

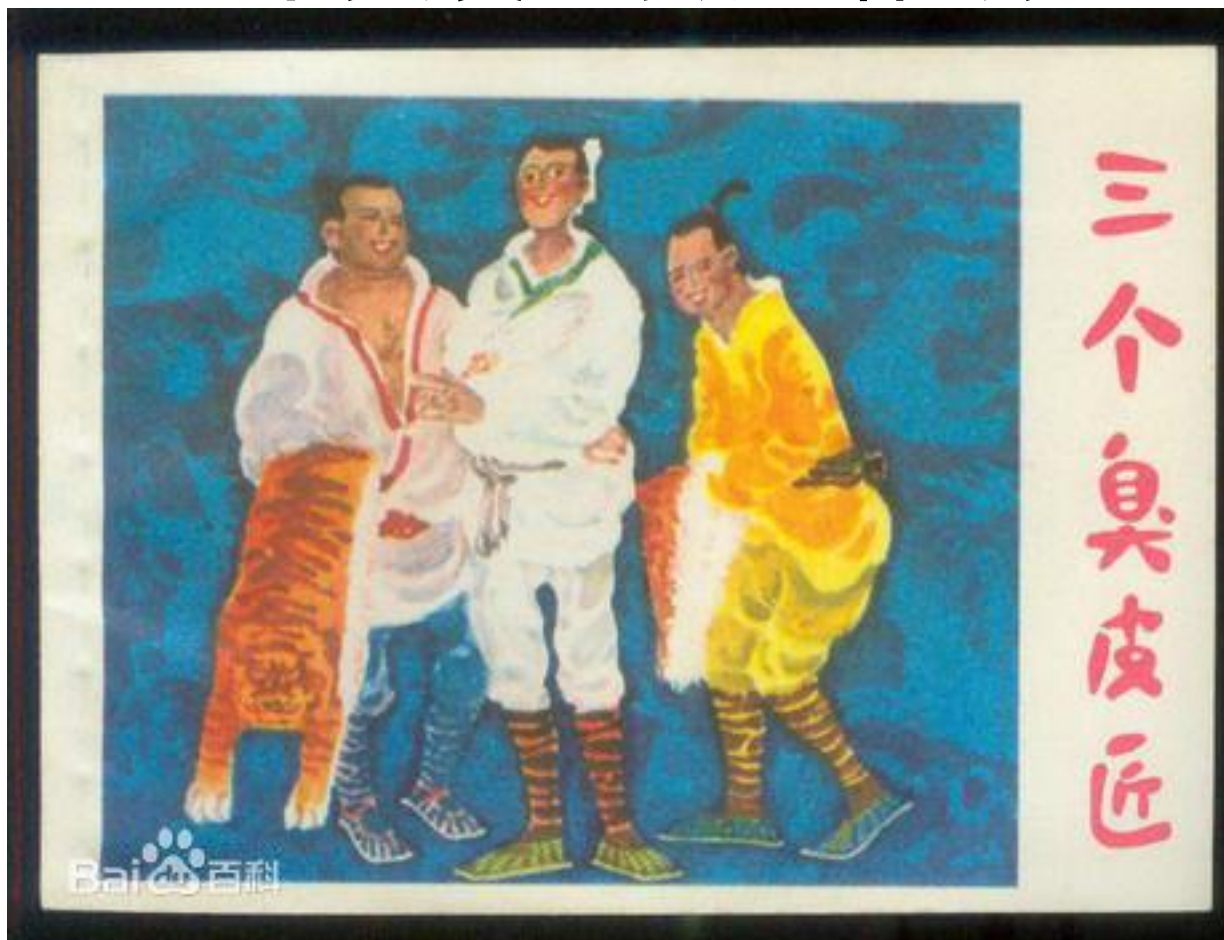
第五章 非线性方法

集成学习 (Ensemble learning)

