



清华大学
Tsinghua University

第八章 Adaboost



提纲

- AdaBoost的起源和基本概念
- AdaBoost算法
- 前向分步训练算法
- AdaBoost 编程
- AdaBoost实时人脸检测算法



Adaboost的起源

- 1984, Kearns和Valiant:
- 强可学习(strongly learnable)和弱可学习(weakly learnable)
 - 在概率近似正确 (probably approximately correct, PAC)学习的框架中, 一个概念 (类), 如果存在一个多项式的学习算法能够学习它, 并且正确率很高, 称这个概念是强可学习的;
 - 一个概念 (类), 如果存在一个多项式的学习算法能够学习它, 学习的正确率仅比随机猜测略好, 则称这个概念是弱可学习的。
- 1989, Schapire, 证明:
 - 在PAC学习的框架下, 一个概念是强可学习的充分必要条件是这个概念是弱可学习。



清华大学

Tsinghua University

Adaboost的起源

- 问题的提出：
- 只要找到一个比随机猜测略好的弱学习算法就可以直接将其提升为强学习算法，而不必直接去找很难获得的强学习算法。



清华大学

Tsinghua University

怎样实现弱学习转为强学习

例如：学习算法A在a情况下失效，学习算法B在b情况下失效，那么在a情况下可以用B算法，在b情况下可以用A算法解决。这说明通过某种合适的方式把各种算法组合起来，可以提高准确率。

为实现弱学习互补，面临两个问题：

- (1) 怎样获得不同的弱分类器？
- (2) 怎样组合弱分类器？



怎样获得不同的弱分类器

- ◆ 使用不同的弱学习算法得到不同基本学习器
 - ◆ 参数估计、非参数估计...
- ◆ 使用相同的弱学习算法，但用不同的参数
 - ◆ K-Mean不同的K，神经网络不同的隐含层...
- ◆ 相同输入对象的不同表示凸显事物不同的特征
- ◆ 使用不同的训练集
 - 装袋 (bagging)
 - 提升 (boosting)



Bagging

- 也称为自举汇聚法(bootstrap aggregating)
 - 从原始数据集选择 S 次后得到 S 个新数据集
 - 新数据集和原数据集的大小相等
 - 每个数据集都是通过在原始数据集中随机选择样本来进行替换而得到的。
 - S 个数据集建好之后，将某个学习算法分别作用于每个数据集就得到 S 个分类器。
 - 选择分类器投票结果中最多的类别作为最后的分类结果。
 - 改进的Bagging算法，如随机森林等。



怎样组合弱分类器

◆ 多专家组合

一种**并行**结构，**所有**的弱分类器都给出各自的预测结果，通过“组合器”把这些预测结果转换为最终结果。eg. **投票 (voting)** 及其变种、混合专家模型

◆ 多级组合

一种**串行**结构，其中下一个分类器只在前一个分类器预测不够准（不够自信）的实例上进行训练或检测。eg. **级联算法 (cascading)**



Adaboost的提出

- 1990年，Schapire最先构造出一种多项式级的算法，即最初的Boost算法；
- 1993年，Drunker和Schapire第一次将神经网络作为弱学习器，应用Boosting算法解决OCR问题；
- 1995年，Freund和Schapire提出了Adaboost(Adaptive Boosting)算法，效率和原来Boosting算法一样，但是不需要任何关于弱学习器性能的先验知识，可以非常容易地应用到实际问题中。



清华大学
Tsinghua University

Adaboost基本概念

AdaBoost

Adaptive Boosting

A learning algorithm

Building a strong classifier a lot of weaker ones



Adaboost基本概念

$$h_1(x) \in \{-1, +1\}$$

$$h_2(x) \in \{-1, +1\}$$

⋮

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

Weak classifiers

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

strong classifier

slightly better than random

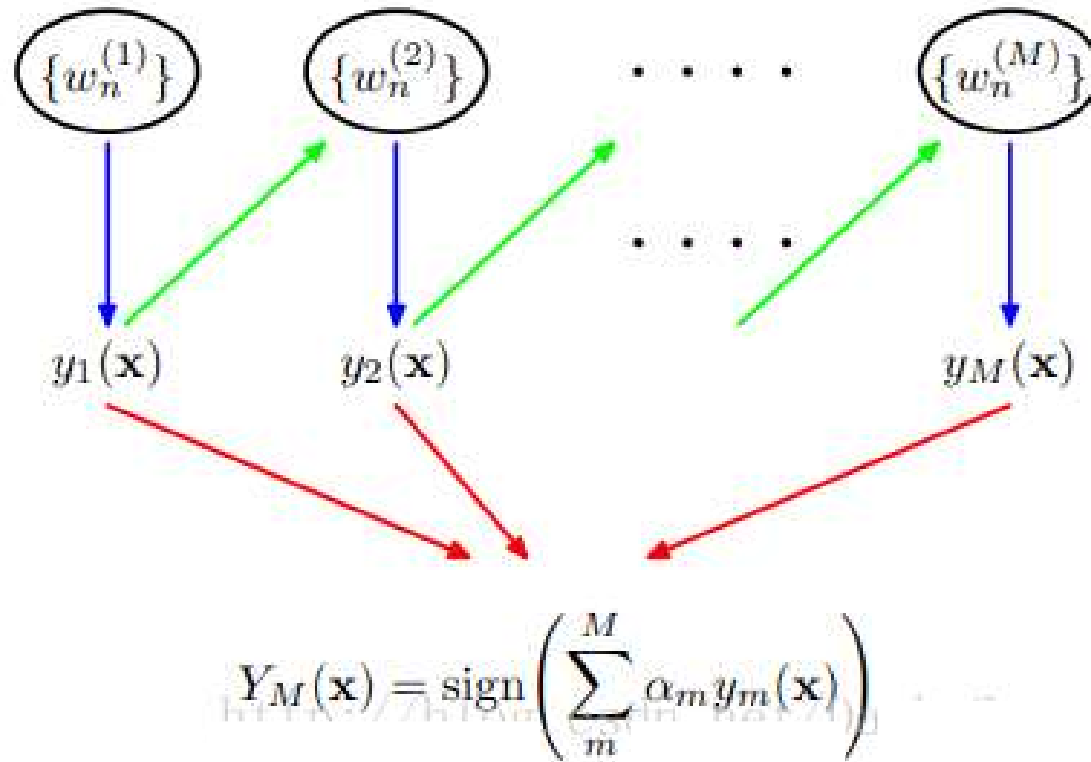


Adaboost基本概念

- 两个问题如何解决：
 - 每一轮如何改变训练数据的权值或概率分布？
 - AdaBoost：提高那些被前一轮弱分类器错误分类样本的权值，降低那些被正确分类样本的权值
- 如何将弱分类器组合成一个强分类器？
 - AdaBoost：加权多数表决，加大分类误差率小的弱分类器的权值，使其在表决中起较大的作用，减小分类误差率大的弱分类器的权值，使其在表决中起较小的作用。



Adaboost基本概念





AdaBoost算法

- 输入：二分类的训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- $x_i \in \mathcal{X} \subseteq \mathbf{R}^n$, $y_i \in \mathcal{Y} = \{-1, +1\}$
- 输出：最终分类器 $G(x)$

- 1 初始化训练数据的起始权值分布

$$D_1 = (w_{11}, \dots, w_{1i}, \dots, w_{1N}) \quad w_{1i} = \frac{1}{N}, \quad i = 1, 2, \dots, N$$



