机器学习

Machine Learning

主讲人：张敏 清华大学长聘副教授

ML 机器学习

MACHINE LEARNING-MIN ZHANG

Unit.09

集成学习

*图片均来自网络或已发表刊物

背景

“*Two heads are better than one.*”

“三个臭皮匠，顶一个诸葛亮”

- 融合多个学习方法的结果来提升效果

  Ensemble learning 集成学习

1. 集成学习简介

要求

	样本1	样本2	样本3
h_1	✓	✓	×
h_2	×	✓	✓
h_3	✓	×	✓
集成	✓	✓	✓

(a) 积极影响

	样本1	样本2	样本3
h_1	✓	✓	×
h_2	✓	✓	×
h_3	✓	✓	×
集成	✓	✓	×

(b) 没有影响

	样本1	样本2	样本3
h_1	✓	×	×
h_2	×	✓	×
h_3	×	×	✓
集成	×	×	×

(c) 负面影响

h_i: 第i个分类器

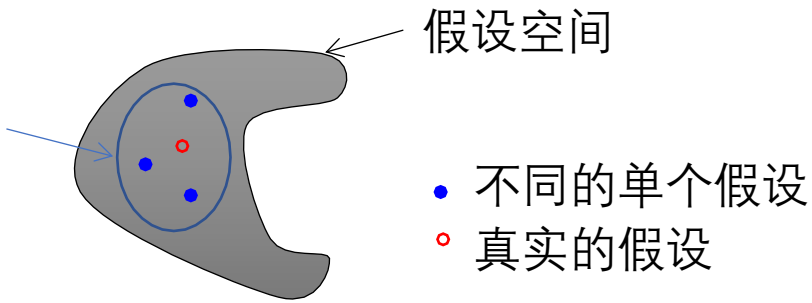
✓: 正确预测　×: 错误预测

集成: 投票集成

分类器需要 效果好 且 不同!

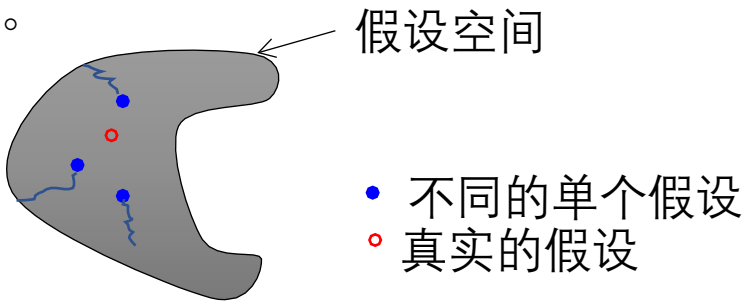
集成学习简介

- 直觉: 把对同一个问题的多个预测结果综合起来考虑的精度，应该比单一学习方法效果好
- 证实: (一些理由)
  - 1. 很容易找到非常正确的 “rules of thumb（经验法则）”，但是很难找到单个的有高准确率的规则
  - 2. 如果训练样本很少，假设空间很大，则存在多个同样精度的假设。选择某一个假设可能在测试集上效果较差。



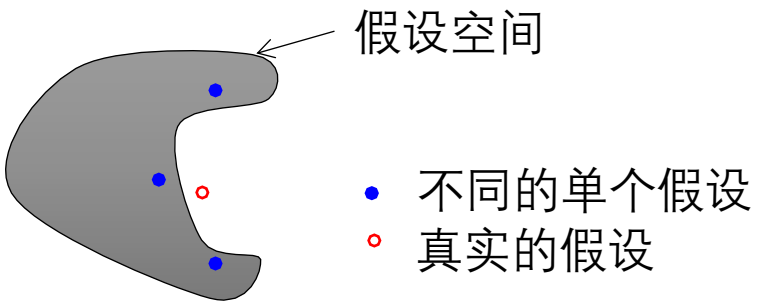
集成学习简介

- 直觉: 把对同一个问题的多个预测结果综合起来考虑的精度，应该比单一学习方法效果好
- 证实: (一些理由)
  - 3. 算法可能会收敛到局部最优解。融合不同的假设可以降低收敛到一个不好的局部最优的风险。或者
  - 在假设空间中穷举地全局搜索代价太大， 所以我们可以结合一些在局部预测比较准确的假设。



集成学习简介

- 直觉: 把对同一个问题的多个预测结果综合起来考虑的精度，应该比单一学习方法效果好
- 证实: (一些理由)
  - 4.由当前算法定义的 假设空间不包括真实的假设, 但做了一些不错的近似



两个概念

- 强学习器: 有高准确度的学习算法
- 弱学习器: 在任何训练集上可以做到比随机预测 略好

$$error = \frac{1}{2} - \gamma$$

我们能否把一个弱学习器增强成一个强学习器?