如何获得一个更好的分类系统？

- 提取更好的特征
- 利用更强大的学习算法
- 更多的训练样本
- 样本在不同空间的映射
- 利用先验知识和上下文信息
- ...
- 分类器的组合

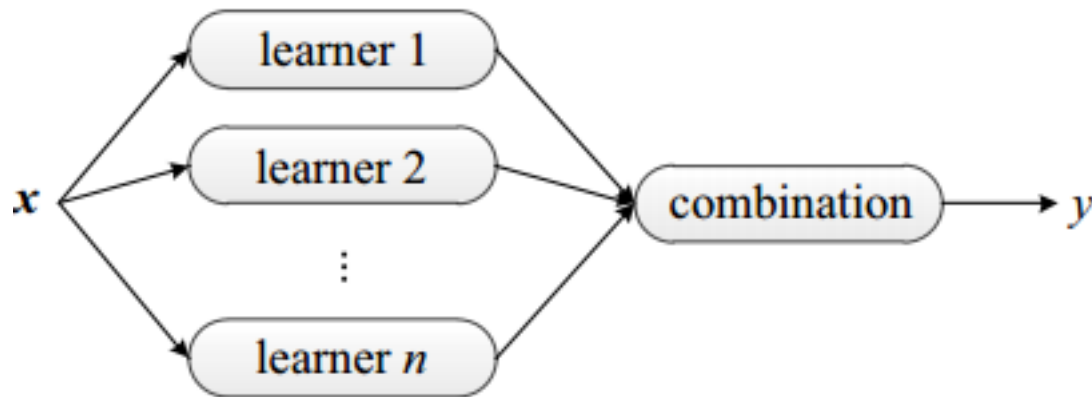
三个臭皮匠赛过诸葛亮



社会生活中常用的原则：民主选举

Ensemble learning

若干个“弱”学习器进行结合，常获得比单一学习器显著优越的性能



基学习器 (base learner)

投票: $g(\mathbf{x}) = \text{sign} \left[\sum_{k=1}^N h_k(\mathbf{x}) \right]$

神经网络、随机森林, ...

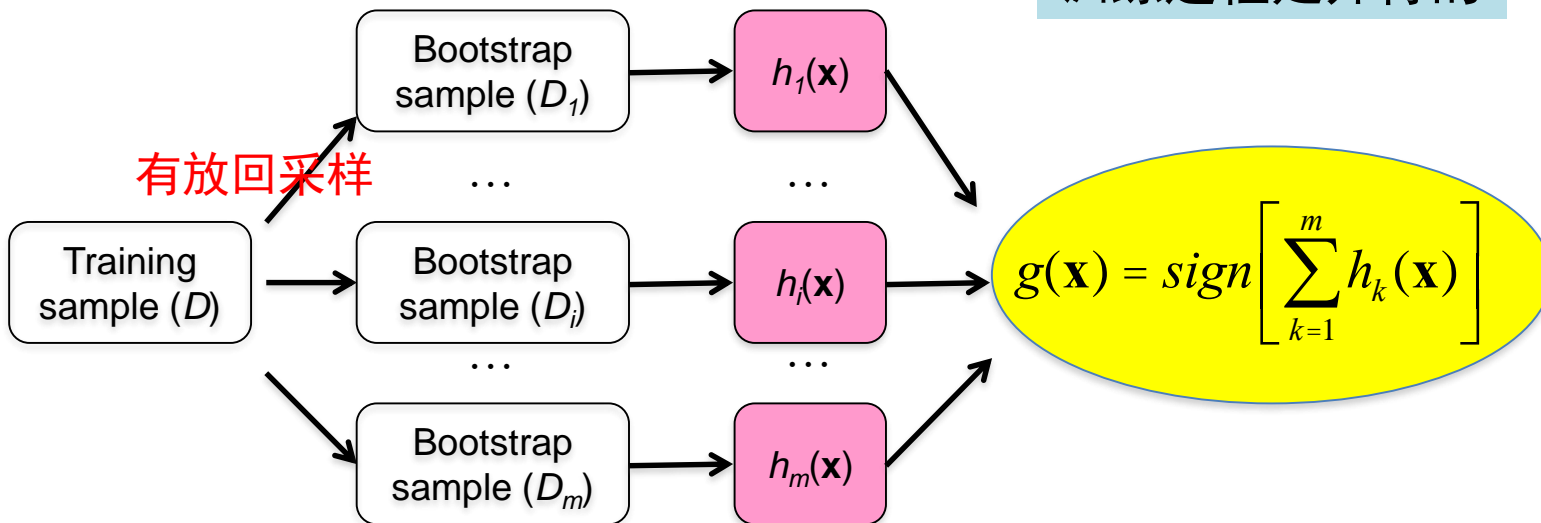
两种思路

- 个体学习器之间**存在**强依赖关系，**串行**生成学习器——**Boosting**
- 个体学习器之间**不存在**强依赖关系，可以**并行**生成学习器——**Bagging, Random Forest**
- 引入扰动：
 - 数据的扰动
 - 属性的扰动
 - 参数的扰动

