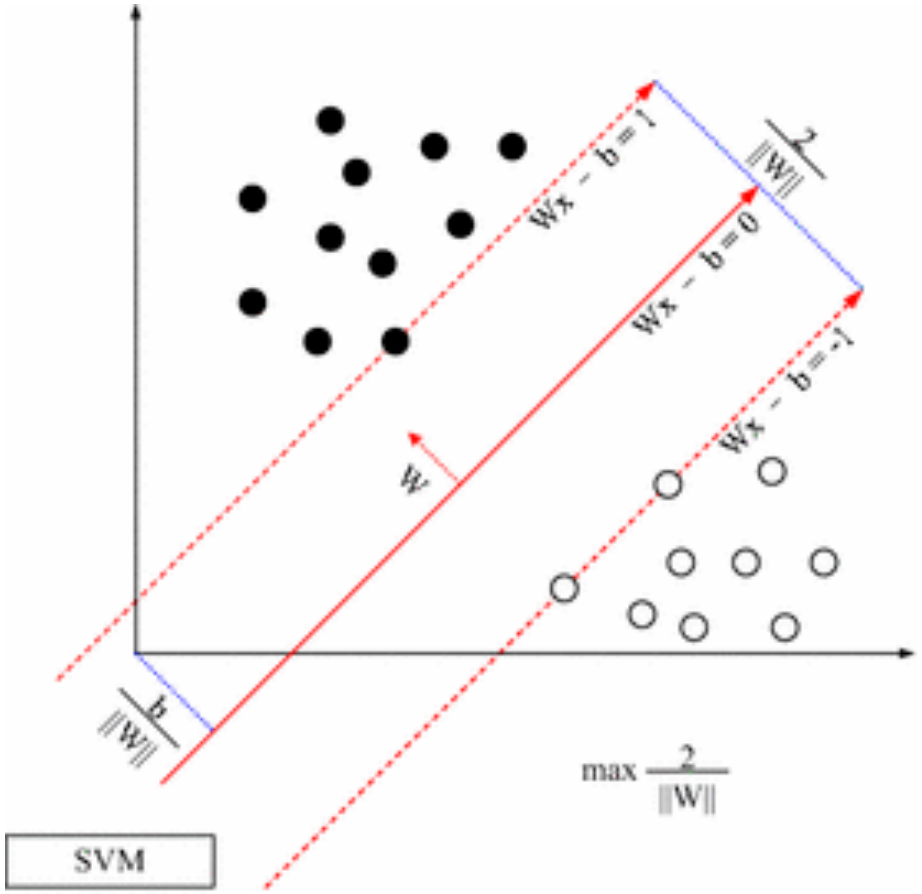
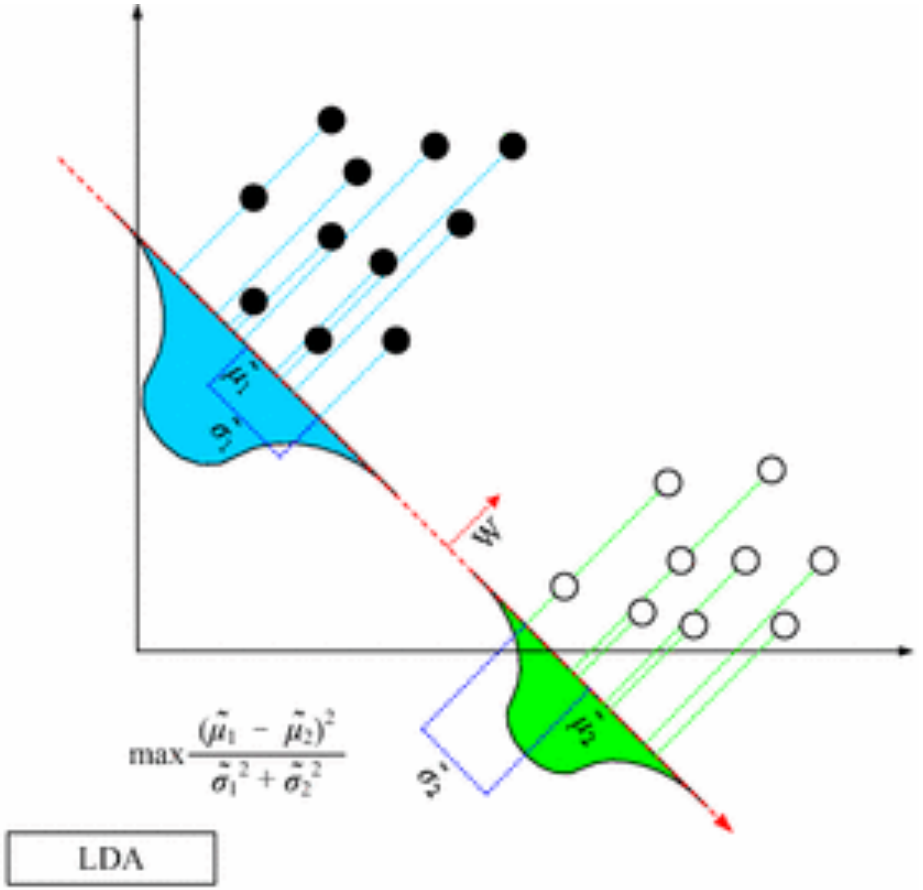
贪心科技
让每个人享受个性化教育服务

LDA, Ensemble Methods

线性判别分析（LDA）
Linear Discriminant Analysis

- LDA 即可以直接作为分类算法，也可以作为特征降维的方法





LDA 作为分类器（1）

- 假设$f_k(x)$为类别 $G = k$ 的数据的条件概率密度函数, π_k 为 为类别 $G = k$ 的数据的先验概率 $\sum_{k=1}^{K} \pi_k = 1, \pi_k = \frac{\mathrm{N_k}}{\mathrm{N}}$. 根据贝叶斯理论
- $P_r(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$
- LDA 模型中假设:
  - $f_k(x) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-u_k)^T\Sigma_k^{-1}(x-u_k)}$
  - $\Sigma_k = \Sigma \, \forall_k$
  - $\mathrm{u_k} = \sum_{\mathrm{g}_i=k} \frac{x_i}{N_k}$
- $\Sigma = \sum_{k=1}^{K} \sum_{\mathrm{g}_i=k} (x_i - u_k)\,(x_i - u_k)^T$
- $\sum_{l=1}^{K} f_l(x)\pi_l$为常数, 可以使用 $1/c'$ 代替, 于是
- $P_r(G = k|X = x) = \frac{f_k(x)\pi_k}{1/c\prime} = \frac{c'\pi_k}{(2\pi)^{\frac{p}{2}}|\Sigma_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-u_k)^T\Sigma_k^{-1}(x-u_k)}$

