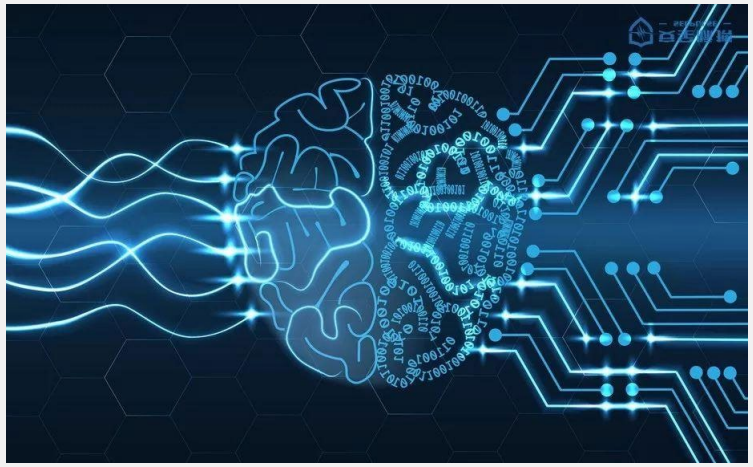


# 机器学习

Machine Learning


授课老师：谭毅华
电　　话：13886021197
办 公 室：科技楼1102
邮　　箱：yhtan@hust.edu.cn

# 第四章：决策树与集成学习

目录
CONTENTS

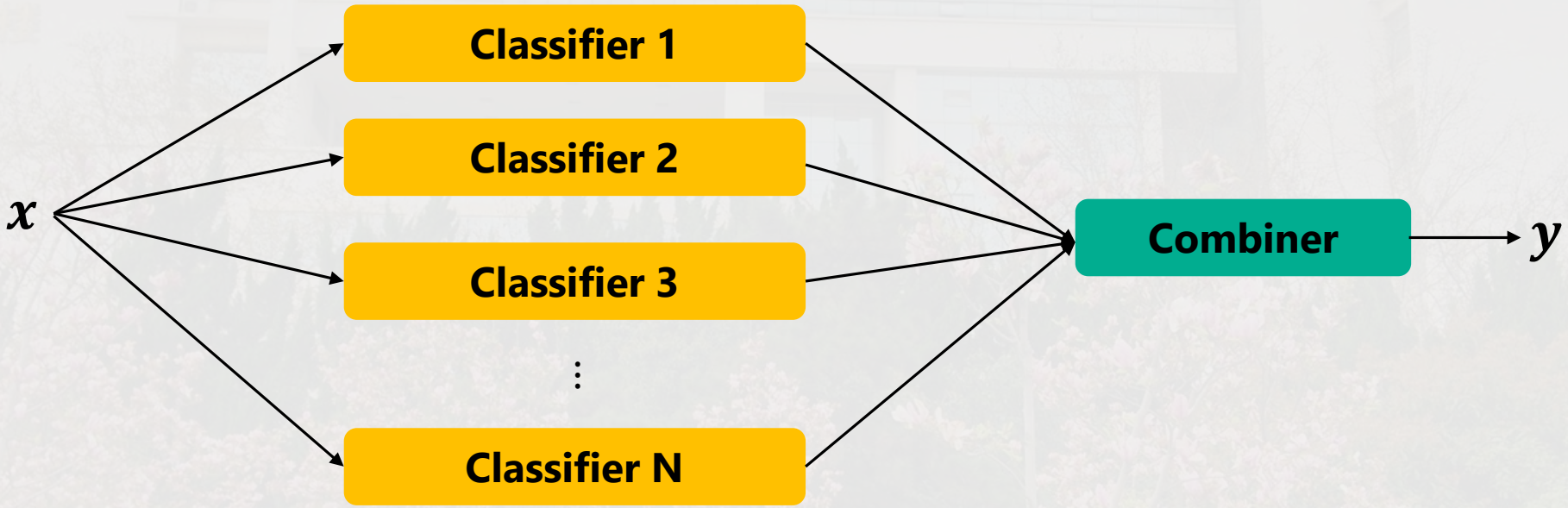
# 生活中的集成学习



# 1、集成学习是什么

集成学习的方法是通过训练多个学习器，并通过学习器的合并来解决一个问题。

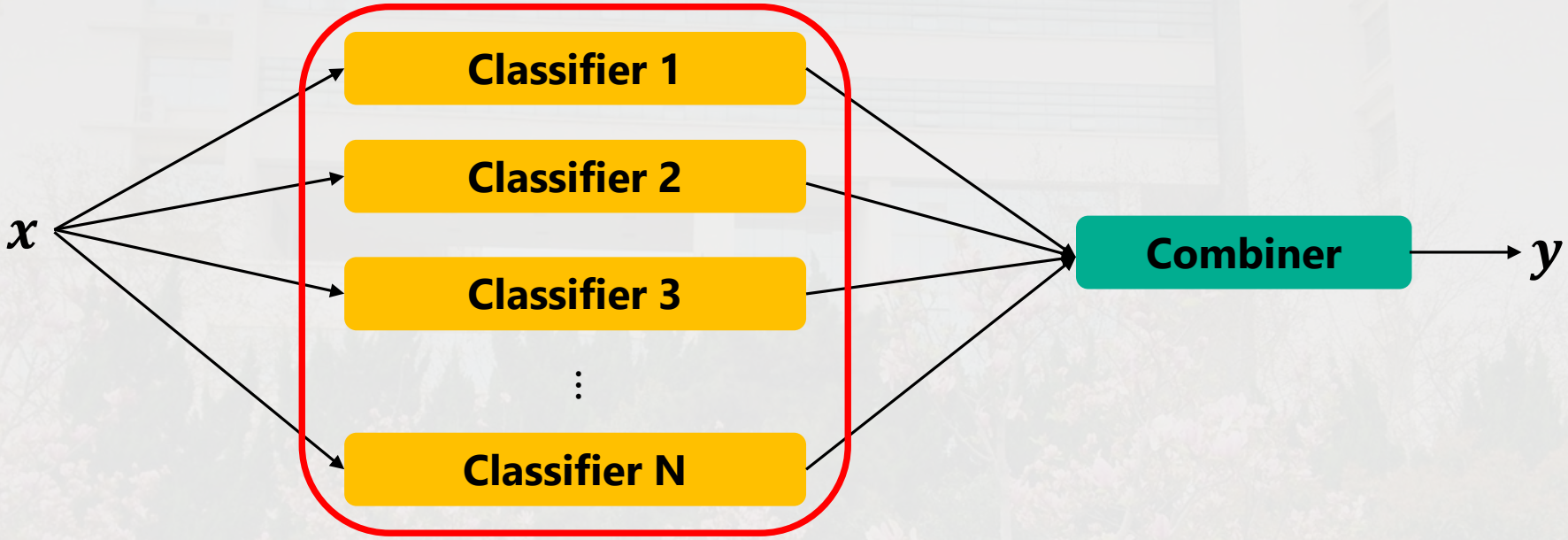
集成学习也被成为基于委员会的学习(committee-based learning)或多分类器系统（multiple classifier system）。



# 集成学习的组成

**集成学习**框架:

- 由多个基学习器(base learner/classifier)构成，而基学习器由基学习算法（base learning algorithm）在训练数据上训练获得，可以是前面我们学到的决策树、简单的分类算法如SVM、或者是复杂的神经网络等各种学习方法。