Bagging (bootstrap aggregating)

- 对训练数据 D 有放回抽样生成 m 个数据集 D_i , ($i=1, \dots, m$)
- 在每一个数据集 D_i 上训练分类器 h_i
- 对各个分类器的结果集成投票

训练过程是并行的



防止过拟合, 减小模型方差

当 D_i 大小与 D 相同时, 大约63.2% $(1-(1-1/n)^n)=1-1/e$ 的样本被抽到, 有的样本出现两次或两次以上

- 随机森林 Random Forests

(Leo Breiman, *Machine Learning*, 45: 5-32, 2001)

(http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm)

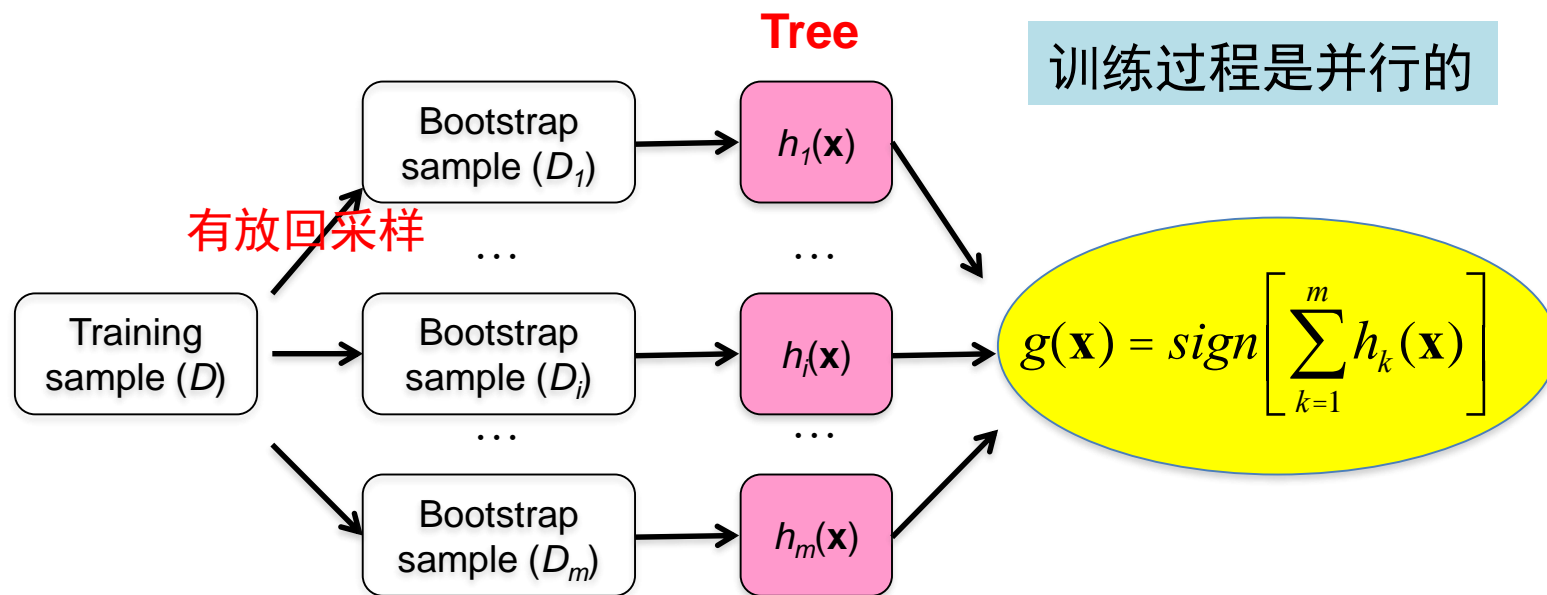
- Many decision trees
→ Random Forest



Leo Breiman (1928-2005)