AdaBoost算法

- 2 对m个弱分类器 $m=1,2, \dots, M$

a、在权值 D_m 下训练数据集，得到弱分类器

$$G_m(x): \mathcal{X} \rightarrow \{-1, +1\}$$

b、计算 G_m 的训练误差

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i)$$

c、计算 G_m 的系数 $\alpha_m = \frac{1}{2} \log \frac{1-e_m}{e_m}$

d、更新训练数据集的权值分布

$$D_{m+1} = (w_{m+1,1}, \dots, w_{m+1,i}, \dots, w_{m+1,N}) \quad w_{m+1,i} = \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i))$$

Z 是规范化因子

$$Z_m = \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$



AdaBoost算法

- 3、构建弱分类器的线性组合,

$$f(x) = \sum_{m=1}^M \alpha_m G_m(x)$$

- 得到最终分类器:

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{m=1}^M \alpha_m G_m(x)\right)$$



AdaBoost算法说明

- 步骤 b $e_m = P(G_m(x_i) \neq y_i) = \sum_{G_m(x_i) \neq y_i} w_{mi} \quad \sum_{i=1}^N w_{mi} = 1$

- 步骤 c $\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}$ 当 $e_m \leq \frac{1}{2}$ 时, $\alpha_m \geq 0$

- 步骤 d
$$w_{m+1,i} = \begin{cases} \frac{w_{mi}}{Z_m} e^{-\alpha_m}, & G_m(x_i) = y_i \\ \frac{w_{mi}}{Z_m} e^{\alpha_m}, & G_m(x_i) \neq y_i \end{cases}$$

- 误分类的样本权值放大
$$e^{2\alpha_m} = \frac{e_m}{1 - e_m}$$



例子：

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

- 初始化 $D_1 = (w_{11}, w_{12}, \dots, w_{110})$
 $w_{1i} = 0.1, \quad i = 1, 2, \dots, 10$
- 对 $m=1$
- a、在权值分布为 D_1 的数据集上，阈值取 2.5，分类误差率最小，基本弱分类器：
$$G_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x > 2.5 \end{cases}$$
- b、 $G_1(x)$ 的误差率：
$$e_1 = P(G_1(x_i) \neq y_i) = 0.3$$
- c、 $G_1(x)$ 的系数：
$$\alpha_1 = \frac{1}{2} \log \frac{1-e_1}{e_1} = 0.4236$$



例子:

- d、更新训练数据的权值分布

$$D_2 = (w_{21}, \dots, w_{2i}, \dots, w_{210})$$

$$w_{2i} = \frac{w_{1i}}{Z_1} \exp(-\alpha_1 y_i G_1(x_i)), \quad i = 1, 2, \dots, 10$$

$$D_2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, \\ 0.1666, 0.1666, 0.1666, 0.0715)$$

$$f_1(x) = 0.4236 G_1(x)$$

- 弱基本分类器 $G_1(x) = \text{sign}[f_1(x)]$ 在更新的数据集上有3个误分类点



例子：

- 对 $m=2$
- a、在权值分布 D_2 上，阈值 $v=8.5$ 时，分类误差率最低

$$G_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x > 8.5 \end{cases}$$

- b、误差率 $e_2 = 0.2143$.
- c、计算 $\alpha_2 = 0.6496$
- d、更新权值分布

$$D_3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.1667, \\ 0.1060, 0.1060, 0.1060, 0.0455)$$

$$f_2(x) = 0.4236G_1(x) + 0.6496G_2(x)$$

- 分类器 $G_2(x)=\text{sign}[f_2(x)]$ 有三个误分类点



例子：

- 对 $m=3$
- a、在权值分布 D_3 上，阈值 $v=5.5$ 时，分类误差率最低

$$G_3(x) = \begin{cases} 1, & x > 5.5 \\ -1, & x < 5.5 \end{cases}$$

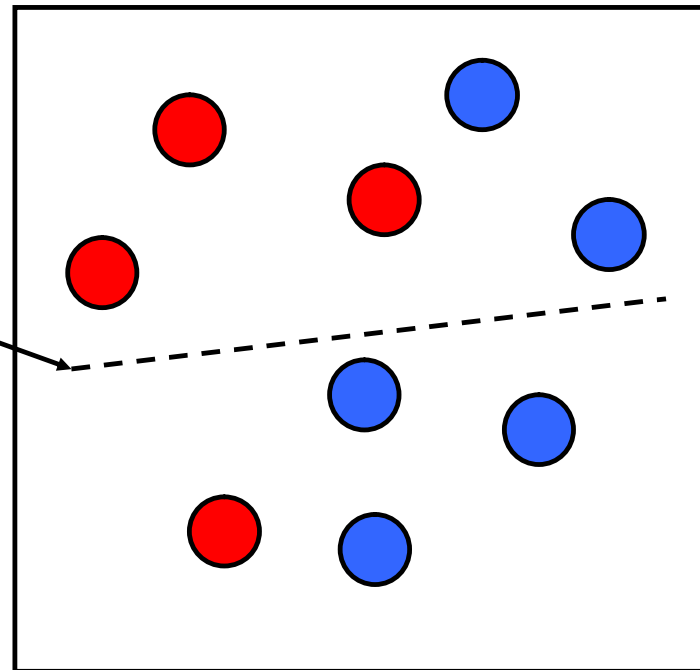
- b、误差率 $e_3 = 0.1820$.
- c、计算 $\alpha_3 = 0.7514$
- d、更新权值分布 $D_4 = (0.125, 0.125, 0.125, 0.102, 0.102, 0.102, 0.065, 0.065, 0.065, 0.125)$
$$f_3(x) = 0.4236G_1(x) + 0.6496G_2(x) + 0.7514G_3(x)$$
- 分类器 $G_3(x) = \text{sign}[f_3(x)]$ 误分类点为 0

$$G(x) = \text{sign}[f_3(x)] = \text{sign}[0.4236G_1(x) + 0.6496G_2(x) + 0.7514G_3(x)]$$