集成学习: 基本想法

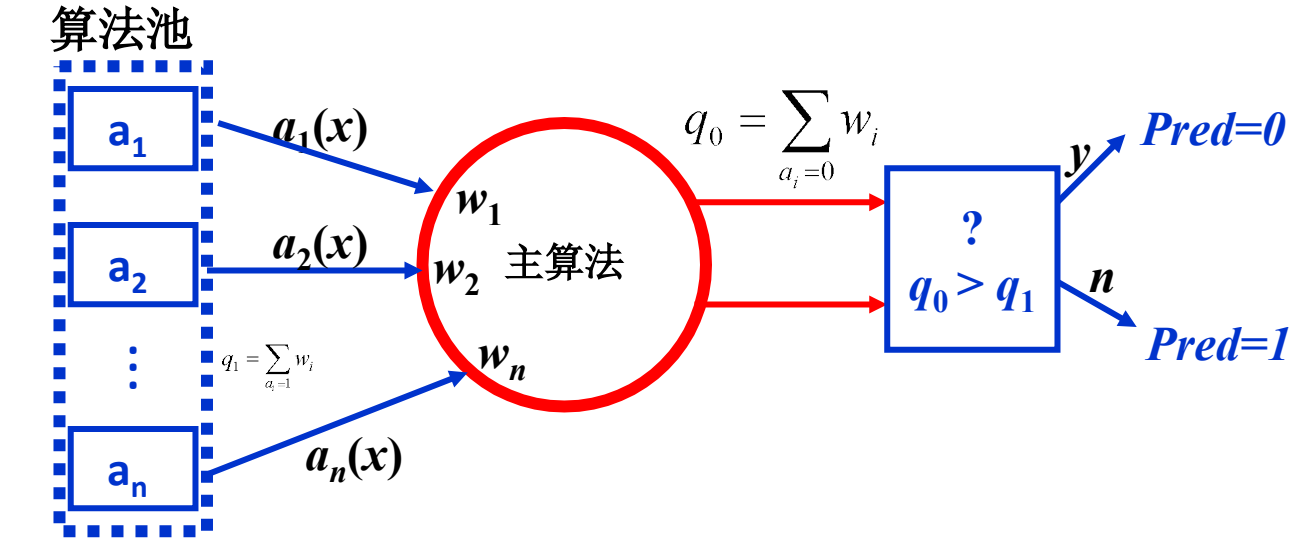
- 有时一个单个分类器 (e.g. 决策树、神经网络…) 表现不好，但是 它们的加权融合表现不错.
- 算法池 中的每一个学习器都有它的权重
- 当需要对一个新的实例作预测时
  - 每个学习器作出自己的预测
  - 然后主算法把这些结果根据权值合并起来，作出最终预测

集成策略

- 平均
  - 简单平均
  - 加权平均
- 投票
  - 多数投票法
  - 加权投票法
- 学习
  - 加权多数
  - 堆叠（Stacking）：层次融合，基学习器的输出作为次学习器的输入

2. Weighted Majority Algorithm (加权多数算法)

加权多数算法 – 预测



假设: 二值输出{0,1}

加权多数算法 – 训练

a_i是算法池A中第i 个预测算法，每个算法对输入X有二值输出$\{0,1\}$

w_i对应a_i的权值

- $\forall\, i$，$w_i \leftarrow 1$
- 对每个训练样本 $<x, c(x)>$
  - 初始化 $q_0 \leftarrow q_1 \leftarrow 0$
  - 对每个算法 a_i
    - 如果 $a_i(x)=0$，那么 $q_0 \leftarrow q_0+w_i$，否则 $q_1 \leftarrow q_1+w_i$
  - 如果 $q_0 > q_1$，则预测 $c(x)=0$，否则预测 $c(x)=1$ （$q_0 = q_1$ 时取任意值）
  - 对每个 $a_i \in A$
    - 如果 $a_i(x) \neq c(x)$，那么 $w_i \leftarrow \beta\, w_i$（$\beta \in [0,1)$ 是惩罚系数）