随机森林 Random Forests

- 利用**bootstrapping**的办法，**训练多棵树**(每次从 N 个训练样本中有放回地随机抽取 N 个样本作为当前训练集)
- 每次**随机抽取 k ($k < p$) 个特征**作为当前节点下决策的备选特征，从中选择特征进行划分 (**split**)
- 最后对每一颗树的预测结果进行汇总投票，作为最终预测



Boosting

- 通过迭代对分类器的输入输出进行加权
 - 开始时所有训练样本同样权重
 - 每一轮学习一个“弱”分类器
 - 根据现有“弱”分类器的加权投票结果对训练样本分布进行调整，对分错的样本给予更多关注
 - 直到“弱”分类器数目达到预先制定的数目 k_{max} ，最终将 k_{max} 个分类器的结果加权组合

最终组合函数：

$$g(\mathbf{x}) = \text{sign} \left[\sum_{k=1}^{k_{\max}} a_k h_k(\mathbf{x}) \right]$$

AdaBoost算法

核心思路：对错分样本进行加权

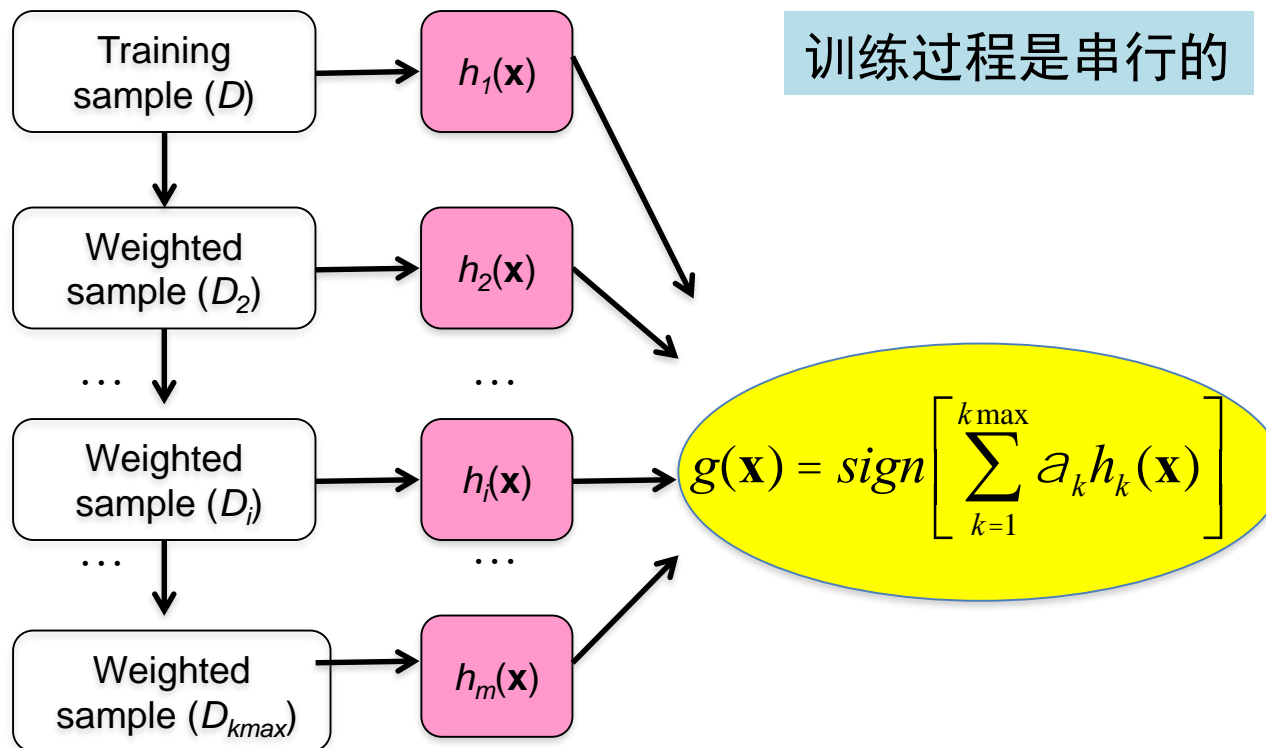
■ Algorithm 1. (AdaBoost)

```
1 begin initialize  $\mathcal{D} = \{\mathbf{x}^1, y_1, \dots, \mathbf{x}^n, y_n\}, k_{\max}, W_1(i) = 1/n, i = 1, \dots, n$   
2    $k \leftarrow 0$   
3   do  $k \leftarrow k + 1$   
4     train weak learner  $C_k$  using  $\mathcal{D}$  sampled according to  $W_k(i)$   
5      $E_k \leftarrow$  training error of  $C_k$  measured on  $\mathcal{D}$  using  $W_k(i)$   
6      $\alpha_k \leftarrow \frac{1}{2} \ln[(1 - E_k)/E_k]$   
7      $W_{k+1}(i) \leftarrow \frac{W_k(i)}{Z_k} \times \begin{cases} e^{-\alpha_k} & \text{if } h_k(\mathbf{x}^i) = y_i \text{ (correctly classified)} \\ e^{\alpha_k} & \text{if } h_k(\mathbf{x}^i) \neq y_i \text{ (incorrectly classified)} \end{cases}$   
8   until  $k = k_{\max}$   
9   return  $C_k$  and  $\alpha_k$  for  $k = 1$  to  $k_{\max}$  (ensemble of classifiers with weights)  
10 end
```

