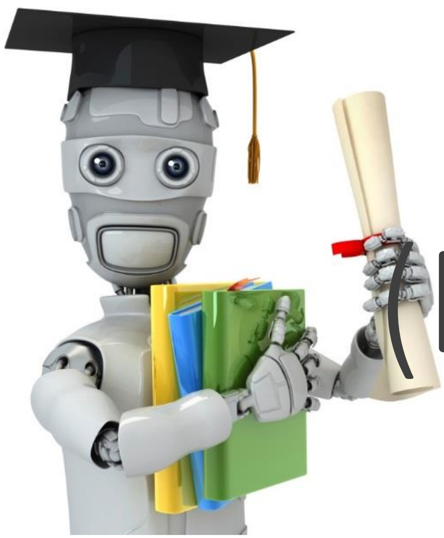


集成学习 (Ensemble Learning)

Machine Learning

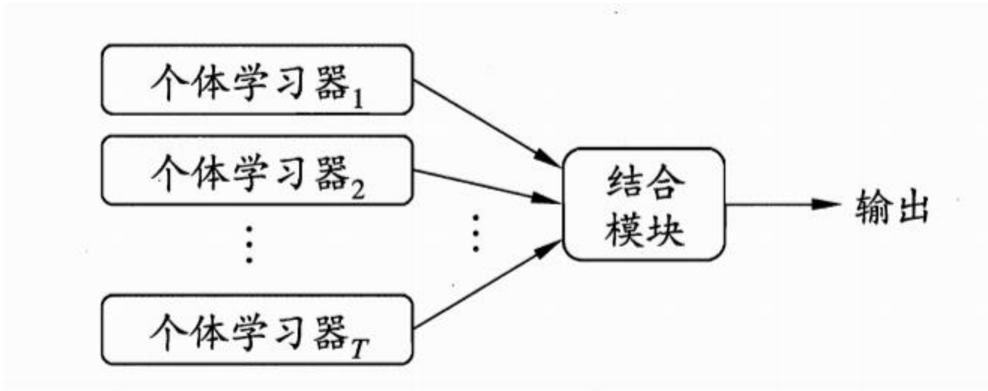
梁毅雄

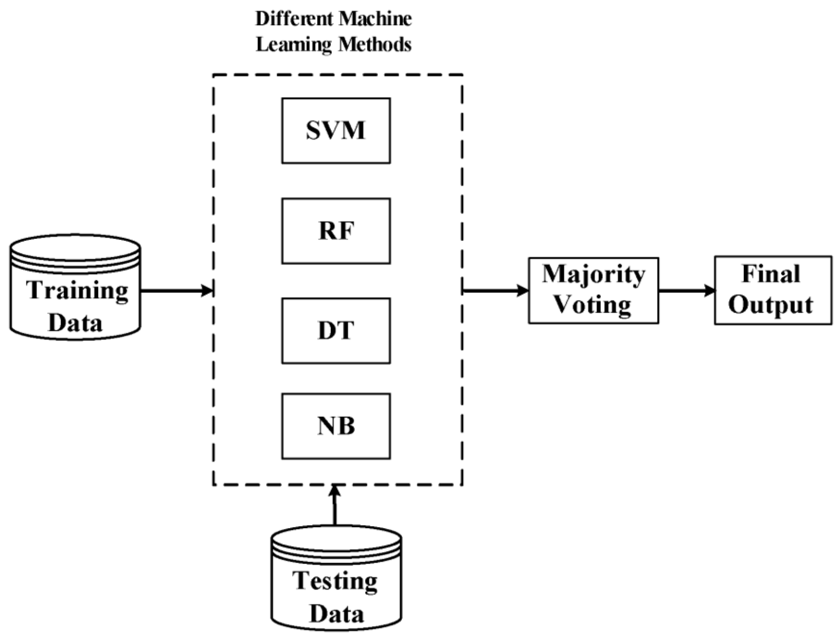
yxliang@csu.edu.cn

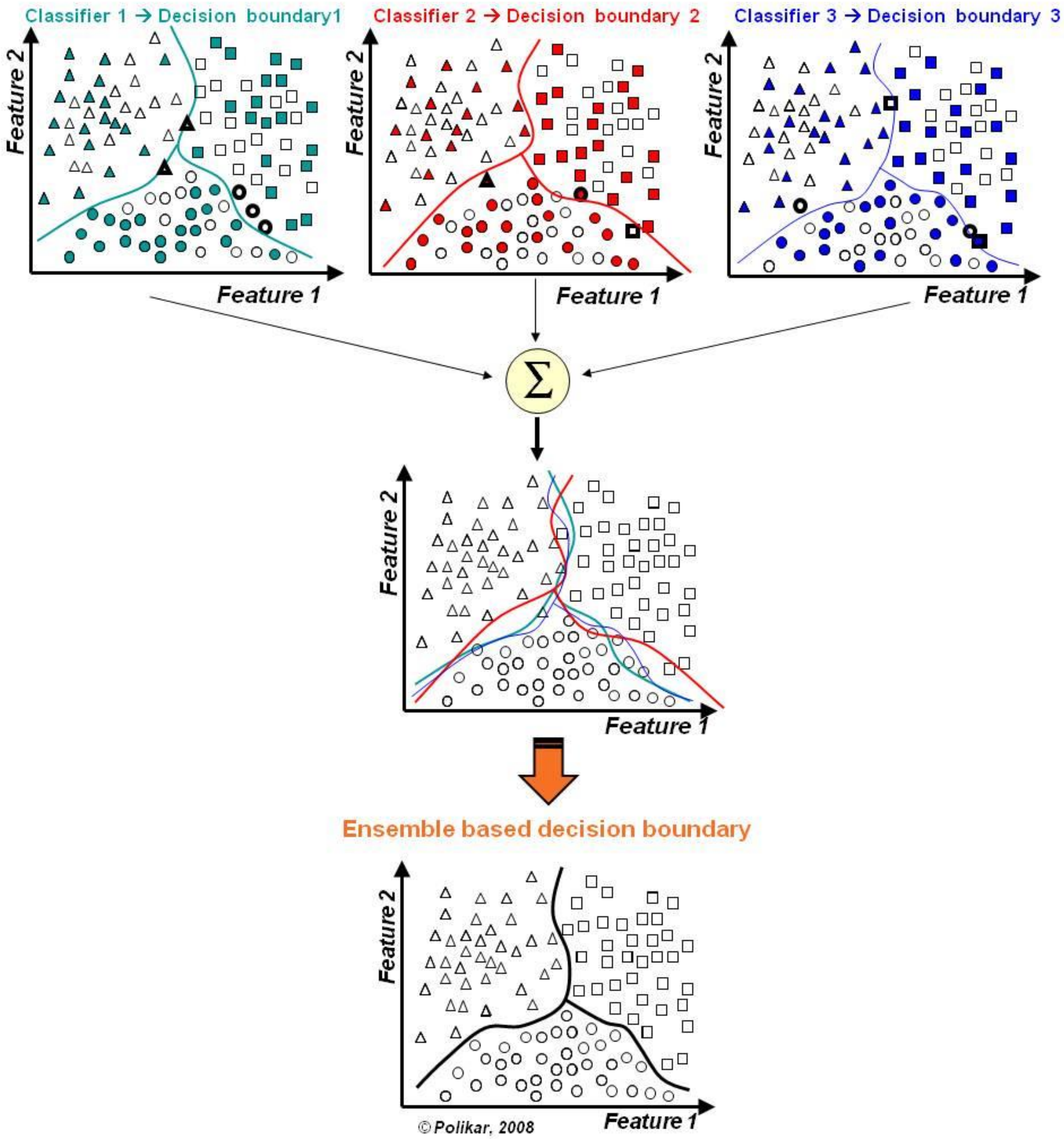
Some materials from Hang Li, Hsuan-Tien Lin and others

集成学习(Ensemble Learning)

