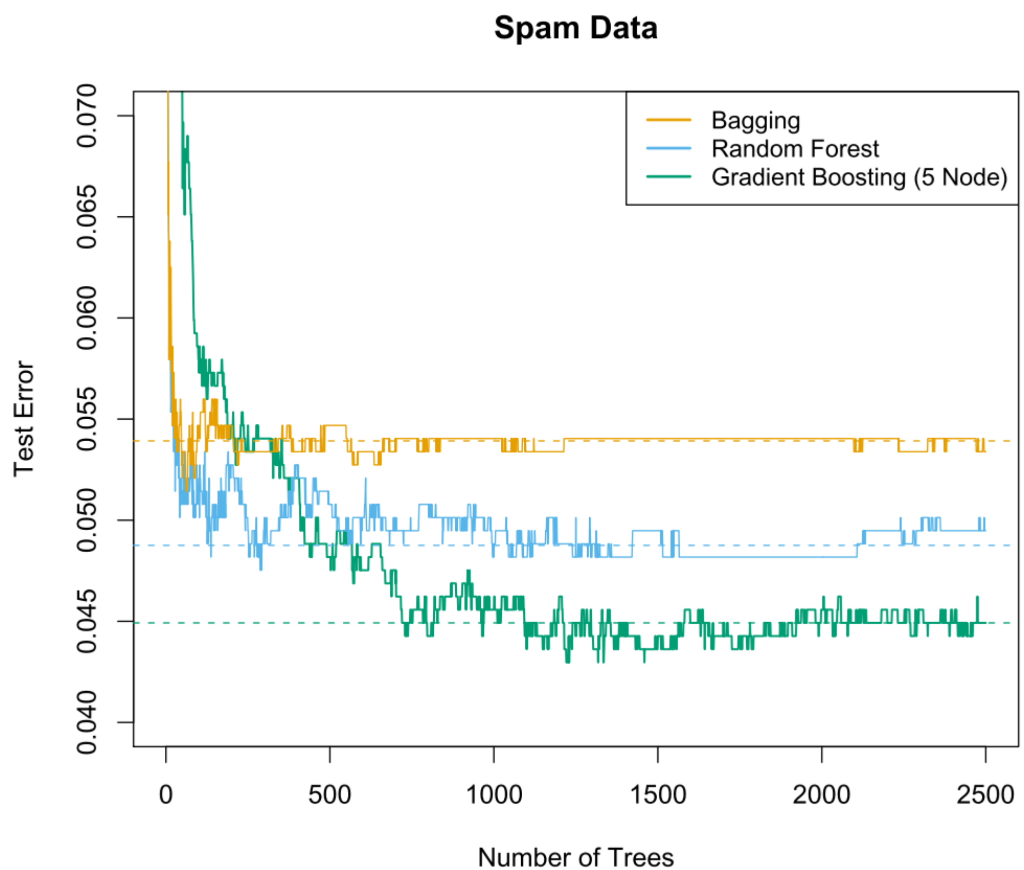
- 对$b = 1$到B：
  - 从训练数据中自举采样n个数据作为自举采样集
  - 根据采样数据生成一个随机树T_b，对每个树的叶节点迭代重复接下来几步，直到到达最小的结点数量：
    - 从p个变量中随机选择m个变量
    - 在m个变量中间选择最好的变量和最佳分裂点
    - 将结点分裂成两个子结点
- 输出新的集成结果$\{T_b\}_{b=1...B}$
- 对新的点x做预测

回归：预测平均值 $$\hat{f}_{\mathrm{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

分类：多数投票 $$\hat{C}_{\mathrm{rf}}^{B}(x) = \text{majority vote}\{\hat{C}_b(x)\}_1^B$$

Algorithm 15.1 of Hastie et al. The elements of statistical learning.

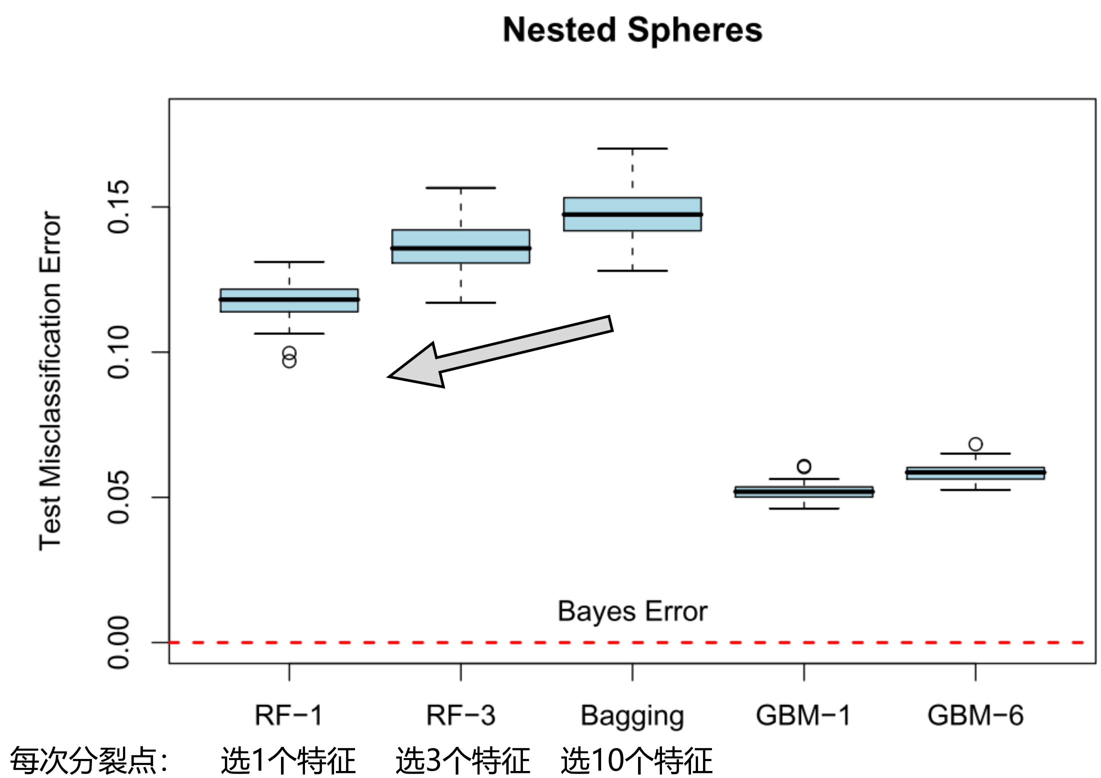
表现对比



1536个测试实例

Fig. 15.1 of Hastie et al. The elements of statistical learning.

表现对比



- 10维嵌套球数据 $Y = \begin{cases} 1 & if\ \sum_{j=1}^{10} X_j^2 > 9.34 \\ -1 & otherwise \end{cases}$
- RF-m: m表示每次分裂时随机选择的变量数量

Fig. 15.2 of Hastie et al. The elements of statistical learning.

广义加性模型

张伟楠 – 上海交通大学

Bagging vs. 随机森林 vs. Boosting

- Bagging仅仅是在有相同权重的自举采样集上对每个预测模型进行训练
- 随机森林通过采样特征的方法尝试去解耦合自举训练预测模型（决策树)
- Boosting算法基于之前的预测模型有策略性地学习和结合接下来的预测模型

加性模型 (Additive Models)

- 加性模型的通用形式

$$F(x) = \sum_{m=1}^{M} f_m(x)$$

- 对于回归问题

$$f_m(x) = \beta_m b(x; \gamma_m) \qquad F_M(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

  - 其他预测模型参数固定，进行最小二乘学习

$$\{\beta_m, \gamma_m\} \leftarrow \underset{\beta,\gamma}{argmin} \mathbb{E}\left[y - \sum_{k \neq m} \beta_k b(x; \gamma_k) - \beta b(x; \gamma)\right]^2$$

  - 之前学习好的预测模型参数固定，逐步进行最小二乘学习

$$\{\beta_m, \gamma_m\} \leftarrow \underset{\beta,\gamma}{argmin} \mathbb{E}[y - F_{m-1}(x) - \beta b(x; \gamma)]^2$$

加性回归模型

- 其他预测模型参数固定，进行最小二乘学习

$$\{\beta_m, \gamma_m\} \leftarrow \underset{\beta,\gamma}{argmin} \mathbb{E}\left[y - \sum_{k \neq m} \beta_k b(x;\gamma_k) - \beta b(x;\gamma)\right]^2$$

  - 本质上讲，这种加性学习相当于把原始值修改如下：

$$y_m \leftarrow y - \sum_{k \neq m} f_k(x)$$

- 之前学习好的预测模型参数固定，逐步进行最小二乘学习

$$\{\beta_m, \gamma_m\} \leftarrow \underset{\beta,\gamma}{argmin} \mathbb{E}[y - F_{m-1}(x) - \beta b(x;\gamma)]^2$$

  - 本质上讲，这种加性学习相当于把原始值修改如下：

$$y_m \leftarrow y - F_{m-1}(x) = y_{m-1} - f_{m-1}(x)$$

加性分类模型

- 对二元分类　$y = \{1, -1\}$

$$F(x) = \sum_{m=1}^{M} f_m(x)$$

$$P(y = 1|x) = \frac{exp(F(x))}{1 + exp(F(x))}$$

$$P(y = -1|x) = \frac{1}{1 + exp(F(x))}$$

- 进行对数变换

$$\log \frac{P(y=1|x)}{1-P(y=1|x)} = F(x)$$

AdaBoost简介

张伟楠 - 上海交通大学

目录

Contents

01

AdaBoost
学习准则

AdaBoost

- 对二元分类，最小化下式

$$J(F) = \mathbb{E}[e^{-yF(x)}]$$

[1998年由Schapire和Singer将其作为误分类误差的上限提出]

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. "Additive logistic regression: a statistical view of boosting." The annals of statistics 28.2 (2000): 337-407.

- 它（几乎）等价于逻辑交叉熵损失函数
  - 对于正（$y = +1$）和负（$y = -1$）标签

$$\begin{aligned}
\mathcal{L}(y,x) &= -\frac{1+y}{2}\log\frac{e^{F(x)}}{1+e^{F(x)}} - \frac{1-y}{2}\log\frac{1}{1+e^{F(x)}} \\
&= -\frac{1+y}{2}\left(F(x) - \log\left(1+e^{F(x)}\right)\right) + \frac{1-y}{2}\log\left(1+e^{F(x)}\right) \\
&= -\frac{1+y}{2}F(x) + \log\left(1+e^{F(x)}\right) \\
&= \log\frac{1+e^{F(x)}}{e^{\frac{1+y}{2}F(x)}} = \begin{cases} \log(1+e^{F(x)}) & if\ y = -1 \\ \log(1+e^{-F(x)}) & if\ y = +1 \end{cases} = \log\left(1+e^{-yF(x)}\right)
\end{aligned}$$

AdaBoost：一个指数规范

- 对二元分类，最小化下式

$$J(F) = \mathbb{E}[e^{-yF(x)}]$$

最优F(x)

$$\mathbb{E}[e^{-yF(x)}] = \int \mathbb{E}\,[e^{-yF(x)}|x]p(x)dx$$

$$\mathbb{E}[e^{-yF(x)}|x] = P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)}$$

$$\frac{\partial \mathbb{E}[e^{-yF(x)}|x]}{\partial F(x)} = -P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)}$$

$$\frac{\partial \mathbb{E}[e^{-yF(x)}|x]}{\partial F(x)} = 0 \Rightarrow F(x) = \frac{1}{2}\log\frac{P(y=1|x)}{P(y=-1|x)}$$