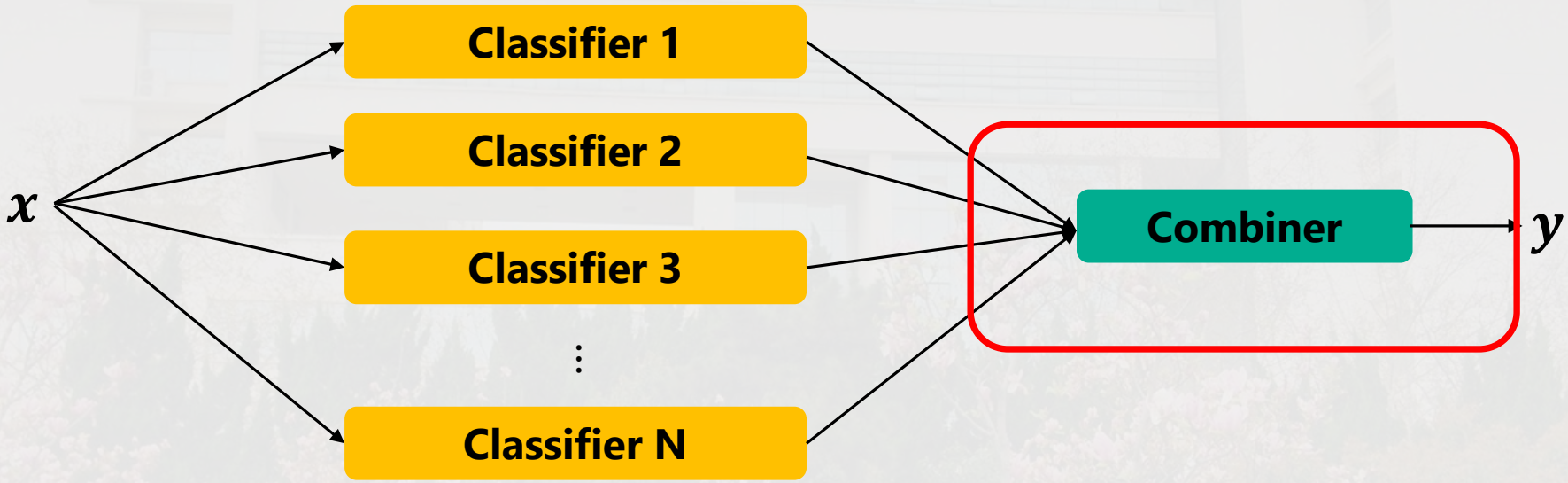

**同质集成 （homogeneous ensemble）**:使用一种基学习算法产生同质的基学习器，即相同种类的学习器。

**异质集成（heterogeneous ensemble）**:多种学习算法训练不同种类的学习器。

# 集成学习的组成

**集成学习**框架:

➢ 集成学习的决策由最终的集成器（Combiner）采用一定的集成策略生成，它使得整个集成算法具有更强的泛化能力




**集成学习受欢迎的原因：**把比随机猜测稍好的弱学习器（weak learner）变成可以精确预测的强学习器（strong learner）。

# 基学习器的训练需要注意的问题

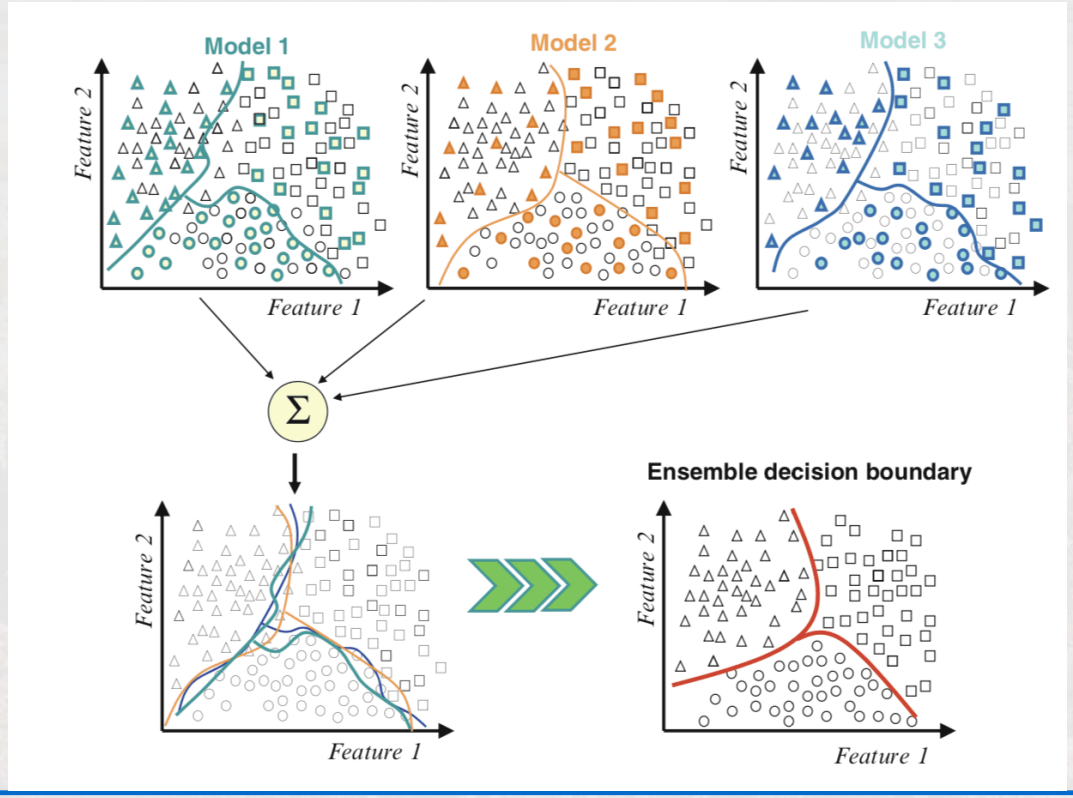
## 集成学习需要生成既相关又不相同的基学习器

不同的参数

不同的训练样本

不同的特征

不同的初始值



# 结合策略

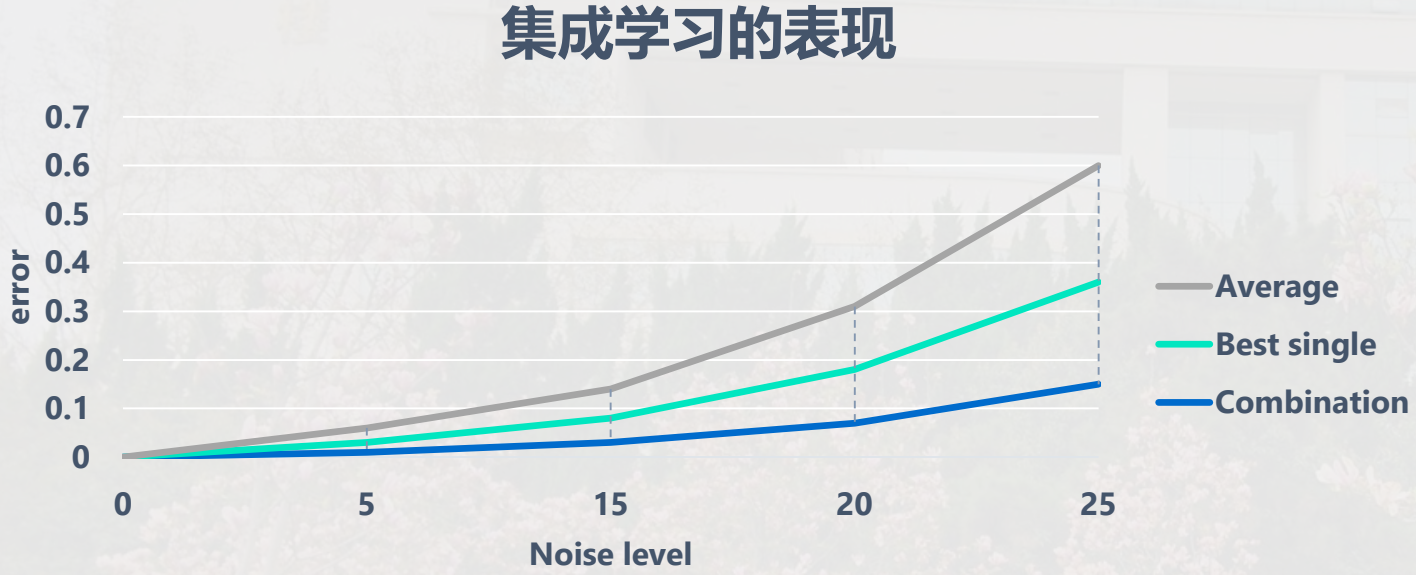
**对基学习器进行结合时，有很多不同的策略：**

- 采用加权平均可以得到比简单平均更好的泛化能力。但是，也有一些研究者认为，对权值进行优化将会导致过配(overfiting)，从而使得集成的泛化能力降低，因此，他们建议使用简单平均。
- 在分类问题上，集成的输出往往由个体学习器投票决定，投票法是一种很早开始就获得广泛使用的方法。(Random forest, Adaboost)

# 1、集成学习是什么

## ➢测验：

一组基学习器的集成结果(Combination)与他们中的最优学习器结果(Best)和平均结果比(Average)，哪个更好(error最小)?




**参考文献：**

实验证明：

*Hansen, L. K., & Salamon, P. (1990). Neural network ensembles.*

理论证明：

Schapire, Robert E. "The strength of weak learnability."

# 2、集成学习的发展史

**1990年**

Hanson和Salamon对神经网络集成进行了研究，第一次提出了神经网络集成的概念。

**1996年**

Freund和Schapire提出了著名的**AdaBoost**(Adaptive Boost)算法,该算法已成为目前最流行的Boosting算法。

**1996年**

Breiman从可重复取样技术(Bootstrap Sampling)入手，提出了著名的**Bagging**方法。

**1996-至今**

XGBoost, GBDT等模型的开发和流行

*Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10), 993-1001*
*Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780), 1612*
*Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.*

# 集成学习的分类

根据基分类器的生成方式，集成学习可以分为两种：串行生成基分类器的“**串行集成方法**”，以及并行生成分类器的“**并行集成方法**”。

**Boosting (Adaboost, GBDT, XGBoost)**

**串行集成方法**

**Bagging (Random Forest)**

**并行集成方法**

# 并行集成方法

# 并行集成算法

**并行集成方法：**其内在原理是利用基学习器之间的独立性，这是因为结合相互独立的基分类器能够显著减小方差（Variance）。

以一个二分类任务为例，输出分别为+1和-1，假设真实的函数为$f(x)$，训练T个基分类器$h_i(t)$，$i = 1, \dots, T$。对于每一个基分类器，假设弱分类时的分类错误的概率$\epsilon < \frac{1}{2}$，存在：

$$P\left(h_i(x) \neq f(x)\right) = \epsilon$$

结合T个分类器，根据Voting的方法来决定最后的分类结果$H(x)$，用符号函数表示，可得

$$H(x) = sign(\sum_{i=1}^{T} h_i(x))$$

集成后的H之后在至少一半的基分类器预测错误时才会犯错。

# 并行集成算法

因此，根据霍夫丁不等式(Hoeffding)，集成后模型的泛化误差为

$$P\big(H(x) \neq f(x)\big) = \sum_{k=0}^{[T/2]} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \leq \exp\left(-\frac{1}{2} T(2\epsilon - 1)^2\right)$$

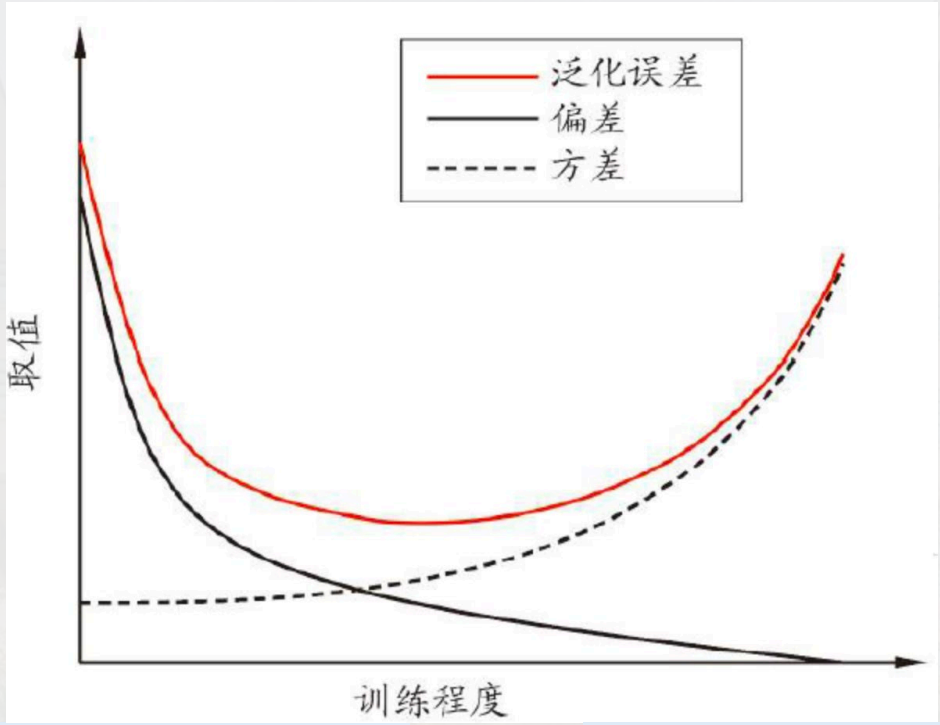
集成器的规模越大，也就是T越大，集成模型的犯错的概率会显著性的呈指数下降，并且随着 $T$ 趋向无穷大而趋于0 。

Note: 由于基分类器由相同的训练数据产生，不是完全独立同分布的，因此很难得到完全独立的基分类器，但可以通过**随机的方法减少他们之间的相关性，并通过集成实现更好的泛化能力。**