$\beta=0$ 时是作用在 A 上的 Halving 算法

3. Bagging

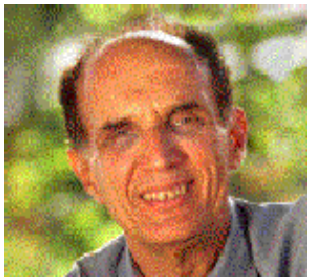
如果我们只有一个弱学习器，如何通过集成来提升它的表现？

直觉

- 则不同的数据上训练可以获得不同的基础模型
- 一个朴素的方法: 从训练集种采样不同的子集且训练不同的基础模型
  - 这些模型会大不相同
  - 但它们的效果可能很差
- 解决办法: 拔靴采样（Bootstrap sampling）

Bagging: 背景

- **Bagging = Bootstrap aggregating**



- Bootstrap: Bradley Efron 在 1993年提出
  - 统计学教授
  - 斯坦福大学
  - 拔靴采样、生物统计学、天体物理中的统计方法

- “我喜欢同时处理应用和理论问题。**统计学的一个好处是你可以在很多领域发挥作用。**我目前的应用包括生物统计和天体物理应用。令人惊讶的是，这两个领域使用的方法是相似的。我在宾夕法尼亚州立大学做了一个叫做 **天体物理学和生物统计学 -- 一对奇特的夫妇** 的演讲，其中提到了这个观点。”

Bagging: 背景

- Bagging = Bootstrap aggregating
- Bootstrap sampling (拔靴法/自举法采样)
  - 给定集合 D， 含有 m 训练样本
  - 通过从D中均匀随机的有放回采样m个样本构建 D_i (drawn with replacement, 取出放回， 有放回采样)
  - 希望 D_i 会 遗漏掉 D 中的某些样本

Bagging: 算法

- Bagging: 由 Breiman 在 1994 年提出
  - 伯克利大学统计学荣誉教授
  - 美国科学院院士
- Bagging 算法

Leo Breiman

For $t = 1, 2, \ldots, T$ *Do*

从S中拔靴采样产生 D_t

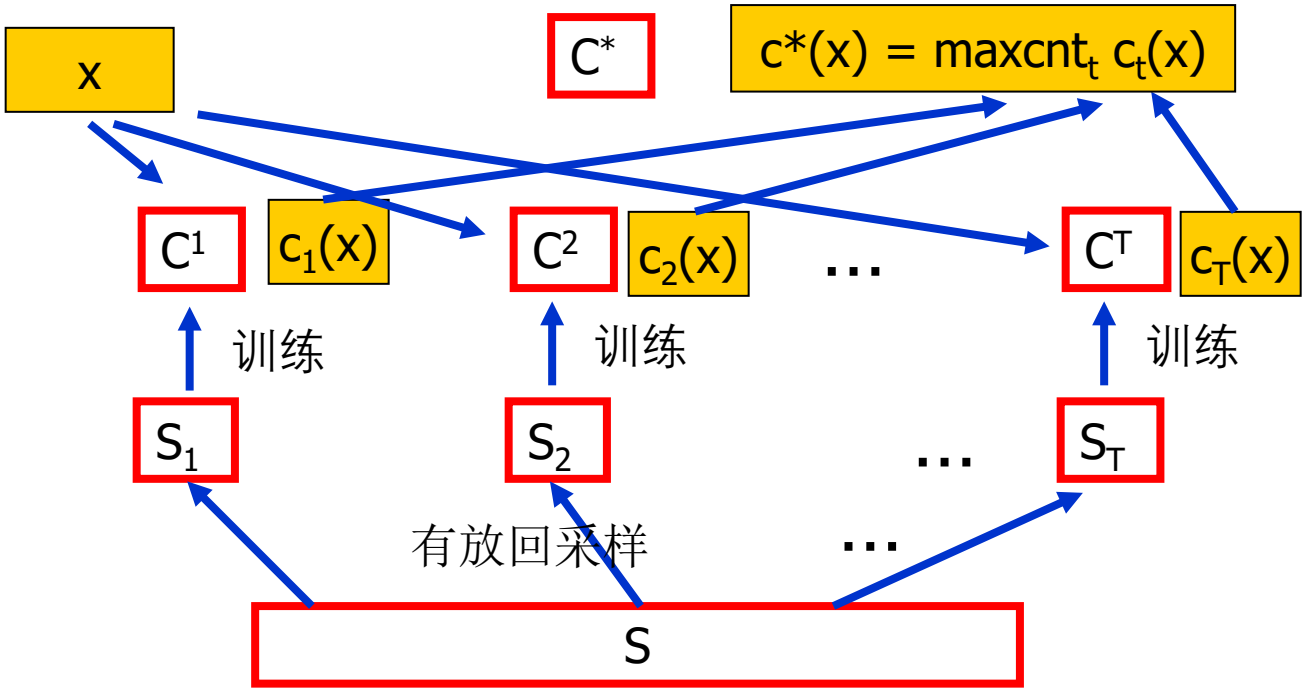
在 D_t 上训练一个分类器H_t

分类一个新的样本$x \in X$ 时，通过对 H_t 多数投票(等权重)