AdaBoost：一个指数规范

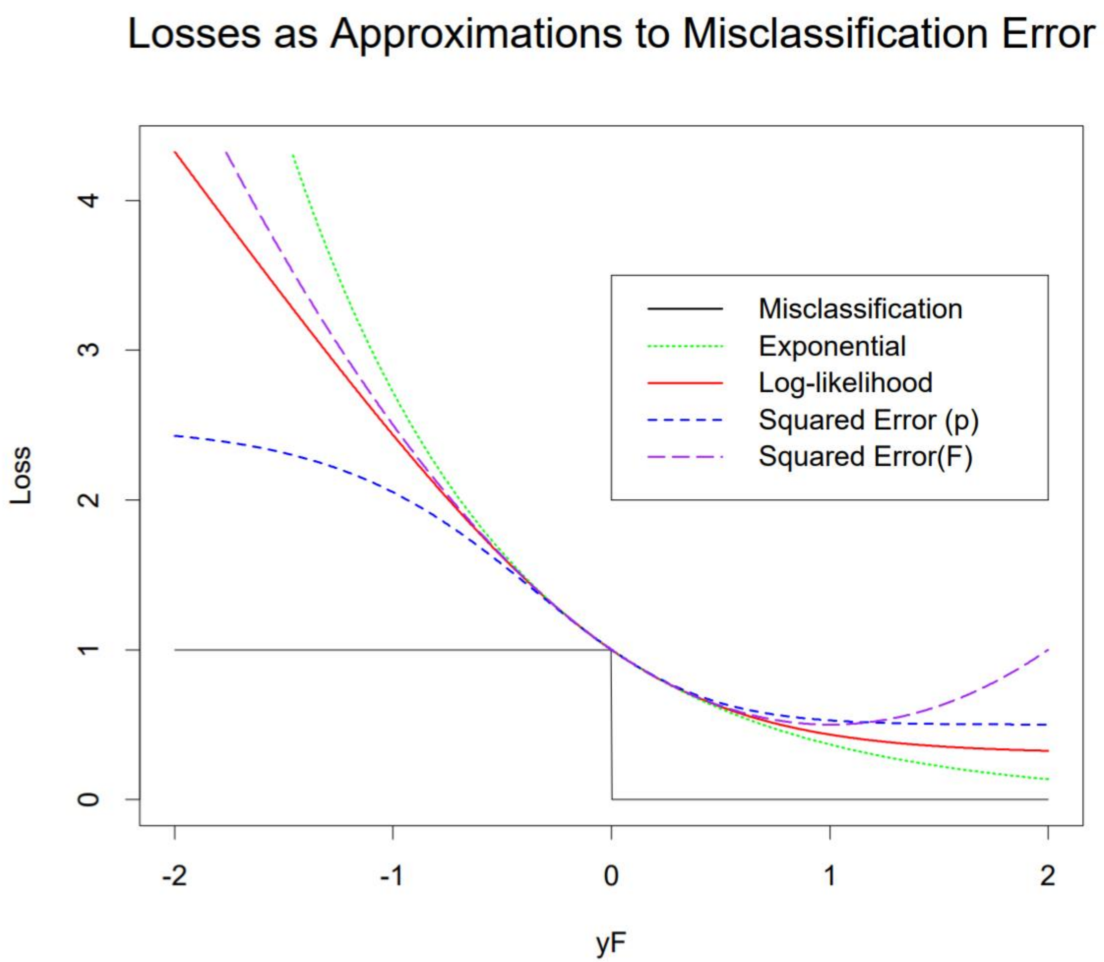
- 最优F(x)

$$\frac{\partial \mathbb{E}[e^{-yF(x)}|x]}{\partial F(x)} = 0 \Rightarrow F(x) = \frac{1}{2}\log\frac{P(y=1|x)}{P(y=-1|x)}$$

$$\Rightarrow P(y=1|x) = \frac{e^{2F(x)}}{1+e^{2F(x)}}$$

- 因此，AdaBoost和逻辑斯谛回归的损失函数几乎相同
- AdaBoost的最优$F(x)$对应于1/2倍的逻辑斯谛回归$F(x)$值

误分类误差



- 指数准则和对数似然（交叉熵）在二阶泰勒级数上是等价的

02
AdaBoost
算法

离散型AdaBoost

- 准则 $J(F)=\mathbb{E}[e^{-yF(x)}]$ $\qquad f(x)=\pm 1$
- 当前估计值 $F(x)$
- 寻找改进的估计值 $F(x)+cf(x)$
- 泰勒级数

$$f(a+x)=f(a)+\frac{f'(a)}{1!}x+\frac{f''(a)}{2!}x^2+\frac{f'''(a)}{3!}x^3+\cdots$$

- 二阶泰勒级数

$$\begin{aligned}J(F+cf)&=\mathbb{E}\left[e^{-y\left(F(x)+cf(x)\right)}\right]\\&\simeq\mathbb{E}\left[e^{-yF(x)}(1-ycf(x)+c^2y^2f(x)^2/2)\right]\\&=\mathbb{E}\left[e^{-yF(x)}(1-ycf(x)+c^2/2)\right]\end{aligned}$$

注意：$y^2=1 \qquad f(x)^2=1$

离散型AdaBoost

- 准则 $J(F) = \mathbb{E}[e^{-yF(x)}]$ $\quad\quad f(x) = \pm 1$

$$J(F+cf) \simeq \mathbb{E}[e^{-yF(x)}(1 - ycf(x) + c^2/2)]$$

- 固定住c求解f

$$\begin{aligned} f &= \underset{f}{argmin}\, J(F+cf) = \underset{f}{argmin} \mathbb{E}\left[e^{-yF(x)}(1 - ycf(x) + c^2/2)\right] \\ &= \underset{f}{argmin}\, \mathbb{E}_w[1 - ycf(x) + c^2/2|x] \\ &= \underset{f}{argmax} \mathbb{E}_w[yf(x)|x] \quad\quad (\text{for } c > 0) \end{aligned}$$

其中加权条件期望为

$$\mathbb{E}_w[yf(x)|x] = \frac{\mathbb{E}\left[e^{-yF(x)}yf(x)\right]}{\mathbb{E}\left[e^{-yF(x)}\right]}$$

权重是在每个数据实例上的归一化误差系数$e^{-yF(x)}$

离散型AdaBoost

- 固定住c求解f $\qquad f(x) = \pm 1$

$$f = \underset{f}{argmin} J(F + cf) = \underset{f}{argmax} \mathbb{E}_w[yf(x)|x] \qquad (for\ c > 0)$$

$$\mathbb{E}_w[yf(x)|x] = \frac{\mathbb{E}[e^{-yF(x)}yf(x)]}{\mathbb{E}[e^{-yF(x)}]}$$ 加权期望

  - 即利用加权的训练数据实例来训练$f(\cdot)$，对应权重与其之前的误差系数$e^{-yF(x)}$成比例

求解答案

$$f(x) = \begin{cases} 1, & \text{if } \mathbb{E}_w(y|x) = P_w(y = 1|x) - P_w(y = -1|x) > 0 \\ -1, & \text{otherwise} \end{cases}$$

离散型AdaBoost

- 准则 $J(F) = \mathbb{E}\left[e^{-yF(x)}\right]$ $f(x) = \pm 1$
- 固定住f求解c

$$c = \underset{c}{argmin} J(F + cf) = \underset{c}{argmin} \mathbb{E}_w[e^{-cyf(x)}]$$

$$\begin{aligned}
\frac{\partial \mathbb{E}_w\left[e^{-cyf(x)}\right]}{\partial c} &= \mathbb{E}_w\left[-e^{-cyf(x)} y f(x)\right] \\
&= \mathbb{E}_w\left[P\big(y \neq f(x)\big) \cdot e^c + \Big(1 - P\big(y \neq f(x)\big)\Big) \cdot (-e^{-c})\right] \\
&= \mathrm{err} \cdot e^c + (1 - \mathrm{err}) \cdot (-e^{-c}) = 0 \\
&\Rightarrow c = \frac{1}{2} \log \frac{1 - \mathrm{err}}{\mathrm{err}}
\end{aligned}$$

$$\boxed{\mathrm{err} = \mathbb{E}_w[1_{[y \neq f(x)]}]}$$

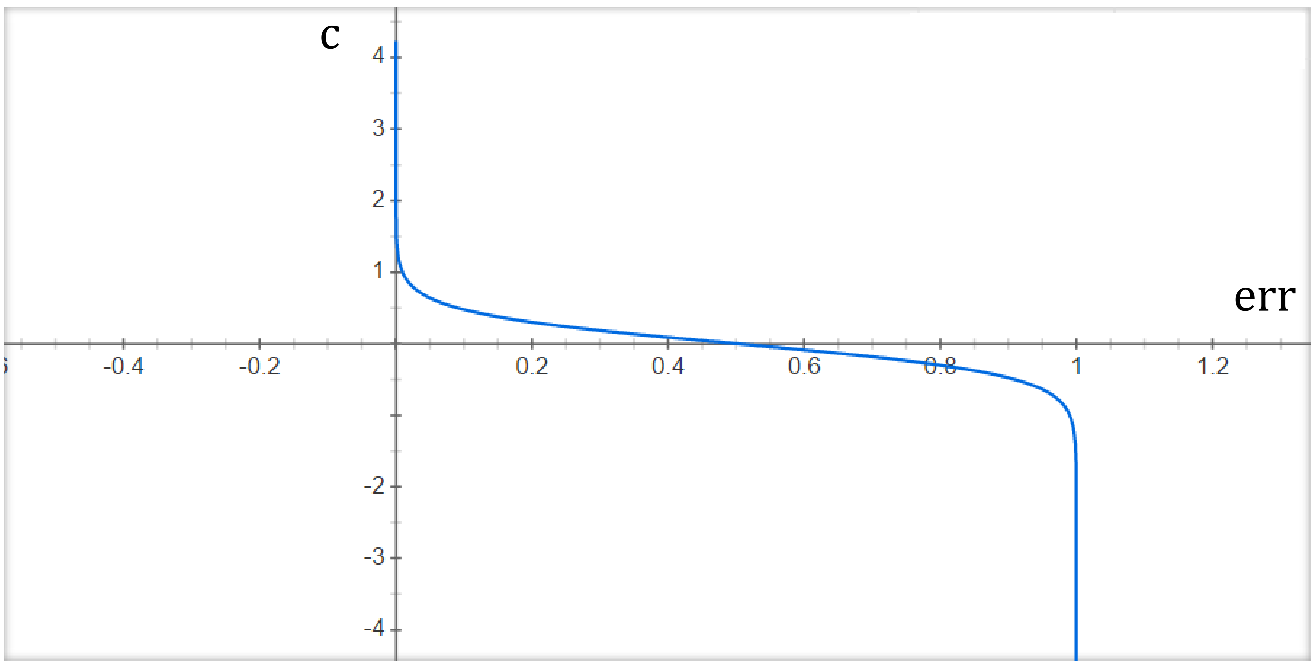
加权实例的总体误差率

离散型AdaBoost

- 准则 $J(F) = \mathbb{E}\left[e^{-yF(x)}\right]$ $f(x) = \pm 1$
- 固定住f求解c

$$c = \frac{1}{2}\log\frac{1-\mathrm{err}}{\mathrm{err}}$$

$$\boxed{\mathrm{err} = \mathbb{E}_w[1_{[y\neq f(x)]}]}$$



离散型AdaBoost

迭代

- 利用加权的训练数据实例来训练$f(\cdot)$，对应权重与其之前的误差系数$e^{-yF(x)}$成比例

$$f(x) = \begin{cases} 1, & \text{if } \mathbb{E}_w(y|x) = P_w(y=1|x) - P_w(y=-1|x) > 0 \\ -1, & \text{otherwise} \end{cases}$$

$$\mathrm{err} = \mathbb{E}_w\left[1_{[y \neq f(x)]}\right]$$

$$F(x) \leftarrow F(x) + \frac{1}{2}\log\frac{1-\mathrm{err}}{\mathrm{err}} f(x)$$

$$c = \frac{1}{2}\log\frac{1-\mathrm{err}}{\mathrm{err}}$$

归一化后消除

$$w(x,y) \leftarrow w(x,y)e^{-cf(x)y} = w(x,y)e^{c\left(2\times 1_{[y\neq f(x)]}-1\right)}$$
$$= w(x,y)exp\left(\log\frac{1-\mathrm{err}}{\mathrm{err}}\left(1_{[y\neq f(x)]}-\frac{1}{2}\right)\right)$$