最终组合函数：

$$g(\mathbf{x}) = \text{sign} \left[\sum_{k=1}^{k_{\max}} a_k h_k(\mathbf{x}) \right]$$

AdaBoost: Adaptive Boosting



Ada Boost (Freund & Schapire, 1997)

Ada Boost for 2 Classes

Initialization step: for each example x , set

$$D(x) = \frac{1}{N}, \text{ where } N \text{ is the number of examples}$$

Iteration step (for $t = 1 \dots T$):

1. Find best weak classifier $h_t(x)$ using weights $D_t(x)$
2. Compute the error rate ϵ_t as

$$\epsilon_t = \sum_{i=1}^N D(x_i) \cdot \mathbf{I}[y_i \neq h_t(x_i)]$$

$$= \begin{cases} 1 & \text{if } y_i \neq h_t(x_i) \\ 0 & \text{otherwise} \end{cases}$$

3. assign weight α_t to classifier h_t in the final hypothesis

$$\alpha_t = \log((1 - \epsilon_t)/\epsilon_t)$$

4. For each x_i , $D(x_i) = D(x_i) \cdot \exp(\alpha_t \cdot \mathbf{I}[y_i \neq h_t(x_i)])$

5. Normalize $D(x_i)$ so that $\sum_{i=1}^N D(x_i) = 1$

$$f_{\text{final}}(x) = \text{sign} \left[\sum \alpha_t h_t(x) \right]$$

Ada Boost: step by step

1. Find best weak classifier $h_t(\mathbf{x})$ using weights $\mathbf{D}(\mathbf{x})$

- Some classifiers accept weighted samples, but most don't
- If the classifier does not take weighted samples, this step is done by sampling from the training samples according to the distribution $\mathbf{D}(\mathbf{x})$



- Draw k samples, each \mathbf{x} with probability equal to $\mathbf{D}(\mathbf{x})$:



Ada Boost: step by step

1. Find best weak classifier $h_t(\mathbf{x})$ using weights $\mathbf{D}(\mathbf{x})$

- Give to the classifier the following re-sampled examples:



- To find the best weak classifier, go through ALL weak classifiers, and find the one that works best (gives smallest error) on the collection above

	$h_1(\mathbf{x})$	$h_2(\mathbf{x})$	$h_3(\mathbf{x})$	$h_m(\mathbf{x})$
errors:	0.46	0.36	0.16		0.43
			the best classifier $h_t(\mathbf{x})$ at iteration t		

Ada Boost: step by step

2. Compute ϵ_t the error rate as

$$\epsilon_t = \sum D(x_i) \cdot I[y_i \neq h_t(x_i)]$$

- where $I[y_i \neq h_t(x_i)] = \begin{cases} 1 & \text{if } y_i \neq h_t(x_i) \\ 0 & \text{otherwise} \end{cases}$



$$\epsilon_t = \frac{1}{4} + \frac{1}{16} = \frac{5}{16}$$

- ϵ_t is simply the weight of all misclassified examples added
 - notice that error rate is computed over original examples, not the re-sampled examples
- If a weak classifier is better than random, then $\epsilon_t < 1/2$