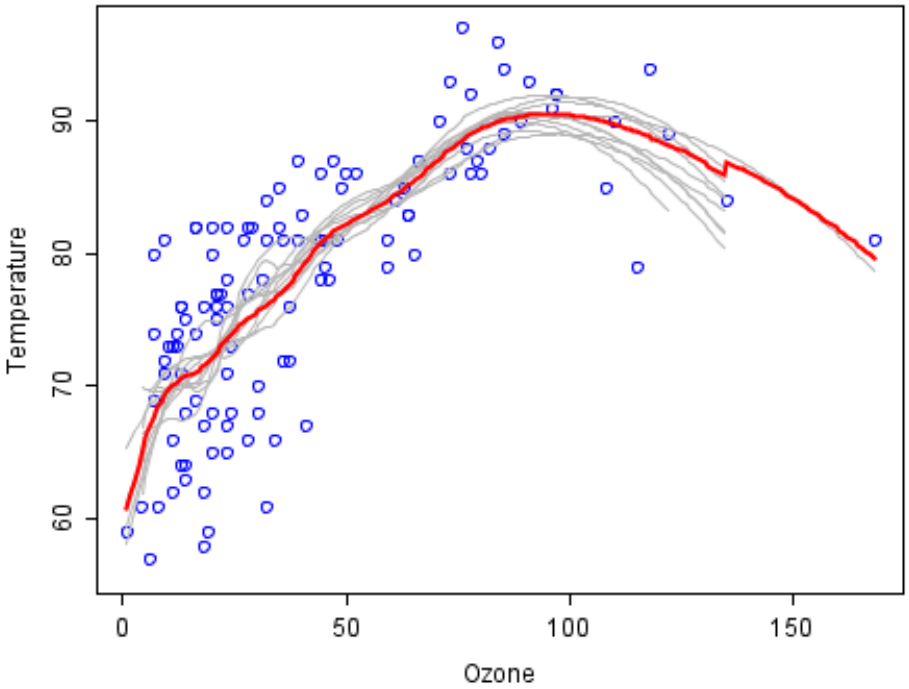
可以给出连续值预测

也可以特定问题上用不同的结合方法

Bagging



Bagging 应用举例



数据集: Rousseeuw 和 Leroy (1986)，关心 臭氧含量 vs. 温度。

100 拔靴采样样本。灰色线条: 初始的10个预测器; 红色线条: 平均。

拔靴采样多少样本?

Table 5.1
Bagged Missclassification Rates (%)

No. Bootstrap Replicates	Missclassification Rate
10	21.8
25	19.5
50	19.4
100	19.4

Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

Bagging: 结果 (接上)

给定样例集S，Breiman重复下列工作100次报告平均结果:

方法 I:

1. 把 S 随机划分成测试集 T(10%) 和训练集 D (90%)
2. 从D中训练 决策树 算法，记 e_S 为它在测试集 T 上的错误率

方法 II:

重复50次: 生成拔靴采样集合D_i，进行决策树学习，记 e_B 为决策树多数投票在测试集T上的错误率 （集成大小 = 50）

Table 1 Missclassification Rate (Percent)

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	22%
soybean	14.5	10.6	27%

Breiman “Bagging Predictors” Berkeley Statistics Department TR#421, 1994

Bagging: 结果 (续)

- 同样的实验，但使用 最近邻分类器 （欧式距离）
- 结果

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	26.1	26.1	0%
heart	6.3	6.3	0%
breast cancer	4.9	4.9	0%
ionosphere	35.7	35.7	0%
diabetes	16.4	16.4	0%
glass	16.4	16.4	0%

- 发生了什么？为什么？

Bagging：讨论

- Bagging 在学习器“不稳定”时有用
  - “关键是预测方法的不稳定性”
  - E.g. 决策树、神经网络
- 为什么？
  - 不稳定: 训练集小的差异可以造成产生的假设大不相同
  - “如果打乱训练集合可以 造成产生的预测器大不相同，则bagging算法可以提升其准确率。”(Breiman 1996)

总结

- 加权多数算法
  - 相同数据集， 不同学习算法
  - 产生多个模型，加权融合
- Bagging
  - 一个数据集， 一个弱分类器
  - 生成多组训练样本 来训练多个模型，然后集成