LDA 作为分类器（2）

- $\frac{c'}{(2\pi)^{\frac{p}{2}}|\Sigma_k|^{\frac{1}{2}}}$ 为常数, 记为 c, 于是
- $P_r(G=k|X=x) = c\pi_k e^{-\frac{1}{2}(x-u_k)^T\Sigma_k^{-1}(x-u_k)}$
- 两边取对数, 得到
- $\log P_r(G=k|X=x) = \log c + \log \pi_k - \frac{1}{2}(x-u_k)^T\Sigma_k^{-1}(x-u_k)$
- 其中 $\log c$ 对所有类别都一样, 于是我们的目标就是找到哪个类别 k 的 $\log P_r(G=k|X=x)$ 最大, 就是最优化如下函数:
- $\log \pi_k - \frac{1}{2}(x-u_k)^T\Sigma_k^{-1}(x-u_k) = \log \pi_k - \frac{1}{2}\left[x^T\Sigma_k^{-1}x + u_k^T\Sigma_k^{-1}u_k\right] + x\Sigma_k^{-1}u_k$
- 其中 $x^T\Sigma_k^{-1}x$ 对所有类别都一样, 在类别判断时可以不计入计算值, 于是, 最优化目标为:
- $\delta_k(x) = x^T\Sigma^{-1}\mathrm{u_k} - \frac{1}{2}\mathrm{u_k^T}\Sigma^{-1}\mathrm{u_k} + log\pi_k$

协方差矩阵

- 协方差**矩阵**，其 *i, j* 位置的元素是第 *i* 个与第 *j* 个随机向量（即随机变量构成的向量）之间的协方差。

假设X是以n个随机变数组成的列向量，

$$\mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$

并且μ_i是X_i的期望值，即，$\mu_i = \mathrm{E}(X_i)$。协方差矩阵的第(i, j)项（第(i, j)项是一个协方差）被定义为如下形式：

$$\Sigma_{ij} = \operatorname{cov}(X_i, X_j) = \mathrm{E}\left[(X_i - \mu_i)(X_j - \mu_j)^{\mathrm{T}} \right]$$

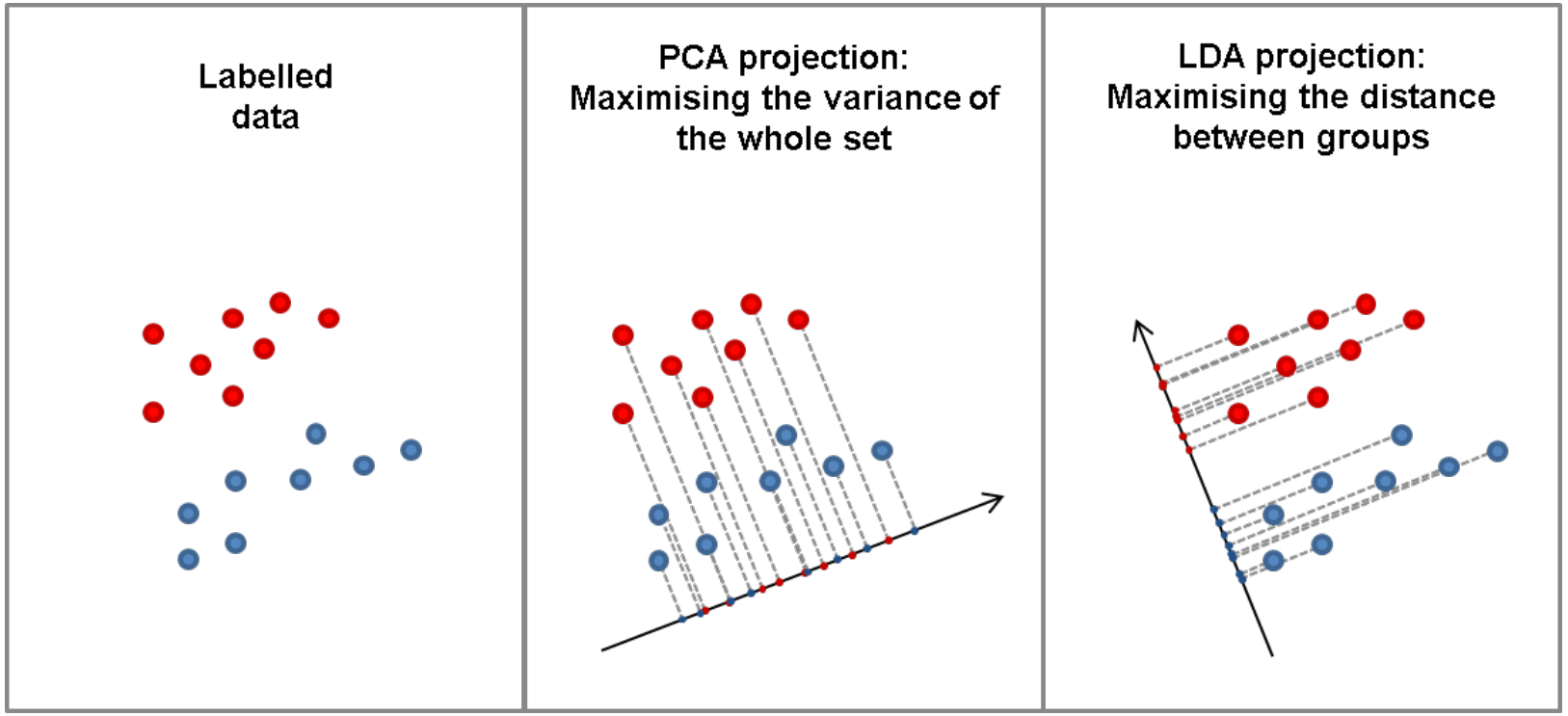
而协方差矩阵为：

$$\Sigma = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}]) (\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\mathrm{T}} \right]$$

$$= \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}$$

LDA 作为特征降维工具

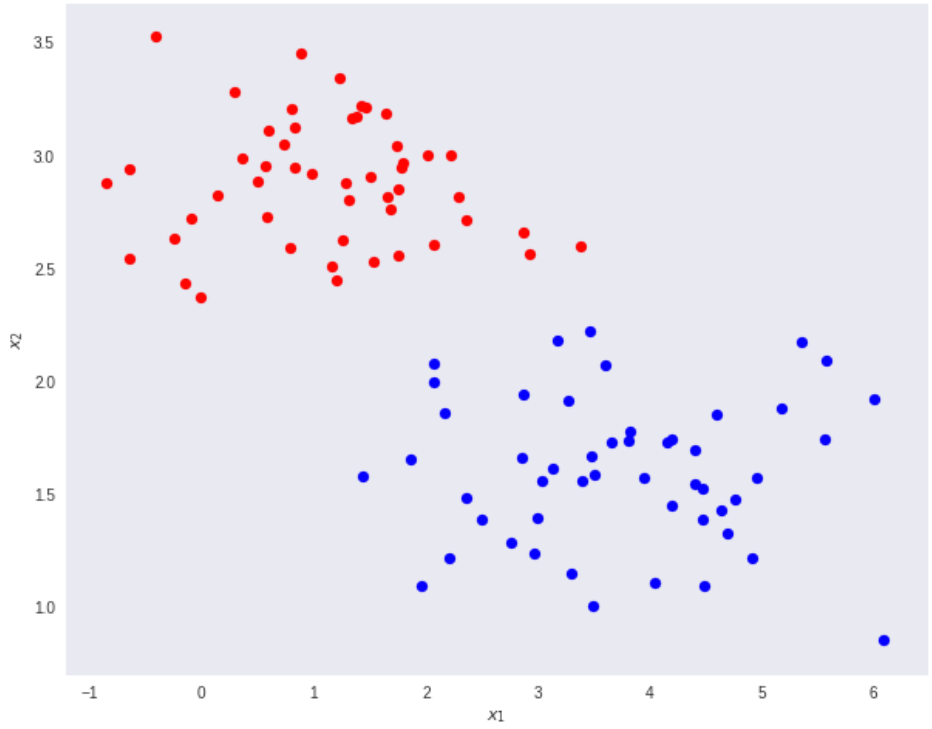
- LDA的原理是，将带上标签的数据（点），通过投影的方法，投影到维度更低的空间中，使得投影后的点，会形成按类别区分，一簇一簇的情况，相同类别的点，将会在投影后的空间中更接近。