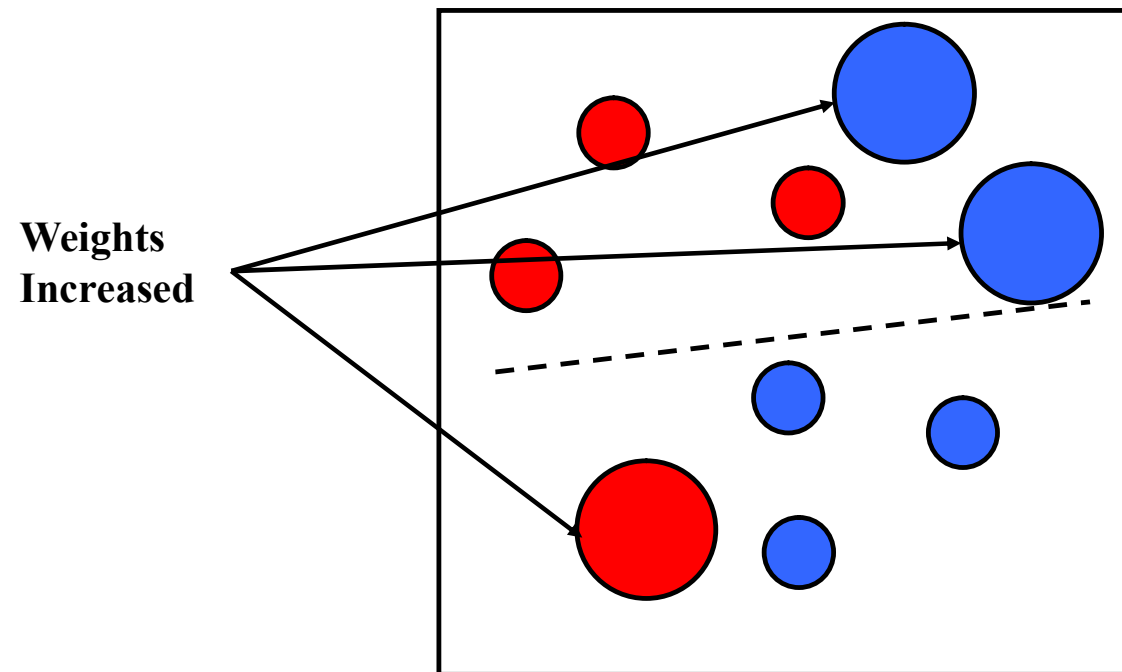
Boosting illustration

**Weak
Classifier 1**



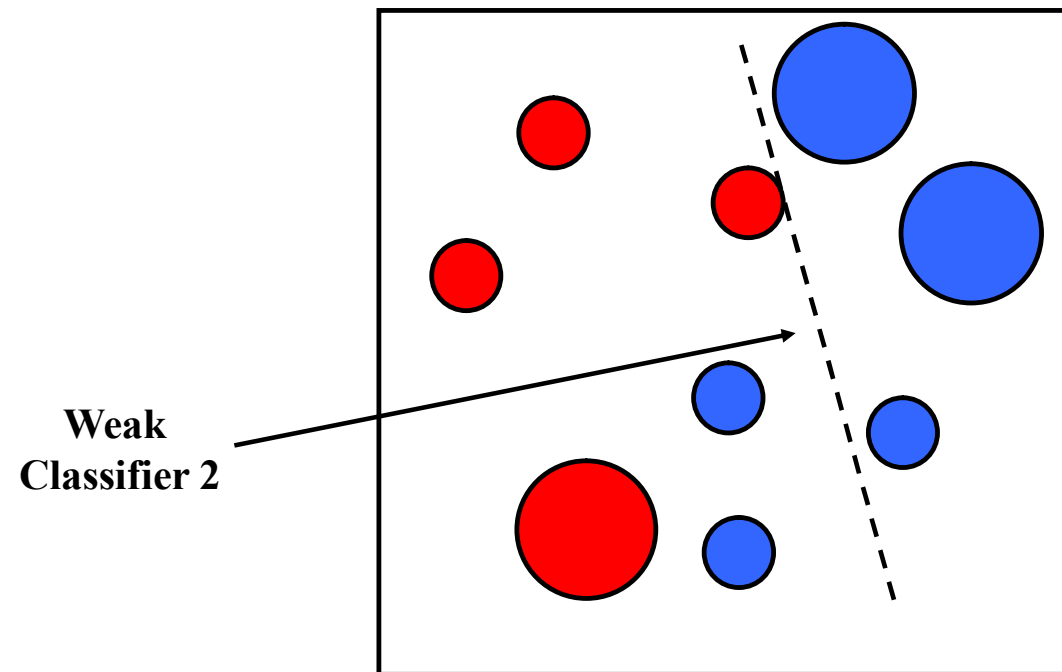


Boosting illustration





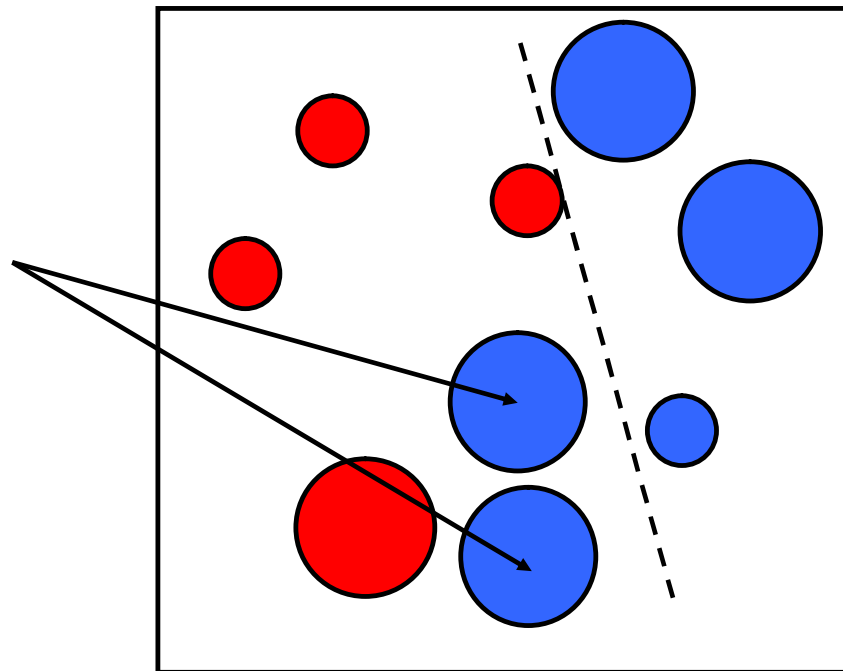
Boosting illustration





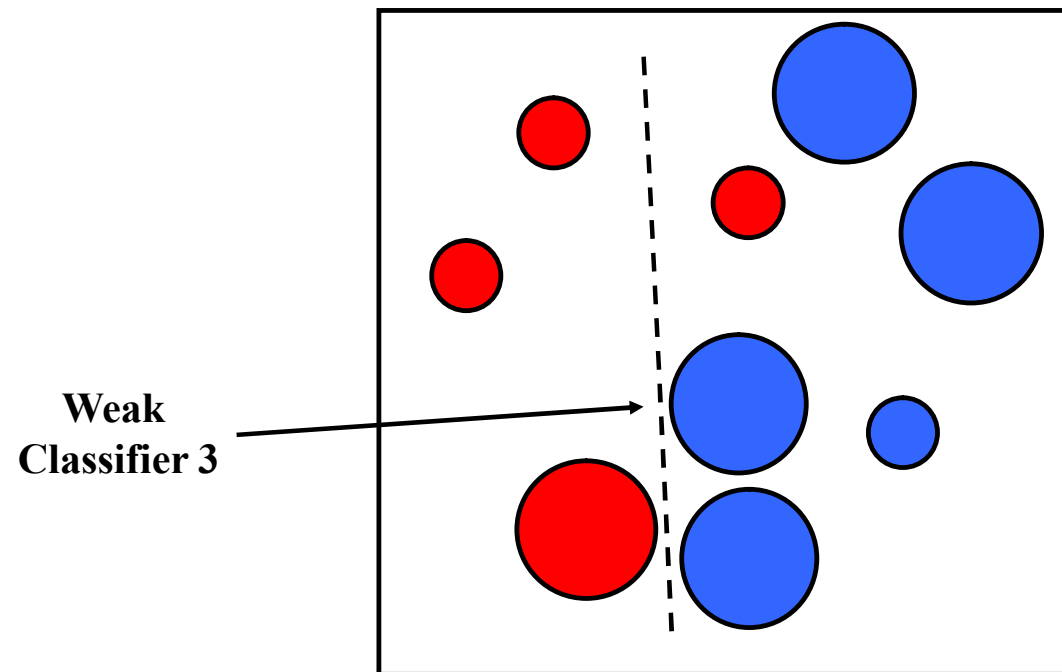
Boosting illustration

**Weights
Increased**





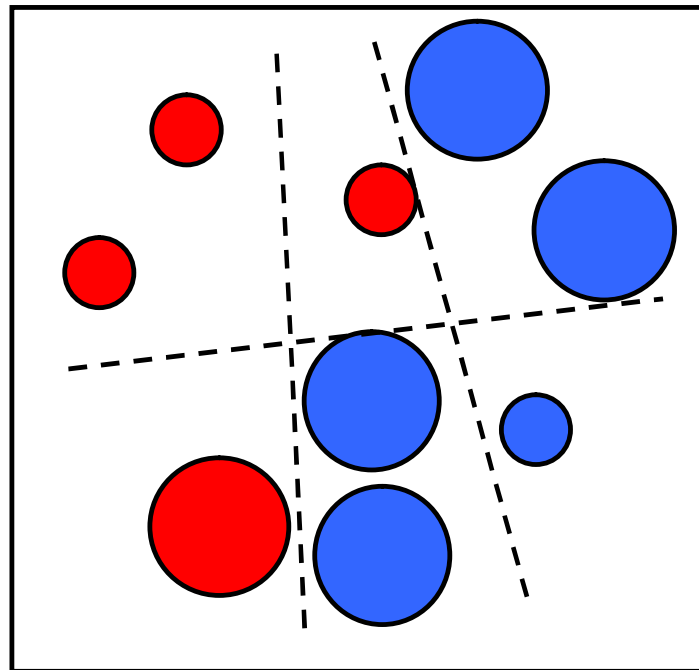
Boosting illustration





Boosting illustration

**Final classifier is
a combination of weak
classifiers**





AdaBoost的训练误差分析

由：

$$Z_m = \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

$$f(x) = \sum_{m=1}^M \alpha_m G_m(x)$$

$$G(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{m=1}^M \alpha_m G_m(x)\right)$$

定理： AdaBoost算法最终分类器的训练误差界为：

$$\frac{1}{N} \sum_{i=1}^N I(G(x_i) \neq y_i) \leq \frac{1}{N} \sum_i \exp(-y_i f(x_i)) = \prod_m Z_m$$



AdaBoost的训练误差分析

$$\frac{1}{N} \sum_{i=1}^N I(G(x_i) \neq y_i) \leq \frac{1}{N} \sum_i \exp(-y_i f(x_i)) = \prod_m Z_m$$

证明：前面部分很明显，
证后面，由

$$w_{mi} \exp(-\alpha_m y_i G_m(x_i)) = Z_m w_{m+1,i} \rightarrow$$

$$\prod_{m=1}^M Z_m \leftarrow$$

$$\begin{aligned} & \frac{1}{N} \sum_i \exp(-y_i f(x_i)) \\ &= \frac{1}{N} \sum_i \exp\left(-\sum_{m=1}^M \alpha_m y_i G_m(x_i)\right) \\ &= \sum_i w_{1i} \prod_{m=1}^M \exp(-\alpha_m y_i G_m(x_i)) \\ &= Z_1 \sum_i w_{2i} \prod_{m=2}^M \exp(-\alpha_m y_i G_m(x_i)) \\ &= Z_1 Z_2 \sum_i w_{3i} \prod_{m=3}^M \exp(-\alpha_m y_i G_m(x_i)) \\ &= \dots \\ &= Z_1 Z_2 \dots Z_{M-1} \sum_i w_{Mi} \exp(-\alpha_M y_i G_M(x_i)) \end{aligned}$$



清华大学

Tsinghua University

AdaBoost的训练误差分析

定理：二分类问题AdaBoost的训练误差界为：

$$\prod_{m=1}^M Z_m = \prod_{m=1}^M [2\sqrt{e_m(1-e_m)}] = \prod_{m=1}^M \sqrt{(1-4\gamma_m^2)} \leq \exp\left(-2\sum_{m=1}^M \gamma_m^2\right)$$
$$\gamma_m = \frac{1}{2} - e_m$$

证明：前面，

$$\begin{aligned} Z_m &= \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i)) \\ &= \sum_{y_i=G_m(x_i)} w_{mi} e^{-\alpha_m} + \sum_{y_i \neq G_m(x_i)} w_{mi} e^{\alpha_m} \\ &= (1-e_m) e^{-\alpha_m} + e_m e^{\alpha_m} \\ &= 2\sqrt{e_m(1-e_m)} = \sqrt{1-4\gamma_m^2} \end{aligned}$$