Ada Boost: step by step

3. assign weight α_t to classifier h_t in the final hypothesis

$$\alpha_t = \log((1 - \epsilon_t)/\epsilon_t)$$

Example from previous slide:

$$\epsilon_t = \frac{5}{16} \Rightarrow \alpha_t = \log \frac{1 - \frac{5}{16}}{\frac{5}{16}} = \log \frac{11}{5} \approx 0.8$$








- Recall that $\epsilon_t < 1/2$
- Thus $(1 - \epsilon_t)/\epsilon_t > 1 \Rightarrow \alpha_t > 0$
- The smaller is ϵ_t , the larger is α_t , and thus the more importance (weight) classifier $h_t(x)$ gets in the final classifier

$$f_{final}(x) = \text{sign} \left[\sum \alpha_t h_t(x) \right]$$

Ada Boost: step by step

4. For each \mathbf{x}_i , $D(\mathbf{x}_i) = D(\mathbf{x}_i) \cdot \exp[\alpha_t \cdot I(y_i \neq h_t(\mathbf{x}_i))]$

Example from previous slide: $\alpha_t = 0.8$

						
1/16	1/4	1/16	1/16	1/4	1/16	1/4
✓	✓	✓	✗	✗	✓	✓
⇓	⇓	⇓	⇓	⇓	⇓	⇓
1/16	1/4	1/16	$(1/16) \exp(0.8)$	$(1/4) \exp(0.8)$	1/16	1/4

- Weight of misclassified examples is increased and the new $D(\mathbf{x}_i)$'s are normalized to be a distribution again

Ada Boost: step by step

5. Normalize $D(x_i)$ so that $\sum D(x_i) = 1$

Example from previous slide:

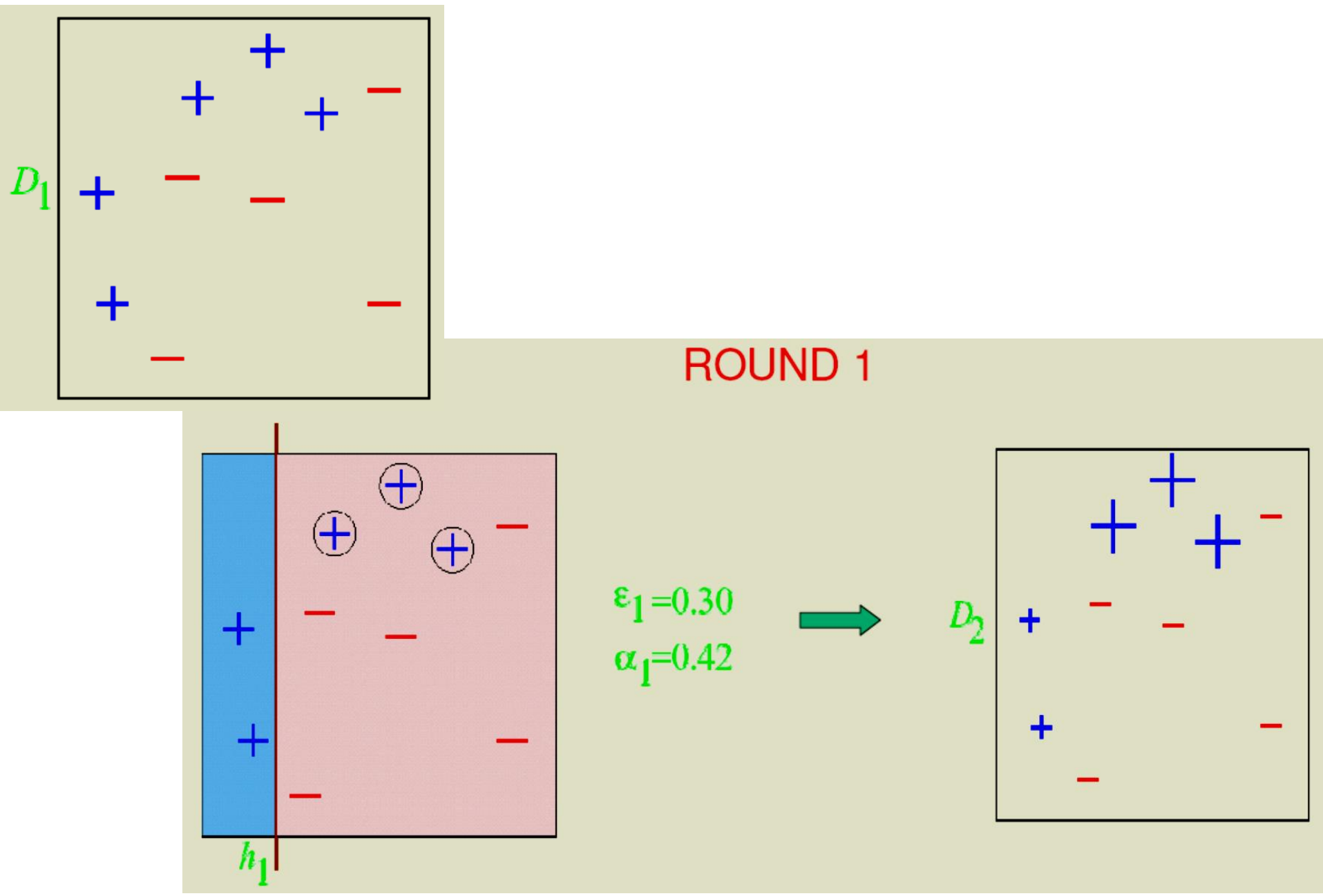


After normalization:

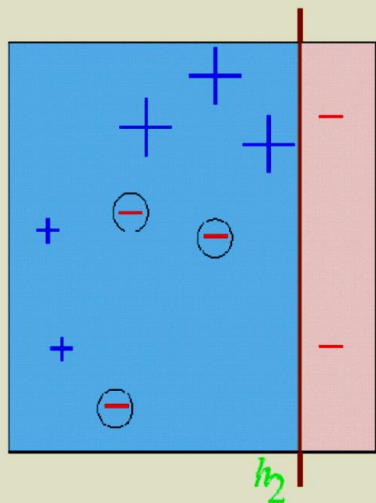


- In matlab, if D is a vector storing weights, $D = D./\text{sum}(D)$

AdaBoost Example (Freund & Schapire)



ROUND 2

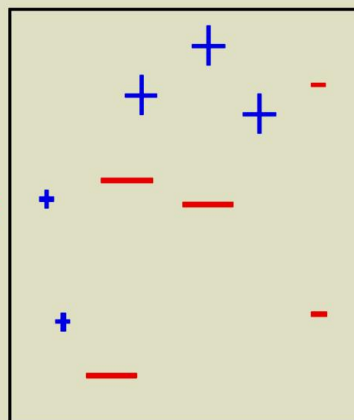


$$\epsilon_2 = 0.21$$

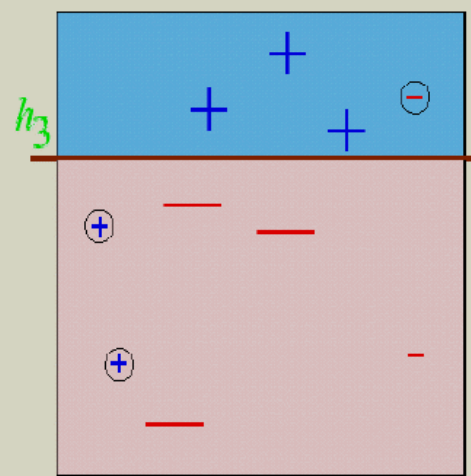
$$\alpha_2 = 0.65$$



D_3



ROUND 3

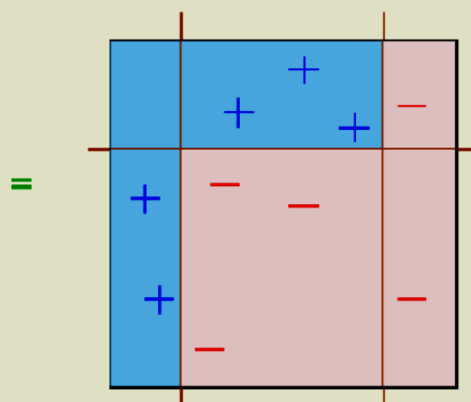


$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

$$f_{\text{FINAL}}(\mathbf{x}) = \text{sign} \left(0.42 \left(\text{Diagram 1} \right) + 0.65 \left(\text{Diagram 2} \right) + 0.92 \left(\text{Diagram 3} \right) \right)$$

Diagram illustrating the final function $f_{\text{FINAL}}(\mathbf{x})$ as a weighted sum of three functions. The first function is $0.42 \times \text{sign}(x_1 - 3)$, the second is $0.65 \times \text{sign}(x_1 - 7)$, and the third is $0.92 \times \text{sign}(x_2 - 4)$.