**Bagging 方法**

# 复习：偏差-方差平衡

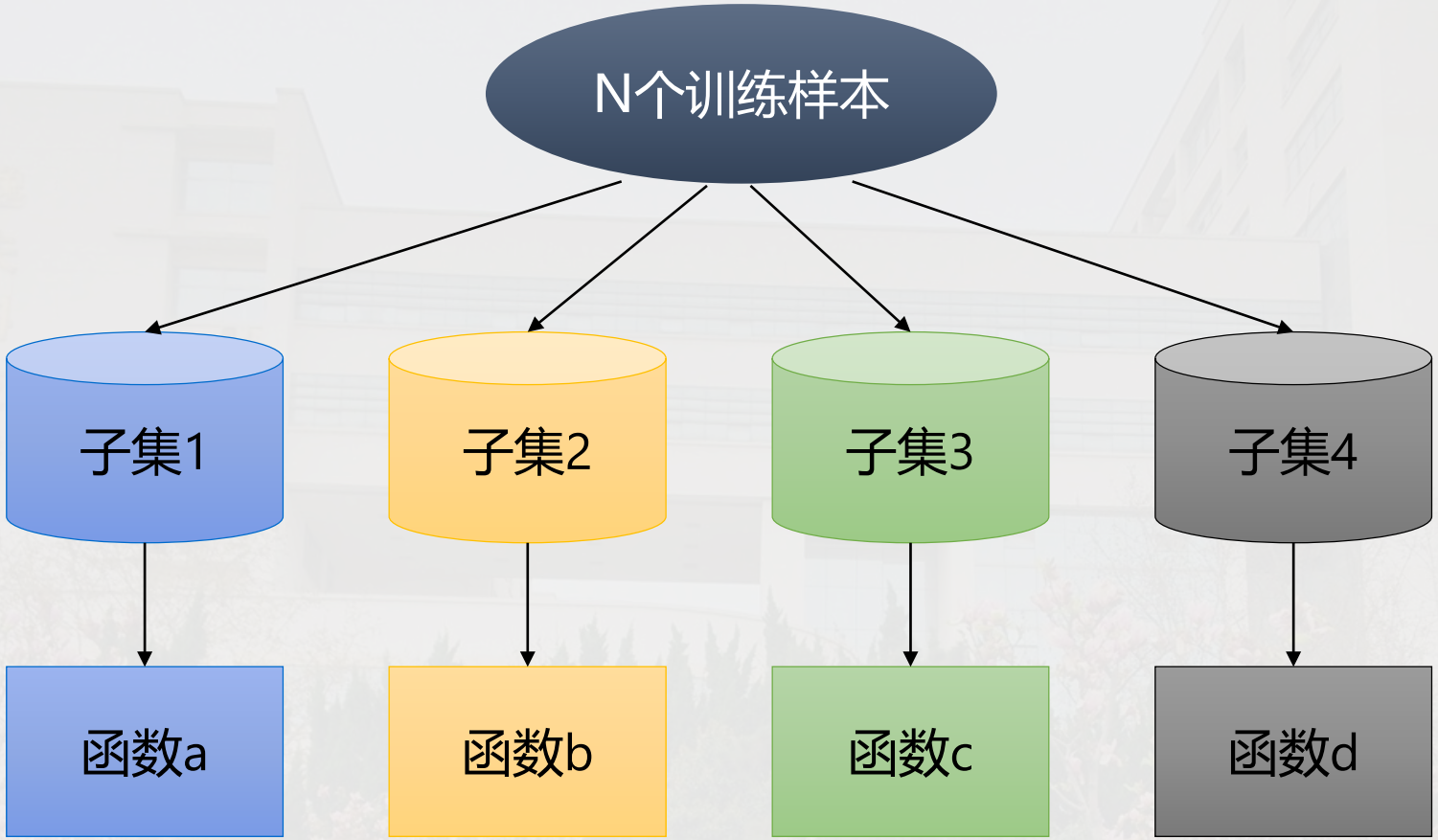
- 偏差与方差是有冲突的，称为“偏差-方差窘境”



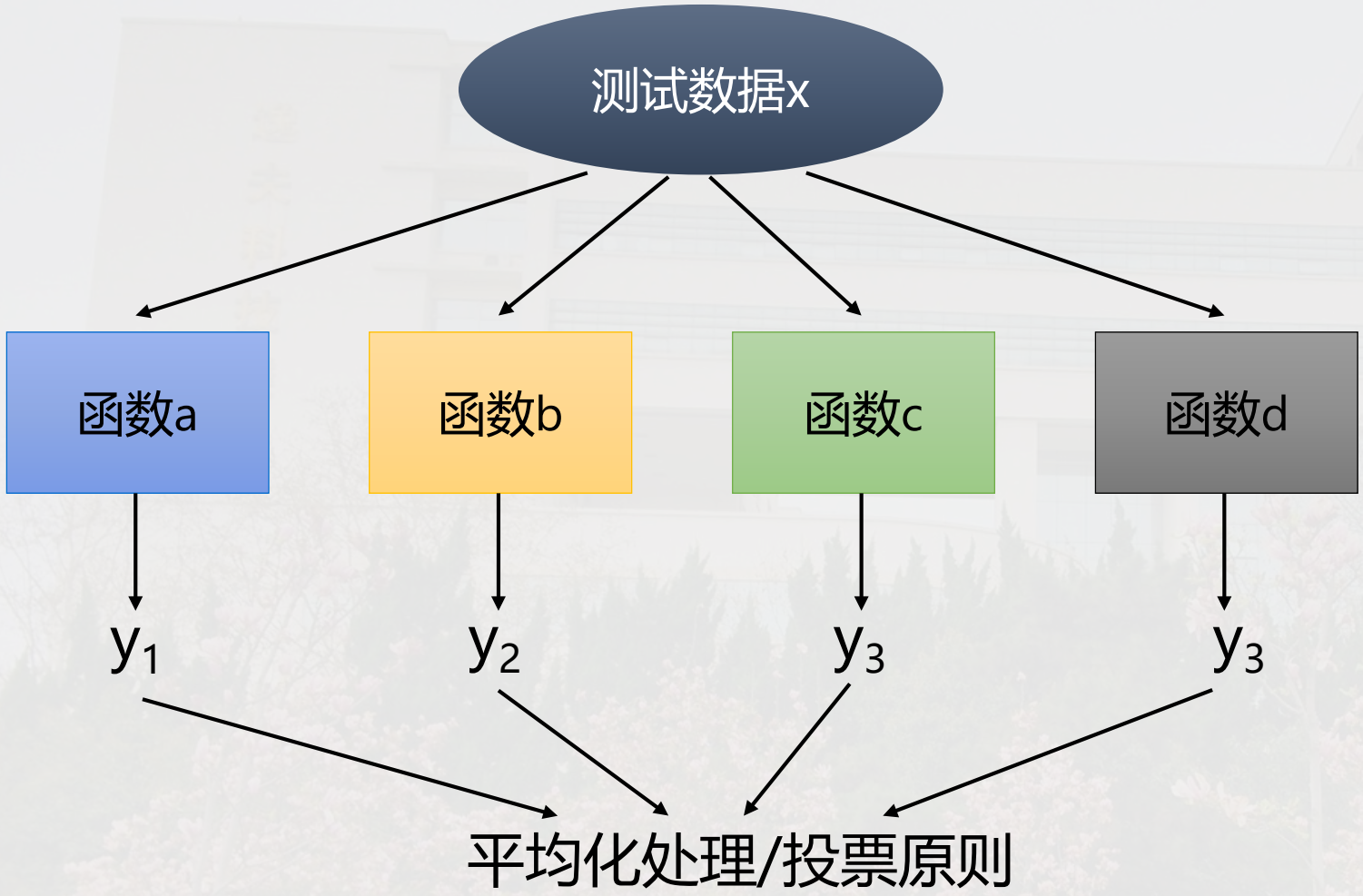

- ❖ 欠拟合，大偏差，小方差
- ❖ 过拟合，小偏差，大方差

- 复杂的模型将具有较大的方差，可平均化处理复杂模型以减少方差。
  - **Bagging可以明显降低方差，在不稳定学习器上格外有效**

# Bagging



# Bagging




- 当模型非常复杂，容易过拟合时，bagging方法很有帮助。如决策树

# Bagging

- Bagging这一名称来源于Bootstrap AGGregatING[Breiman, 1996]的缩写,是一种自助聚合的方法

Bagging采用**自助采样**生成不同的基分类器。

*生成不同的训练样本的同时又可以服从类似的分布*

# 包外样本（Out-of-bag sample）

- Bootstrap赋予了Bagging一个额外的优势。给定m个训练样本，第$i$个样本每次被选中的概率为$\frac{1}{m}$，选了m次，当m趋近于无穷的时候，没有被选中的概率为：