AdaBoost的训练误差分析

定理：二分类问题AdaBoost的训练误差界为：

$$\prod_{m=1}^M Z_m = \prod_{m=1}^M [2\sqrt{e_m(1-e_m)}] = \prod_{m=1}^M \sqrt{(1-4\gamma_m^2)} \leq \exp\left(-2\sum_{m=1}^M \gamma_m^2\right)$$
$$\gamma_m = \frac{1}{2} - e_m$$

证明：后面，

由 e^x 和 $\sqrt{1-x}$ 在 $x=0$ 的泰劳展开得：

$$\sqrt{(1-4\gamma_m^2)} \leq \exp(-2\gamma_m^2)$$

进而得证。



AdaBoost的训练误差分析

定理：如果存在 $\gamma > 0$ ，对所有的 m 有 $\gamma_m \geq \gamma$ ，则

$$\frac{1}{N} \sum_{i=1}^N I(G(x_i) \neq y_i) \leq \exp(-2M\gamma^2)$$

即：训练误差为指数下降。

注意：AdaBoost算法不需要知道下界 γ ，这正是Freund与Schapire设计AdaBoost时所考虑的，与一些早期的提升方法不同，AdaBoost具有适应性，即它能适应弱分类器各自的训练误差率，这也是它的名称(适应的提升)的由来，Ada是Adaptive的简写



清华大学

Tsinghua University

AdaBoost算法的解释

- AdaBoost
 - 模型：加法模型
 - 损失函数：指数函数
 - 学习算法：前向分步算法的二分类学习算法



前向分步算法

- 加法模型 (additive model)

基函数的参数

基函数的系数

基函数

$$f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$$
$$f(x) = \sum_{m=1}^M \alpha_m G_m(x)$$

- 给定训练数据和损失函数 $L(y, f(x))$, 学习加法模型 $f(x)$ 成为经验风险极小化即损失函数极小化问题:

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^N L\left(y_i, \sum_{m=1}^M \beta_m b(x_i; \gamma_m)\right)$$

复杂的优化问题



前向分步算法

- 前向分步算法 Forward stagewise algorithm 求解思路：
 - 根据学习的是加法模型，如果能够从前向后，每一步只学习一个基函数及其系数，逐步逼近优化目标函数式，
 - 具体，每步只需优化如下损失函数：

$$\min_{\beta, \gamma} \sum_{i=1}^N L(y_i, \beta b(x_i; \gamma))$$



$$\min_{\beta_m, \gamma_m} \sum_{i=1}^N L\left(y_i, \sum_{m=1}^M \beta_m b(x_i; \gamma_m)\right)$$



前向分步算法

- 输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- 损失函数 $L(y, f(x))$ 基函数集 $\{b(x; \gamma)\}$
- 输出：加法模型 $f(x)$
- 1 初始化 $f_0(x) = 0$
- 2 对 $m=1, 2, \dots, M$
 - a、极小化损失函数 $(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
 - 得到参数 β_m, γ_m
 - b、更新 $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$
- 3 得到加法模型 $f(x) = f_M(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$