- 集成学习：
  - 构建多个学习器一起结合来完成具体的学习任务
  - 也称为Multi-Classifier System, Committee-Based Learning
  - 学习器可以是同类型的，也可以是不同类型
  - 通过将多个学习器进行结合，常可获得比单一学习器显著优越的泛化性能，对“弱学习器”尤为明显（三个臭皮匠，顶个诸葛亮）

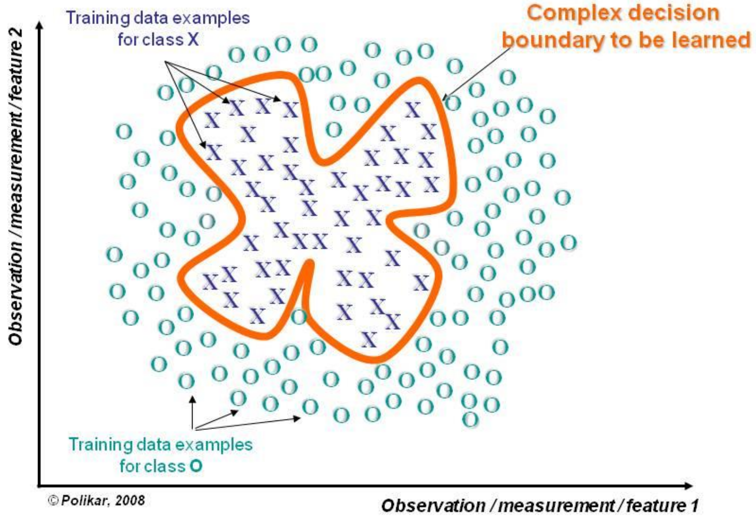


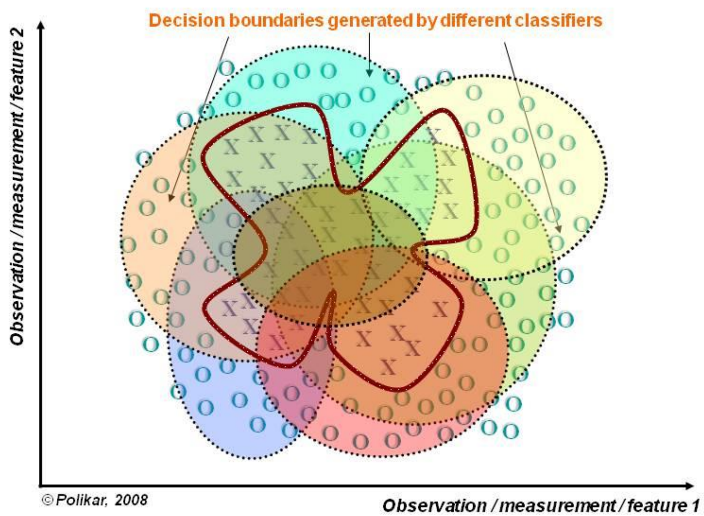




http://www.scholarpedia.org/article/Ensemble_learning

集成学习(Ensemble Learning)



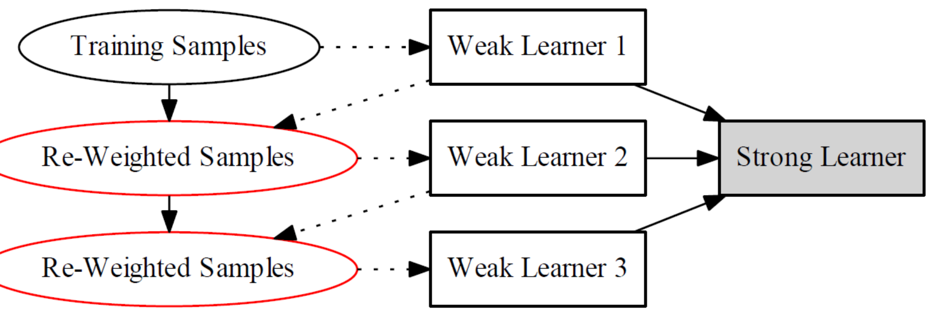


http://www.scholarpedia.org/article/Ensemble_learning

集成学习(Ensemble Learning)

- 集成学习
  - 理论分析指出：假设各分类器的错误率相互独立，则随着集成中个体分类器数目*T*的增大，集成的错误率将指数级下降
  - 现实中个体学习器是为解决同一个问题训练出来的，不可能相互独立
  - 如何产生并结合“好而不同”的个体学习器是集成学习研究的核心
- 集成学习分类
  - 个体学习器间存在强依赖关系，必须串行生成的序列化方法。代表：Boosting（AdaBoost, Gradient Boosting Machine)
  - 个体学习器间不存在强依赖关系，可同时生成的并行化方法。代表：Bagging和随机森林（Random Forest）

AdaBoost



AdaBoost

$$\underbrace{\underbrace{\text{Ada}}_{\text{Adaptive}}\ \underbrace{\text{Boost}}_{\text{Boosting}}}_{\text{A learning algorithm}}$$

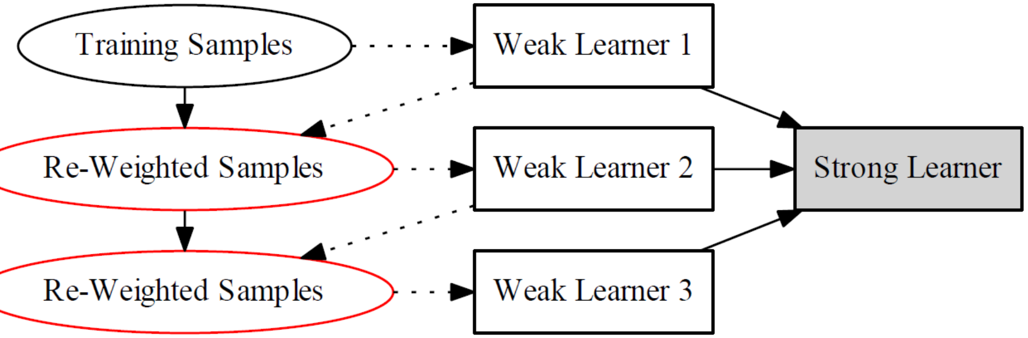
Building a strong classifier a lot of weaker ones

$$\left.\begin{array}{l} h_1(x) \in \{-1,+1\} \\ h_2(x) \in \{-1,+1\} \\ \vdots \\ h_T(x) \in \{-1,+1\} \end{array}\right\} \quad H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Weak classifiers   strong classifier