离散型AdaBoost算法

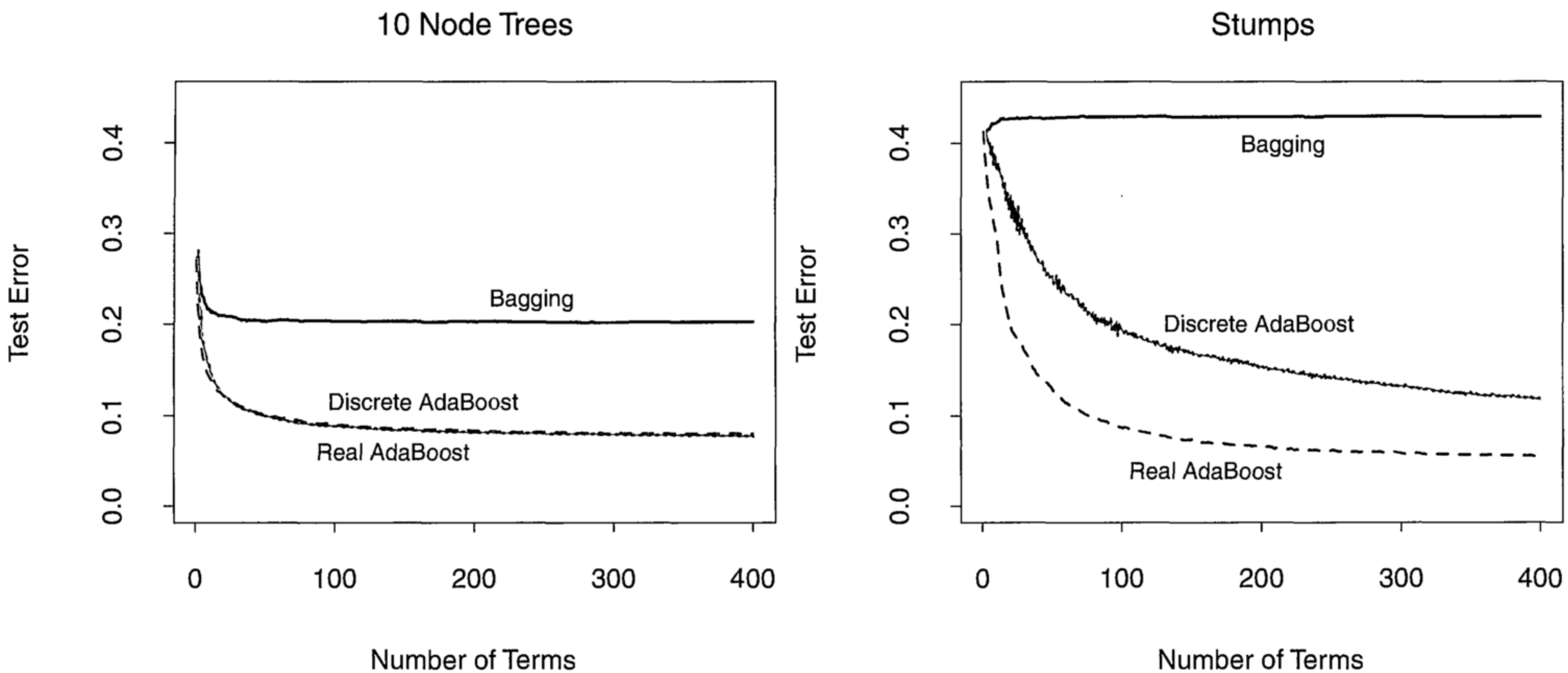
1. 等权重初始化每一个样本权重$w_i = \frac{1}{N}, i = 1, \ldots, N$
2. 从$m = 1$到M,
   I. 用权重w_i在训练数据上训练分类器$f_m(x) \in \{-1,1\}$
   II. 计算误差$\mathrm{err} = \mathbb{E}_w[1_{[y \neq f(x)]}], c = \frac{1}{2}\log\frac{1-\mathrm{err}}{\mathrm{err}}$
   III. 更新权重$w_i \leftarrow w_i \exp\left[c_m 1_{[y_i \neq f_m(x_i)]}\right], i = 1, \ldots, N$，并重新归一化使$\sum_i w_i = 1$
3. 输出分类结果$\mathrm{sign}[\sum_{m=1}^{M} c_m f_m(x)]$

实数型AdaBoost算法

1. 等权重初始化每一个样本权重$w_i = \frac{1}{N}, i = 1, \dots, N$
2. 从$m = 1$到M,
   I. 用权重w_i在训练数据上训练分类器得到一个类别概率估计$p_m(x) = \hat{P}_w(y = 1|x) \in [0,1]$
   II. 令$f_m(x) \leftarrow \frac{1}{2} \log p_m(x)/(1 - p_m(x)) \in R$
   III. 令$w_i \leftarrow w_i exp[-y_i f_m(x_i)], i = 1, \dots, N$，并重新归一化使$\sum_i w_i = 1$
3. 输出分类结果$\mathrm{sign}[\sum_{m=1}^{M} f_m(x)]$

- 实数型Adaboost使用类别概率估计$p_m(x)$来计算实数值贡献$f_m(x)$

Bagging vs. Boosting



Stump：只有两个叶节点的单层分叉树

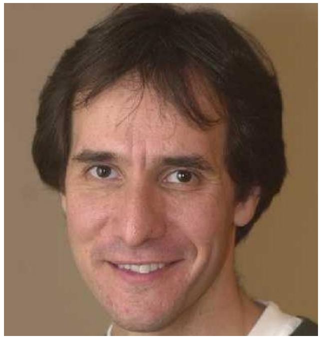
推荐周志华教授文章：
Boosting学习理论的探索 —— 一个跨越30年的故事
https://www.jiqizhixin.com/articles/2020-04-16-10

提升算法简史

张伟楠 – 上海交通大学

提升算法简史

- 1990年-Schapire证明，弱学习器总是可以通过在输入数据的过滤版本上训练另外两个分类器来提高其性能
  - 弱学习器是一种生成两元分类器的算法，其性能保证（以高概率）明显优于随机分类



Robert Schapire

- 具体地，
  - 在有N个样本的原始数据上训练分类器h_1
  - 然后在一组新的N个样本上训练分类器h_2，其中有一半样本是被h_1错误分类的
  - 然后在h_1和h_2分类结果不一致的N个样本上训练h_3
  - 提升分类器h_B =多数投票(h_1, h_2, h_3)
  - 事实证明，h_B比h_1性能更高

提升算法简史

- 1995年 - Freund提出了一种“多数提升”的变种方法，将许多弱学习器同时结合起来并改善了Schapire简单提升算法的性能
  - 这两种算法都要求弱学习器有固定的错误率
- 1996年 - Freund和Schapire提出了AdaBoost
  - 去除了固定错误率的要求
- 1996~1998 - Freund，Schapire和Singer以泛化误差的上界形式提出了一些理论来支持他们的算法
  - 但是边界太松散而不具有实际意义
  - 提升算法的实际表现远远超过了理论边界
- 2003年Freund，Schapire因AdaBoost获得哥德尔奖

Yoav Freund

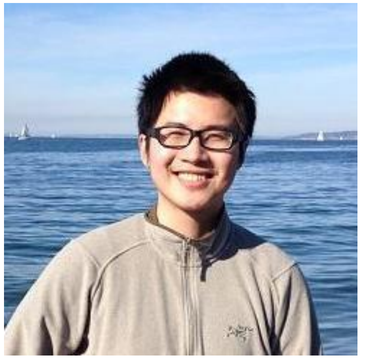
GBDT梯度提升决策树

讲师：张伟楠 – 上海交通大学

GBDT梯度提升决策树

梯度上升决策树

- 决策树的提升算法
  - $f_m(x)$是一个决策树模型
  - 许多变体，诸如GBRT，boosted trees，GBM



- 强烈推荐陈天奇的教程
  - https://web.njit.edu/~usman/courses/cs675_fall16/BoostedTree.pdf
  - http://devdoc.net/bigdata/xgboost-doc-0.81/tutorials/model.html
  - https://xgboost.readthedocs.io/en/latest/

GBDT梯度提升决策树

增量树

$$J^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \Omega(f_t)$$

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

Objective w.r.t. f_t $\quad J^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$

- 生成下一棵树f_t来最小化包含惩罚项$\Omega(f_t)$的损失函数$J^{(t)}$

$$\min_{f_t} J^{(t)}$$

GBDT梯度提升决策树

泰勒级数逼近

Objective w.r.t. f_t $$J^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

- 泰勒级数

$$f(a+x) = f(a) + \frac{f'(a)}{1!}x + \frac{f''(a)}{2!}x^2 + \frac{f'''(a)}{3!}x^3 + \cdots$$

- 定义梯度

$$g_i = \nabla_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right) \qquad h_i = \nabla^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$

- 目标函数近似

$$J^{(t)} \simeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

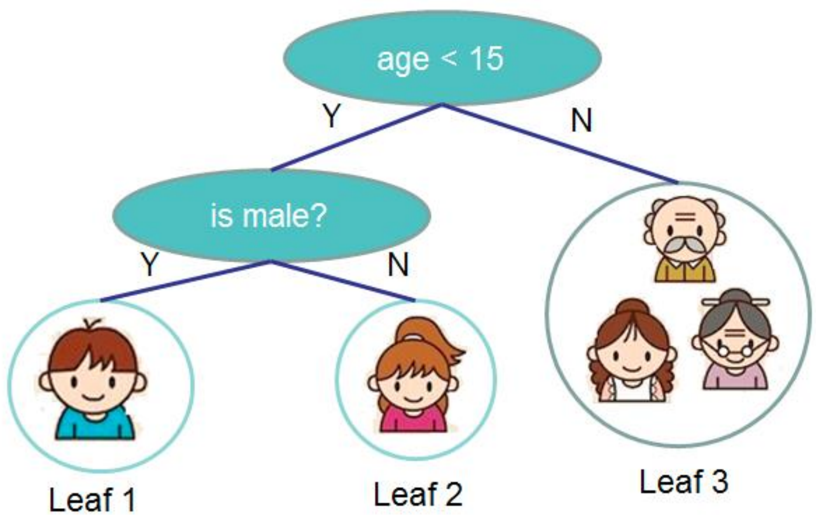
GBDT梯度提升决策树

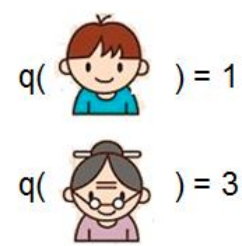
树结构复杂度惩罚

- 叶子节点预测值的变化程度

$$f_t(x) = w_{q(x)}, w \in \mathbb{R}^T, q: \mathbb{R}^d \mapsto \{1,2,\dots,T\}$$

  - 其中，T为叶子节点的数目





$w_1 = +2$ $\quad w_2 = +0.1$ $\quad w_3 = -1$

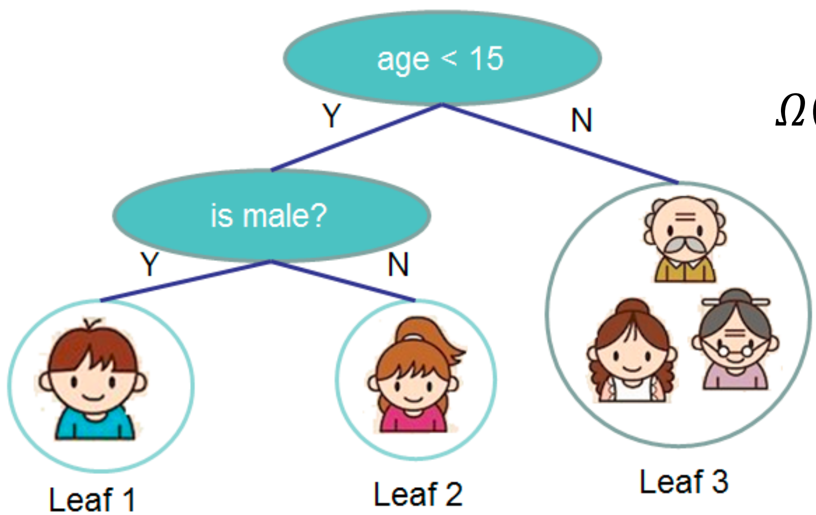
GBDT梯度提升决策树

树结构复杂度惩罚

- 定义树结构的复杂度惩罚为

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

  - 其中，T为叶子节点的数目，w_j为叶子节点的权重



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

$w_1 = +2$　　$w_2 = +0.1$　　$w_3 = -1$

GBDT梯度提升决策树

重写目标函数

- 考虑惩罚项

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

- 目标函数

$$J^{(t)} \simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t)$$

$$= \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 + const$$ 展开惩罚项

以叶子做加和 $$= \sum_{j=1}^{T}\left[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2\right] + \gamma T + const$$

$$g_i = \nabla_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$
$$h_i = \nabla^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$$