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^{m}=\frac{1}{e}$$

对于任意一个基分类器，训练时的原始训练数据中由大约36.8%的样本未被使用

- 未被选中的样本（即包外样本）可用于验证基分类器的性能。

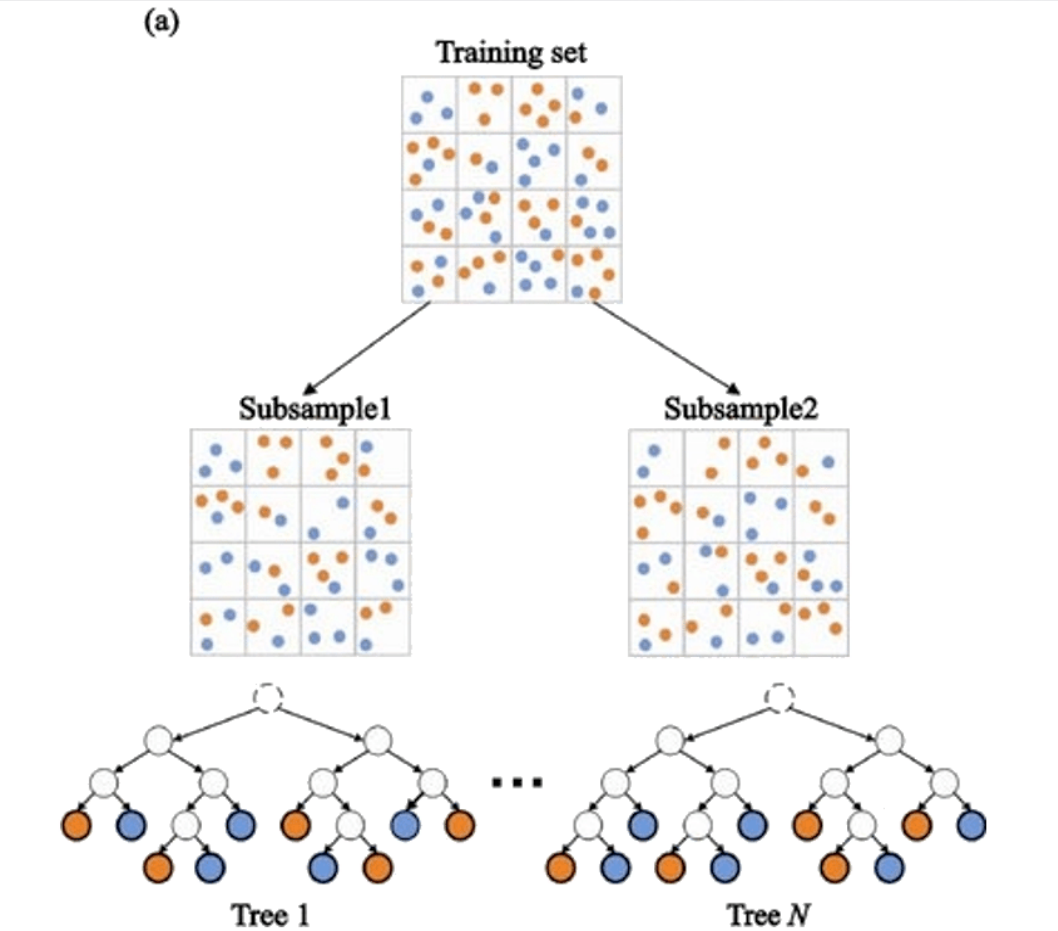
# 随机森林

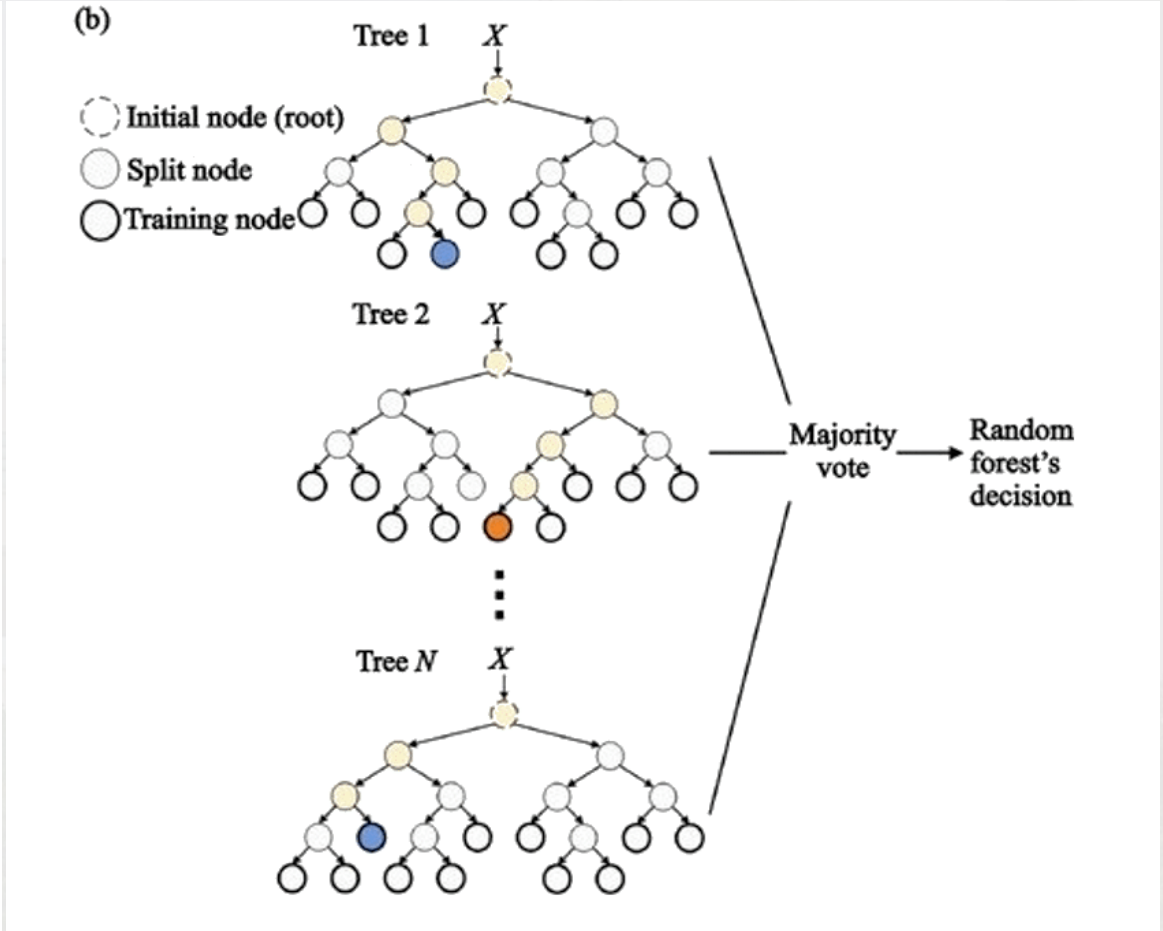
- 随机森林(Random Forest，简称RF) 是Bagging 的一个扩展变体。RF 是在以决策树为基学习器构建Bagging 集成的基础上，进一步在决策树的训练过程中引入了**随机属性选择**.

  - ✓ 传统决策树在选择划分属性时是在当前结点的属性集合(假定有d个属性) 中选择一个最优属性；

  - ✓ 而在RF 中，对基决策树的每个结点，先从该结点的属性集合中**随机选择一个包含*K* 个属性的子集**，然后再从这个子集中选择一个最优属性用于划分。

# 随机森林




- 对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含**K 个属性的子集**，然后再从这个子集中选择一个最优属性用于划分

# • 随机森林算法实现

**Input:** Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size $K$.

**Process:**

1. $N \leftarrow$ create a tree node based on $D$;
2. **if** *all instances in the same class* **then return** $N$
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** $\mathcal{F}$ *is empty* **then return** $N$
5. $\tilde{\mathcal{F}} \leftarrow$ select $K$ features from $\mathcal{F}$ randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of $D$ with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of $D$ with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters $(D_l, K)$;
11. $N_r \leftarrow$ call the process with parameters $(D_r, K)$;
12. **return** $N$

**Output:** A random decision tree

# 随机森林与Bagging的预测效果比较

- A comparison of Random Forest and Bagging for different numbers of trees (2020) 本文比较了在不同数据集上Random forest 和Bagging的区别

More specifically, the 8 data sets we consider in this study are of a small size (inferior to 1000 observations). Secondly, they have less than 10 explanatory variables. Finally, the explained variable in all these data sets takes only two values (ex. 0 and 1).

We conclude that despite the number of trees used to develop our algorithms, there are no significant differences in terms of Accuracy between the two algorithms. In the following section we present and explain these two algorithms. In section 3 we talk about the train/test split and cross-validation. In section 4 we display our methodology and the related research. In section 5 we present our results. In the last section we state the conclusions of our research.

1. Haberman Dataset (https://www.openml.org/d/43)
2. visualizing_environmental dataset (https://www.openml.org/d/678)
3. fri_c3_100_5 dataset (https://www.openml.org/d/611)
4. hayes-roth(https://www.openml.org/d/329)
5. blood-transfusion dataset (https://www.openml.org/d/1464)
6. Diabetes dataset(https://www.openml.org/d/37)
7. Immunotherapy dataset(https://archive.ics.uci.edu/ml/datasets/Immunotherapy+Dataset)
8. Cryotherapy Dataset (https://archive.ics.uci.edu/ml/datasets/Cryotherapy+Dataset+)

✓ 准确率区别不大
✓ RF训练效率更高

# 随机森林与Bagging的预测效果比较

## 五次10-fold cross validation, 对预测结果取均值

Table 1: Habberman ([6]) dataset —> 3 explanator

| Algorithm/Trees | 2 | 5 | 10 | 20 | 100 |
|---|---|---|---|---|---|
| Random Forest | 72.7 | 73.33 | 74.2 | 73.6 | 74.0 |
| Bagging | 73.52 | 72.43 | 73.4 | 73.76 | 74.1 |
| Significant | No | No | No | No | No |

Table 2: visualizing_environmental ([6]) dataset —> 3 exp

| Algorithm/Trees | 2 | 5 | 10 | 20 | 10( |
|---|---|---|---|---|---|
| Random Forest | 64.89 | 66.83 | 64.52 | 65.92 | 66.4 |
| Bagging | 64.58 | 64.82 | 67.48 | 66.68 | 66.8 |
| Significant | No | No | No | No | Nc |

Table 3: fri_c3_100_5 ([6]) dataset —> 5 explanator

| Algorithm/Trees | 2 | 5 | 10 | 20 | 100 |
|---|---|---|---|---|---|
| Random Forest | 71.2 | 72.6 | 76 | 76.2 | 77.2 |
| Bagging | 70.2 | 74.4 | 75.2 | 76.8 | 77.6 |
| Significant | No | No | No | No | No |

Table 6: Diabetes ([6]) dataset —> 8 explanatory variables and 768 instances

| Algorithm/Trees | 2 | 5 | 10 | 20 | 100 | 200 | 500 | 1000 | 4000 |
|---|---|---|---|---|---|---|---|---|---|
| Random Forest | 72.89 | 74.86 | 75.61 | 75.59 | 75.8 | 75.62 | 75.67 | 75.75 | 75.3 |
| Bagging | 75.2 | 75.9 | 75.72 | 75.82 | 75.64 | 76.2 | 75.62 | 75.8 | 75.75 |
| Significant | No | No | No | No | No | No | No | No | No |

Table 7: Immunotherapy ([15,18,19]) dataset —> 7 explanatory variables and 90 instances

| Algorithm/Trees | 2 | 5 | 10 | 20 | 100 | 200 | 500 | 1000 | 4000 |
|---|---|---|---|---|---|---|---|---|---|
| Random Forest | 80.89 | 81.56 | 83.33 | 84 | 83.56 | 84 | 83.78 | 84 | 83.56 |
| Bagging | 82.89 | 82.67 | 84.67 | 84.89 | 84.22 | 83.78 | 83.33 | 82.89 | 82.89 |
| Significant | No | No | No | No | No | No | No | No | No |

Table 8: Cryotherapy ([15,18,19]) dataset: —> 6 explanatory variables and 90 instances

| Algorithm/Trees | 2 | 5 | 10 | 20 | 100 | 200 | 500 | 1000 | 4000 |
|---|---|---|---|---|---|---|---|---|---|
| Random Forest | 82.88 | 85.78 | 86.67 | 86 | 89.56 | 88.22 | 90 | 90.89 | 91.33 |
| Bagging | 83.33 | 84.44 | 84.89 | 85.11 | 85.11 | 85.11 | 85.33 | 84.22 | 83.78 |
| Significant | No | No | No | No | No | No | No | No | No |