slightly better than random

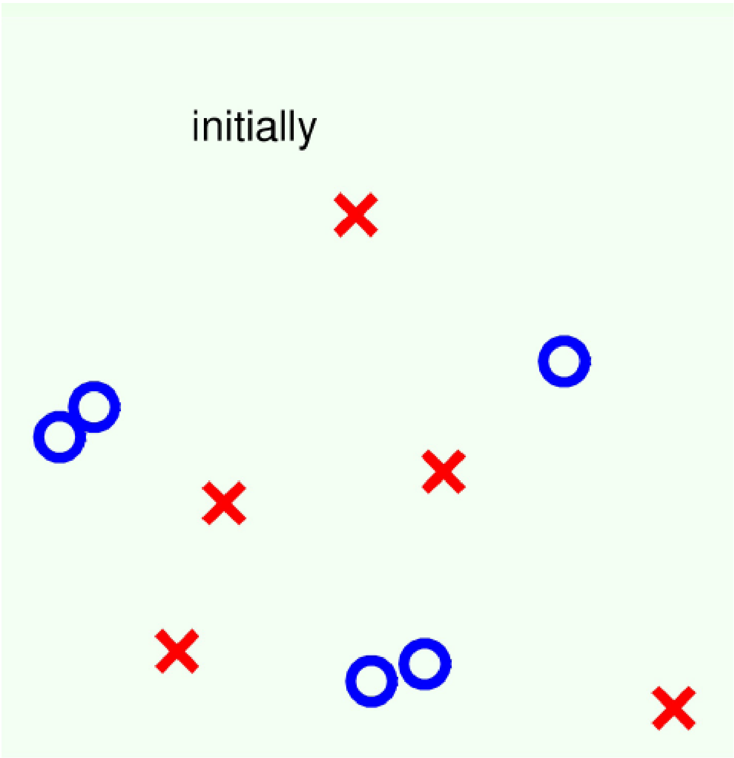
AdaBoost

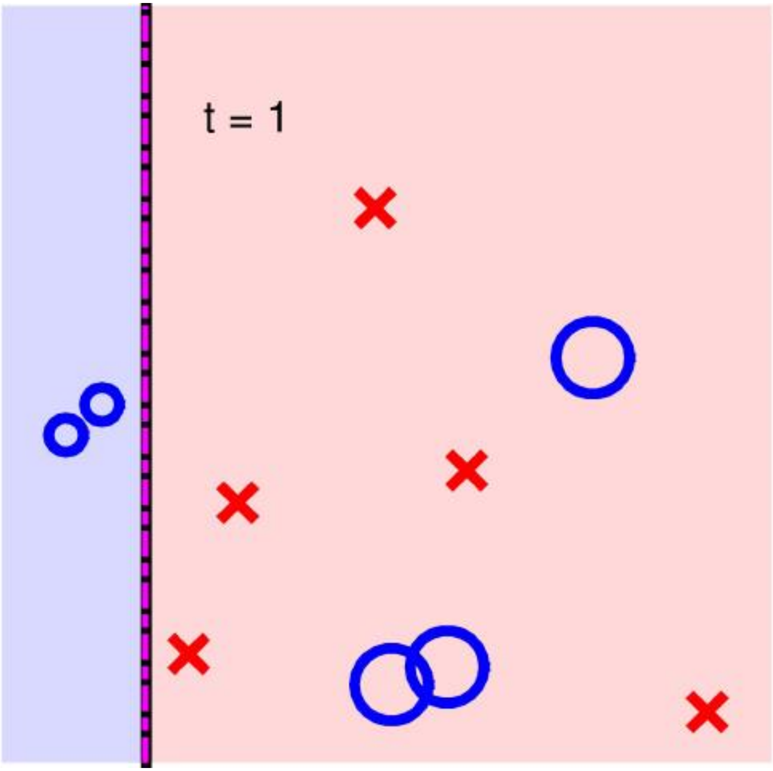


- AdaBoost的主要思想：
  - 先训练出一个基学习器；
  - 根据该学习器的表现对训练样本分布进行调整，使得现有基学习器做错的样本在后续学习器的训练中受到更多关注；
  - 基于调整后的样本分布来训练下一个基学习器；
  - 如此重复进行直至基学习器数目达到事先指定的值*T*；
  - 最终将这*T*个基学习器进行加权结合

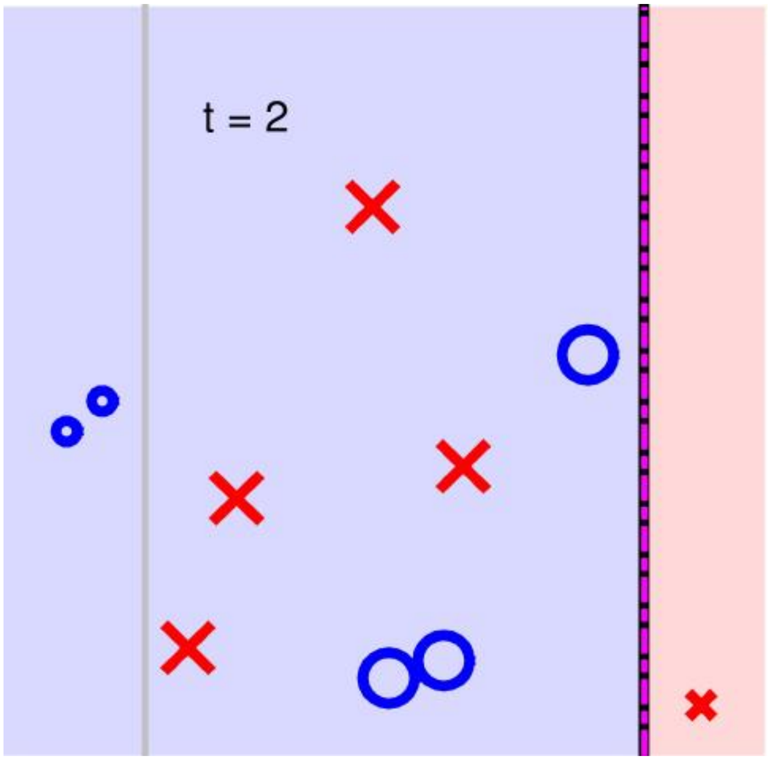
$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