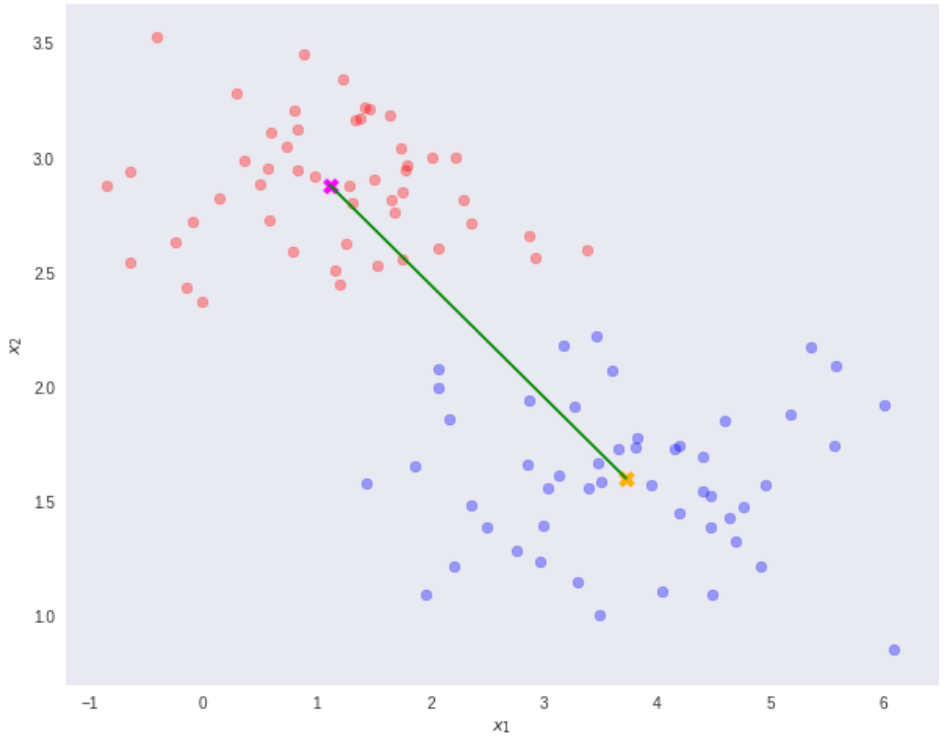


LDA 降维直观理解(1)

- 计算每个类别的数据的中心

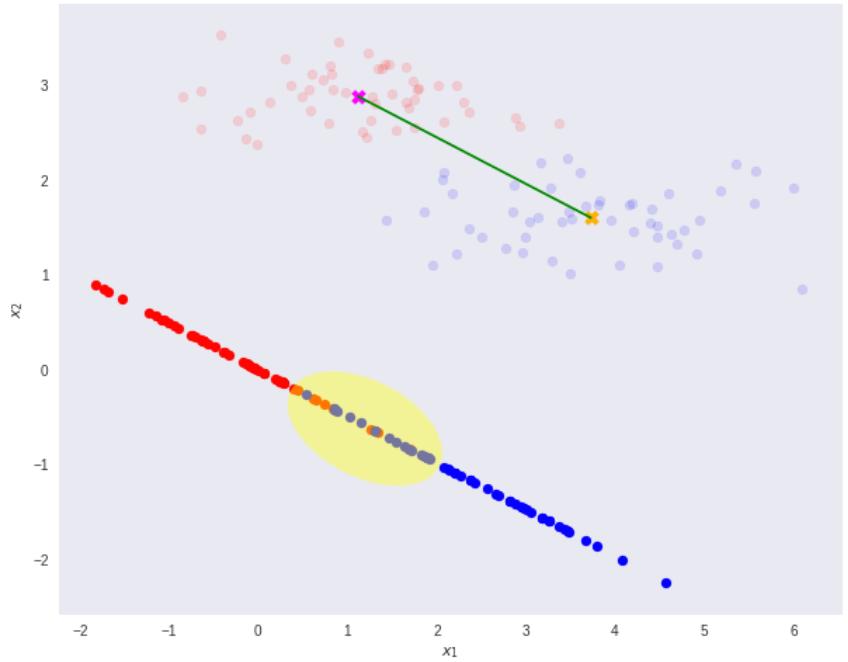
$$m_1 = \frac{1}{N_1}\sum_{n\in C1} x_n \qquad m_2 = \frac{1}{N_2}\sum_{n\in C2} x_n \qquad (1)$$

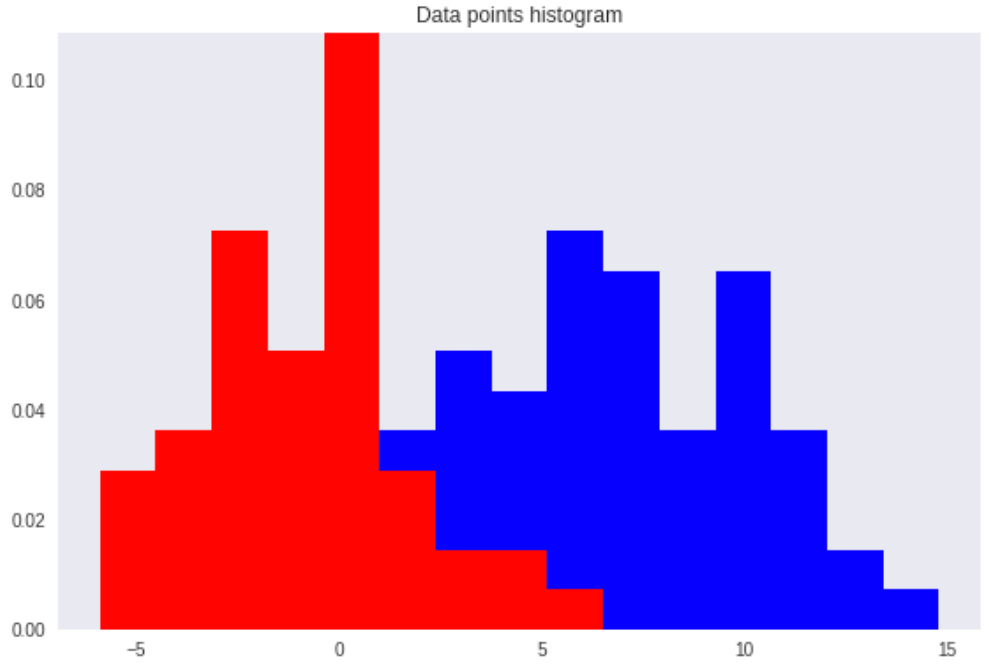




LDA 降维直观理解(2)