前向分步算法

- 定理:
- AdaBoost算法是前向分步加法算法的特例, 模型是由基本分类器组成的加法模型, 损失函数是指数函数

- 证明: 前面部分很明显: $f(x) = \sum_{m=1}^M \alpha_m G_m(x)$
- 后面部分证明损失函数是指数函数 (exponential loss function)

$$L(y, f(x)) = \exp[-yf(x)]$$

- 假设经过m-1轮迭代前向分步算法得到 $f_{m-1}(x)$:

$$\begin{aligned} f_{m-1}(x) &= f_{m-2}(x) + \alpha_{m-1} G_{m-1}(x) \\ &= \alpha_1 G_1(x) + \cdots + \alpha_{m-1} G_{m-1}(x) \end{aligned}$$

- 在第m轮迭代得到 α_m , $G_m(x)$ 和 $f_m(x)$ $f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$



前向分步算法

- 在第m轮迭代得到

$$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$$

- 目标: $(\alpha_m, G_m(x)) = \arg \min_{\alpha, G} \sum_{i=1}^N \exp[-y_i (f_{m-1}(x_i) + \alpha G(x_i))]$

- 即: $(\alpha_m, G_m(x)) = \arg \min_{\alpha, G} \sum_{i=1}^N \bar{w}_{mi} \exp[-y_i \alpha G(x_i)]$

$$\bar{w}_{mi} = \exp[-y_i f_{m-1}(x_i)]$$

不依赖 α 和 G ,依赖于 $f_{m-1}(x)$,随着每一轮迭代改变

- 现证使上式最小的 α_m^* 和 $G_m^*(x)$
- 就是AdaBoost算法得到的 α_m 和 $G_m(x)$



前向分步算法

- 首先, 求 $G_m^*(x)$; 对任意 $\alpha > 0$, 使式

$$(\alpha_m, G_m(x)) = \arg \min_{\alpha, G} \sum_{i=1}^N \bar{w}_{mi} \exp[-y_i \alpha G(x_i)]$$

- 最小的 $G(x)$ 由下式得到:

$$G_m^*(x) = \arg \min_G \sum_{i=1}^N \bar{w}_{mi} I(y_i \neq G(x_i))$$

$$\bar{w}_{mi} = \exp[-y_i f_{m-1}(x_i)]$$

- 即 $G_m^*(x)$ 为AdaBoost算法的基本分类器 $G_m(x)$ 为使第 m 轮加权训练数据分类误差率最小的基本分类器



前向分步算法

- 求 α_m^* 由
$$\begin{aligned} & \sum_{i=1}^N \bar{w}_{mi} \exp[-y_i \alpha G(x_i)] \\ &= \sum_{y_i=G_m(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G_m(x_i)} \bar{w}_{mi} e^{\alpha} \\ &= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^N \bar{w}_{mi} I(y_i \neq G(x_i)) + e^{-\alpha} \sum_{i=1}^N \bar{w}_{mi} \end{aligned}$$
- 将已求得的 $G_m^*(x)$ 代入, 对 α 求导并使导数为0,

- 得:
$$\alpha_m^* = \frac{1}{2} \log \frac{1 - e_m}{e_m}$$

$$e_m = \frac{\sum_{i=1}^N \bar{w}_{mi} I(y_i \neq G_m(x_i))}{\sum_{i=1}^N \bar{w}_{mi}} = \sum_{i=1}^N w_{mi} I(y_i \neq G_m(x_i))$$

与AdaBoost的 α_m 完全一致



前向分步算法

- 每一轮样本权值的更新，由：

$$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$$

$$\bar{w}_{mi} = \exp[-y_i f_{m-1}(x_i)]$$

- 可得：
$$\bar{w}_{m+1,i} = \bar{w}_{m,i} \exp[-y_i \alpha_m G_m(x)]$$
- 与AdaBoost的权值更新一致，相差规范化因子，因而等价。



提升树

- 提升树是以分类树或回归树为基本分类器的提升方法；提升树被认为是统计学习中性能最好的方法之一。
- 提升树模型 (boosting tree)
 - 提升方法实际采用：加法模型(即基函数的线性组合)与前向分步算法，以决策树为基函数；
 - 对分类问题决策树是二叉分类树，
 - 对回归问题决策树是二叉回归树，
 - 基本分类器 $x < v$ 或 $x > v$ ，可以看作是由一个根结点直接连接两个叶结点的简单决策树，即所谓的决策树桩(decision stump)。

$$f_M(x) = \sum_{m=1}^M T(x; \Theta_m)$$

$T(x; \Theta_m)$ 表示决策树； Θ_m 为决策树的参数； M 为树的个数



提升树算法

- 前向分步算法：
- 首先确定初始提升树： $f_0(x) = 0$
- 第 m 步的模型： $f_m(x) = f_{m-1}(x) + T(x; \Theta_m)$
- 其中， $f_{m-1}(x)$ 为当前模型，通过经验风险极小化确定下一棵决策树的参数 θ_m

$$\hat{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

- 由于树的线性组合可以很好地拟合训练数据，即使数据中的输入与输出之间的关系很复杂也是如此，所以提升树是一个高功能的学习算法。



提升树算法

- 针对不同问题的提升树学习算法，使用的损失函数不同：
 - 用平方误差损失函数的回归问题，
 - 用指数损失函数的分类问题，
 - 用一般损失函数的一般决策问题。
- 对二类分类问题：提升树算法只需将AdaBoost算法中的基本分类器限制为二类分类树即可，这时的提升树算法是AdaBoost算法的特殊情况。
- 讨论回归问题提升树：



提升树算法

- 回归问题提升树:
- 已知训练数据集: $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, $x_i \in \mathcal{X} \subseteq \mathbf{R}^n$
- \mathcal{X} 为输入空间, \mathcal{Y} 为输出空间, $y_i \in \mathcal{Y} \subseteq \mathbf{R}$
- 将 \mathcal{X} 划分为 J 个互不相交的区域 R_1, R_2, \dots, R_J , 并且在每个区域上确定输出的常量 c_j , 那么, 树可表示为:

$$T(x; \Theta) = \sum_{j=1}^J c_j I(x \in R_j)$$

$$\Theta = \{(R_1, c_1), (R_2, c_2), \dots, (R_J, c_J)\}$$

- J 是回归树的复杂度即叶结点个数



提升树算法

- 前向分步算法： $f_0(x) = 0$
 $f_m(x) = f_{m-1}(x) + T(x; \Theta_m), \quad m = 1, 2, \dots, M$
 $f_M(x) = \sum_{m=1}^M T(x; \Theta_m)$

- 在前向分步算法的第 m 步，给定当前 f_{m-1} 需求解

$$\hat{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

- 得到第 m 棵树的参数 $\hat{\Theta}_m$
- 采用平方损失函数时： $L(y, f(x)) = (y - f(x))^2$

$$\begin{aligned} &L(y, f_{m-1}(x) + T(x; \Theta_m)) \\ &= [y - f_{m-1}(x) - T(x; \Theta_m)]^2 = [r - T(x; \Theta_m)]^2 \end{aligned}$$



清华大学

Tsinghua University

回归问题的提升树算法

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$

$$x_i \in \mathcal{X} \subseteq \mathbf{R}^n, y_i \in \mathcal{Y} \subseteq \mathbf{R}$$

输出: 提升树 $f_M(x)$.