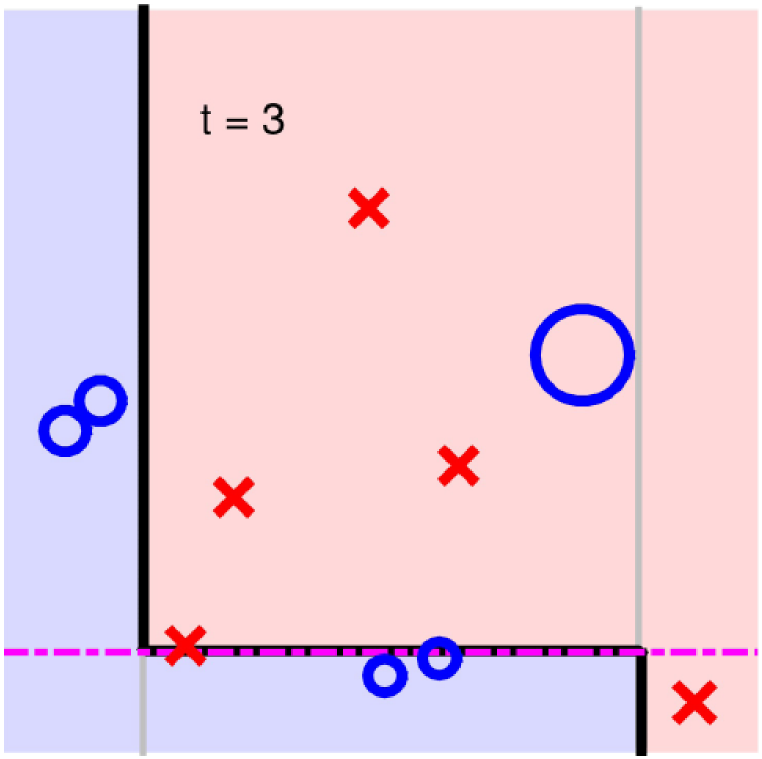
AdaBoost Example



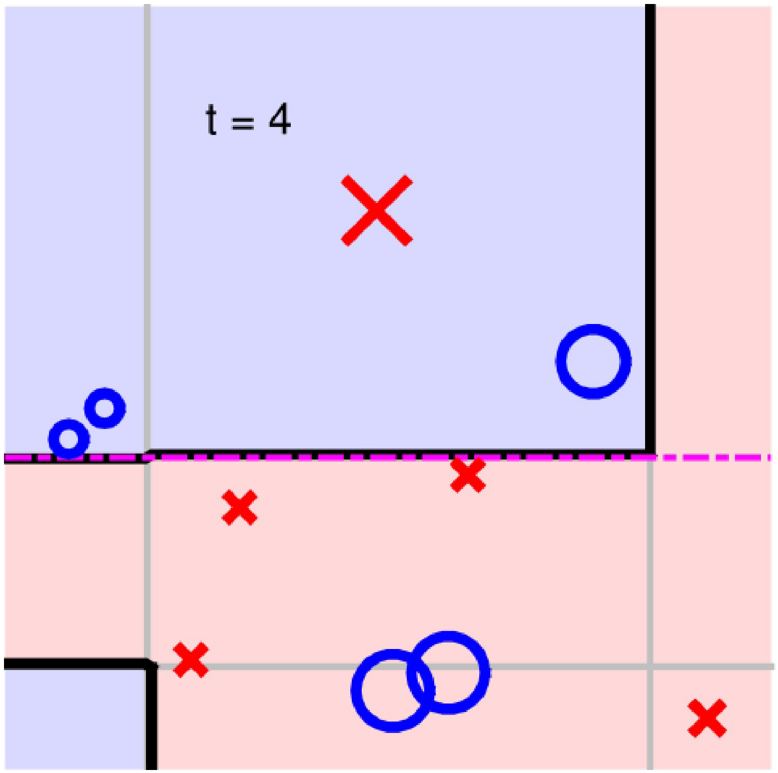


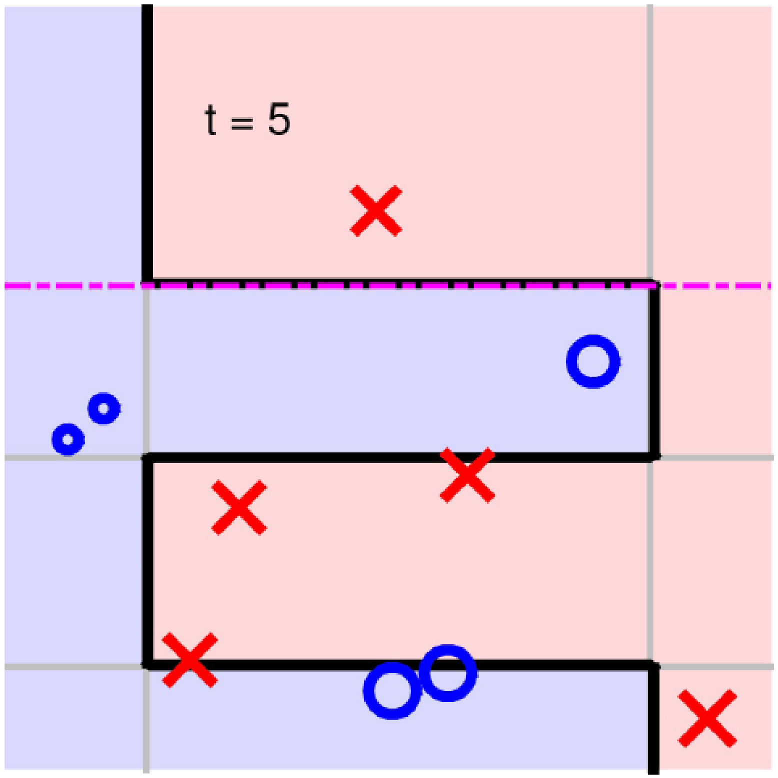
AdaBoost Example



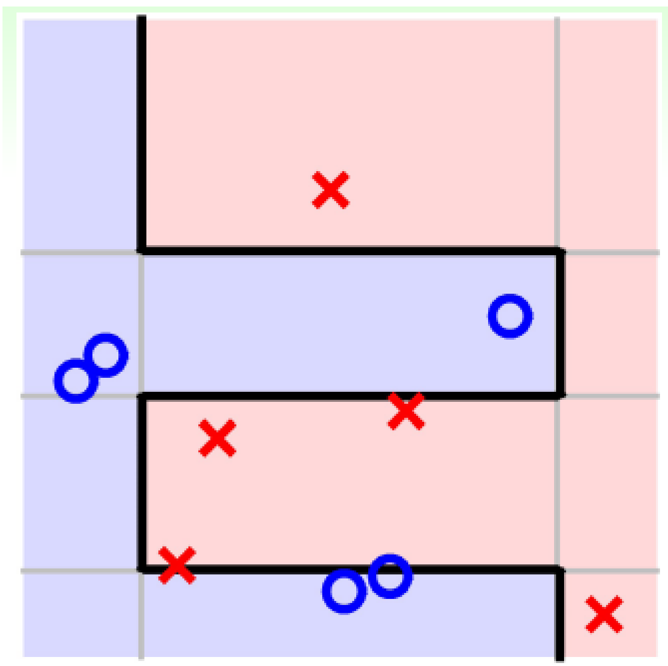


AdaBoost Example





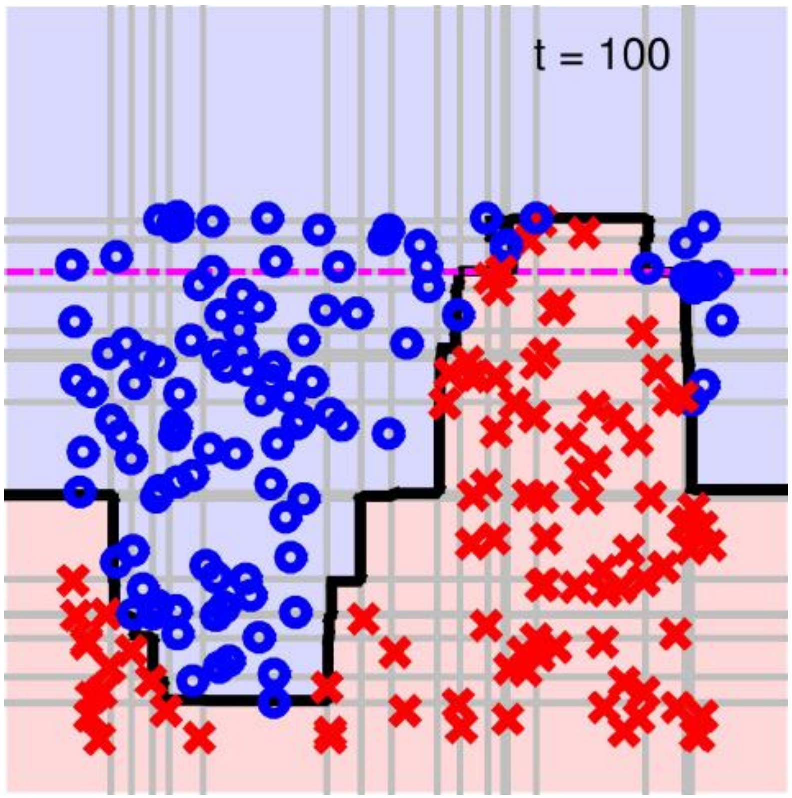
AdaBoost Example



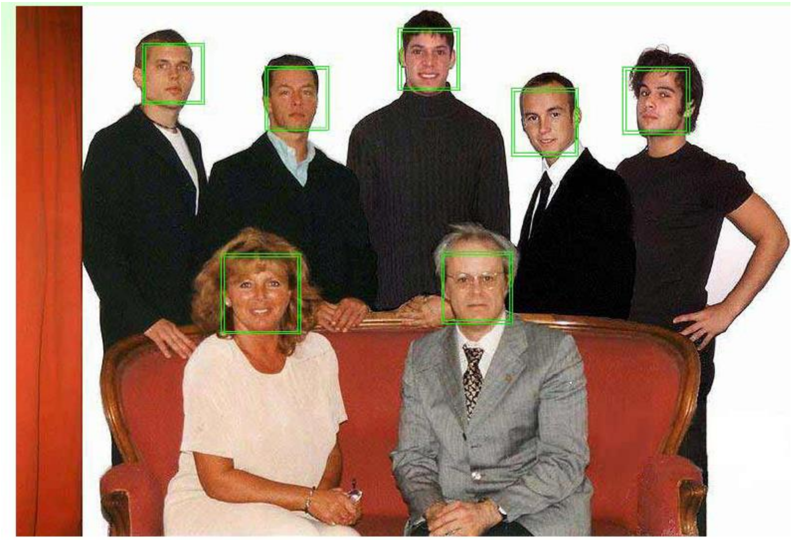
- 基分类器：h_t (like vertical/horizontal lines)
- 强分类器：H (like black curve)
- 每次在学习h_t的时候，更关注分类器h_{t-1}错分的样本
- 从偏差-方差分解的角度看，AdaBoost主要关注降低错误率（即降低偏差），因此AdaBoost能基于分类性能相当弱的学习器构建出分类性能很强的分类器。

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

AdaBoost Example



AdaBoost-Stump: **non-linear yet efficient**



世界首个实时人脸检测系统,可进行特征选择

AdaBoost算法

输入：训练数据集$\{x^{(i)}, y^{(i)}\}_{i=1}^{m}, x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{-1, +1\}$, 基分类器学习算法