# 随机森林

➢ 测验：

下列哪一项不是随机森林的特点：

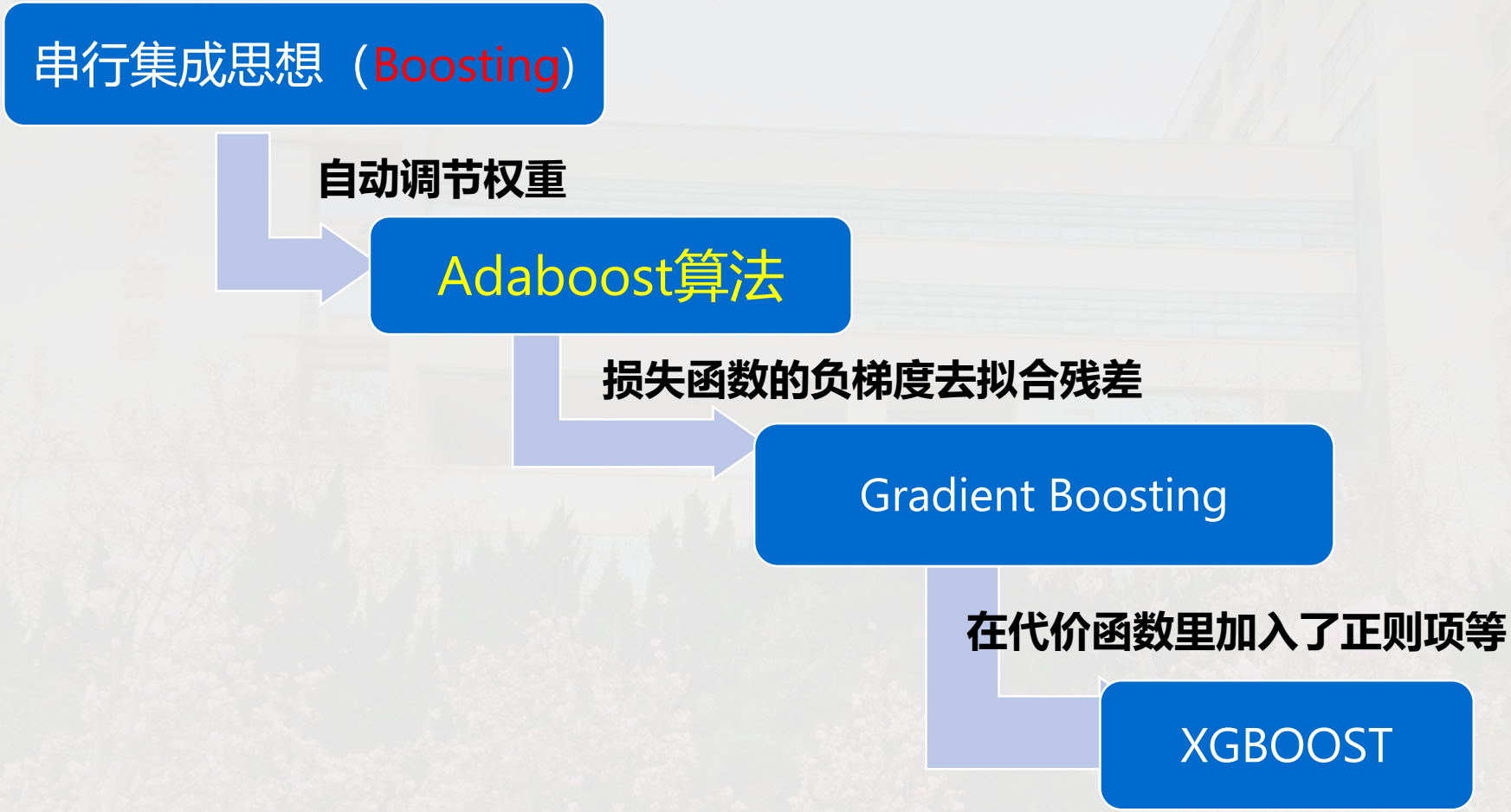
A. 它是基于决策树的集成学习方法

B. 引入了随机特征选择

✓ C. 它是一种串行集成方法

D. 样本选择时采用Bootstrap的采样方式

# 串行集成方法


串行集成思想（Boosting）
自动调节权重
Adaboost算法
损失函数的负梯度去拟合残差
Gradient Boosting
在代价函数里加入了正则项等
XGBOOST

# Boosting

Training data:
$\{(x^1, \hat{y}^1), \cdots, (x^n, \hat{y}^n), \cdots, (x^N, \hat{y}^N)\}$
$\hat{y} = \pm 1$ (binary classification)

- Boosting是一类将弱学习器提升为强学习器的算法
  - ✓ 弱学习算法：在训练数据上生成错误率略小于50%的分类器。
  - ✓ Boosting后，可获得错误率为0%的分类器。

- Boosting的工作框架：
  - 获得第一个分类器$f_1(x)$
  - 寻找另一个函数$f_2(x)$来**帮助**$f_1(x)$
    - 然而，如果$f_2(x)$与$f_1(x)$相似，则用处不大
    - 我们希望$f_2(x)$与$f_1(x)$**互补**（怎么做？）
  - 获得第二个分类器$f_2(x)$ ……
  - 最后，组合所有分类器
- 分类器是按顺序学习的

# Boosting

- 怎么获得不同的分类器?
  - ✓ 使用不同的训练数据集来进行训练
- 怎么获得**不同的训练集**?
  - ✓ 重新采样训练数据以形成新的集合
  - ✓ **重新赋权**训练数据，改变样本分布，形成新的训练数据集
  - ✓ 在实际操作中，只更改代价/目标函数

$(x^1, \hat{y}^1, u^1)$ $u^1 = 1$ 0.4

$(x^2, \hat{y}^2, u^2)$ $u^2 = 1$ 2.1

$(x^3, \hat{y}^3, u^3)$ $u^3 = 1$ 0.7

$$L(f) = \sum_n l\left(f(x^n), \hat{y}^n\right)$$

$$L(f) = \sum_n u^n \cdot l(f(x^n), \hat{y}^n)$$

# Adaboost

- 核心思想：在使得f1训练失败的训练集上训练f2
- 如何找到使f1训练失败的训练集？

$\varepsilon 1$：f1在训练集上的错误率

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1}$$

$$Z_1 = \sum_n u_1^n$$

$$\varepsilon_1 < 0.5$$

更新$u_1^n$到$u_2^n$样本的权重，使

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

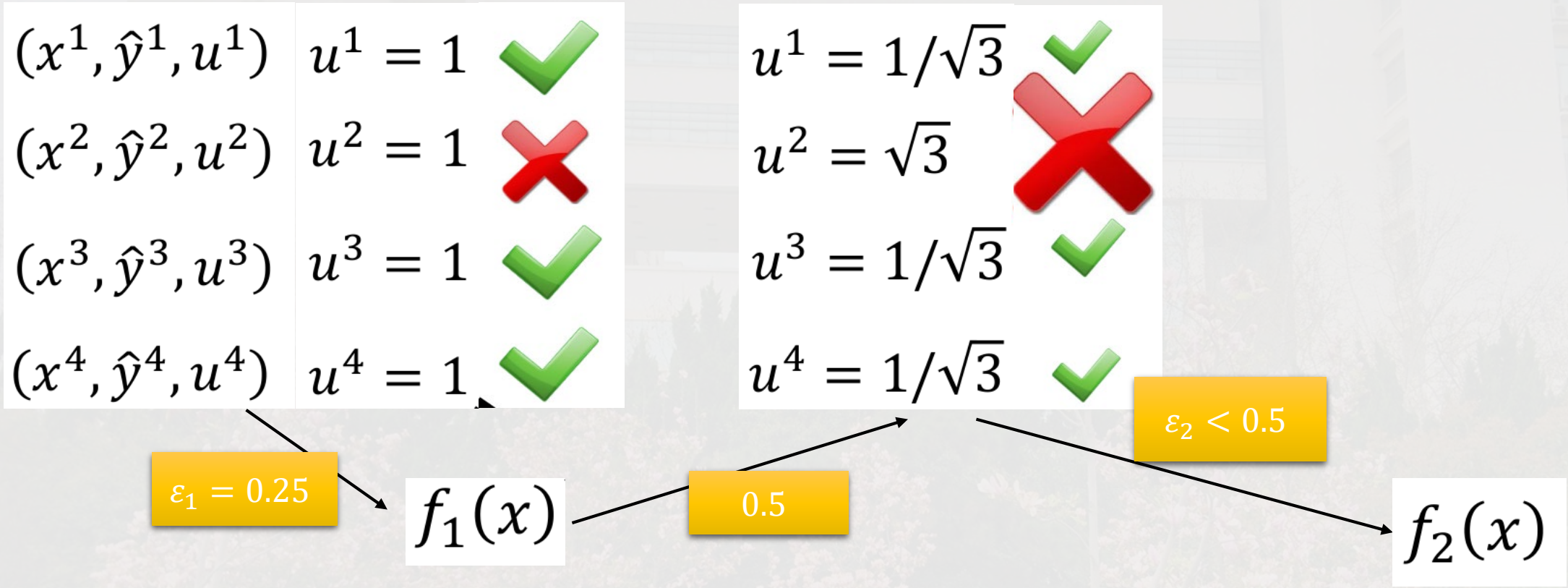
对于新的权重，f1的性能是随机的

基于新权重$u_2^n$，训练f2

# Adaboost

想法：在f1训练失败的训练集上训练f2

如何找到f1训练失败的训练集？



# 新权重下的训练数据

想法：在f1训练失败的训练集上训练f2

如何找到f1训练失败的训练集?

如果$x^n$被f1错误分类： $u_2^n = u_1^n * d_1$（增强）

如果$x^n$被f1正确分类： $u_2^n = u_1^n / d_1$（减小）

基于样本权重$u_2^n$，训练f2分类器

$d_1$的值是什么?