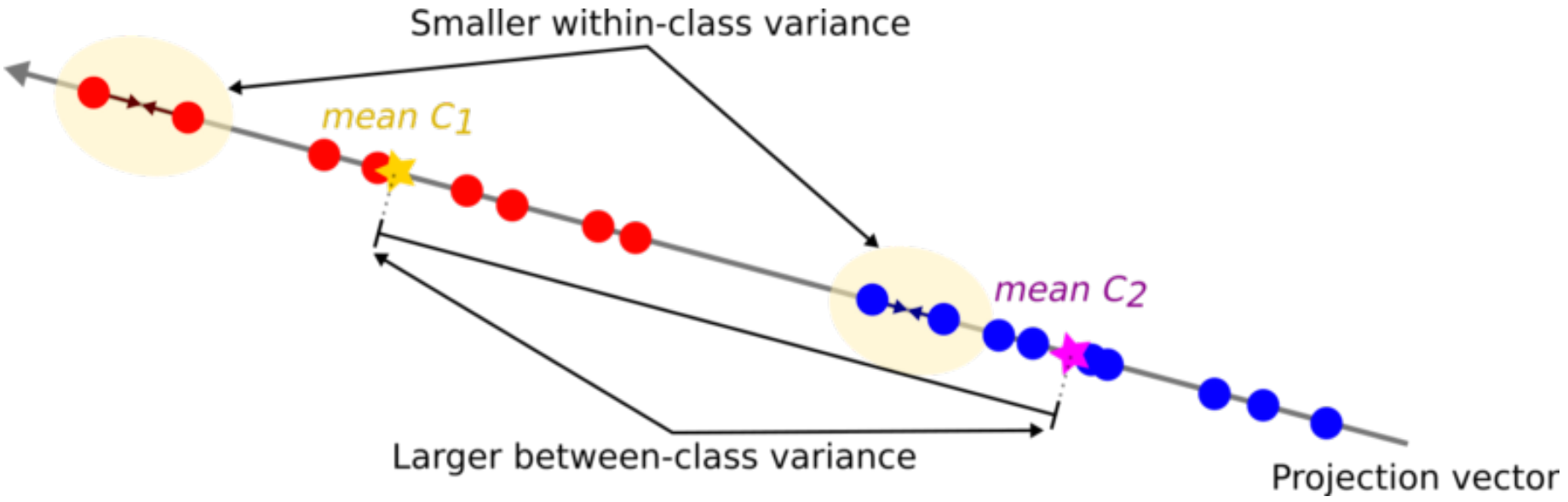
- 如果将数据直接投影到连接两个中心的向量 W 上(降维), 并非最优, 数据有重叠. 原来可线性分割的数据, 降维后无法线性分割了.





LDA 降维直观理解(3)

- Fisher's Linear Discriminant: 降维的数据满足两个特征来减少重叠:
  - 不同类别数据降维之后相互之间的差异大
  - 同一类别数据降维之后相互之间的差异小



LDA 降维公式推导（1）

最优化问题

$$\max_w J(w) = \frac{w^T S_B w}{w^T S_w w}$$

$$S_B = \sum_c (\boldsymbol{\mu}_c - \bar{\mathbf{x}})(\boldsymbol{\mu}_c - \bar{\mathbf{x}})^T$$

$$S_W = \sum_c \sum_{i \in c} (\mathbf{x}_i - \boldsymbol{\mu}_c)(\mathbf{x}_i - \boldsymbol{\mu}_c)^T$$

转换为带约束最优化问题

$$\begin{aligned} \min_{\mathbf{w}} \quad & -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} \\ \text{s.t.} \quad & \mathbf{w}^T S_W \mathbf{w} = 1 \end{aligned}$$

拉格朗日函数

$$\mathcal{L}_P = -\frac{1}{2}\mathbf{w}^T S_B \mathbf{w} + \frac{1}{2}\lambda(\mathbf{w}^T S_W \mathbf{w} - 1)$$

KKT 条件(最优解时,对w的梯度为0)

$$S_B \mathbf{w} = \lambda S_W \mathbf{w} \quad \Rightarrow \quad S_W^{-1} S_B \mathbf{w} = \lambda \mathbf{w}$$

如果 $S_w^{-1} S_B$ 为对称矩阵, 求w的问题就是解特征向量的问题, 但是这个情况下不一定对称, 此问题为广义特征向量问题. 已知S_B为对称正定矩阵, 可以写为$S_B^{\frac{1}{2}} S_B^{\frac{1}{2}}$, 计算$S_B^{\frac{1}{2}}$的方式为:

LDA 降维公式推导（2）

$$S_B\mathbf{w} = \lambda S_W\mathbf{w} \quad \Rightarrow \quad S_W^{-1}S_B\mathbf{w} = \lambda\mathbf{w}$$

- 如果 $S_w^{-1}S_B$ 为对称矩阵, 求w的问题就是解特征向量的问题, 但是这个情况下不一定对称, 此问题为广义特征向量问题. 已知 S_B 为对称正定矩阵, 可以写为 $S_B^{\frac{1}{2}}S_B^{\frac{1}{2}}$, 计算 $S_B^{\frac{1}{2}}$ 的方式为: 对 S_B 做特征值分解 $S_B = U\Lambda U^T \rightarrow S_B^{\frac{1}{2}} = U\Lambda^{\frac{1}{2}}U^T$. 定义 $v = S_B^{\frac{1}{2}}w$, 可以得到:
- $S_B^{\frac{1}{2}}S_w^{-1}S_B^{\frac{1}{2}}v = \lambda v$
- 由于 $S_B^{\frac{1}{2}}S_w^{-1}S_B^{\frac{1}{2}}$ 是一个对称正定矩阵, 这个问题变为特征值问题. 找到 λ_k 和 v_k 以后, 可以得到:
- $w_k = S_B^{-\frac{1}{2}}v_k$

如何证明协方差矩阵一定半正定

- 协方差矩阵的定义为 $\Sigma = \mathrm{E}((X-\mu)(X-\mu)^T)$
- 显然协方差矩阵是对称的, 下面只要证明对于任意的非零向量u:

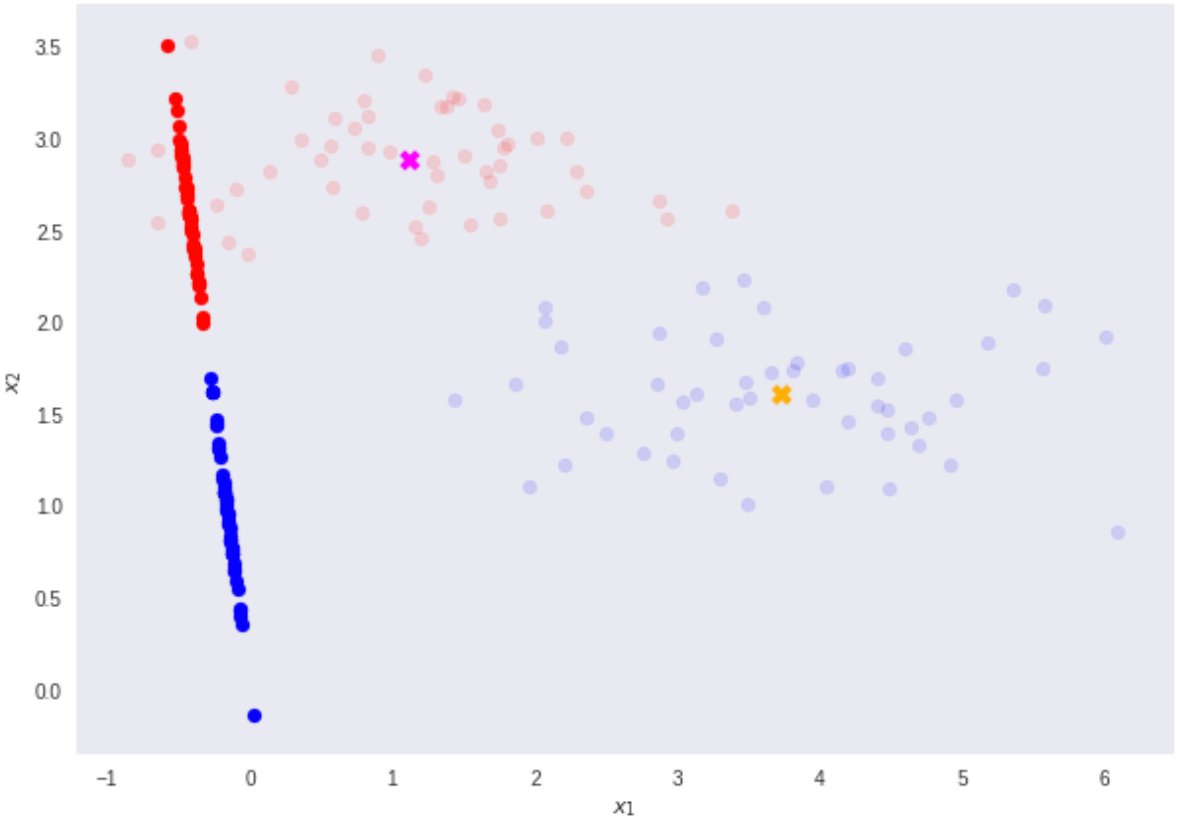
$$u\Sigma u^T \geq 0$$

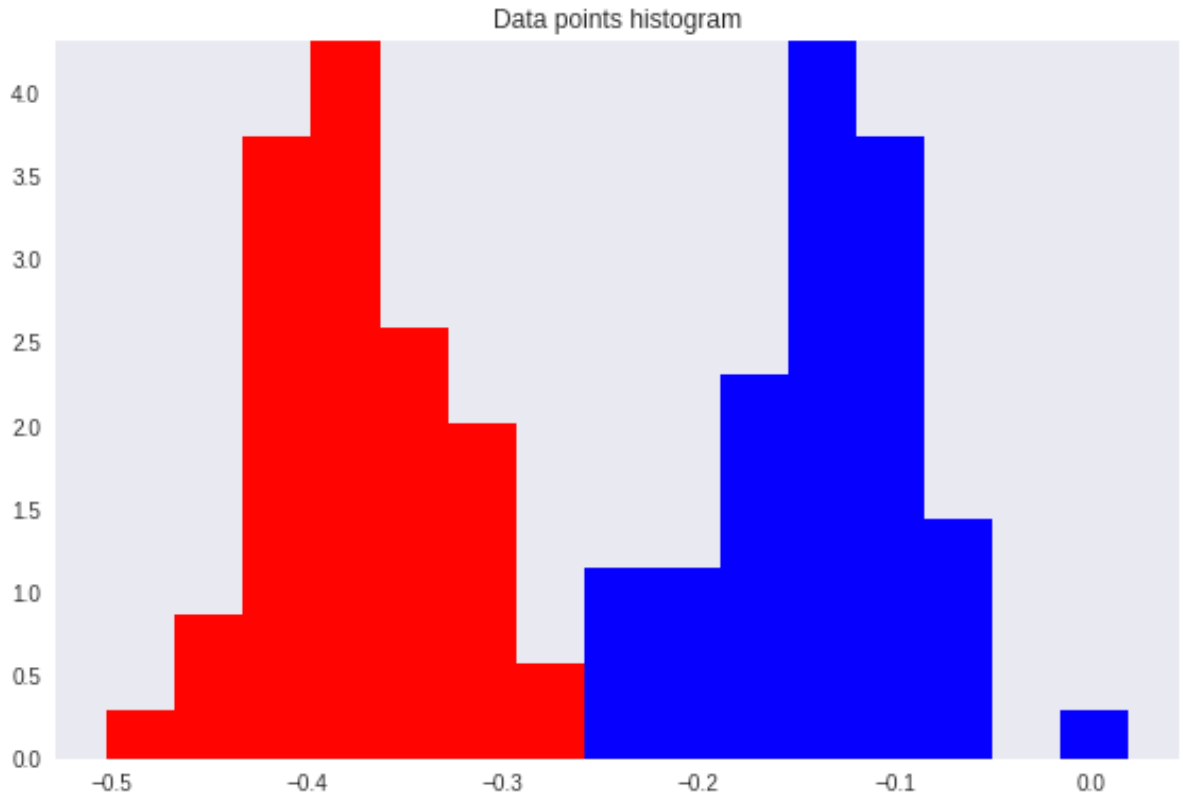
- 证明:

$$u\Sigma u^T = u\mathrm{E}((X-\mu)(X-\mu)^T)u^T = \mathrm{E}((u(X-\mu))((u(X-\mu))^T)) = \mathrm{E}((u(X-\mu))^2) \geq 0$$

- 一个数的平方的期望当然是非负的所以协方差是半正定

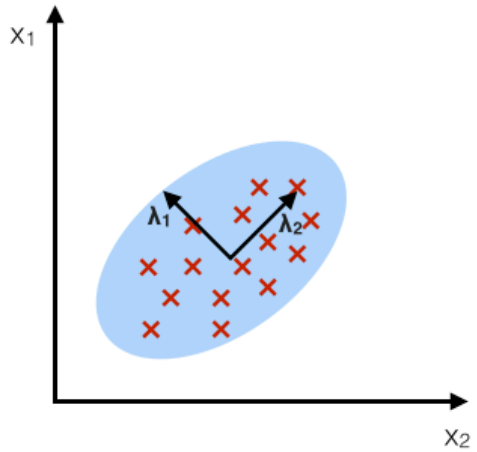
LDA 降维


3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
x2
−1
0
1
2
3
4
5
6
x1

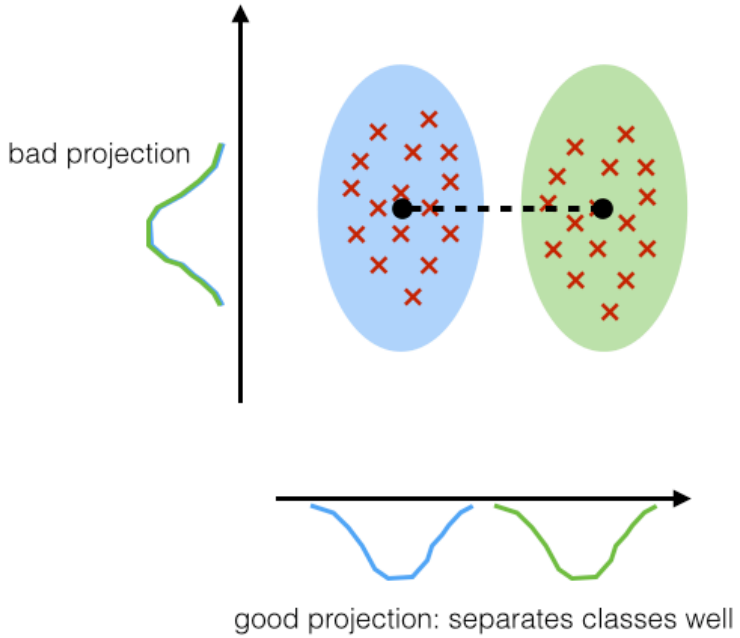

Data points histogram
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−0.4
−0.3
−0.2
−0.1
0.0