1. 初始化训练数据的权值分布 $\mathcal{D}_1(x^{(i)}) = \frac{1}{m}$
2. for $t = 1$ to T
   - 使用具有权值分布$\mathcal{D}_t$的训练集进行学习，得到分类器$h_t(x)$
   - 计算$h_t(x)$在当前训练集上的分类误差:
     $\epsilon_t = P_{x \sim \mathcal{D}_t}[h_t(x) \neq y] = \sum_{y^{(i)} \neq h_t(x^{(i)})} \mathcal{D}_t(x^{(i)})$
   - 若$\epsilon_t > 0.5$, break；否则由下式计算该分类器的权重:
     $$\boxed{\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}}$$
   - 更新样本的权重:
     $$\boxed{\mathcal{D}_{t+1}(x^{(i)}) = \frac{1}{Z_t} \mathcal{D}_t(x^{(i)}) \exp[-\alpha_t y^{(i)} h_t(x^{(i)})]}$$
     其中$Z_t = \sum_i \mathcal{D}_t(x^{(i)}) \exp[-\alpha_t y^{(i)} h_t(x^{(i)})]$是归一化因子，使得$\mathcal{D}_{t+1}$仍然为分布

加性模型：

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

输出：最终的强分类器$H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

AdaBoost算法的推导

假设函数： $H(x)=\sum_{t=1}^{T}\alpha_t h_t(x)$

损失函数（指数损失）：$\ell(H(x),y)=\exp(-yH(x))$

第一个分类器h_1是直接通过直接基于初始数据分布用基学习算法可得。此后迭代生产h_t和对应的权重α_t, 应使得$\alpha_t h_t$最小化指数损失

$$\begin{aligned}\ell_t(\alpha_t)&=E_{x\sim\mathcal{D}_t}\exp[-y\alpha_t h_t(x)]\\&=e^{-\alpha_t}P_{x\sim\mathcal{D}_t}[h_t(x)=y]+e^{\alpha_t}P_{x\sim\mathcal{D}_t}[h_t(x)\neq y]=(1-\epsilon_t)e^{-\alpha_t}+\epsilon_t e^{\alpha_t}\end{aligned}$$

令 $\frac{\partial \ell_t}{\partial \alpha_t}=-e^{-\alpha_t}(1-\epsilon_t)+e^{\alpha_t}\epsilon_t=0$, 有

$$\alpha_t=\frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

AdaBoost算法的推导(Optional)

假设函数： $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

损失函数（指数损失）：$\ell(H(x), y) = \exp(-yH(x))$

在得到分类器H_{t-1}后，分类器h_t应能纠正H_{t-1}的错误，即应最小化

$$\ell_t(H_{t-1} + h_t|\mathcal{D}) = E_{x\sim\mathcal{D}} \exp[-y(H_{t-1}(x) + h_t(x))] = E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)} e^{-yh_t(x)}$$

根据泰勒公式,$e^x = 1 + x^1 + x^2/2! + \cdots$，和$y^2 = 1, h_t^2(x) = 1$有

$$e^{-yh_t(x)} \approx 1 - yh_t(x) + \frac{1}{2}y^2h_t^2(x) = \frac{3}{2} - yh_t(x)$$

$$y \in \{-1, +1\}$$

$$h_t(x) \in \{-1, +1\}$$

$$h_t(x) = \arg\min_h E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)} e^{-yh_t(x)}$$

$$\approx \arg\max_h E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)} yh_t(x) = \arg\max_h E_{x\sim\mathcal{D}} \left[\frac{e^{-yH_{t-1}(x)}}{E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)}} yh_t(x) \right]$$

AdaBoost算法的推导(Optional)

$$h_t(x) = \arg\min_h E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)} e^{-yh_t(x)}$$

$$\approx \arg\max_h E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)} yh_t(x) = \arg\max_h E_{x\sim\mathcal{D}} \left[\frac{e^{-yH_{t-1}(x)}}{E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)}} yh_t(x) \right]$$

令$\mathcal{D}_t$表示分布

$$\mathcal{D}_t(x) = \frac{\mathcal{D}(x) e^{-yH_{t-1}(x)}}{E_{x\sim\mathcal{D}} e^{-yH_{t-1}(x)}}$$

$$y \in \{-1, +1\}$$

$$h_t(x) \in \{-1, +1\}$$

根据期望的定义

$$yh_t(x) = 1 - 2\mathbb{I}[y \neq h_t(x)]$$

$$h_t(x) = \arg\max_h E_{x\sim\mathcal{D}_t}[yh_t(x)]$$

$$= \arg\min_h E_{x\sim\mathcal{D}_t} \mathbb{I}[y \neq h_t(x)]$$

即此时最佳的h_t应在分布 $\mathcal{D}_t$下最小化分类误差.

AdaBoost算法的推导(Optional)

考虑到$\mathcal{D}_t$和$\mathcal{D}_{t+1}$的关系，有

$$
\begin{aligned}
\mathcal{D}_{t+1}(x) &= \frac{\mathcal{D}(x)e^{-yH_t(x)}}{E_{x\sim\mathcal{D}}e^{-yH_t(x)}} \\
&= \frac{\mathcal{D}(x)e^{-yH_{t-1}(x)}e^{-y\alpha_t h_t(x)}}{E_{x\sim\mathcal{D}}e^{-yH_t(x)}} \\
&= \mathcal{D}_t(x)e^{-y\alpha_t h_t(x)}\frac{E_{x\sim\mathcal{D}}e^{-yH_{t-1}(x)}}{E_{x\sim\mathcal{D}}e^{-yH_t(x)}}
\end{aligned}
$$

$$
\mathcal{D}_t(x) = \frac{\mathcal{D}(x)e^{-yH_{t-1}(x)}}{E_{x\sim\mathcal{D}}e^{-yH_{t-1}(x)}}
$$

AdaBoost

如何选择基学习器h_t？

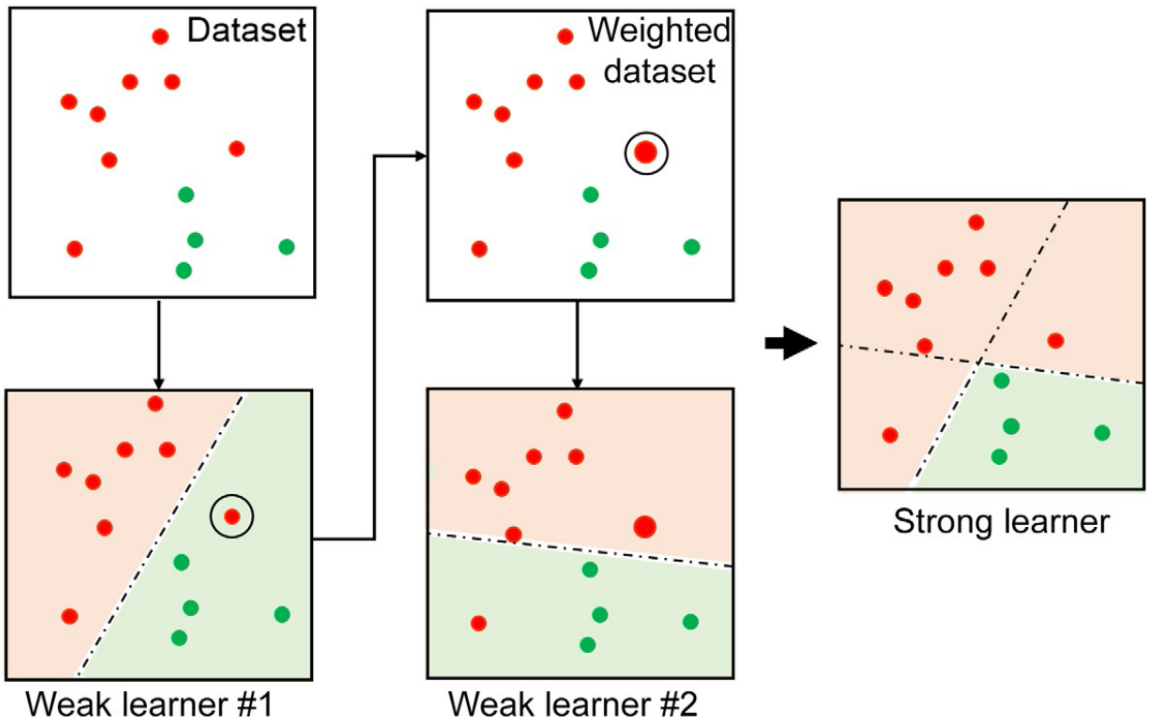
- A popular choice: decision stump (决策桩)

$$h_{s,i,\theta} = s\,\mathrm{sign}(x_i - \theta)$$

- 三个参数 feature i, threshold θ, direction s
- 物理意义：2D平面上的水平或者垂直线
- 非常容易求解: $O(n \cdot m \log m)$

AdaBoost

- 除了Decision Stump外，是否可以采用其他分类器？



Misra, S., & Li, H. (2019). Noninvasive fracture characterization based on the classification of sonic wave travel times. *Machine Learning for Subsurface Characterization*, 243.