- I_j是样例的集合$\{i|q(x_i) = j\}$

GBDT梯度提升决策树

重写目标函数

- 目标函数

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

- 为了简化公式书写，定义$G_j = \sum_{i\in I_j} g_i$，$H_j = \sum_{i\in I_j} h_i$

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

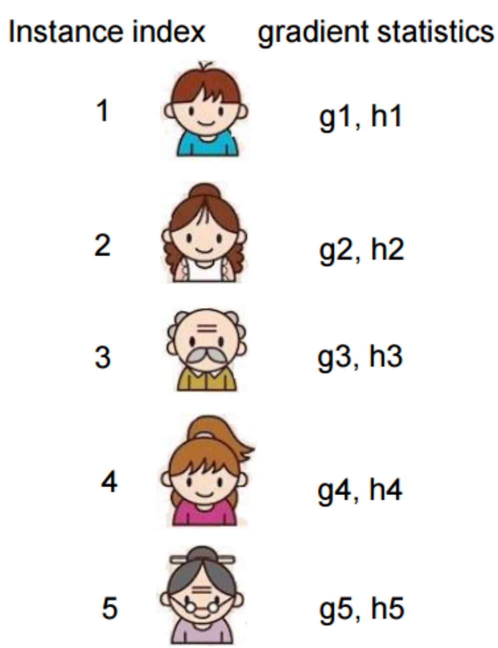
- 对于固定树结构$q: \mathbb{R}^d \mapsto \{1,2,\dots,T\}$
- 其闭式解为

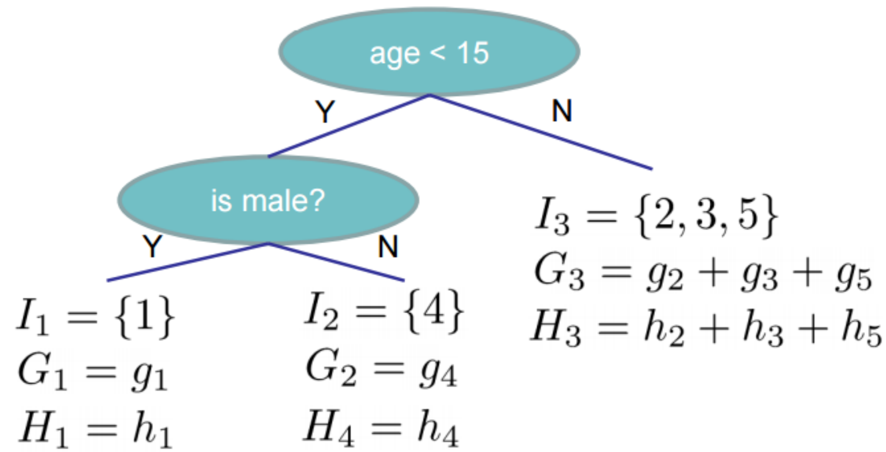
用于衡量一个树结构的好坏

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

GBDT梯度提升决策树

结构打分计算





$$J^{(t)} = -\frac{1}{2}\sum_{j=1}^{3}\frac{G_j^2}{H_j+\lambda} + \gamma 3$$

数值越小，树结构越好

注意：现在并没有着眼最大化基尼不纯度或者信息增益

GBDT梯度提升决策树

寻找最优树结构

- 特征和分裂点
- 贪心地生长树结构
  - 从树结构深度为0处开始
  - 对于每一个叶子节点，尝试增加一次分裂之后，其目标函数的变化为

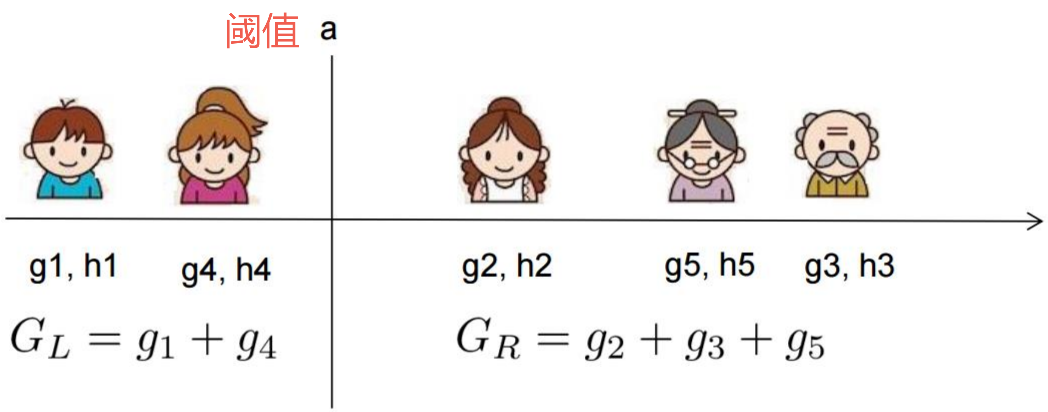
$$\text{Gain} = \frac{1}{2}\left(\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{左子树分数}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{右子树分数}} - \underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}_{\text{无分裂分数}}\right) - \underbrace{\gamma}_{\text{新叶子惩罚}}$$

  - 由于最后的惩罚项，引入分裂以后有可能获得非正数的增益

GBDT梯度提升决策树

高效寻找最优分裂

- 对于被选特征j，将数据进行升序排序



- 仅需要获得每一边的g和h，即可计算

$$\text{Gain} = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right) - \gamma$$

- 从左往右对于排序好的样例进行遍历，便能够决定此特征的最佳分裂点

GBDT梯度提升决策树

分裂点搜寻算法

- 对于每一个节点，枚举所有可选特征
  - 对于每一个特征，根据特征数值对样例进行排序
  - 以线性时间进行样例遍历并且寻找该特征之下的最佳分裂点
  - 对于所有特征都寻找出最佳分裂点
- 生长一棵深度为K的树所需时间复杂度
  - 复杂度为$O(ndK\log n)$:
    - 对于每一层，需要$O(n\log n)$的时间复杂度进行排序
    - 对于d个特征，需要在K层都进行排序操作
  - 可以进行进一步优化（例如：近似方法或将排序好的特征进行缓存）
  - 可以扩展到大规模数据集

GBDT梯度提升决策树

XGBoost

- 最有影响力以及最有效的GBDT工具包

https://xgboost.readthedocs.io

T Chen, C Guestrin. XGBoost: A Scalable Tree Boosting System. KDD 2016.

Zhi-hua Zhou, Ji Feng. Deep Forest: Towards an Alternative to Deep Neural Networks. IJCAI 2017.

本节材料来自
南京大学周志华教授

深度森林

讲师：张伟楠 – 上海交通大学

探寻深度学习的本质

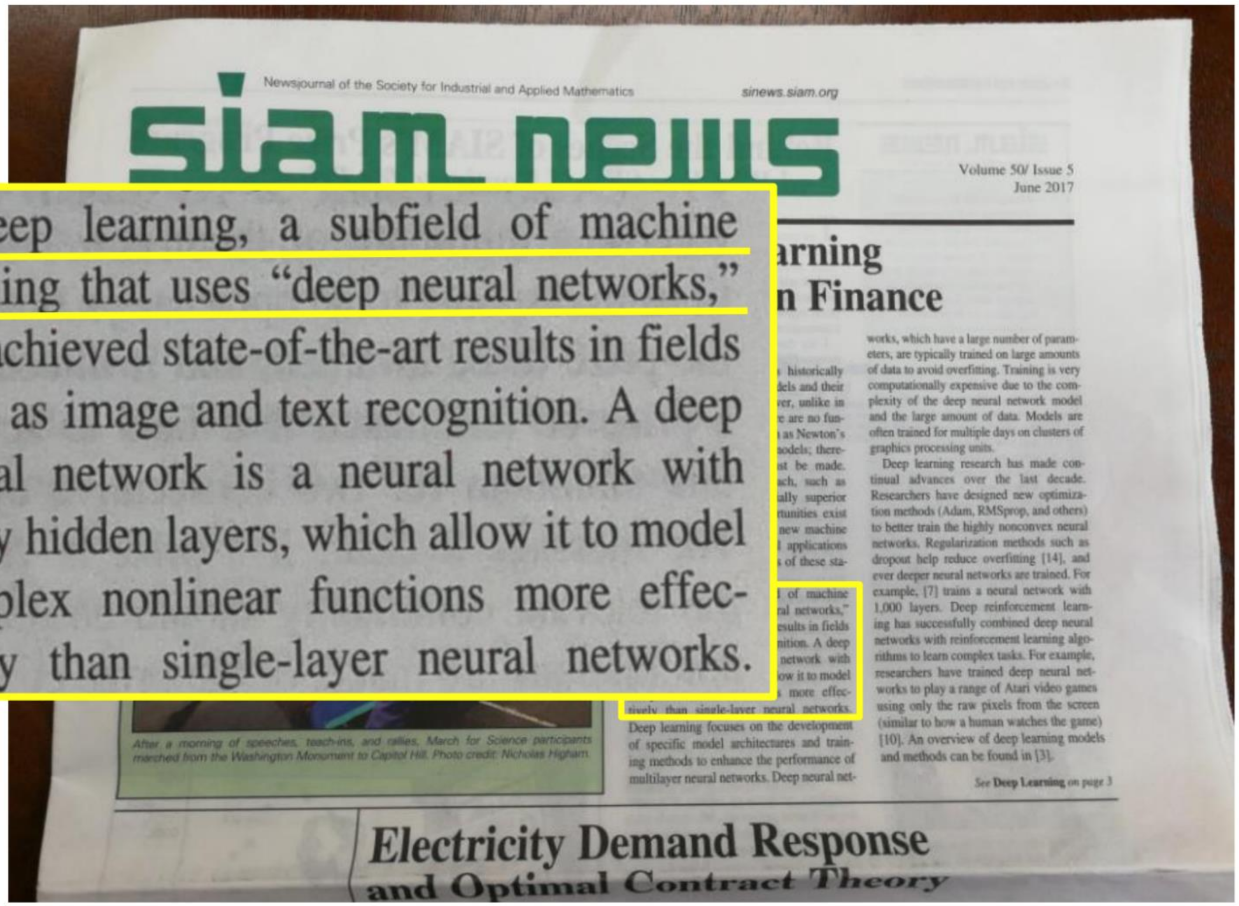
深度学习

- 近年来，深度学习取得了巨大成功

- 然而，“深度学习”到底是什么？
  - 现在而言，深度学习=深度神经网络（DNNs）
  - SIAM (Society for Industrial and Applied Mathematics)的例子

探寻深度学习的本质

深度学习


Newsjournal of the Society for Industrial and Applied Mathematics sinews.siam.org

siam news

Volume 50/ Issue 5
June 2017

...arning
...n Finance

Deep learning, a subfield of machine learning that uses "deep neural networks," has achieved state-of-the-art results in fields such as image and text recognition. A deep neural network is a neural network with many hidden layers, which allow it to model complex nonlinear functions more effectively than single-layer neural networks.

After a morning of speeches, teach-ins, and rallies, March for Science participants marched from the Washington Monument to Capitol Hill. Photo credit: Nicholas Higham.