(1) 初始化 $f_0(x) = 0$

(2) 对 $m = 1, 2, \dots, M$

(a) 按式 (8.27) 计算残差

$$r_{mi} = y_i - f_{m-1}(x_i), \quad i = 1, 2, \dots, N$$

(b) 拟合残差 r_{mi} 学习一个回归树, 得到 $T(x; \Theta_m)$

(c) 更新 $f_m(x) = f_{m-1}(x) + T(x; \Theta_m)$

(3) 得到回归问题提升树 $f_M(x) = \sum_{m=1}^M T(x; \Theta_m)$



例：• X的取值范围[0.5, 10.5], y的取值范围： [5.0,10.10], 用树桩做基函数；

x_i	1	2	3	4	5	6	7	8	9	10
y_i	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

• 求f1(x)回归树T1(x),
$$\min_s \left[\min_{c_1} \sum_{x_i \in R_1} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2} (y_i - c_2)^2 \right]$$

- 求切分点s: $R_1 = \{x | x \leq s\}$, $R_2 = \{x | x > s\}$
- 容易求得在R1,R2内部使平方损失误差达到最小值的 c_1, c_2 :

$$c_1 = \frac{1}{N_1} \sum_{x_i \in R_1} y_i, \quad c_2 = \frac{1}{N_2} \sum_{x_i \in R_2} y_i$$



例： • 各切分点：

1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, 9.5

$$m(s) = \min_{c_1} \sum_{x_i \in R_1} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2} (y_i - c_2)^2$$

当 $s = 1.5$ 时, $R_1 = \{1\}$, $R_2 = \{2, 3, \dots, 10\}$, $c_1 = 5.56$, $c_2 = 7.50$,

$$m(s) = \min_{c_1} \sum_{x_i \in R_1} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2} (y_i - c_2)^2 = 0 + 15.72 = 15.72$$

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
$m(s)$	15.72	12.07	8.36	5.78	3.91	1.93	8.01	11.73	15.74

- 回归树T1
$$T_1(x) = \begin{cases} 6.24, & x < 6.5 \\ 8.91, & x \geq 6.5 \end{cases}$$
$$f_1(x) = T_1(x)$$



例：

	$r_{2i} = y_i - f_1(x_i)$									
x_i	1	2	3	4	5	6	7	8	9	10
r_{2i}	-0.68	-0.54	-0.33	0.16	0.56	0.81	-0.01	-0.21	0.09	0.14

- 用 f_1 拟合数据的平方误差：
$$L(y, f_1(x)) = \sum_{i=1}^{10} (y_i - f_1(x_i))^2 = 1.93$$

- 第二步：求 T_2 ，
$$T_2(x) = \begin{cases} -0.52, & x < 3.5 \\ 0.22, & x \geq 3.5 \end{cases}$$

$$f_2(x) = f_1(x) + T_2(x) = \begin{cases} 5.72, & x < 3.5 \\ 6.46, & 3.5 \leq x < 6.5 \\ 9.13, & x \geq 6.5 \end{cases}$$

$$L(y, f_2(x)) = \sum_{i=1}^{10} (y_i - f_2(x_i))^2 = 0.79$$



清华大学

Tsinghua University

例: $T_3(x) = \begin{cases} 0.15, & x < 6.5 \\ -0.22, & x \geq 6.5 \end{cases} \quad L(y, f_3(x)) = 0.47$

$$T_4(x) = \begin{cases} -0.16, & x < 4.5 \\ 0.11, & x \geq 4.5 \end{cases} \quad L(y, f_4(x)) = 0.30$$

$$T_5(x) = \begin{cases} 0.07, & x < 6.5 \\ -0.11, & x \geq 6.5 \end{cases} \quad L(y, f_5(x)) = 0.23$$

$$T_6(x) = \begin{cases} -0.15, & x < 2.5 \\ 0.04, & x \geq 2.5 \end{cases}$$

$$f_6(x) = f_5(x) + T_6(x) = T_1(x) + \cdots + T_5(x) + T_6(x)$$

$$= \begin{cases} 5.63, & x < 2.5 \\ 5.82, & 2.5 \leq x < 3.5 \\ 6.56, & 3.5 \leq x < 4.5 \\ 6.83, & 4.5 \leq x < 6.5 \\ 8.95, & x \geq 6.5 \end{cases}$$

$$L(y, f_6(x)) = \sum_{i=1}^{10} (y_i - f_6(x_i))^2 = 0.17$$



梯度提升

- 提升树利用加法模型与前向分步算法实现学习的优化过程，当损失函数是平方损失和指数损失函数时，每一步优化是很简单的，但对一般损失函数而言，往往每一步优化并不容易。
- 针对这一问题，Freidmao提出了梯度提升(gradient boosting)算法。这是利用最速下降法的近似方法，其关键是利用损失函数的负梯度在当前模型的值

$$-\left[\frac{\partial L(y, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$

- 作为回归问题提升树算法中的残差的近似值，拟合一个回归树。



梯度提升算法

(1) 初始化 $f_0(x) = \arg \min_c \sum_{i=1}^N L(y_i, c)$

(2) 对 $m=1, 2, \dots, M$

(a) 对 $i=1, 2, \dots, N$, 计算 $r_{mi} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=f_{m-1}(x)}$

(b) 对 r_{mi} 拟合一个回归树, 得到第 m 棵树的叶结点区域 R_{mj}

(c) 对 $j=1, 2, \dots, J$, 计算 $c_{mj} = \arg \min_c \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c)$

(d) 更新 $f_m(x) = f_{m-1}(x) + \sum_{j=1}^J c_{mj} I(x \in R_{mj})$

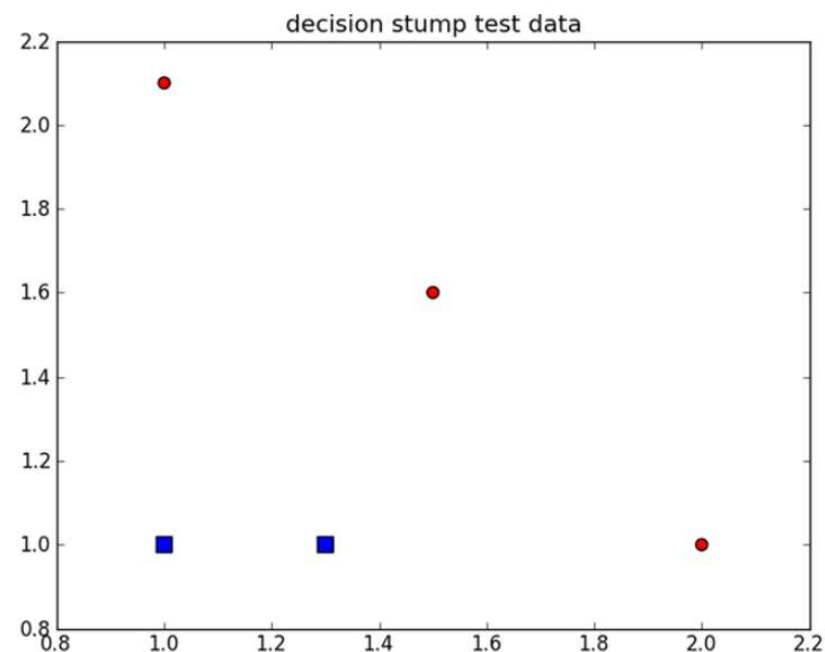
(3) 得到回归树

$$\hat{f}(x) = f_M(x) = \sum_{m=1}^M \sum_{j=1}^J c_{mj} I(x \in R_{mj})$$



Adaboost的实现

```
def loadSimpData():  
    datMat = matrix([[ 1. ,  2.1],  
                      [ 2. ,  1.1],  
                      [ 1.3,  1. ],  
                      [ 1. ,  1. ],  
                      [ 2. ,  1. ]])  
    classLabels = [1.0, 1.0, -1.0, -1.0, 1.0]  
    return datMat,classLabels
```