Gradient Boosting Machine (GBM)

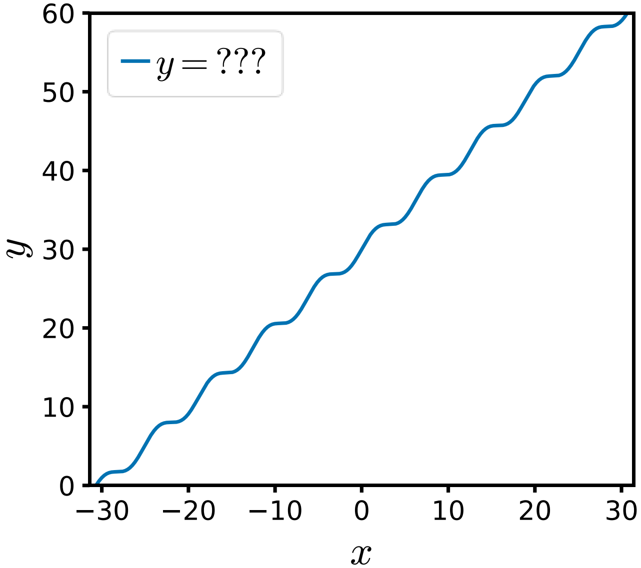
AdaBoost采用指数损失(with binary-output hypothesis h)，对应的目标函数可以表示为

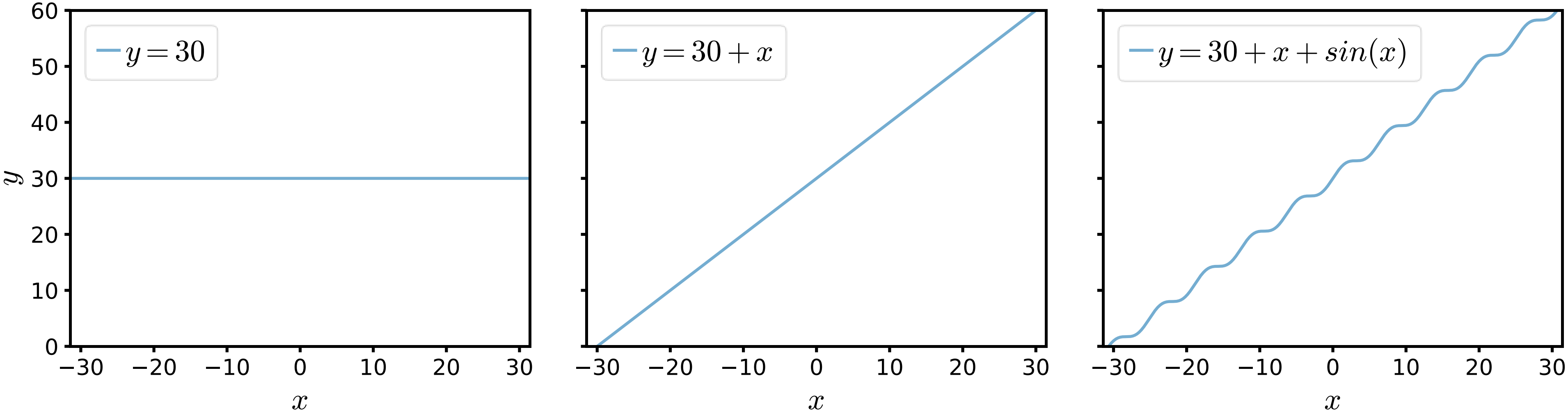
$$
\begin{aligned}
(\alpha_t, h_t) &= \arg\min_{\alpha,h} \frac{1}{m} \sum_{i=1}^{m} \exp\left[-y^{(i)} \left(H_{t-1}(x^{(i)}) + \alpha h(x^{(i)})\right)\right] \\
&= \arg\min_{\alpha,h} \frac{1}{m} \sum_{i=1}^{m} \exp\left[-y^{(i)} \left(\sum_{\tau=1}^{t-1} \alpha_\tau h_\tau(x^{(i)}) + \alpha h(x^{(i)})\right)\right]
\end{aligned}
$$

GBM仍采用加性模型：$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$，但拓展为可以采用其他任意损失$\ell$(如前面介绍过的平方损失、交叉熵损失等) with any hypothesis h，对应的目标函数可以表示为

$$
(\alpha_t, h_t) = \arg\min_{h,\alpha} \frac{1}{m} \sum_{i=1}^{m} \ell\left(\sum_{\tau=1}^{t-1} \alpha_\tau h_\tau(x^{(i)}) + \alpha h(x^{(i)}), y^{(i)}\right)
$$

加性模型



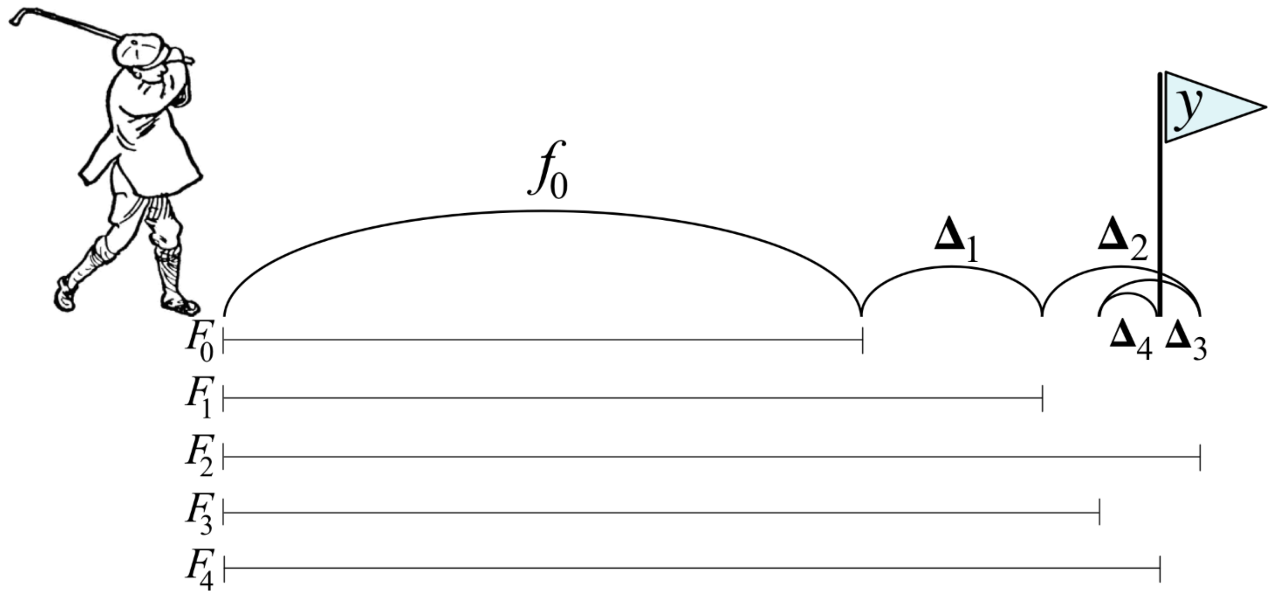


$$y = H(x) = h_1(x) + h_2(x) + h_3(x)$$

$$\begin{aligned} h_1(x) &= 30 \\ h_2(x) &= x \\ h_3(x) &= sin(x) \end{aligned}$$