See Deep Learning on page 3

Electricity Demand Response and Optimal Contract Theory

探寻深度学习的本质

深度模型的一种解释

- 增加模型复杂度→增强学习能力
  - 增加隐层单元数（模型宽度）
  - 增加隐层的层数（模型深度）

增加隐层的层数更为高效：
- 增加了具有激活函数的单元数
- 增加了函数的嵌入深度

- 增加模型复杂度→ 加大过拟合风险以及训练难度
  - 针对过拟合：提供大量训练数据
  - 针对训练：提供高效的算力机器

当经过多层的反向传播，误差的梯度将会发散，收敛到稳定状态将会非常困难：
- 传统的反向传播算法将会失效，需要使用许多技巧进行训练

探寻深度学习的本质

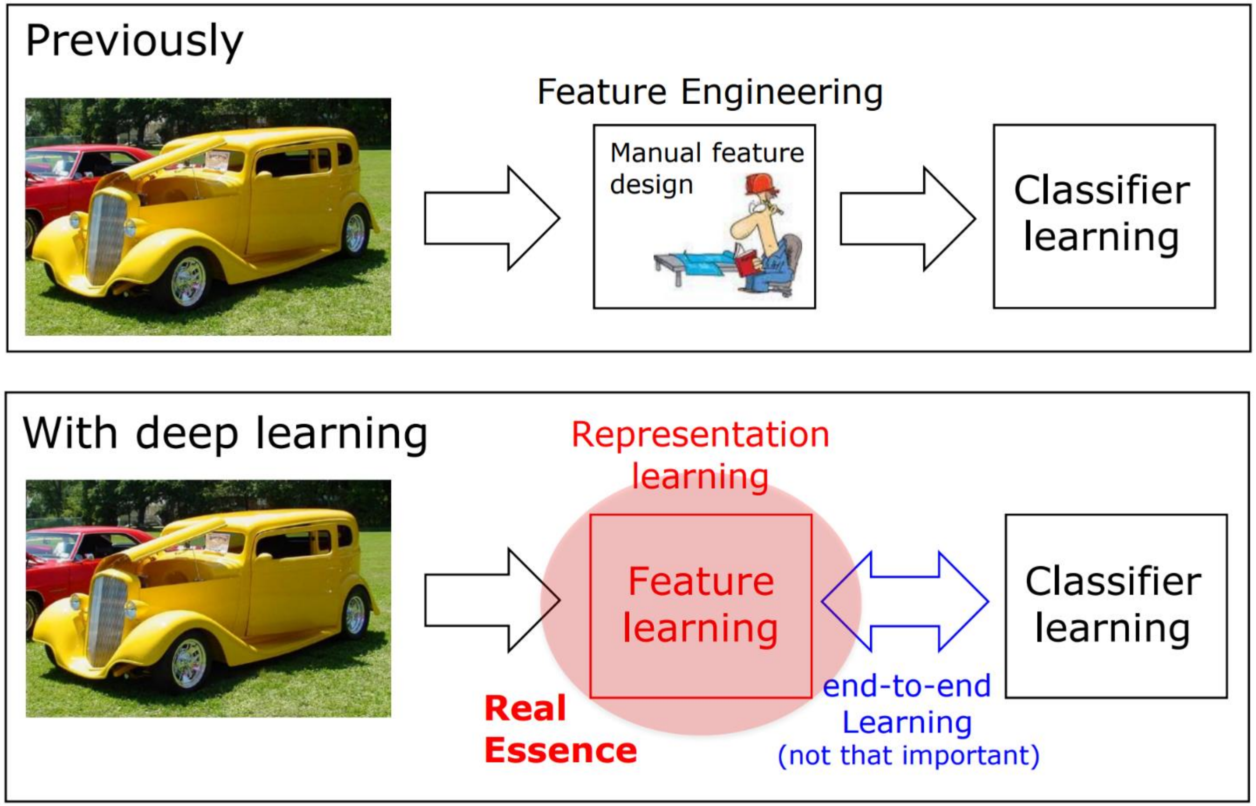
DNNs成功的关键

- 庞大的训练数据
  - 减小过拟合风险最为简单有效的方式
- 高效的算力机器
  - 针对大模型：没有GPU的加速，DNNs无法如此成功
- 训练技巧
  - 启发式，甚至是无法解释的
  - 使用传统的反向传播算法，误差的梯度可能会在多层传播以后发散，很难收敛到稳定状态

探寻深度学习的本质

DNNs的本质

- 表示学习



探寻深度学习的本质

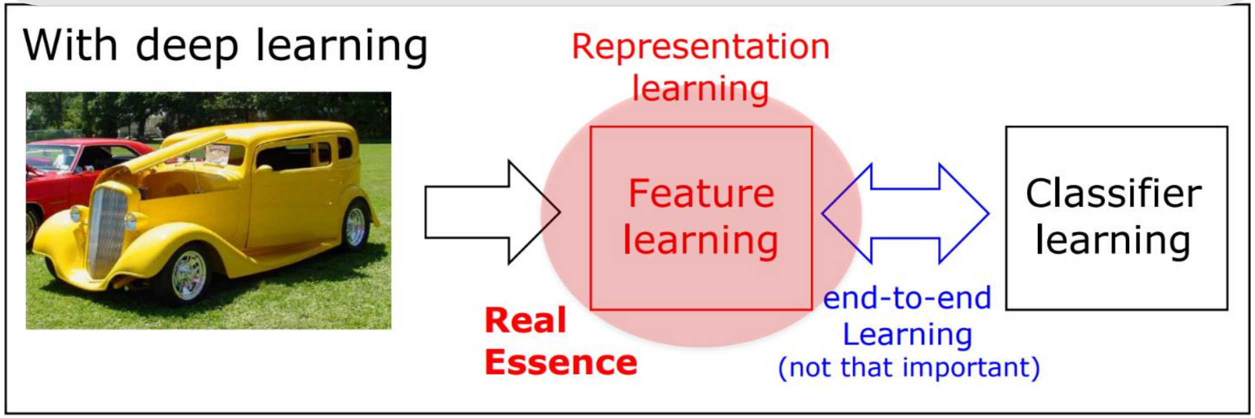
DNNs的本质

- 表示学习

对于何种类型的任务有效?

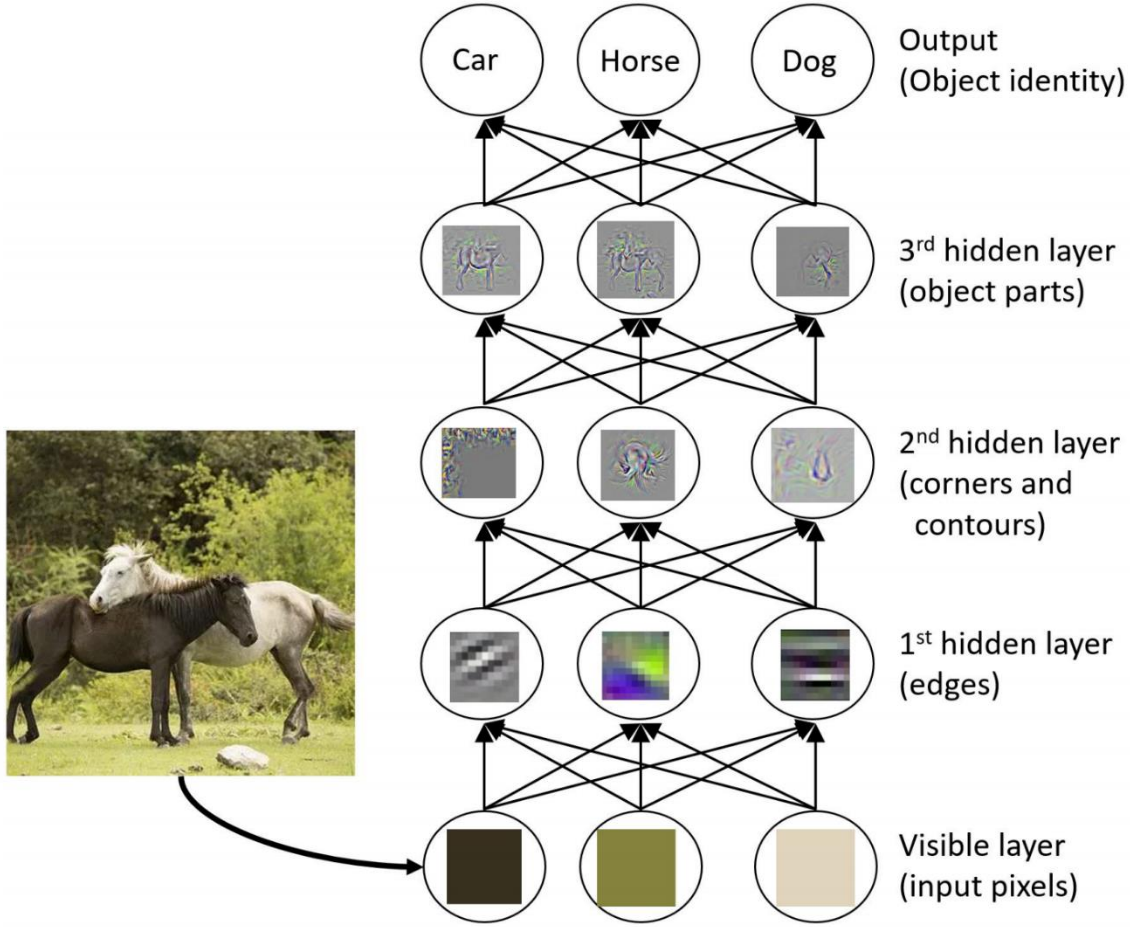
数据的“原始表示”（例如，图像的像素点）

距离对任务而言“足够好的表示”很远



探寻深度学习的本质

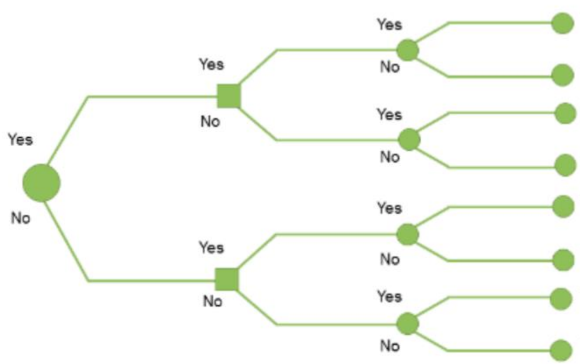
逐层学习很关键



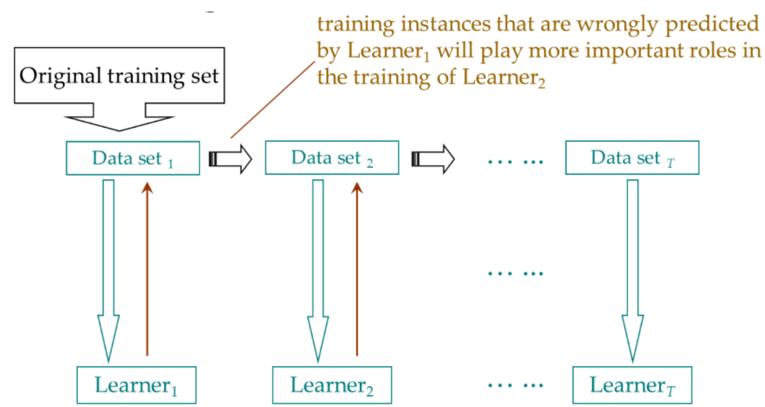
探寻深度学习的本质

其他模型

- 决策树?



- 提升算法?