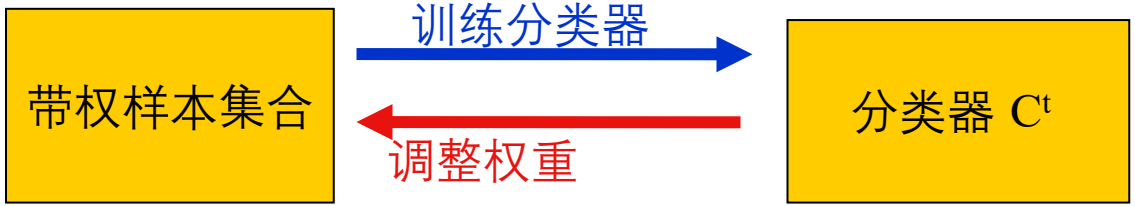
是否有一个集成算法能够考虑 学习中数据的差异性

?

4. Boosting

Boosting：基本想法

- “从失败中学习”
- 基本想法：
  - 给每个样本一个权值
  - T 轮迭代，在每轮迭代后增大错误分类样本的权重
    - 更关注于“难”样本



Valiant Leslie G.

- 哈佛大学
- 美国科学院院士
- 理论计算科学
- 2010 图灵奖

Boosting 背景

- [Kearns&Valiant'88]
  提出寻找boosting算法的开放问题
- [Schapire'89], [Freund'90]
  第一个 多项式时间的 boosting 算法
- [Drucker, Schapire & Simard '92]
  第一个使用boosting算法的实验
- [Freund & Schapire '95]
  - 提出 AdaBoost 算法
  - 相比于之前的boosting算法在实际使用中优势很大
- 使用AdaBoost的实验，继续发展理论和算法 (使用不那么弱的学习器，等等)

AdaBoost

- 初始给每个样本赋相等权重为 *1/N*;
- *For t = 1, 2, ..., T Do*
  - 生成一个假设 C_t;
  - 计算错误率 ϵ_t:

    ϵ_t = 所有错误分类的样本权重和
  - $$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$$
  - 更新每个样本的权重:

    正确分类: $W_{new} = W_{old} * e^{-\alpha_t}$

    错误分类: $W_{new} = W_{old} * e^{\alpha_t}$
  - 归一化 权重 (权重和=1);
- 融合所有假设 C_t， 各自投票权重为 α_t

这里是二分类!

AdaBoost.M1

Vs. AdaBoost

- 初始给每个样本赋相等权重为 *1/N* ;
- *For t = 1, 2, ..., T Do*
  - 生成一个假设 C_t;
  - 计算错误率 ϵ_t :

    ϵ_t= 所有错误分类样本权重和

    如果 $\epsilon_t > 0.5$, 则退出循环

    选择一个更好的 C_t

    NA
  - $\beta_t = \epsilon_t/(1-\epsilon_t)$

    $\alpha_t = 1/2 \ln((1-\epsilon_t)/\epsilon_t)$
  - 更新 每个样本的权重:

    正确 分类: $W_{new} = W_{old} * \beta_t$

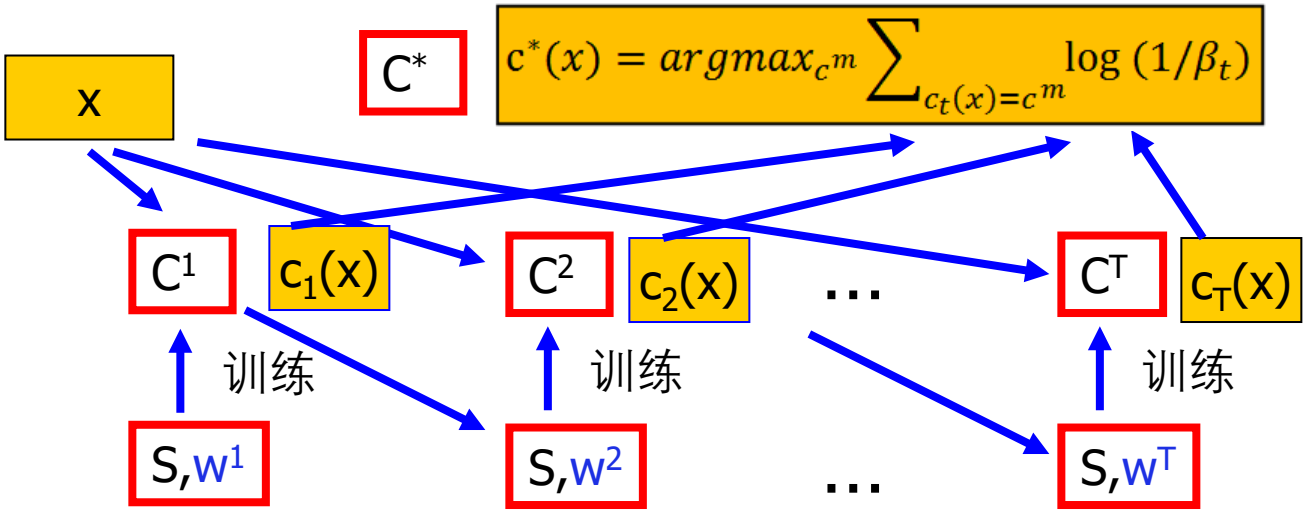
    错误 分类: $W_{new} = W_{old}$

    $W_{new} = W_{old} * e^{-\alpha_t}$

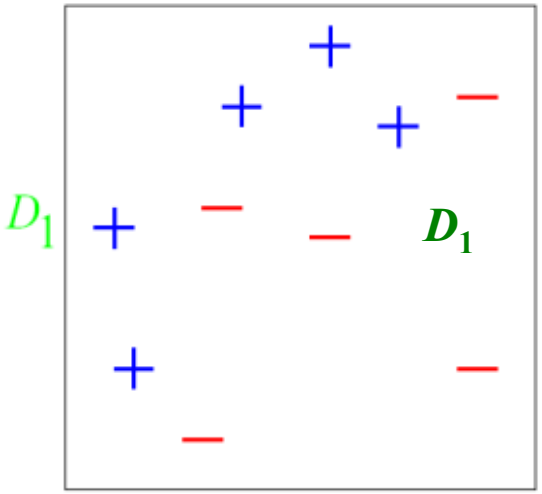
    $W_{new} = W_{old} * e^{\alpha_t}$
  - 归一化权重(权重和=1);
- 融合所有假设 C_t， 各自投票权重为 $\log[1/\beta_t]$

α_t

Boosting


x
C*
$c^*(x) = argmax_{c^m} \sum_{c_t(x)=c^m} \log(1/\beta_t)$
C1
c1(x)
C2
c2(x)
...
CT
cT(x)
训练
训练
训练
S,w1
S,w2
...
S,wT