https://explained.ai/gradient-boosting/L2-loss.html

加性模型



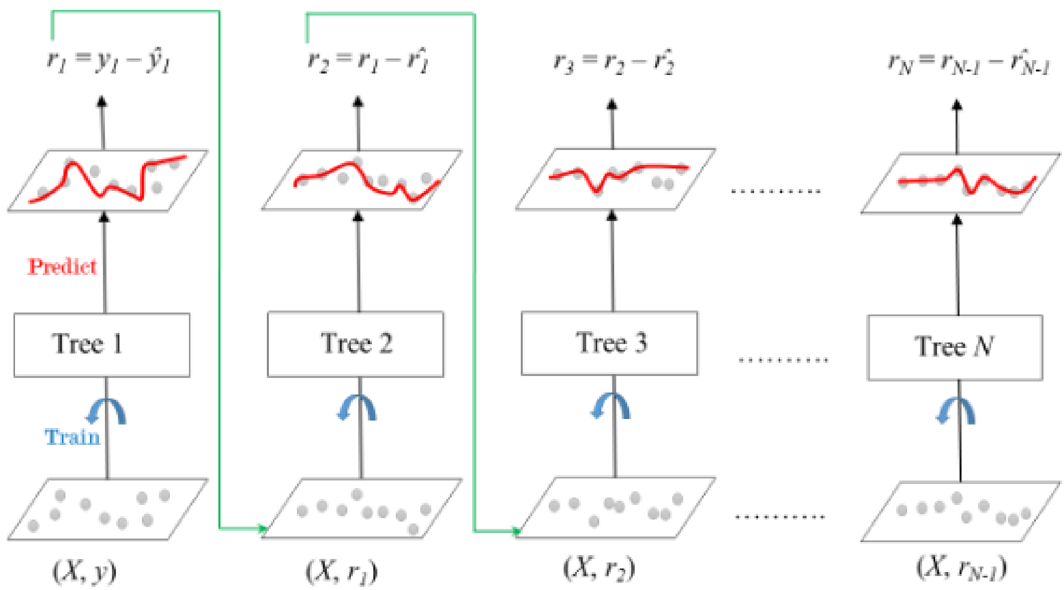
$$
\begin{aligned}
\hat{y} &= f_0(x) + \Delta_1(x) + \Delta_2(x) + \ldots + \Delta_M(x) \\
&= f_0(x) + \sum_{m=1}^{M} \Delta_m(x) \\
&= F_M(x)
\end{aligned}
$$

$$
\begin{aligned}
F_0(x) &= f_0(x) \\
F_m(x) &= F_{m-1}(x) + \Delta_m(x)
\end{aligned}
$$

https://explained.ai/gradient-boosting/L2-loss.html

Gradient Boosting Decision Tree (GBDT)

- GBM一般采用决策树（或回归树）作为基学习器，称为Gradient Boosting Decision Tree (GBDT)，



https://www.geeksforgeeks.org/ml-gradient-boosting/

Gradient Boosting Machine (GBM)

- 针对不同问题使用不同的损失函数：
  - 用指数损失函数的分类问题
  - 用平方误差损失函数的回归问题

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

直观上，新学习得到的分类器h_t应能最大限度降低损失，GBM的思想是分类器h_t应能沿着损失函数负梯度降低损失函数的值。这里分类器h_t常采用CART树，即每次学习一棵CART树(回归树)$h_t(x)$，去拟合样本余量(Sample Residuals) $\tilde{y}^{(i)}$:

$$\tilde{y}^{(i)} = -\left[\frac{\partial \ell(H, y^{(i)})}{\partial H}\right]_{H=H_{t-1}(x^{(i)})}$$

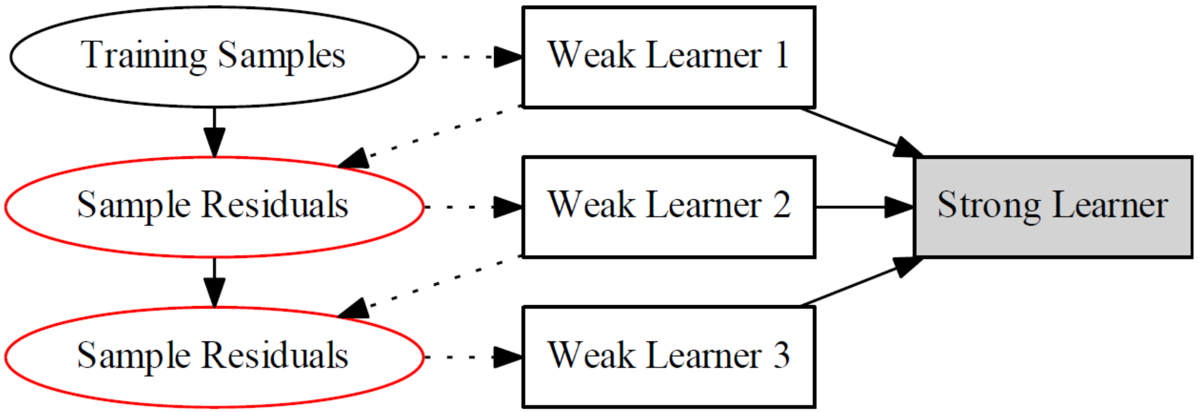
Gradient Boosting Decision Tree (GBDT)

输入：训练样本集 $\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$, 基学习器的个数$T$和损失函数$\ell(\cdot,\cdot)$

- 初始化h_1, for $t = 2$ to T
- 计算Sample Residuals: $\tilde{y}^{(i)} = -\frac{\partial \ell(H, y^{(i)})}{\partial H}\big|_{H=H_{t-1}(x^{(i)})}$
- 基于$\{x^{(i)}, \tilde{y}^{(i)}\}_{i=1}^{m}$ 训练CART回归树$h_t(x)$:
  $h_t(x) = \arg\min_h \frac{1}{m}\sum_{i=1}^{m}[\tilde{y}^{(i)} - h(x^{(i)})]^2$
- 计算对应CART树$h_t(x)$的权重（一维线性搜索）
  $\alpha_t = \arg\min_\alpha \sum_{i=1}^{m} \ell[y^{(i)}, H_{t-1}(x^{(i)}) + \alpha h_t(x^{(i)})]$
- $H_t(x) = H_{t-1}(x) + \alpha_t h_t(x)$
- 输出$H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

- 初始化
  - 随机初始化
  - 用训练样本中的统计量进行初始化
  - 用其他模型的预测值进行初始化
  - GBDT对初始化不敏感，但好的初始化能加速
- 存在其他改进算法，如Kaggle杀器：XGBoost, LigntGBM等