PCA vs LDA

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



LDA 降维(两个类别)

- 假设包含两组类别的数据: $\chi_1 = \{x_1^1, \dots, x_{l_1}^1\}$ 和$\chi_2 = \{x_2^2, \dots, x_{l_2}^2\}$,
- 假设类别 i 的均值为$m_i = \frac{1}{l_i}\sum_{j=1}^{l_i} x_j^i$
- LDA 的解就是使得如下公式取得最大值时候的 w:
- $J(w) = \frac{w^T S_B w}{w^T S_w w} = \frac{(w^T m_1 - w^T m_2)(m_1^T w - m_2^T w)}{w^T \sum_{x\in\chi_1}(x-m_i)(x-m_i)^T w + w^T \sum_{x\in\chi_2}(x-m_i)(x-m_i)^T w} = \frac{[w^T(m_1-m_2)][(m_1^T-m_2^T)w]}{w^T\Sigma_1 w + w^T\Sigma_2 w} = \frac{[w^T(m_1-m_2)]^2}{w^T\Sigma_1 w + w^T\Sigma_2 w}$
- $S_B = (m_1 - m_2)(m_1 - m_2)^T$
- $S_w = \sum_{i=1,2}\sum_{x\in\chi_i}(x - m_i)(x - m_i)^T$
- $w \propto (\Sigma_1 + \Sigma_2)^{-1}(m_1 - m_2)$
- 使用均值的投影$w^T m_1$和$w^T m_2$的均值作为分类阈值
- $\mathrm{T} = \frac{1}{2}w(m_1 - m_2)$

Kernel Fisher LDA

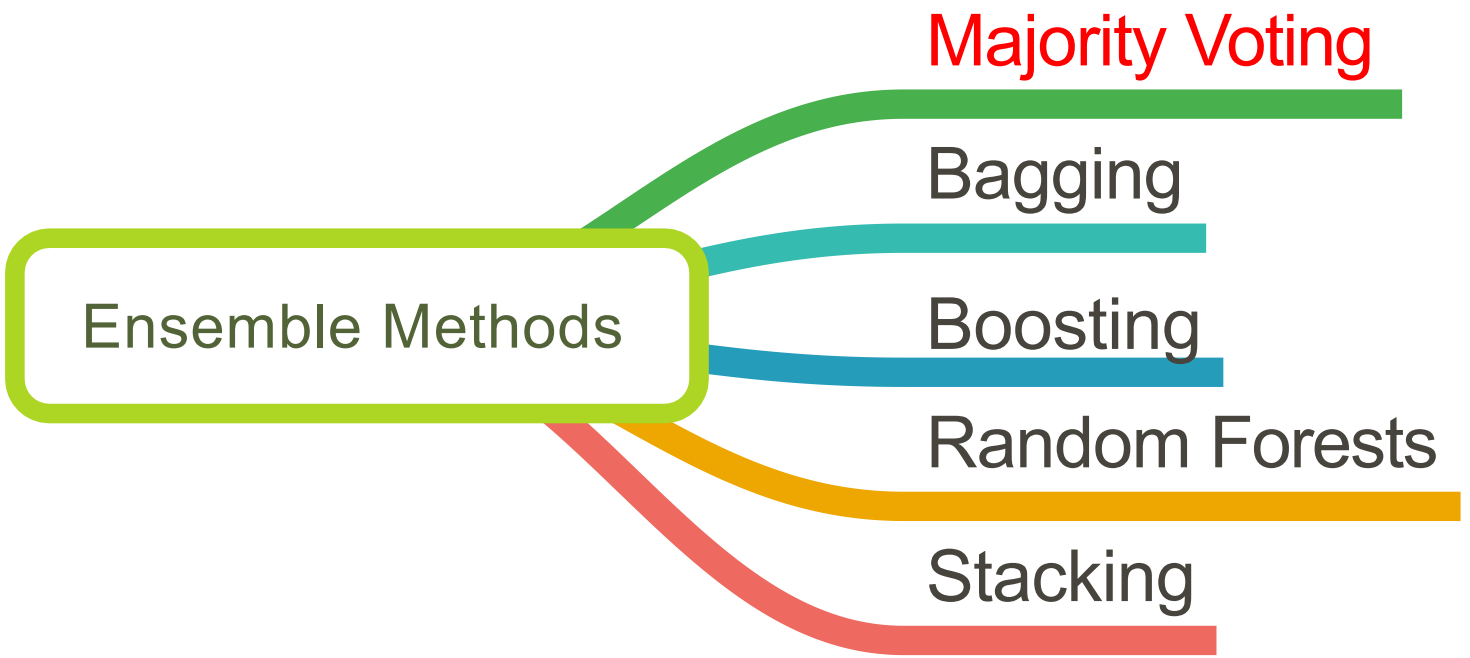
- 假设Φ为一个非线性映射到特征空间$\mathcal{F}$, 在特征中间中LDA就是最优化:
- $\max_w J(w) = \frac{w^T S_B^\Phi w}{w^T S_w^\Phi w}$
- 其中$w \in \mathcal{F}$, S_B^Φ和S_w^Φ为定义在$\mathcal{F}$中的矩阵
  - $S_B^\Phi = (m_1^\Phi - m_2^\Phi)(m_1^\Phi - m_2^\Phi)^{\mathrm{T}}$
  - $S_w^\Phi = \sum_{i=1,2} \sum_{x \in \chi_i} (\Phi(x) - m_i^\Phi)(\Phi(x) - m_i^\Phi)^{\mathrm{T}}$
  - $m_i^\Phi = \frac{1}{l_i} \sum_{j=1}^{l_i} \Phi(x_j^i)$
- 借鉴SVM的思路, 避免直接投影到特征空间$\mathcal{F}$, 而利用Mercer Kernels. 使用$k(x,y) = (\Phi(x), \Phi(y))$替代$\langle x, y \rangle$.
- w一定可以表示为所有训练数据的线性组合, 假设 $w = \sum_i \alpha_i \Phi(x_i)$