- 逐层计算
  - 复杂度不够
  - 总是针对原始特征

- 非逐层计算
  - 复杂度不够
  - 总是针对原始特征

探寻深度学习的本质

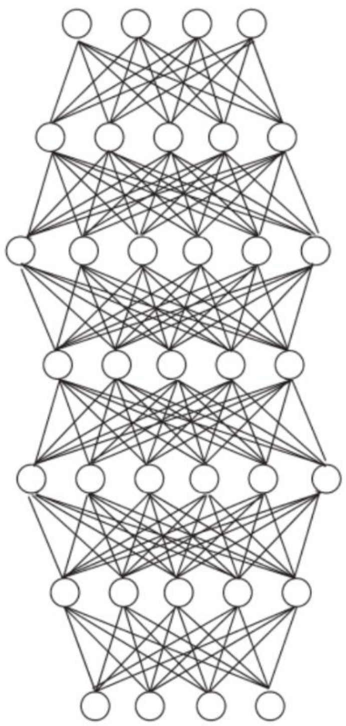
深度模型的关键

- 逐层处理
- 特征变换
- 足够的模型复杂度

探寻深度学习的本质

使用神经网络

- 过多的超参数
  - 调参需要许多小技巧，特别是跨任务训练时
  - 很难复现他人的结果，
    - 例如：当大家都使用CNNs时，他们其实也在使用不同的模型，因为卷积层的结构可能不尽相同
- 当结构固定以后，模型复杂度也随之确定，这一复杂度往往超过了需求
- 需要大量训练数据
- 理论分析十分困难
- 黑盒问题
- …



探寻深度学习的本质

除此以外

- 在许多的任务中，DNNs并没有优越性，有时甚至NNs带来不足
  - 数据不同维度的连续/离散性质、不同的规模、有缺失
  - 在许多Kaggle竞赛任务中，随机森林或者XGBoost往往是参赛者的最终选择

探寻深度学习的本质

巨大的挑战

是否可以使用不可微的模块
来实现深度学习？

- 这个挑战在理解层面十分关键：
  - 深度模型？ =DNNs
  - 不可微模块可以做深度学习吗？ （不使用反向传播）
  - 可以让深度模型在更多任务上胜出吗?
  - …

探寻深度学习的本质

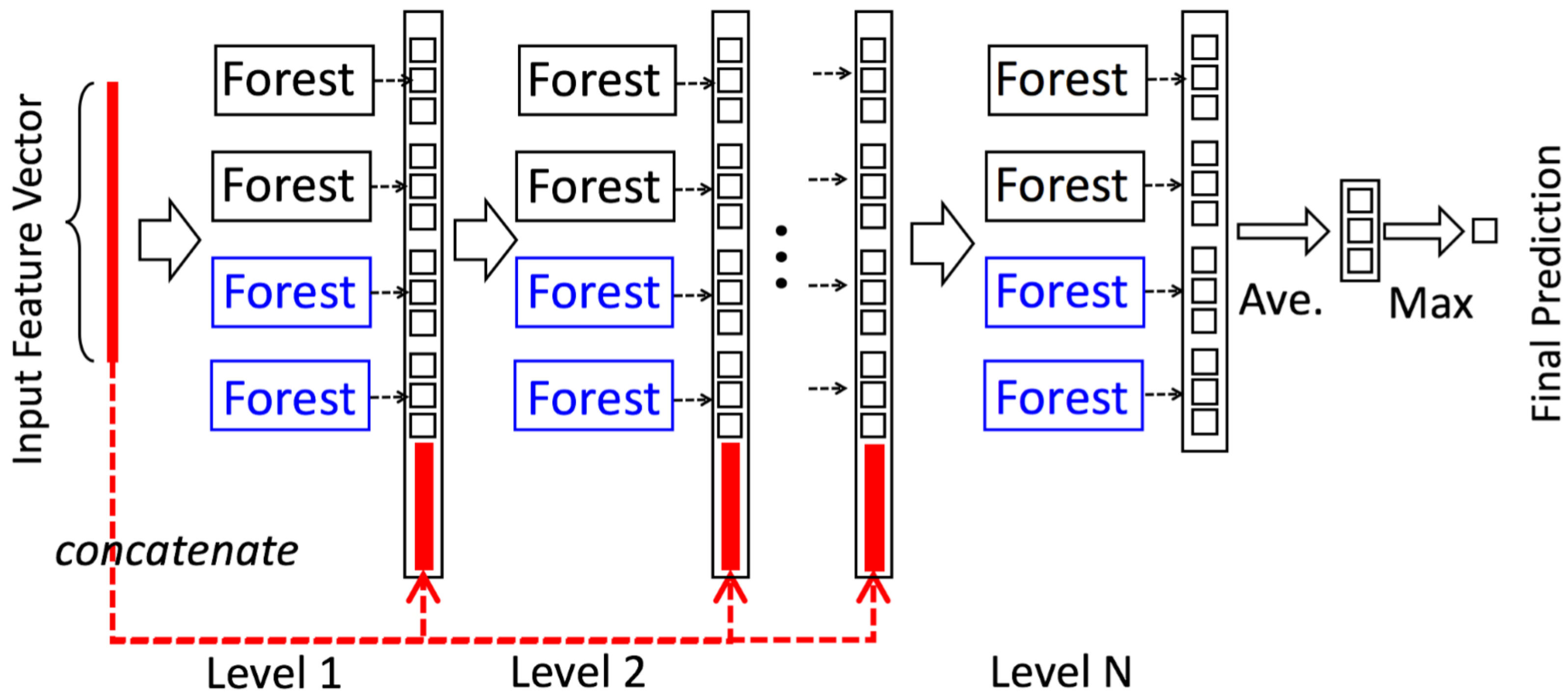
灵感启发

- 设计的模型需要具有
  - 逐层处理
  - 特征变换
  - 足够的模型复杂度
- 如何设计？深度森林是一种可能的方案

深度森林

灵感启发

- 用堆叠森林模型来实现：逐层处理、特征变换、足够的模型复杂度



多粒度级联森林

讲师：张伟楠 – 上海交通大学

多粒度级联森林

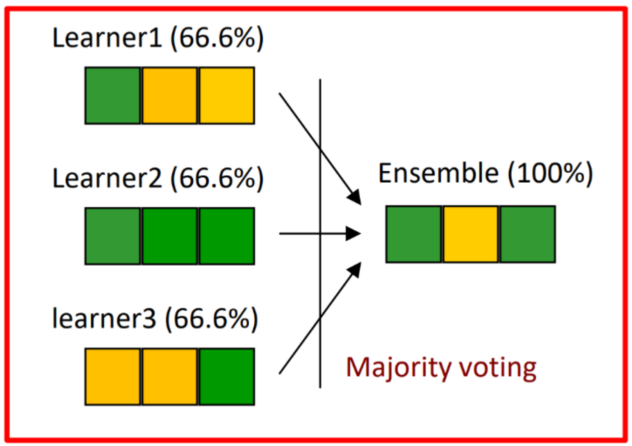
gcForest简介

- 多粒度级联森林gcForest(multi-Grained Cascade Forest)
  - 一种决策树森林（集成学习）方法
  - 与DNNs相比，在多个任务上表现都具有很强竞争力
  - 超参数减少明显
    - 更加容易设定参数
    - 默认设定在多个任务上都可以很有效
  - 自适应的模型复杂度
    - 根据数据集自动决定
    - 可以在小数据集上应用
  - …

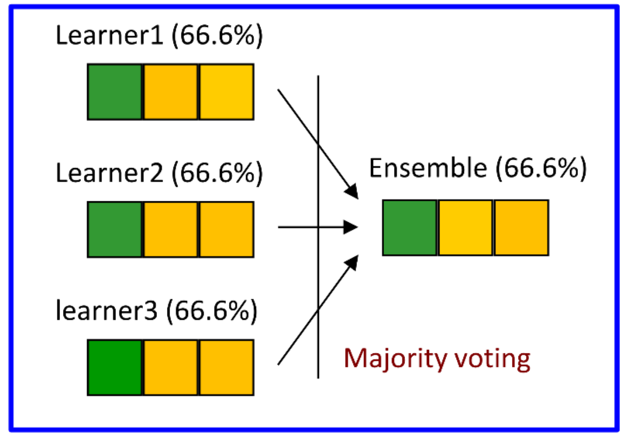
多粒度级联森林

好的集成方式

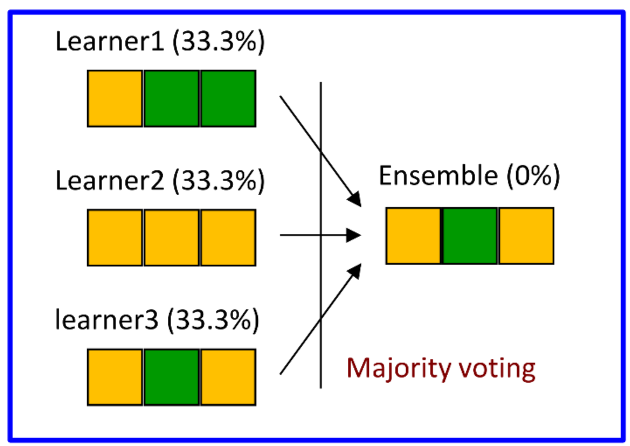
- 直觉而言：



集成的确会有帮助



个体必须存在差异



个体必须不能太差

多粒度级联森林

差异生成

- 根据误差-分歧分解（Error-ambiguity decomposition）

  [Krogh & Vedelsby, NIPS’95]:

$$\boxed{E} = \boxed{\bar{E}} - \boxed{\bar{A}}$$

集成误差　平均个体误差　平均个体分歧

- 单个学习模型越准确越具有差异，集成模型表现越佳

- 需要注意
  - “分歧”并没有一个形式化定义
  - 误差-分歧分解是在使用平方损失的回归问题中推导得到的

多粒度级联森林

差异性是关键

- 基本思想：加入更多的随机性
- 主要策略：
  - 数据采样的处理
    - Bagging中的自举采样
    - Boosting中的重要性采样
  - 输入特征的处理
    - 在随机子空间中进行特征采样
  - 可学习参数的处理
    - NN的随机初始化Random Initialization [Kolen &Pollack, NIPS’91]
    - 负相关学习Negative Correlation [Liu & Yao, NNJ 1999]
  - 输出表示的处理
    - ECOC [Dietterich & Bakiri, JAIR 1995]
    - 输出翻转Flipping Output [Breiman, MLJ 2000]

这些策略并非一直有效：

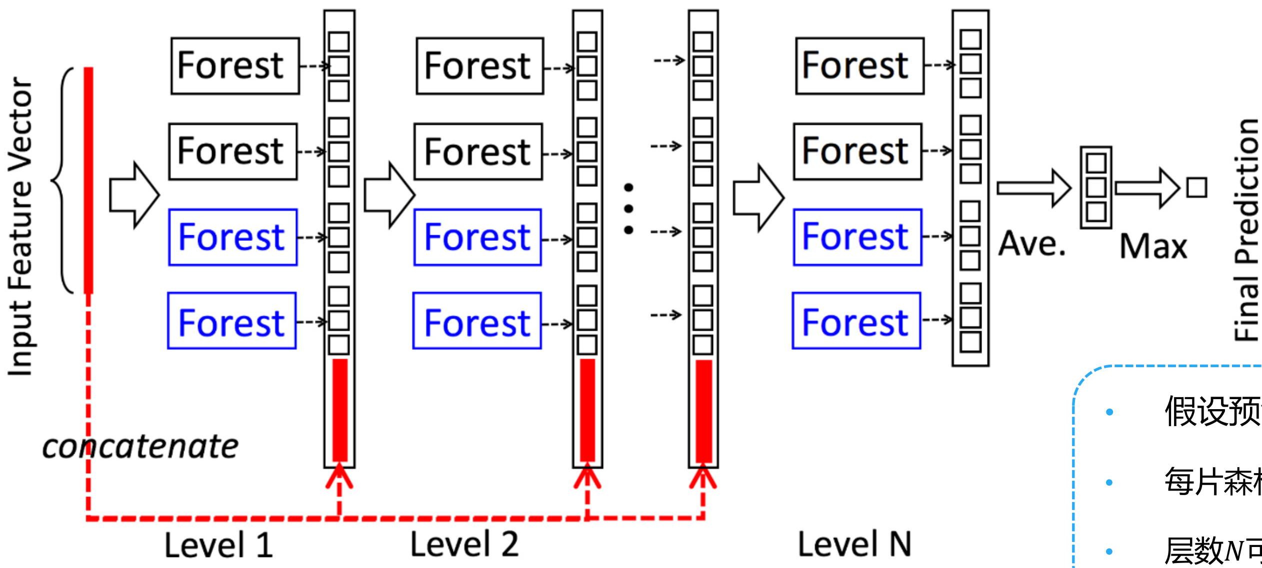
- 数据采样的处理对于“静态学习器”，如线性分类器、SVMs并不生效

采用多种策略结合：

- 随机森林
- FASBIR [Zhou & Yu, TSMCB 2005]