# 新权重下的训练数据

$d_1$的值是什么?

$$Z_1 = \sum_n u_1^n$$

$$Z_2 = \sum_n u_2^n$$

若$f_1(x^n) \neq \hat{y}_n$，则 $u_2^n = u_1^n d_1$

若$f_1(x^n) = \hat{y}_n$，则 $u_2^n = u_1^n / d_1$

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1}$$

$$\frac{\sum_n u_2^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_2} = 0.5$$

$$= \sum_{f_1(x^n) \neq \hat{y}_n} u_1^n d_1$$

$$\frac{\sum_{f_1(x^n) \neq \hat{y}_n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}_n} u_1^n / d_1}{\sum_{f_1(x^n) \neq \hat{y}_n} u_1^n d_1} = 2$$

$$Z_2 = \sum_{f_1(x^n) \neq \hat{y}_n} u_2^n + \sum_{f_1(x^n) = \hat{y}_n} u_2^n = \sum_{f_1(x^n) \neq \hat{y}_n} u_1^n d_1 + \sum_{f_1(x^n) = \hat{y}_n} u_1^n / d_1$$

# 新权重下的训练数据

**$d_1$的值是什么?**

$$Z_1 = \sum_n u_1^n$$

$$Z_2 = \sum_n u_2^n$$

若$f_1(x^n) \neq \hat{y}_n$，则$u_2^n = u_1^n d_1$

若$f_1(x^n) = \hat{y}_n$，则$u_2^n = u_1^n / d_1$

$$\varepsilon_1 = \frac{\sum_n u_1^n \delta(f_1(x^n) \neq \hat{y}^n)}{Z_1}$$

$$\frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1}{\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1} = 1$$

$$\frac{\sum_{f_1(x^n)\neq\hat{y}_n} u_1^n d_1 + \sum_{f_1(x^n)=\hat{y}_n} u_1^n / d_1}{\sum_{f_1(x^n)\neq\hat{y}_n} u_1^n d_1} = 2$$

$$\sum_{f_1(x^n)=\hat{y}^n} u_1^n / d_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n d_1$$

$$\frac{1}{d_1} \underbrace{\sum_{f_1(x^n)=\hat{y}^n} u_1^n}_{Z_1(1-\varepsilon_1)} = d_1 \underbrace{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}_{Z_1\varepsilon_1}$$

$$Z_1(1-\varepsilon_1)/d_1 = Z_1\varepsilon_1 \, d_1$$

错误率：$\varepsilon_1 = \frac{\sum_{f_1(x^n)\neq\hat{y}^n} u_1^n}{Z_1}$ 则$Z_1\varepsilon_1 = \sum_{f_1(x^n)\neq\hat{y}^n} u_1^n$

$$d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} > 1$$

# AdaBoost算法

- 给定如下训练数据：

$$\{(x^1, \hat{y}^1, u_1^1), \cdots, (x^n, \hat{y}^n, u_1^n), \cdots, (x^N, \hat{y}^N, u_1^N)\}$$

$\hat{y} = \pm 1$（二分类），$u_1^n = 1$（相等权重）

- 对于t=1,...,T:
  - 用权重$\{u_t^1, \ldots, u_t^N\}$来训练弱分类器$f_t(x)$
  - $\varepsilon_t$为$f_t(x)$在权重$\{u_t^1, \ldots, u_t^N\}$上训练的错误率
  - 对于$n = 1, \ldots, N$:
    - 如果$x_n$被$f_t(x)$错误分类：$u_{t+1}^n = u_t^n * d_t = u_t^n * \exp(\alpha_t)$
    - Else:

$$u_{t+1}^n = u_t^n / d_t = u_t^n * \exp(-\alpha_t)$$

$$d_t = \sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n \leftarrow u_t^n \times \exp(-\hat{y}^n f_t(x^n)\ \alpha_t)$$

# AdaBoost算法

- 根据上述算法得到一组函数$f_1(x), \ldots, f_t(x), \ldots, f_T(x)$
- 如何整合他们?

误差$\varepsilon_t$越小,
最终投票权重越大

•统一的权重: $H(x) = sign(\sum_{t=1}^{T} f_t(x))$

•不统一的权重: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$\epsilon^t = 0.1$

$\alpha^t = 1.10$

$\epsilon^t = 0.4$

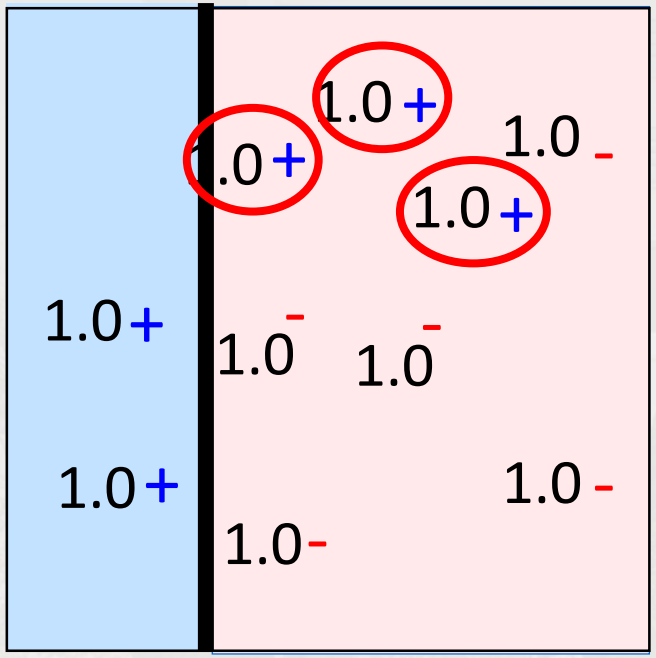
$\alpha^t = 0.20$

# AdaBoost

T=3, weak classifier = decision stump

➢例

- t=1

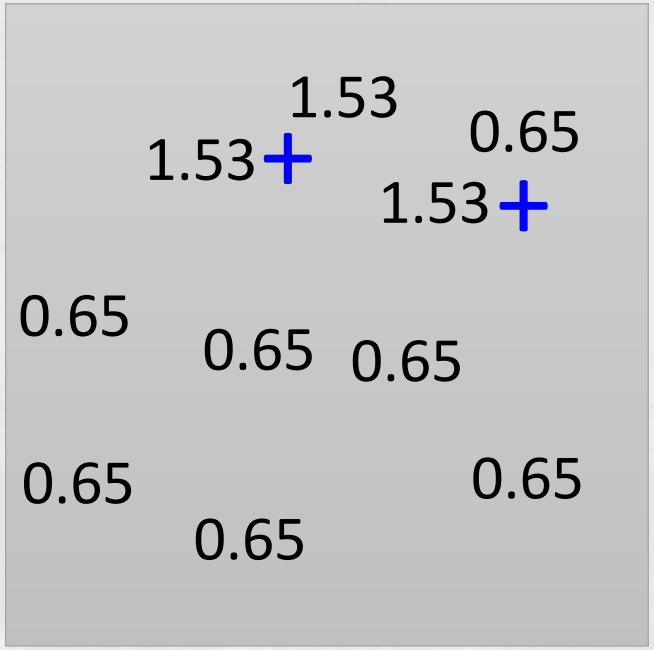



$h_1(x)$

$\varepsilon_1 = 0.30$

$d_1 = 1.53$

$\alpha_1 = 0.42$



# AdaBoost

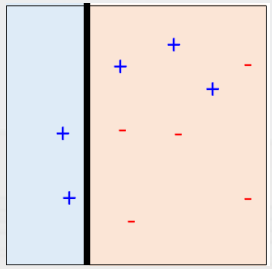
T=3, weak classifier = decision stump （决策树桩）

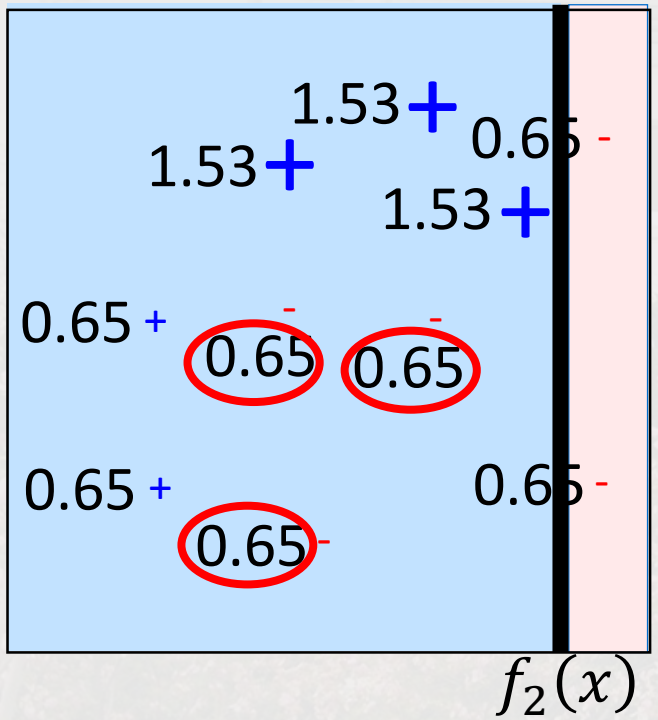
➢例

- t=2

$f_1(x)$:

$\alpha_1 = 0.42$



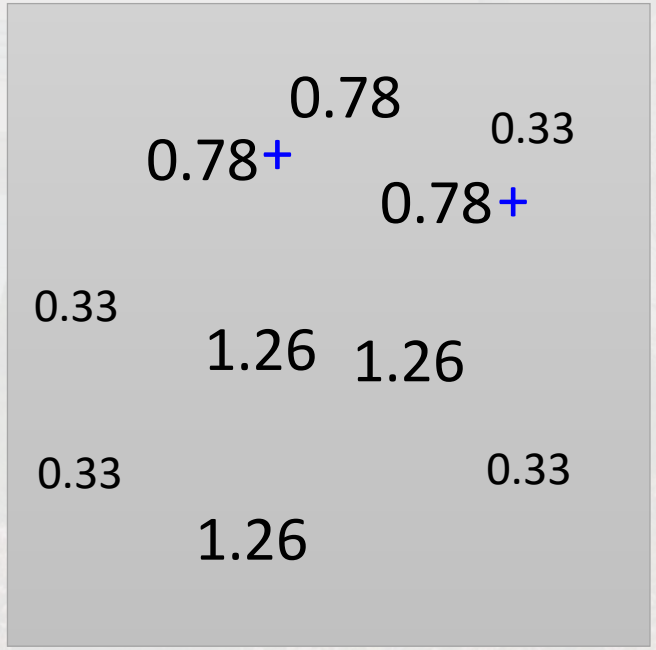



$f_2(x)$

$\varepsilon_2 = 0.21$

$d_2 = 1.94$

$\alpha_2 = 0.66$



# AdaBoost

采用标准的权值更新规则

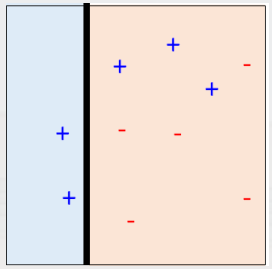
T=3, weak classifier = decision stump

➢例

- t=3

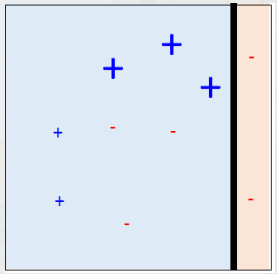
$f_1(x)$:

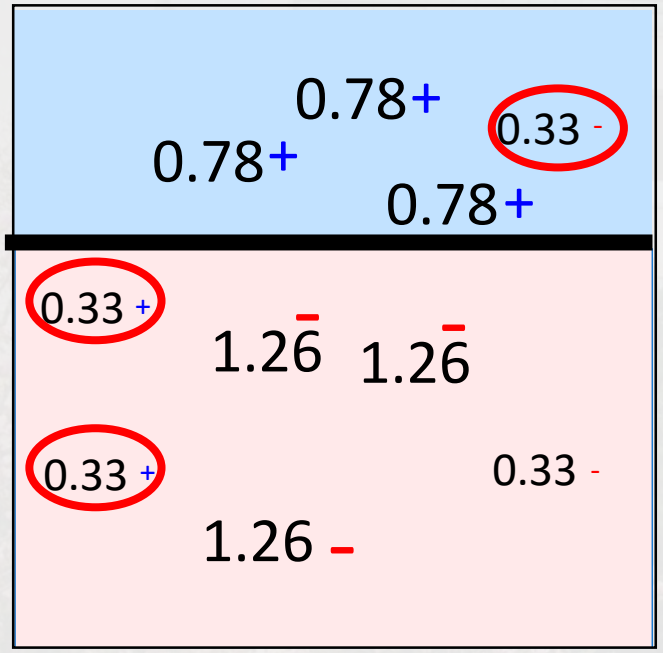
$\alpha_1 = 0.42$



$f_2(x)$ :

$\alpha_2 = 0.66$





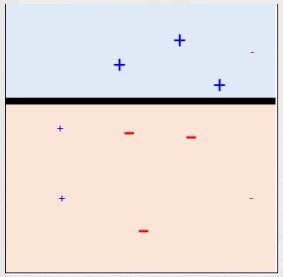

$\varepsilon_3 = 0.13$

$d_3 = 2.59$

$\alpha_3 = 0.95$

$f_3(x)$ :

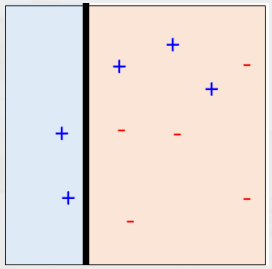
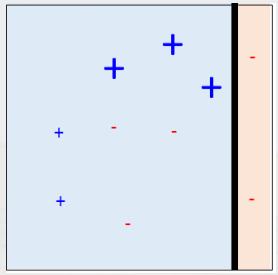
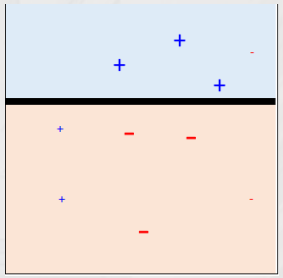
$\alpha_3 = 0.95$

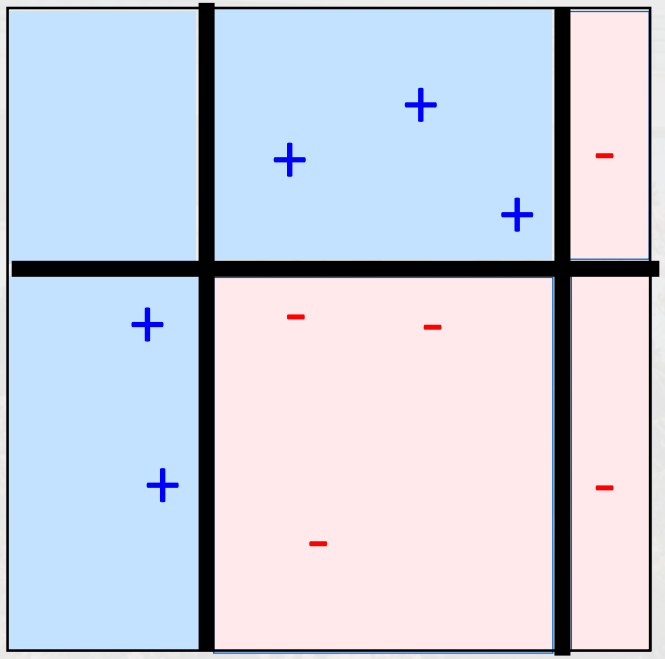


# AdaBoost

➢例

• 最终的分类器: $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$

$sign($ 0.42  + 0.66  + 0.95  $)$



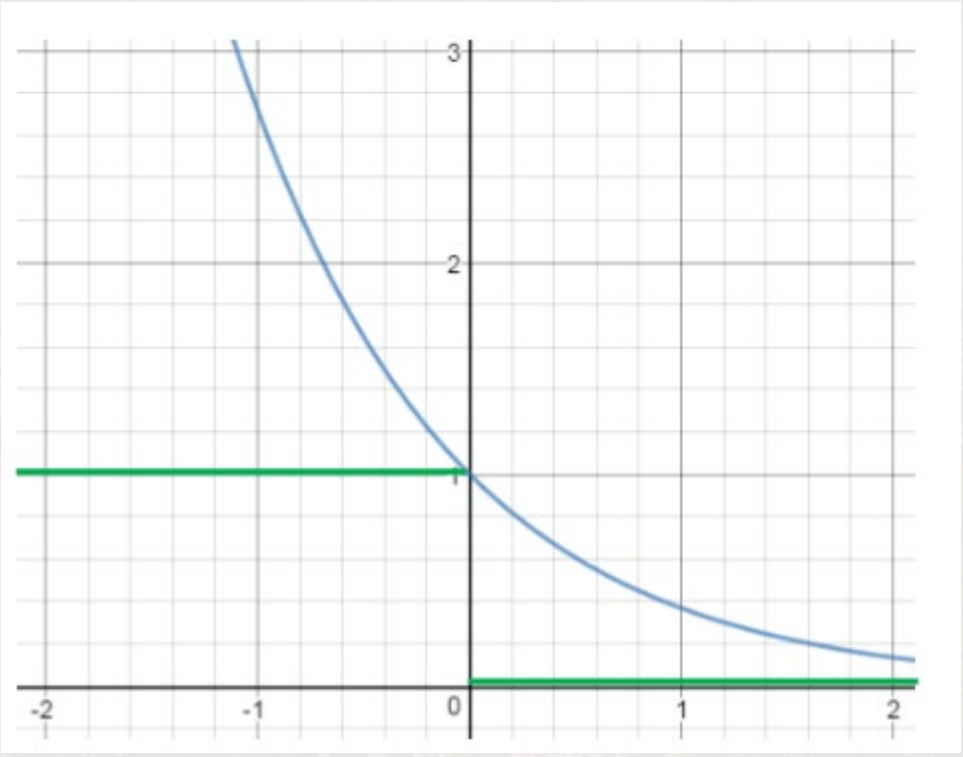
# AdaBoost

- 随着基学习器个数$T$的增加，最终分类器的训练误差越来越小
  - ✓ 最终的分类器： $H(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$ $g(x)$
    - $\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$
  - ✓ 训练数据错误率：

$$= \frac{1}{N}\sum_{n} \delta(H(x^n) \neq \hat{y}^n)$$

$$= \frac{1}{N}\sum_{n} \delta(\hat{y}^n g(x^n) < 0)$$

$$\leq \frac{1}{N}\sum_{n} exp(-\hat{y}^n g(x^n))$$




$\hat{y}^n g(x^n)$

# AdaBoost

- 训练数据错误率 $\leq \frac{1}{N}\sum_{n}\exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

- $Z_t$：训练 $f_t$ 的训练数据权重的总和，则：

$$Z_{T+1} = \sum_{n} u_{T+1}^n$$

$$u_1^n = 1$$

$$u_{t+1}^n = u_t^n \times exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$u_{T+1}^n = \prod_{t=1}^{T} exp(-\hat{y}^n f_t(x^n)\alpha_t)$$

$$Z_{T+1} = \sum_{n}\prod_{t=1}^{T} exp(-\hat{y}^n f_t(x^n)\alpha_t) = \sum_{n} exp\left(-\hat{y}^n \underbrace{\sum_{t=1}^{T} f_t(x^n)\alpha_t}_{g(x)}\right)$$

# AdaBoost

$$g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$$

$$\alpha_t = ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

- 训练数据错误率 $\leq \frac{1}{N}\sum_n \exp(-\hat{y}^n g(x^n)) = \frac{1}{N} Z_{T+1}$

$Z_1 = N$ （初始权重都相同=1）

$Z_{t+1} = Z_t \varepsilon_t e^{\alpha_t} + Z_t(1-\varepsilon_t)e^{-\alpha_t}$

$= Z_t\ \varepsilon_t\sqrt{(1-\varepsilon_t)/\varepsilon_t} + Z_t(1-\varepsilon_t)\sqrt{\varepsilon_t/(1-\varepsilon_t)}$

$= Z_t \times 2\sqrt{(1-\varepsilon_t)\varepsilon_t}$

$$Z_t\varepsilon_t = \sum_{f_t(x^n)\neq\hat{y}^n} u_t^n$$

$$Z_t(1-\varepsilon_t) = \sum_{f_t(x^n)=\hat{y}^n} u_t^n$$

$$Z_{T+1} = N\prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

- 训练数据错误率 $\leq \prod_{t=1}^{T} 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$