Kernel Fisher LDA（续）

- 根据m_i^Φ的定义, $w^T m_i^\Phi = \frac{1}{l_i}\sum_{j=1}^{l}\sum_{k}^{l_i}\alpha_j k(x_j, x_k^i) = \boldsymbol{\alpha^T M_i}$
- $(\boldsymbol{M_i})_{\boldsymbol{j}} = \frac{1}{l_i}\sum_{k}^{l_i}\alpha_j k(x_j, x_k^i)$
- 于是 定义$\boldsymbol{M} = (\boldsymbol{M_1} - \boldsymbol{M_2})(\boldsymbol{M_1} - \boldsymbol{M_2})^T$, 则
- $w^T S_B^\Phi w = \boldsymbol{\alpha^T M \alpha}$
- 定义 $\boldsymbol{N} = \sum_{i=1,2} K_j\left(I - 1_{l_j}\right)K_j^T$, 其中 K_j 为 $l \times l_i$ 矩阵, 为第 j 个类别的Kernel Matrix, $\left(K_j\right)_{\mathrm{nm}} = k\left(x_n, x_m^j\right)$. I 为单位矩阵, 1_{l_j}为所有元素值为$1/l_j$的矩阵, 则
- $w^T S_w^\Phi w = \boldsymbol{\alpha^T N \alpha}$
- 于是问题为最优化
- $\max_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \frac{\boldsymbol{\alpha^T M \alpha}}{\boldsymbol{\alpha^T N \alpha}}$
- 同理可以使用LDA的方法解 $\boldsymbol{\alpha}$, Kernel LDA的投影方法为:
- $\langle w, \Phi(x)\rangle = \sum_{i=1}^{l}\alpha_i k(x_i, x)$
- $\boldsymbol{N}$为对 l 个数据做 l 为的协方差矩阵, l 很可能大于数据的特征数量, 这样的话 $\boldsymbol{N}$ 不一定可逆, 可以使用 $\boldsymbol{N}_\lambda = \boldsymbol{N} + \lambda I$ 代替$\boldsymbol{N}$. 这种替代可以理解为:
  - 当λ足够大, $\boldsymbol{N}_\lambda$一定会为正定, 可逆
  - 可以看作是正则化regularization

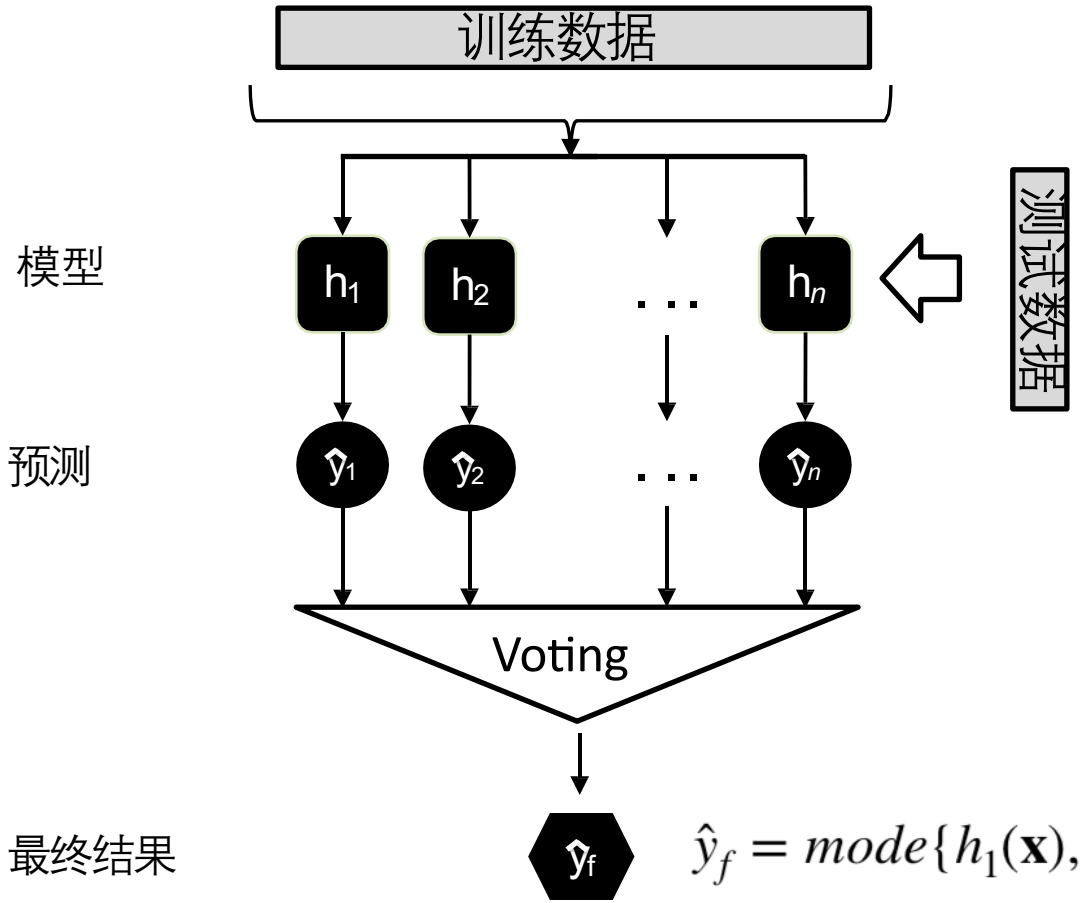
集成学习方法
Ensemble Methods

集成方法一览


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

Majority Voting

Majority Vote Classifier



$$\hat{y}_f = mode\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_n(\mathbf{x})\}$$

where $h_i(\mathbf{x}) = \hat{y}_i$

Why Majority Vote?

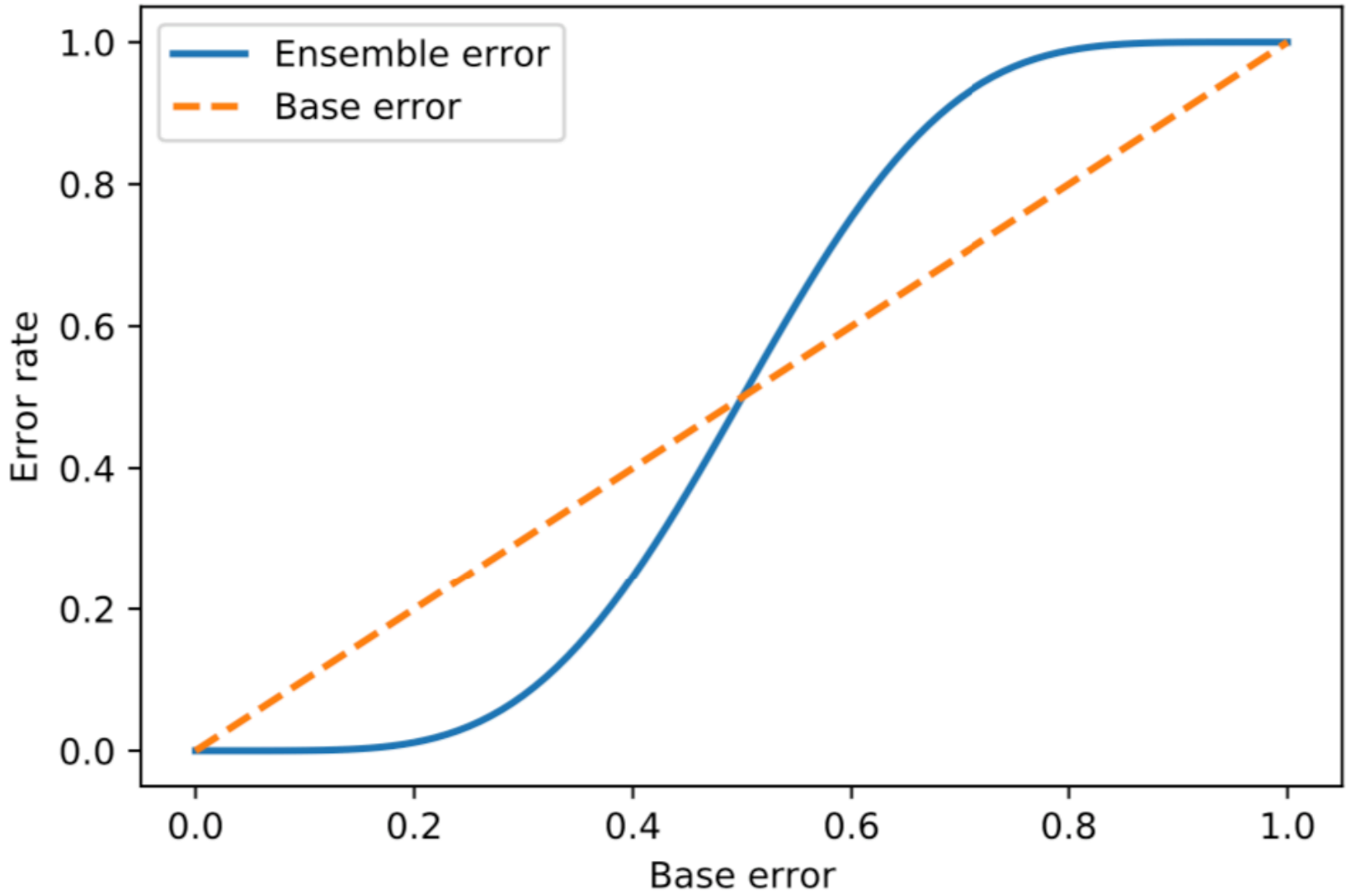
- 假设有 *n* 个独立的分类模型, 每一个的错误率都为ϵ
- 独立意味着误差是不相关的
- 假设是二元分类的场景
- 假设每一个分类模型的错误率只是比随机猜测好一点 (错误率低于 0.5)