单层决策树生成函数

```
def stumpClassify(dataMatrix, dimen, threshVal, threshIneq):  
    retArray = ones((shape(dataMatrix)[0], 1))  
    if threshIneq == 'lt':  
        retArray[dataMatrix[:, dimen] <= threshVal] = -1.0  
    else:  
        retArray[dataMatrix[:, dimen] > threshVal] = -1.0  
    return retArray
```




清华大学

Tsinghua University

单层决策树生成函数

```
def buildStump(dataArr,classLabels,D):  
    dataMatrix = mat(dataArr); labelMat = mat(classLabels).T  
    m,n = shape(dataMatrix)  
    numSteps = 10.0; bestStump = {}; bestClasEst = mat(zeros((m,1)))  
    minError = inf  
    for i in range(n):  
        rangeMin = dataMatrix[:,i].min(); rangeMax = dataMatrix[:,i].max()  
        stepSize = (rangeMax-rangeMin)/numSteps  
        for j in range(-1,int(numSteps)+1):  
            for inequal in ['lt', 'gt']:  
                threshVal = (rangeMin + float(j) * stepSize)  
                predictedVals = \  
                    stumpClassify(dataMatrix,i,threshVal,inequal)  
                errArr = mat(ones((m,1)))  
                errArr[predictedVals == labelMat] = 0  
                weightedError = D.T*errArr
```




清华大学

Tsinghua University

单层决策树生成函数

```
#print "split: dim %d, thresh %.2f, thresh inequal:
      %s, the weighted error is %.3f" %\
      (i, threshVal, inequal, weightedError)
if weightedError < minError:
    minError = weightedError
    bestClasEst = predictedVals.copy()
    bestStump['dim'] = i
    bestStump['thresh'] = threshVal
    bestStump['ineq'] = inequal
return bestStump,minError,bestClasEst
```



清华大学

Tsinghua University

单层决策树生成函数

```
>>> D = mat(ones((5,1))/5)
>>> adaboost.buildStump(datMat,classLabels,D)
split: dim 0, thresh 0.90, thresh inequal: lt, the weighted error is 0.400
split: dim 0, thresh 0.90, thresh inequal: gt, the weighted error is 0.600
split: dim 0, thresh 1.00, thresh inequal: lt, the weighted error is 0.400
split: dim 0, thresh 1.00, thresh inequal: gt, the weighted error is 0.600
.
.
split: dim 1, thresh 2.10, thresh inequal: lt, the weighted error is 0.600
split: dim 1, thresh 2.10, thresh inequal: gt, the weighted error is 0.400
({'dim': 0, 'ineq': 'lt', 'thresh': 1.3}, matrix([[ 0.2]]), array([[ -1.],
[ 1.],
[ -1.],
[ -1.],
[ 1.]])
```



完整AdaBoost算法的实现

```
def adaBoostTrainDS(dataArr, classLabels, numIt=40):  
    weakClassArr = []  
    m = shape(dataArr)[0]  
    D = mat(ones((m,1))/m)  
    aggClassEst = mat(zeros((m,1)))  
    for i in range(numIt):  
        bestStump, error, classEst = buildStump(dataArr, classLabels, D)  
        print "D:", D.T  
        alpha = float(0.5*log((1.0-error)/max(error, 1e-16)))  
        bestStump['alpha'] = alpha  
        weakClassArr.append(bestStump)  
        print "classEst: ", classEst.T  
        expon = multiply(-1*alpha*mat(classLabels).T, classEst)  
        D = multiply(D, exp(expon))  
        D = D/D.sum()  
        aggClassEst += alpha*classEst
```



清华大学

Tsinghua University

完整AdaBoost算法的实现

```
print "aggClassEst: ",aggClassEst.T
    aggErrors = multiply(sign(aggClassEst) !=
                        mat(classLabels).T,ones((m,1)))
    errorRate = aggErrors.sum()/m
    print "total error: ",errorRate,"\n"
    if errorRate == 0.0: break
return weakClassArr

>>> classifierArray = adaboost.adaBoostTrainDS(datMat,classLabels,9)
D: [[ 0.2  0.2  0.2  0.2  0.2]]
classEst: [[-1.  1. -1. -1.  1.]]
aggClassEst: [[-0.69314718  0.69314718 -0.69314718 -0.69314718
               0.69314718]]
total error: 0.2

D: [[ 0.5   0.125  0.125  0.125  0.125]]
classEst: [[ 1.  1. -1. -1. -1.]]
aggClassEst: [[ 0.27980789  1.66610226 -1.66610226 -1.66610226
               -0.27980789]]
total error: 0.2
```



清华大学

Tsinghua University

完整AdaBoost算法的实现

```
D: [[ 0.28571429  0.07142857  0.07142857  0.07142857  0.5      ]]  
classEst: [[ 1.  1.  1.  1.  1.]]  
aggClassEst: [[ 1.17568763  2.56198199 -0.77022252 -0.77022252  
               0.61607184]]  
total error: 0.0
```



```
>>> classifierArray
[{'dim': 0, 'ineq': 'lt', 'thresh': 1.3, 'alpha': 0.69314718055994529},
 {'dim': 1, 'ineq': 'lt', 'thresh': 1.0, 'alpha': 0.9729550745276565},
 {'dim': 0, 'ineq': 'lt', 'thresh': 0.90000000000000002, 'alpha':
  0.89587973461402726}]
```




测试算法

- 基于AdaBoost的分类函数

```
def adaClassify(datToClass, classifierArr):  
    dataMatrix = mat(datToClass)  
    m = shape(dataMatrix)[0]  
    aggClassEst = mat(zeros((m,1)))  
    for i in range(len(classifierArr)):  
        classEst = stumpClassify(dataMatrix, classifierArr[i]['dim'],\  
                                   classifierArr[i]['thresh'],\  
                                   classifierArr[i]['ineq'])  
        aggClassEst += classifierArr[i]['alpha']*classEst  
    print aggClassEst  
    return sign(aggClassEst)
```



测试算法

- 基于AdaBoost的分类函数

```
>>> reload(adaboost)
<module 'adaboost' from 'adaboost.py'>
>>> datArr,labelArr=adaboost.loadSimpData()
>>> classifierArr = adaboost.adaBoostTrainDS(datArr,labelArr,30)
>>> adaboost.adaClassify([0, 0],classifierArr)
[[-0.69314718]]
[[-1.66610226]]
[[-2.56198199]]
matrix([[ -1.]])
>>> adaboost.adaClassify([[5, 5],[0,0]],classifierArr)
[[ 0.69314718]
.
.
[-2.56198199]]
matrix([[ 1.]
        [-1.]])
```




在马疝病数据集上应用Adaboost

- 加载数据

```
def loadDataSet(fileName):  
    numFeat = len(open(fileName).readline().split('\t'))  
    dataMat = []; labelMat = []  
    fr = open(fileName)  
    for line in fr.readlines():  
        lineArr = []  
        curLine = line.strip().split('\t')  
        for i in range(numFeat-1):  
            lineArr.append(float(curLine[i]))  
        dataMat.append(lineArr)  
        labelMat.append(float(curLine[-1]))  
    return dataMat, labelMat
```



清华大学

Tsinghua University

在马疝病数据集上应用Adaboost

```
>>> datArr,labelArr = adaboost.loadDataSet('horseColicTraining2.txt')
>>> classifierArray = adaboost.adaBoostTrainDS(datArr,labelArr,10)
total error: 0.284280936455
total error: 0.284280936455
```

```
total error: 0.230769230769
>>> testArr,testLabelArr = adaboost.loadDataSet('horseColicTest2.txt')
>>> prediction10 = adaboost.adaClassify(testArr,classifierArray)
To get the number of misclassified examples type in:

>>> errArr=mat(ones((67,1)))
>>> errArr[prediction10!=mat(testLabelArr).T].sum()
16.0
```