多粒度级联森林

级联森林结构

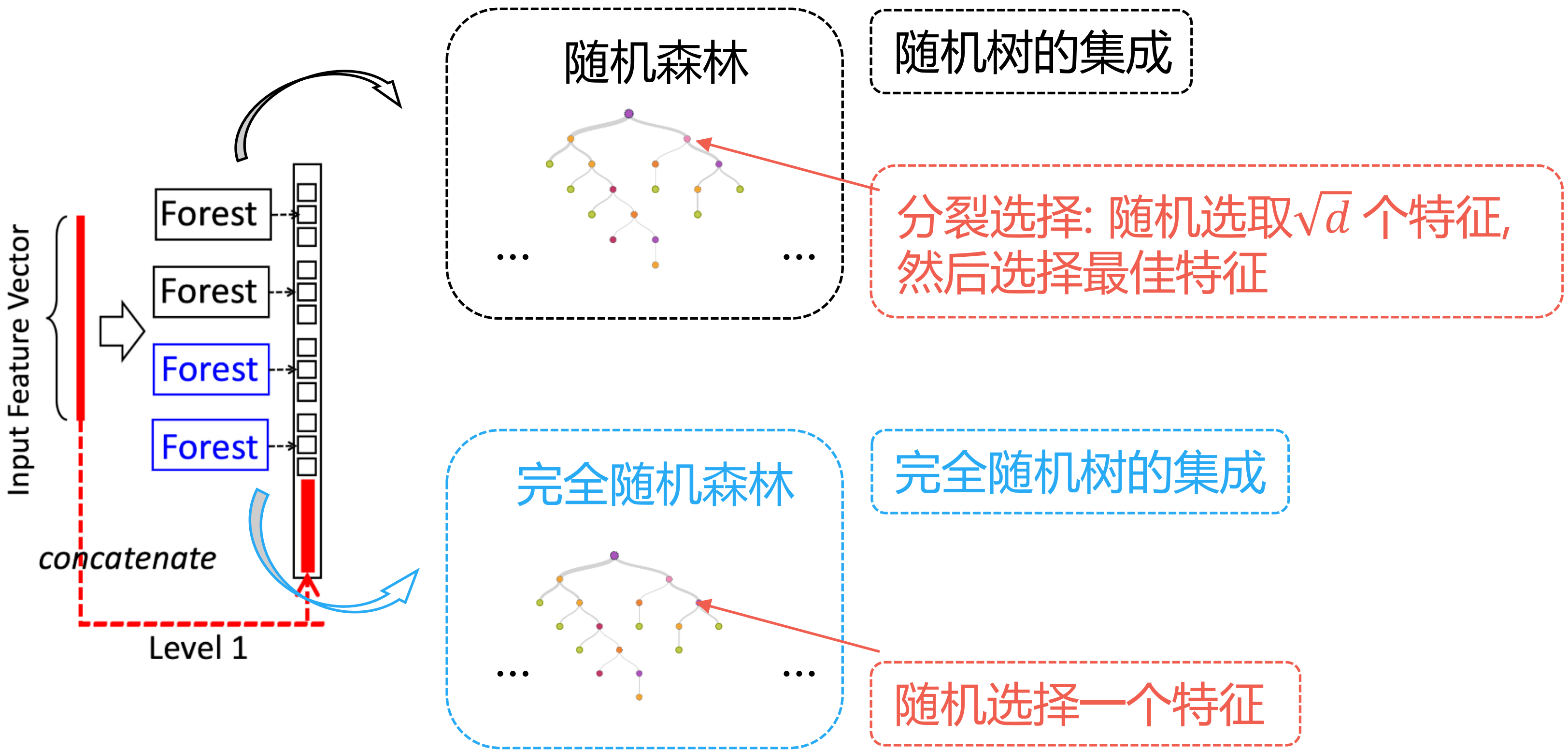


- 假设预测三个类别
- 每片森林输出一个三维的类别向量
- 层数*N*可以由交叉验证决定

- 将多片森林的输出作为新的输入，传入另一层的多片森林：
  - 类似于Stacking [Wolpert, NNJ 1992; Breiman, MLJ 1996]，著名集成学习方法
  - Stacking一般只对一到两层进行，多了容易过拟合，其并不能应用于深度模型中

多粒度级联森林

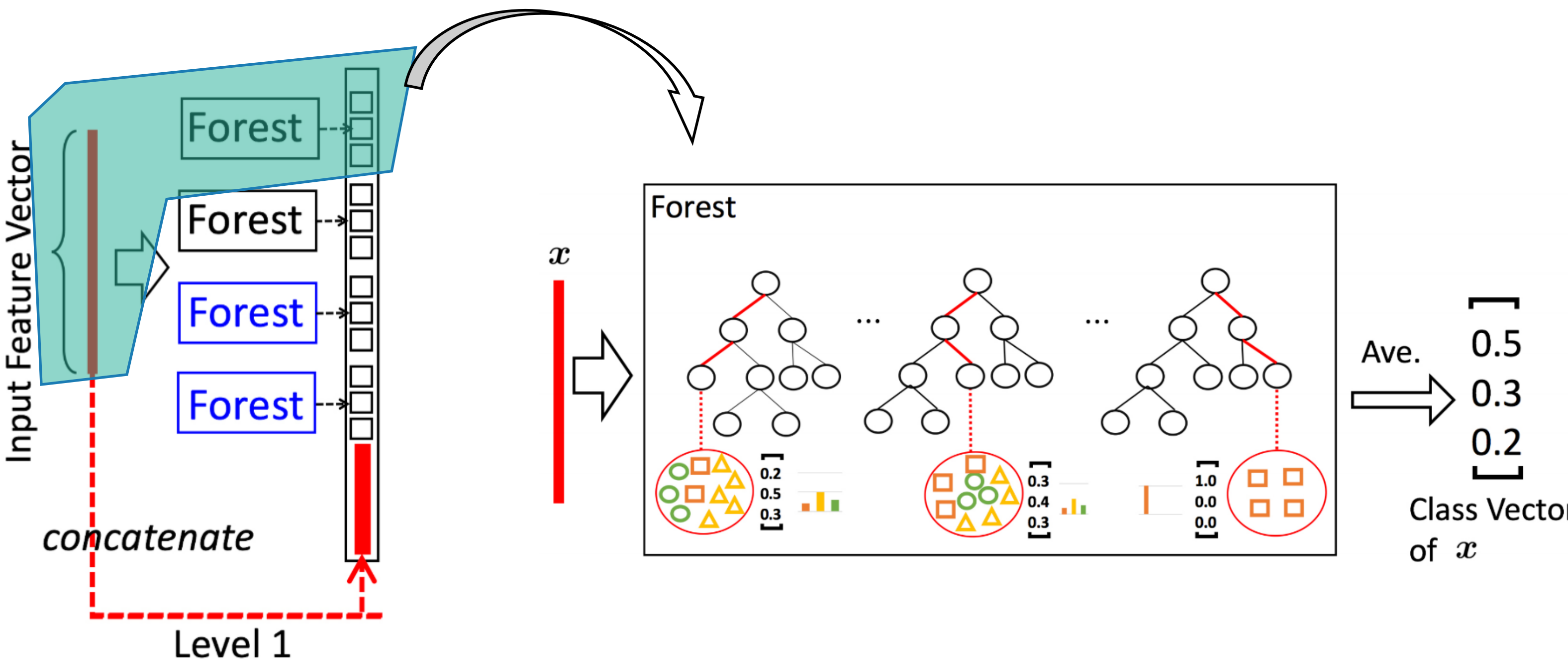
集成模型的集成



- 采用不同种类的森林：鼓励集成模型的集成时的多样性

多粒度级联森林

类别向量的生成



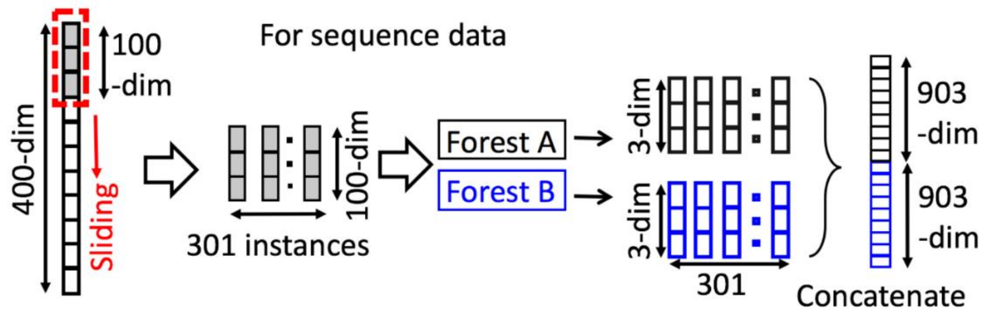
- 对于k分类任务，每片森林输出k维向量，对应每个类别的分类概率

多粒度级联森林

滑动窗口扫描

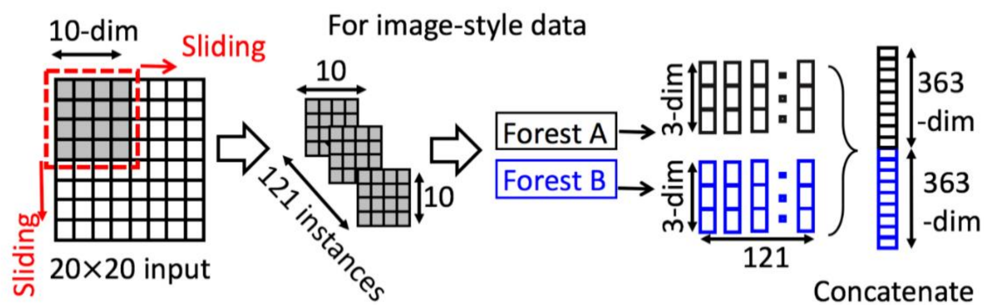
- 受到CNNs以及RNNs的启发，挖掘空间及序列的关联

重新表示



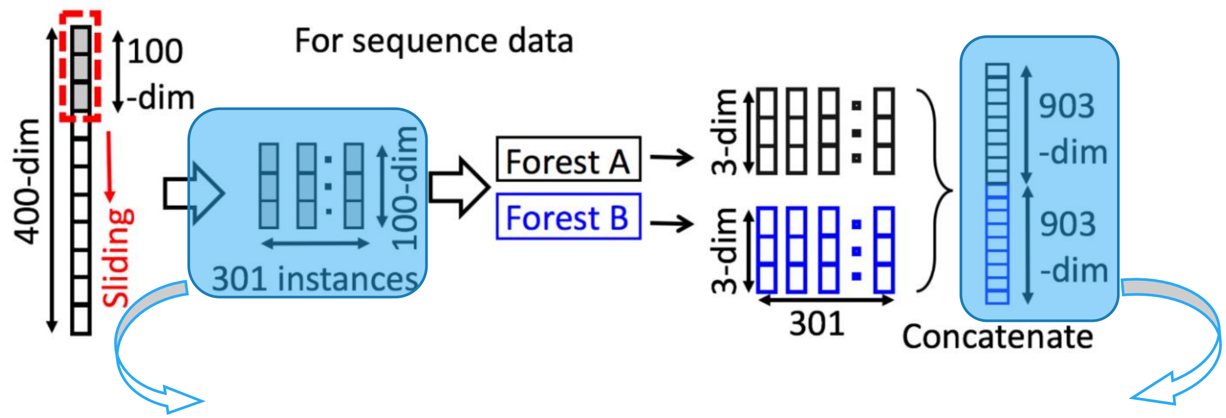
所有301个样例都具有与原来样例相同的标签

输出翻转Flipping Output[Breiman, MLJ 2000]相关,一种鼓励更多的集成差异的输出处理



多粒度级联森林

滑动窗口扫描



针对高维数据 ...

过多的样例以及过大的向量长度如何处理?