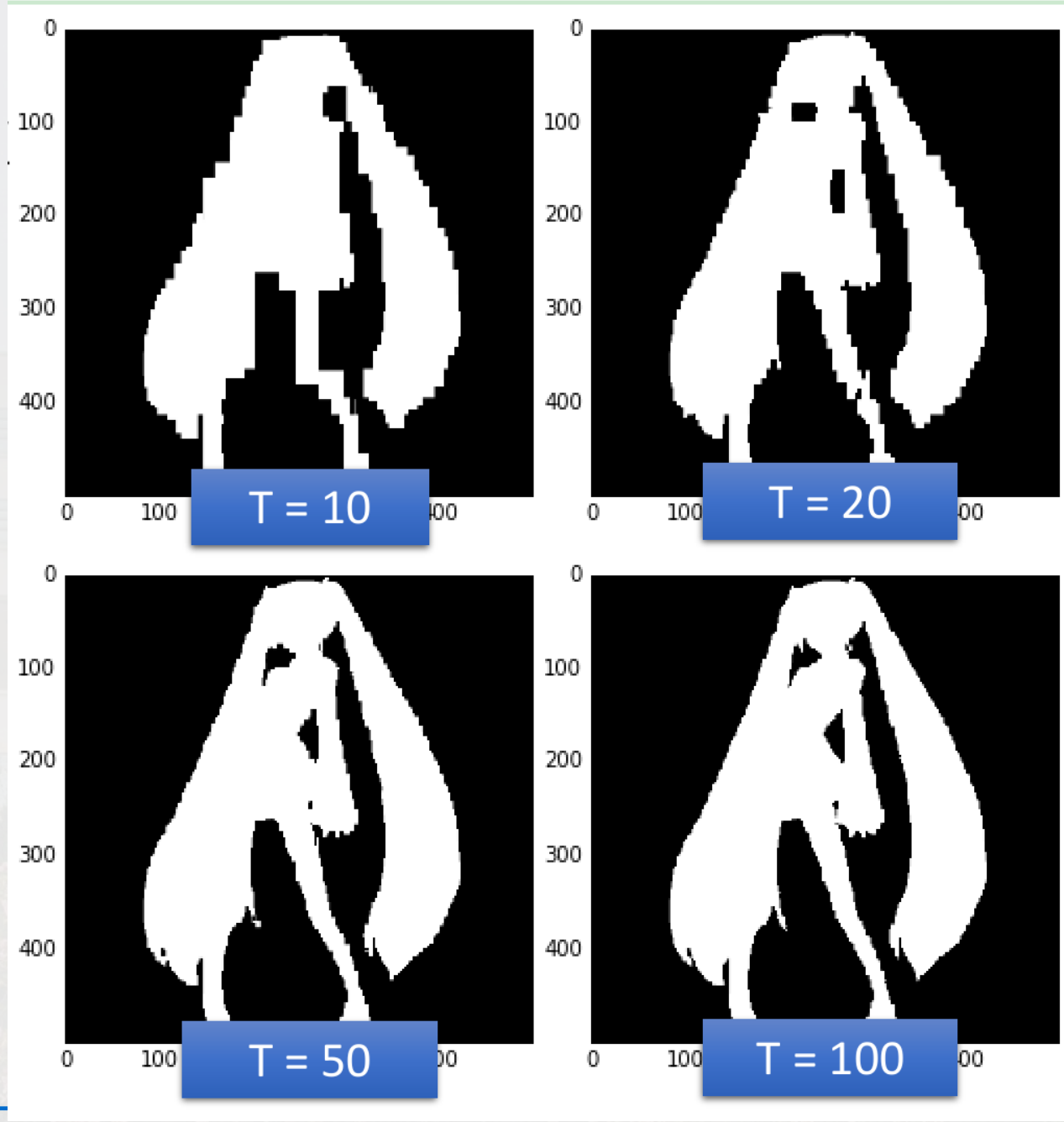
随着T的增加，变得越来越小

# 实验：关于Miku的函数

Adaboost+决策树

（深度=5）

https://ai.hgcitech.com/courseMake/97



# 拓展资料

- Introduction of Adaboost:
  - Freund; Schapire (1999). "A Short Introduction to Boosting“
- Multiclass/Regression
  - Y. Freund, R. Schapire, “A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.
  - Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In Proceedings of the Eleventh Annual Conference on Computational Learning Theory, pages 80–91, 1998.
- Gentle Boost
  - Schapire, Robert; Singer, Yoram (1999). "Improved Boosting Algorithms Using Confidence-rated Predictions".

# Adaboost总结

**优点：**

- 模型简单容易操作，不用重新设计模型
- 分类错误率上界随着训练增加而稳定下降（理论可证）
- 在一定程度上不存在过拟合

**缺点：**

- α值是局部最优
- 对噪声比较敏感
- 模型的可解释性较差

# Boosting的一般形式

- Boosting的工作流程可写为如下更一般的情形
  - ✓ 初始函数：$g_0(x) = 0$
  - ✓ 从t=1到T：
    - ✓ 寻找一个函数$f_t(x)$和$\alpha_t$来提高函数$g_{t-1}(x)$的性能
      - $g_{t-1}(x) = \Sigma_{i=1}^{t-1}\alpha_i f_i(x)$
    - ✓ $g_t(x) = g_{t-1}(x) + \alpha_t f_t(x)$
  - ✓ 输出：$H(x) = sign\big(g_T(x)\big)$

- $g(x)$的学习目标：使如下损失函数最小

$$L(g) = \sum_n l\big(\hat{y}^n, g(x^n)\big) = \sum_n exp(-\hat{y}^n g(x^n))$$

# Gradient Boosting

- 找到一个$g(x)$，使 $L(g)=\sum_n exp(-\hat{y}^n g(x^n))$ 最小

  •如果已经获得$g(x)=g_{t-1}(x)$，如何去更新$g(x)$?

采用梯度下降思想：

$$g_t(x)=g_{t-1}(x)-\eta\left.\frac{\partial L(g)}{\partial g(x)}\right|_{g(x)=g_{t-1}(x)}$$

同一方向

$$\sum_n exp(-\hat{y}^n g_{t-1}(x^n))(\quad \hat{y}^n)$$

Boosting角度：$g_t(x)=g_{t-1}(x)+\alpha_t f_t(x)$

# Gradient Boosting

$$f_t(x) \Longleftrightarrow \sum_n exp(-\hat{y}^n g_t(x^n))(\hat{y}^n)$$

同一方向

- 我们希望找到一个$f_t(x)$，使如下函数最大

最小化误差

$$\sum_n exp(-\hat{y}^n g_{t-1}(x^n))(\hat{y}^n) f_t(x^n)$$

样本权重$u^n_t$　　同一符号

$$u_t^n = exp\left(-\hat{y}^n g_{t-1}(x^n)\right) = exp\left(-\hat{y}^n \sum_{i=1}^{t-1} \alpha_i f_i(x^n)\right)$$

$$= \prod_{i=1}^{t-1} exp\left(-\hat{y}^n \alpha_i f_i(x^n)\right)$$

我们在Adaboost中获得的权重

# Gradient Boosting

- 找到一个g(x)，使 $L(g)=\sum_n exp(-\hat{y}^n g(x^n))$ 最小

$$g_t(x)=g_{t-1}(x)+\alpha_t f_t(x)$$

$\alpha_t$是一个类似于学习率的参数

固定$f_t(x)$找出使$L(g_t)$最小的$\alpha_t$

$$L(g)=\sum_n exp\left(-\hat{y}^n\left(g_{t-1}(x)+\alpha_t f_t(x)\right)\right)$$

$$=\sum_n exp\left(-\hat{y}^n g_{t-1}(x)\right)exp\left(-\hat{y}^n \alpha_t f_t(x)\right)$$

$$=\sum_{\hat{y}^n\neq f_t(x)} exp\left(-\hat{y}^n g_{t-1}(x^n)\right)exp(\alpha_t)+\sum_{\hat{y}^n=f_t(x)} exp\left(-\hat{y}^n g_{t-1}(x^n)\right)exp(-\alpha_t)$$

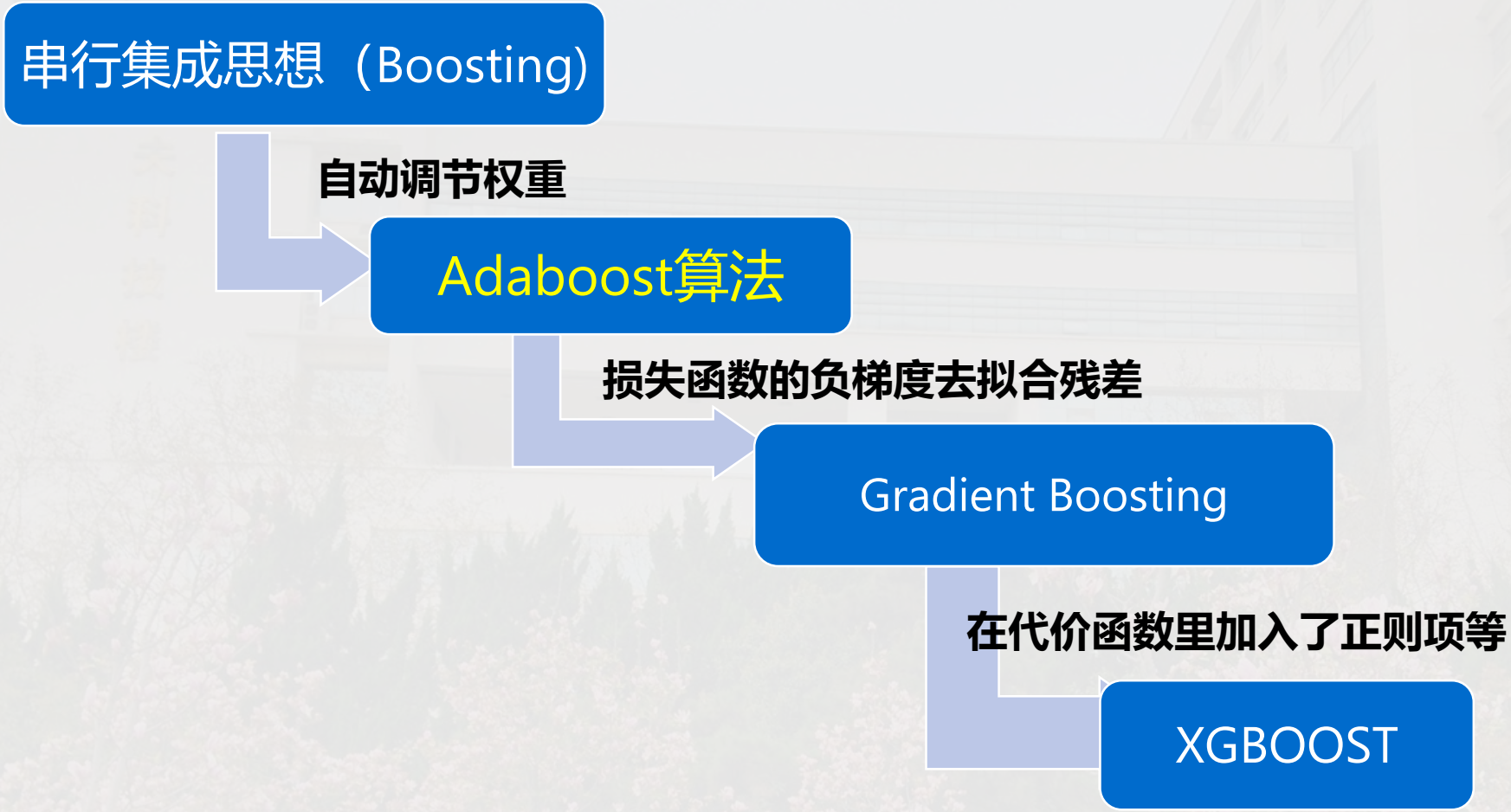
找到一个$\alpha_t$，使

$$\frac{\partial L(g)}{\partial \alpha_t}=0$$

$$\alpha_t=ln\sqrt{(1-\varepsilon_t)/\varepsilon_t}$$

Adaboost!

# 串行集成方法总结


串行集成思想（Boosting）
自动调节权重
Adaboost算法
损失函数的负梯度去拟合残差
Gradient Boosting
在代价函数里加入了正则项等
XGBOOST

# Bagging和Boosting总结

## ➢测验： Bagging和Boosting有什么不同

**1）样本选择上：**

Bagging：训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的。

Boosting：每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化。而权值是根据上一轮的分类结果进行调整。

**2）样例权重：**

Bagging：使用均匀取样，每个样例的权重相等

Boosting：根据错误率不断调整样例的权值，错误率越大则权重越大。

**3）预测函数：**

Bagging：所有预测函数的权重相等。

Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。

**4）并行计算：**

Bagging：各个预测函数可以并行生成。

Boosting：各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果。

# 小结

- 理解集成学习框架和思路
- 掌握Bagging和随机森林的集成方法
- 掌握AdaBoost的工作流程：权重的计算，分类误差
- 理解Gradient Boosting的思想