在马疝病数据集上应用Adaboost

Number of Classifiers	Training Error	Test Error
1	0.28	0.27
10	0.23	0.24
50	0.19	0.21
100	0.19	0.22
500	0.16	0.25
1000	0.14	0.31
10000	0.11	0.33

overfitting



SVM和AdaBoost的关系

- 相似之处
 - 把弱分类器看成SVM的核函数。
 - 按照最大化某个最小间隔的方式重写AdaBoost算法。
- 不同
 - 二者所定义的间隔计算方式有所不同，导致结果不同。
 - 高维空间下，二者间差异会更加明显。



非均衡分类问题

- 1、马疮病
- 2、垃圾邮件
- 3、癌症检测
- 分类性能度量指标：正确率、召回率、ROC曲线
- 混淆矩阵 Confu

		Predicted		
		Dog	Cat	Rat
Actual	Dog	24	2	5
	Cat	2	27	0
	Rat	4	2	30



非均匀分类问题

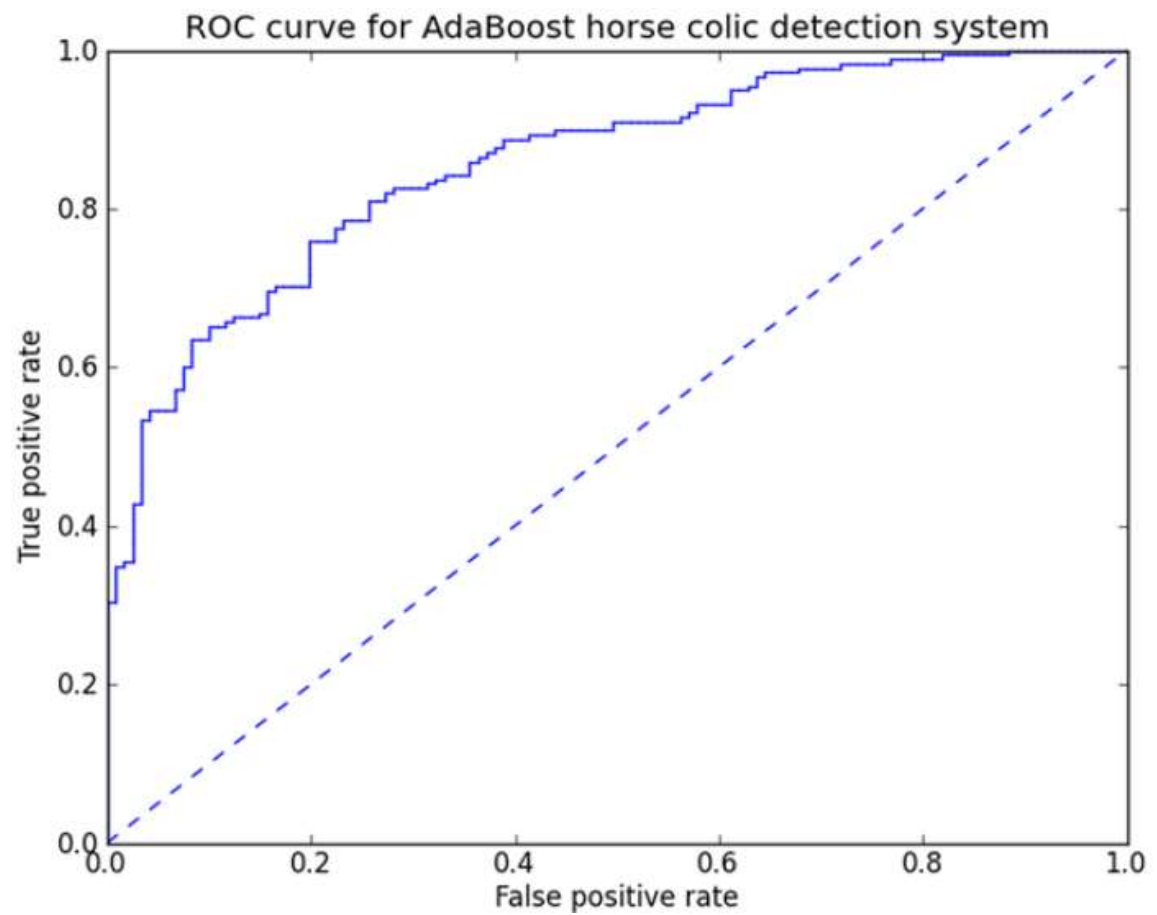
- 二分类问题的混淆矩阵

		redicted	
		+1	-1
Actual	+1	True Positive (TP)	False Negative (FN)
	-1	False Positive (FP)	True Negative (TN)

- Precision 正确率= $TP/(TP+FP)$
- Recall 召回率= $TP/(TP+FN)$
- 假阳率= $FP/(FP+TN)$
- 真阳率= $TP/(TP+FN)$



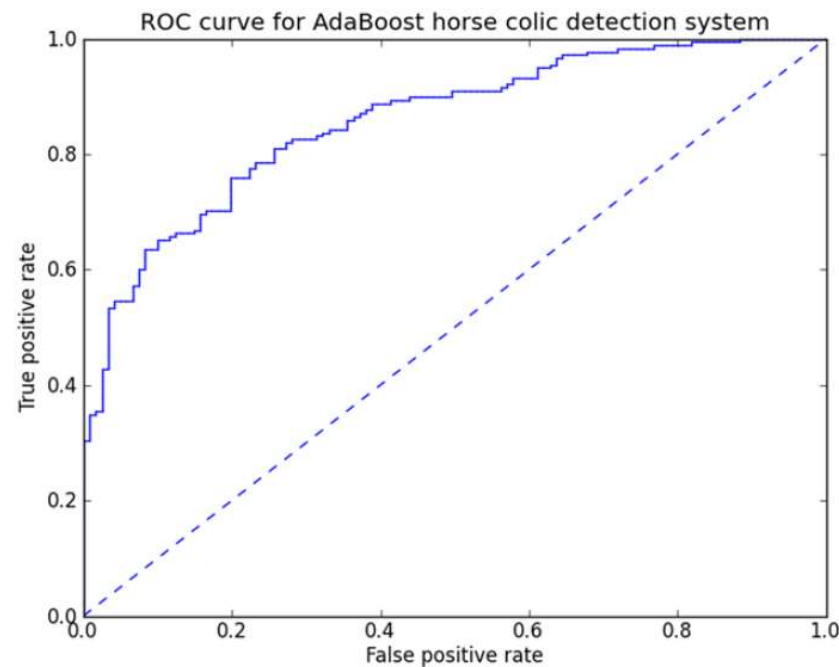
非均匀分类问题





非均匀分类问题


- 接收者操作特性 (receiver operating Characteristic)
- 成本效益 (cost-versus-benefit)
- 曲线下面积 (Area Under the Curve, AUC)





基于代价函数的分类器决策控制

		Predicted	
		+1	-1
Actual	+1	0	1
	-1	1	0



		Predicted	
		+1	-1
Actual	+1	-5	1
	-1	50	0



处理非均衡数据的数据抽样方法

- 欠抽样 (undersampling)
- 过抽样 (oversampling)



清华大学
Tsinghua University

• END