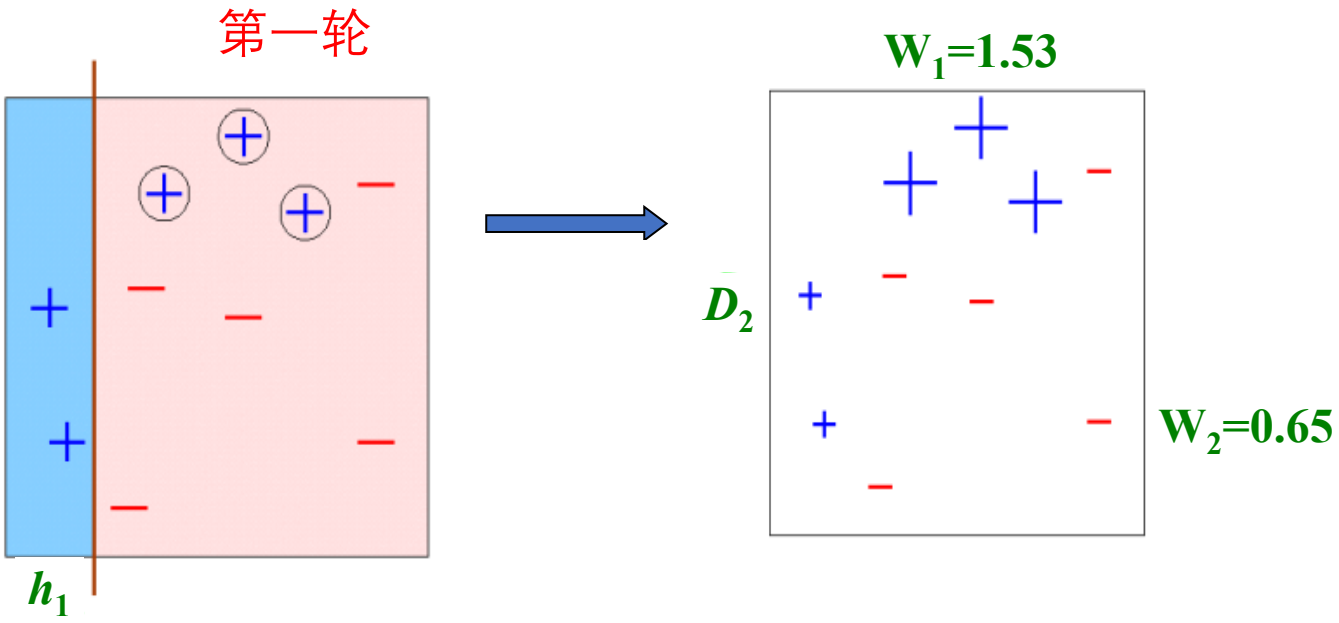
AdaBoost 举例



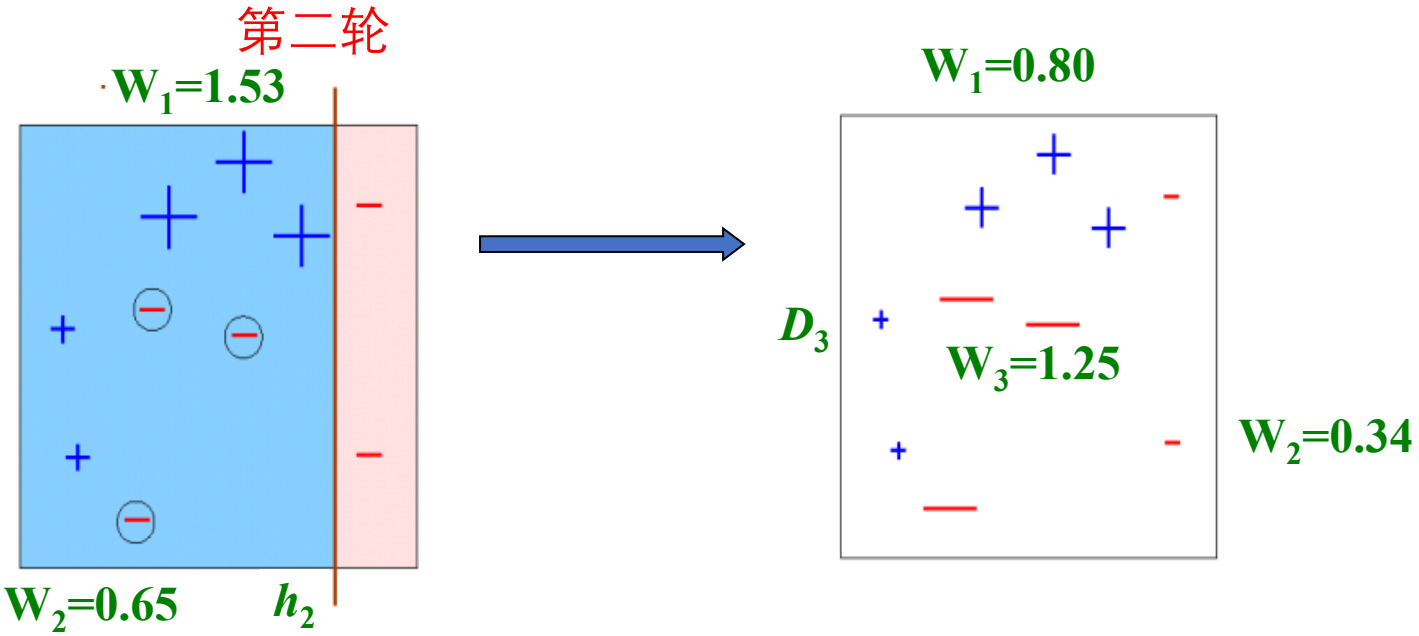
原始训练集: 所有样本相同权重

-- 来自Yoav Freund, Rob Schapire, **A Tutorial on Boosting**

AdaBoost 举例

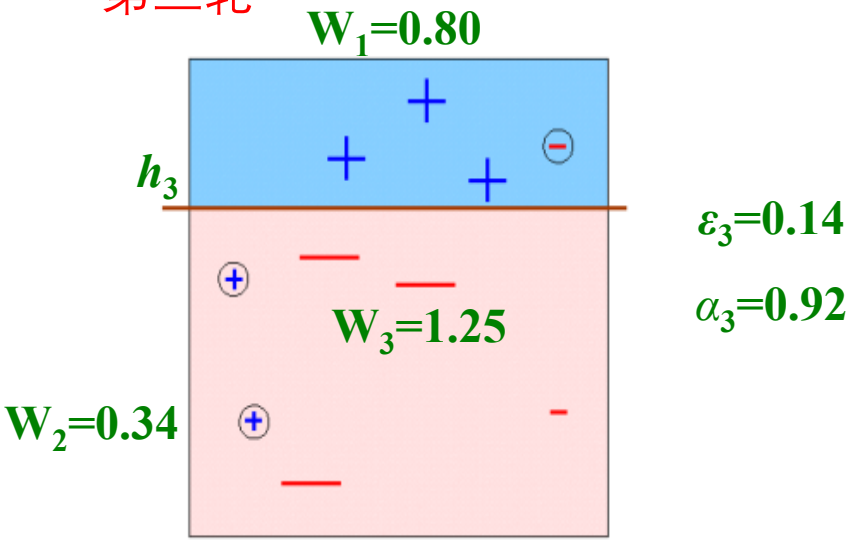


AdaBoost 举例



AdaBoost 举例

第三轮

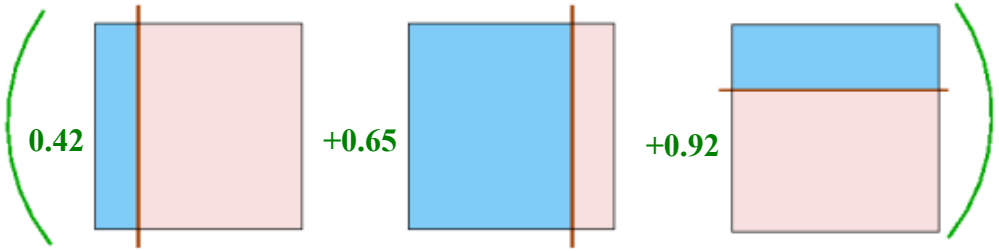


$\varepsilon_3=0.14$

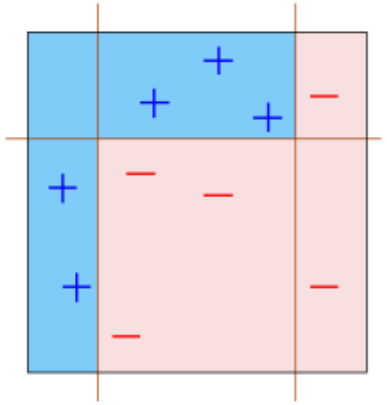
$\alpha_3=0.92$

AdaBoost 举例：最终假设

H_{final}=sign



=



AdaBoost在实际使用时的优点

- (非常) 快
- 简单 + 易编程实现
- 只有 一个参数 要调 (T)
- 没有 先验知识
- 灵活: 可以和 任何分类器 结合(NN, C4.5, …)
- 被证明是有效的 (尤其是 弱学习器)
  - 转变了思路:

    现在目标是 仅仅需要寻找比随机猜测好一些的假设就可以了

AdaBoost 注意事项

- 性能依赖于 数据 & 弱学习器
- 在下列情况中使用AdaBoost会 失效
  - 弱学习器 太复杂 (过拟合)
  - 弱学习器 太弱 ($\alpha_t \to 0$ 太快),
    - 欠拟合
    - 边界太窄 → 过拟合