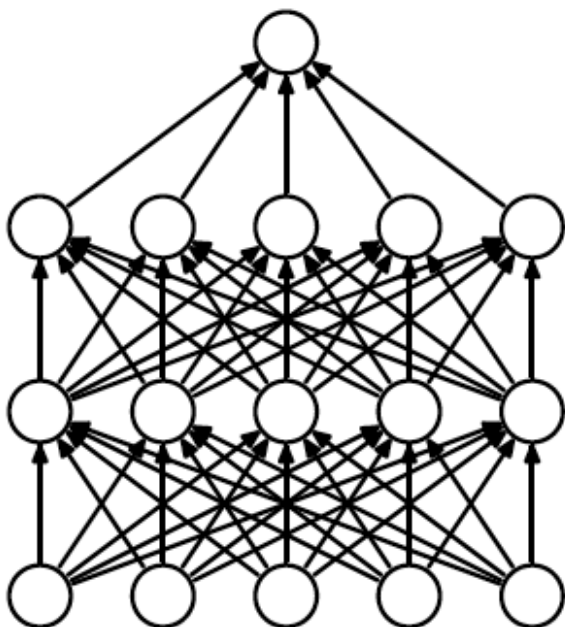
$$f_{\text{final}}(\mathbf{x}) = \text{sgn} [0.42 \text{sgn}(x_1 - 3) + 0.65 \text{sgn}(x_1 - 7) + 0.92 \text{sgn}(x_2 - 4)]$$

Boosting方法的特点

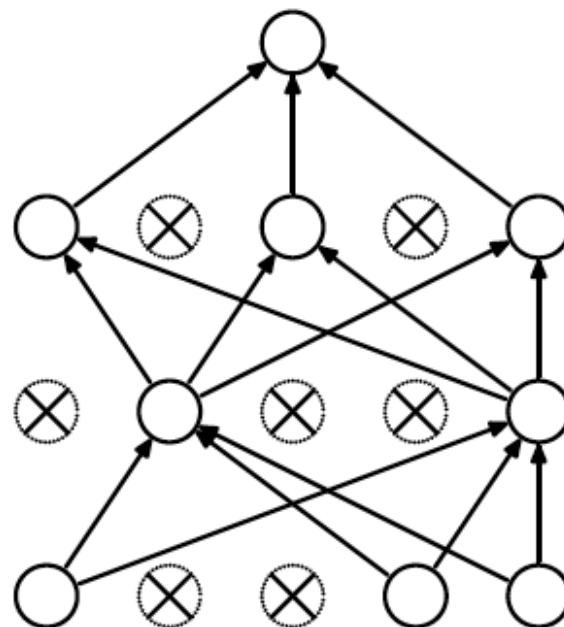
- 优点：
 - 快速
 - 简单
 - 只有一个参数 k_{\max}
 - 灵活：可以与任何分类器使用，只需要比随机好的“弱”分类器
- 缺点：
 - “弱”分类器不能太复杂（容易过拟合）
 - 对噪声敏感，例如会给标记错误的样本过大的权重

神经网络防止过拟合：Dropout

每次训练时随机把部分节点参数置零
相当于利用很多不同结构网络的结果的集合



(a) Standard Neural Net



(b) After applying dropout.

Dropout: A Simple Way to Prevent Neural Networks from Overfitting
Journal of Machine Learning Research 15 (2014) 1929-1958

小结

- 将‘弱’分类器组合可以得到推广性更好的分类器
- 串行与并行（Boosting vs. Bagging, RF）
- 引入扰动（样本，特征，...）
- 通常对不稳定的分类器（比如决策树）效果好，但不局限于树