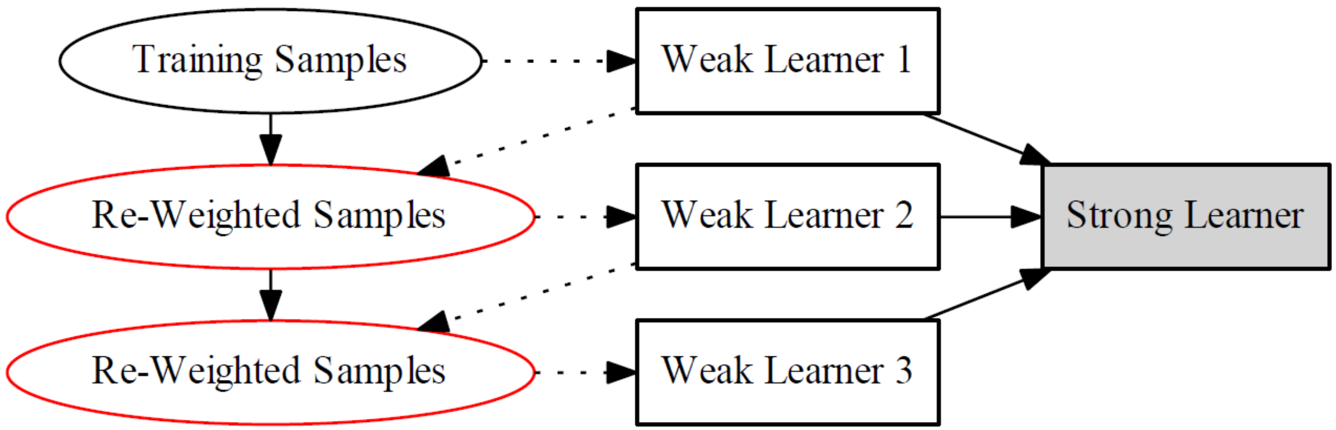
Gradient Boosting Machine

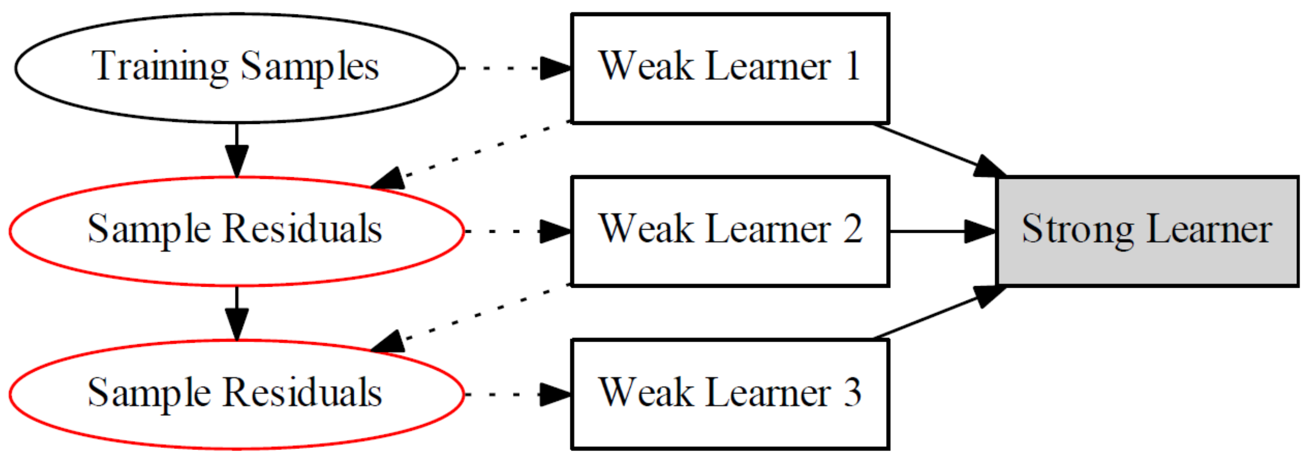


同样为了克服过拟合，可加入正则项

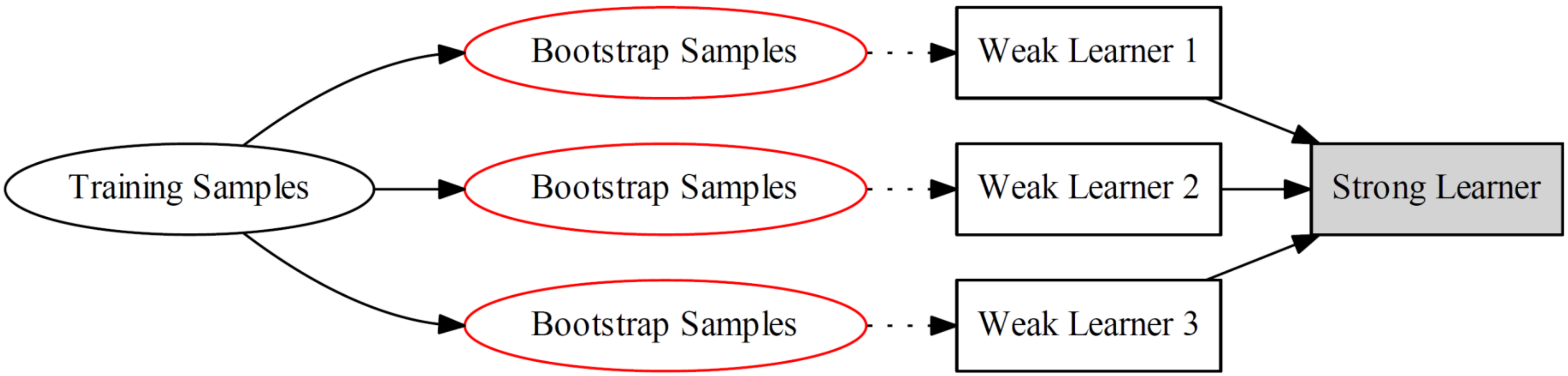
$$
\begin{aligned}
(\alpha_t, h_t) &= \arg\min_{\alpha, h} \frac{1}{m} \sum_{i=1}^{m} \ell\left(\sum_{\tau=1}^{t-1} \alpha_\tau h_\tau(x^{(i)}) + \alpha h(x^{(i)}), y^{(i)}\right) + \lambda R(h) \\
&= \arg\min_{\alpha, h} \frac{1}{m} \sum_{i=1}^{m} \ell\left(H_{t-1}(x^{(i)}) + \alpha h(x^{(i)}), y^{(i)}\right) + \lambda R(h)
\end{aligned}
$$

AdaBoost vs. GBM





Bagging


Training Samples
Bootstrap Samples
Bootstrap Samples
Bootstrap Samples
Weak Learner 1
Weak Learner 2
Weak Learner 3
Strong Learner

Bagging

- 自助采样(Bootstrap Sampling):指任何一种有放回的均匀抽样，也就是说，每当选中一个样本，它等可能地被再次选中并被再次添加到训练集中
- Bagging: 利用自助采样得到*T*组训练样本集，分别利用这些训练样本集训练*T*个分类器(CART or SVM or others)，最后进行投票集成
- 从Bias-Variance分解的角度看，Bagging主要关注降低方差

Bagging

Algorithm: Bagging

Input:

- Training data S with correct labels $\omega_i \Omega = \{\omega_1,...,\omega_C\}$ representing C classes
- Weak learning algorithm **WeakLearn,**
- Integer T specifying number of iterations.
- Percent (or fraction) F to create bootstrapped training data

Do t=1, ..., T

1. Take a bootstrapped replica S_t by randomly drawing F percent of S.
2. Call **WeakLearn** with S_t and receive the hypothesis (classifier) h_t.
3. Add h_t to the ensemble, $\mathcal{E}$.

End

Test: Simple Majority Voting – Given unlabeled instance **x**

1. Evaluate the ensemble $\mathcal{E} = \{h_1,...,h_T\}$ on **x**.
2. Let $v_{t,j} = \begin{cases} 1, & \text{if } h_t \text{ picks class } \omega_j \\ 0, & \text{otherwise} \end{cases}$ be the vote given to class ω_j by classifier h_t. (1)
3. Obtain total vote received by each class, $V_j = \sum_{t=1}^{T} v_{t,j}$ $j = 1,...,C$. (2)
4. Choose the class that receives the highest total vote as the final classification.

Polikar, © 2008

随机森林(Random Forest)

- 基本思想:
  - 充分利用“随机”的思想来增加各分类器的多样性(Diversity)
- “随机”体现在两个方面:
  - 基于自助采样法来随机选择训练样本
  - 随机选择特征（或属性）
- Random Forest (RF) = Bagging + Fully-Grown CART with Random-Subspace
- 特点:
  - 可高度并行化
  - 继承了CART的优点
  - 克服了完全生长树的缺点

决策融合策略

- 平均法
- 加权平均法
- 投票法
  - 绝大多数投票(Majority Voting)：超过半数，则决策，否则拒绝
  - 少数服从多数(Plurality Voting): 预测为得票最多的标记
- 学习法
  - 用各学习器的输出生成新的训练数据，再去训练一个学习器(如线性SVM等)

Thanks!

Any questions?