- 过去的实验表明，AdaBoost 似乎 很容易受到噪声的影响

5. 讨论

Bagging vs. Boosting

- 训练集合
  - Bagging: 随机选择样本， 各轮训练集相互独立
  - Boosting: 与前轮的学习结果有关， 各轮训练集并不独立
- 预测函数
  - Bagging: 没有权重; 便于并行
  - Boosting: 权重变化 呈指数; 只能顺序进行

Bagging vs. Boosting (接上)

- 效果
  - 实际中，bagging几乎总是有效
  - 平均地说，Boosting比bagging好一些，但boosting算法也较常出现损害系统性能的情况
  - 对 稳定模型来说bagging效果不好，Boosting可能仍然有效
  - Boosting 可能在有噪声数据上带来性能损失。Bagging没有这个问题

重新调权 vs. 重新采样

- Reweighting 调整样本权重可能更难处理
  - 一些学习方法无法使用样本权重
  - 很多常用工具包不支持训练集上的权重
- Resampling 我们可以重采样来代替:
  - 对数据使用boostrap抽样，抽样时根据每个样例的权值确定其被抽样的概率
- 一般重新调权效果会更好一些 但 重新抽样更容易实现

Bagging & boosting 应用

- 互联网内容过滤
- 图像识别
- 手写识别
- 语音识别
- 文本分类
- ……

总结

- 集成学习简介
- 方法
  - 加权多数算法
  - Bagging
    - Boostrap采样
  - Boosting
- 更多讨论
  - Bagging vs. boosting
  - 重新调权 vs. 重新采样