特征采样

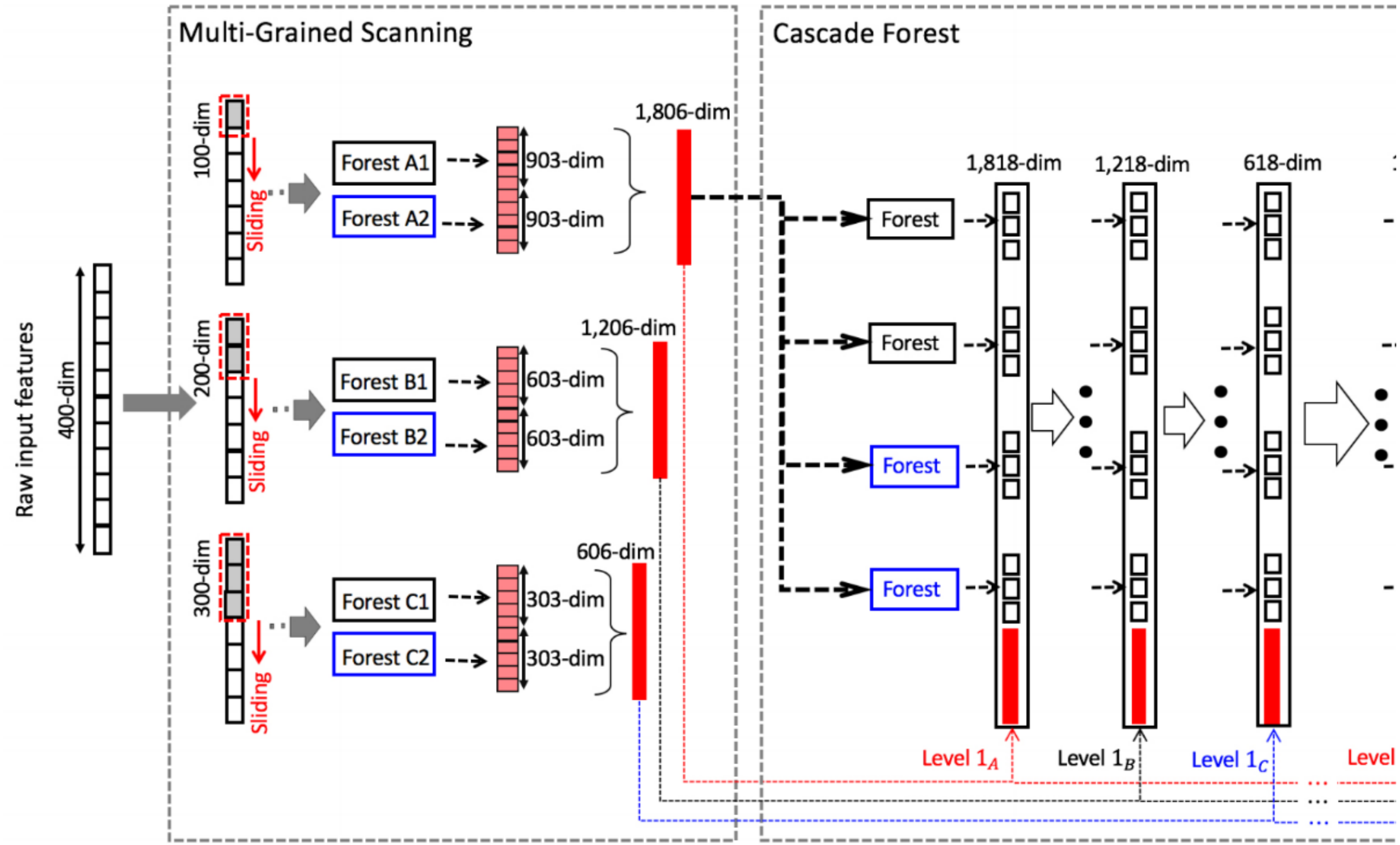
(例如，对于样例进行降采样)

与Random Subspace [Ho, TPAMI 1998]相关，一种鼓励更多集成差异的特征处理方法

- 完全随机树不依赖于分裂特征选取
- 随机森林对于分裂特征的扰动十分敏感

多粒度级联森林

多粒度

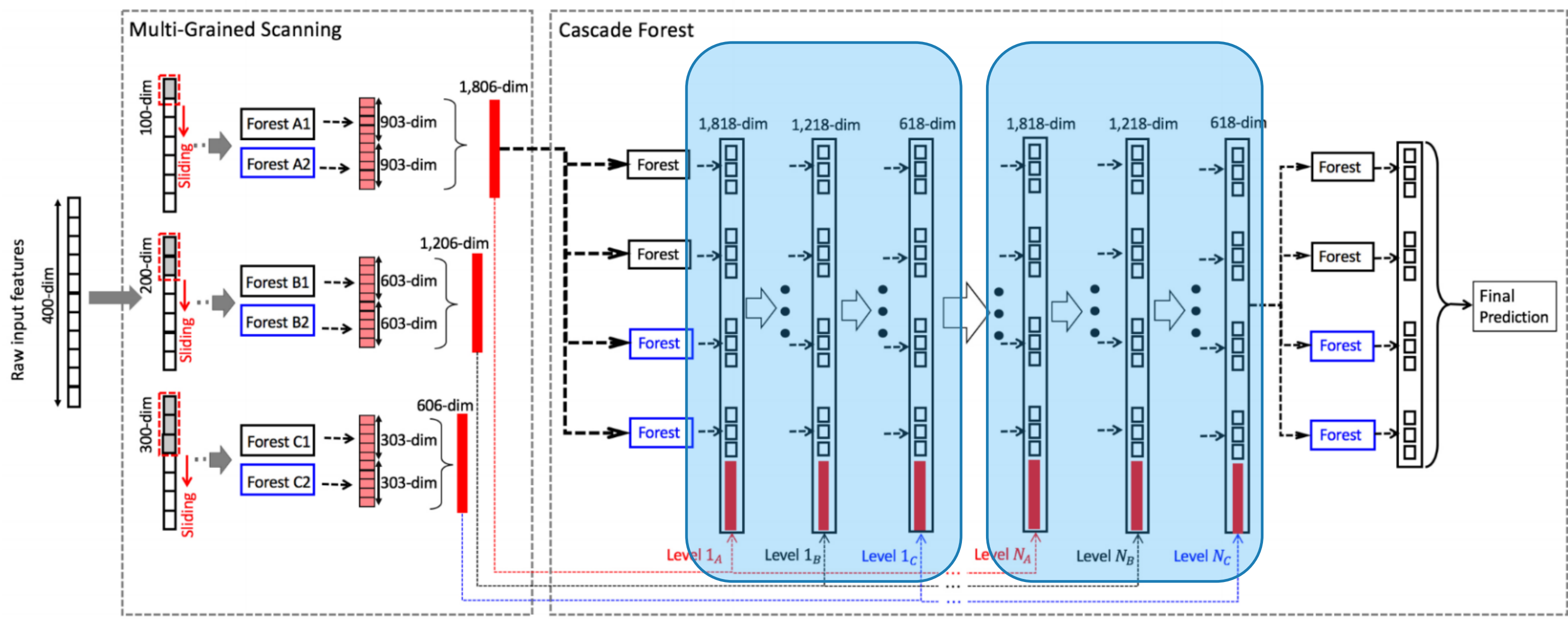


Multi-Grained Scanning
Cascade Forest
Raw input features
400-dim
100-dim
200-dim
300-dim
Sliding
Forest A1
Forest A2
Forest B1
Forest B2
Forest C1
Forest C2
903-dim
903-dim
603-dim
603-dim
303-dim
303-dim
1,806-dim
1,206-dim
606-dim
Forest
Forest
Forest
Forest
1,818-dim
1,218-dim
618-dim
Level 1A
Level 1B
Level 1C
Level

多粒度级联森林

总体架构

级联间的级联



粒度扫描：

- 每片森林500棵树
- 树的生长：直到纯的叶子或者深度=100
- 滑动窗口大小$[d/16],[d/8],[d/4]$

级联：

- 每片森林500棵树
- 树的生长：直到纯的叶子

多粒度级联森林

超参数

Deep neural networks (e.g., convolutional neural networks)	gcForest
Type of activation functions:	Type of forests:
Sigmoid, ReLU, tanh, linear, etc.	Completely-random tree forest, random forest, etc.
Architecture configurations:	Forest in multi-grained scanning:
No. Hidden layers: ?	**No. Forests**: {2}
No. Nodes in hidden layer: ?	**No. Trees in each forest**: {500}
No. Feature maps: ?	Tree growth: till pure leaf, or reach depth 100
Kernel size: ?	**Sliding window size**: $\{\lfloor d/16\rfloor, \lfloor d/8\rfloor, \lfloor d/4\rfloor\}$
Optimization configurations:	Forest in cascade:
Learning rate: ?	**No. Forests**: {8}
Dropout: {0.25/0.50}	**No. Trees in each forest**: {500}
Momentum: ?	Tree growth: till pure leaf
L1/L2 weight regularization penalty: ?	
Weight initialization: Uniform, glorot_normal, glorot_uni, etc.	
Batch size: {32/64/128}	

- 在实验中：
  - gcForest对于所有数据都使用相同的超参数
  - DNNs对于每个数据集都进行了精细的调参

多粒度级联森林

实验结果

- 图像识别（MNIST）

gcForest	**99.26%**
LeNet-5	99.05%
Deep Belief Net	98.75% [23]
SVM (rbf kernel)	98.60%
Random Forest	96.80%

[Hinton *et al.*, 2006]

- 人脸识别（ORL）

	5 image	7 images	9 images
gcForest	**91.00%**	**96.67%**	**97.50%**
Random Forest	91.00%	93.33%	95.00%
CNN	86.50%	91.67%	95.00%
SVM (rbf kernel)	80.50%	82.50%	85.00%
kNN	76.00%	83.33%	92.50%

多粒度级联森林

实验结果

音乐分类（GTZAN）

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

情感分类（IMDB）

gcForest	**89.16%**
CNN	89.02% [26]
MLP	88.04%
Logistic Regression	88.62%
SVM (linear kernel)	87.56%
Random Forest	85.32%

[Kim et al., 2014]

手掌运动识别（sEMG）

gcForest	**71.30%**
LSTM	45.37%
MLP	38.52%
Random Forest	29.62%
SVM (rbf kernel)	29.62%
Logistic Regression	23.33%

低维数据（特征：16，14，8）

	LETTER	ADULT	YEAST
gcForest	**97.40%**	**86.40%**	**63.45%**
Random Forest	96.50%	85.49%	61.66%
MLP	95.70%	85.25%	55.60%

多粒度级联森林

更多实验

- 图像分类（CIFAR-10）
  - 彩色32×32的图片
  - 训练：50000张10类的图片；测试：10000张图片

ResNet	93.57% [20]
AlexNet	83.00% [30]
gcForest(gbdt)	**69.00%**
gcForest(5grains)	**63.37%**
Deep Belief Net	62.20% [31]
gcForest(default)	**61.78%**
Random Forest	50.17%
MLP	42.20% [1]
Logistic Regression	37.32%
SVM (linear kernel)	16.32%

使用GBDT替代最后一层

使用5个滑动窗口

使用默认参数

- gcForest已经是非DNN方法中效果最好的方法
  - DNNs方法在图像相关的任务中效果极好
  - 深度森林在许多其他任务中会更有帮助

多粒度级联森林

gcForest和深度森林

- gcForest作为集成模型的成功应用
  - 差异性是集成模型的关键
  - gcForest几乎使用了所有策略来增加模型的差异性

- gcForest只是深度森林方法的开始
  - 深度森林有许多可能性去探索
  - 深度森林在许多其他任务中可以尝试

总结集成学习

- Bagging有效的原因是与原始模型相比偏差相同但是降低了方差
- 集成学习通过构建和组合多个‘不同的’学习器，达到更好的效果
  - 多样化的学习器
- 随机森林通过随机化节点分裂时的候选特征集合，从而增大集成学习的单模型多样性
- Boosting算法则直接通过最小化损失函数作为目标来构建下一个学习器，GBDT则是通过构建下一棵树来最小化当前的损失
- 深度森林则从深度这个维度进一步探索森林在堆叠的构架下树模型能否取得深度学习的卓越效果

THANK YOU