$$\forall \epsilon_i \in \{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}, \epsilon_i < 0.5$$

- 那么 k 个模型输出都是错误类别, 造成最终是错误类别的概率

$$P(k) = \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k} \qquad k > \lceil n/2 \rceil$$

$$\epsilon_{ens} = \sum_{k}^{n} \binom{n}{k} \epsilon^k (1-\epsilon)^{n-k}$$



$$\epsilon_{ens} = \sum_{k=6}^{11} \binom{11}{k} 0.25^k (1 - 0.25)^{11-k} = 0.034$$

"Soft" Voting

$$\hat{y} = \arg\max_{j} \sum_{i=1}^{n} w_i p_{i,j}$$

$p_{i,j}$: *第 i 个分类模型输出类别为 j 的概率*

w_i: *第 i 个分类模型的权重, 例如* $w_i = 1/n, \forall w_i \in \{w_1, \ldots, w_n\}$

"Soft" Voting $\hat{y} = \arg\max_j \sum_{i=1}^{n} w_i p_{i,j}$

Binary classification example

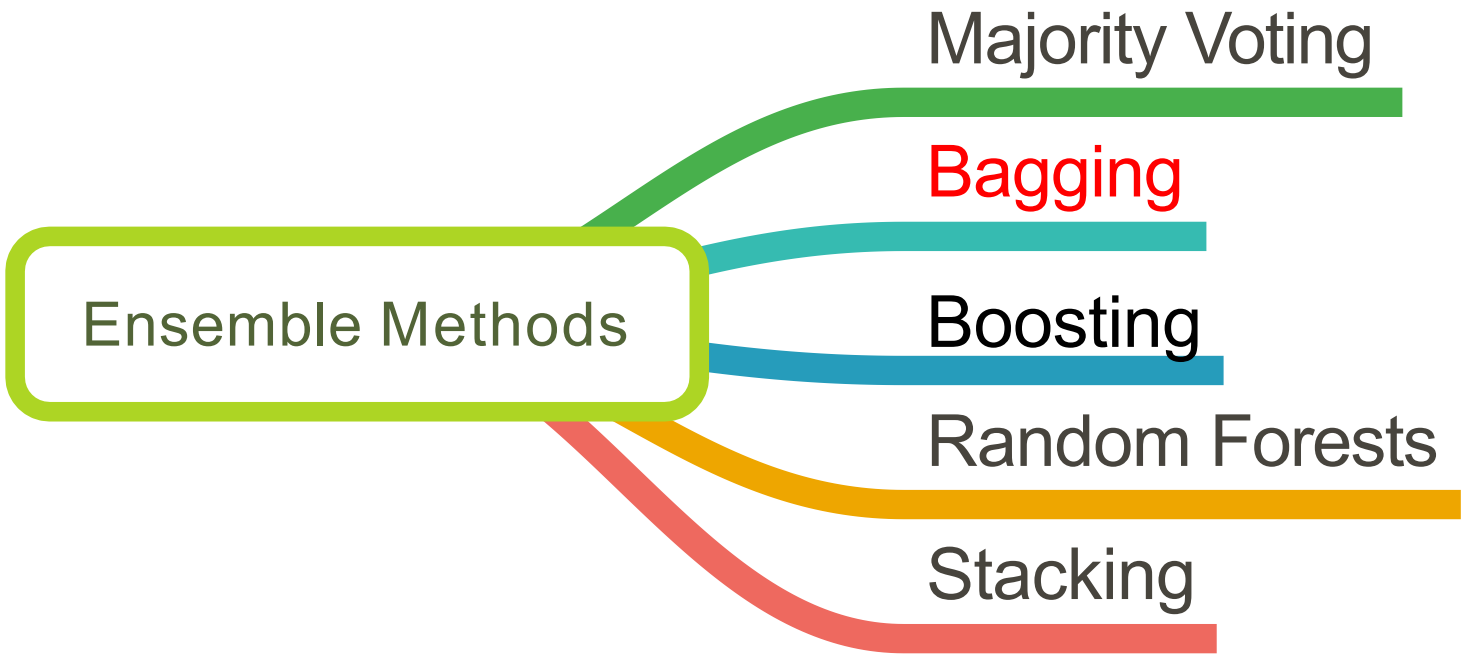
$j \in \{0,1\}$ $h_i(i \in \{1,2,3\})$

$$h_1(\mathbf{x}) \rightarrow [0.9,0.1]$$
$$h_2(\mathbf{x}) \rightarrow [0.8,0.2]$$
$$h_3(\mathbf{x}) \rightarrow [0.4,0.6]$$

$$p(j = 0 \,|\, \mathbf{x}) = 0.2 \cdot 0.9 + 0.2 \cdot 0.8 + 0.6 \cdot 0.4 = 0.58$$

$$p(j = 1 \,|\, \mathbf{x}) = 0.2 \cdot 0.1 + 0.2 \cdot 0.2 + 0.6 \cdot 0.6 = 0.42$$

$$\hat{y} = \arg\max_j \left\{ p(j = 0 \,|\, \mathbf{x}),\ p(j = 1 \,|\, \mathbf{x}) \right\}$$


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

Bagging

(Bootstrap Aggregating)

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Bagging

(Bootstrap Aggregating)

Algorithm 1 Bagging

1: Let n be the number of bootstrap samples
2:
3: **for** i=1 to n **do**
4: Draw bootstrap sample of size m, D_i
5: Train base classifier h_i on D_i
6: $\hat{y}=mode\{h_1(\mathbf{x}), \ldots, h_n(\mathbf{x})\}$

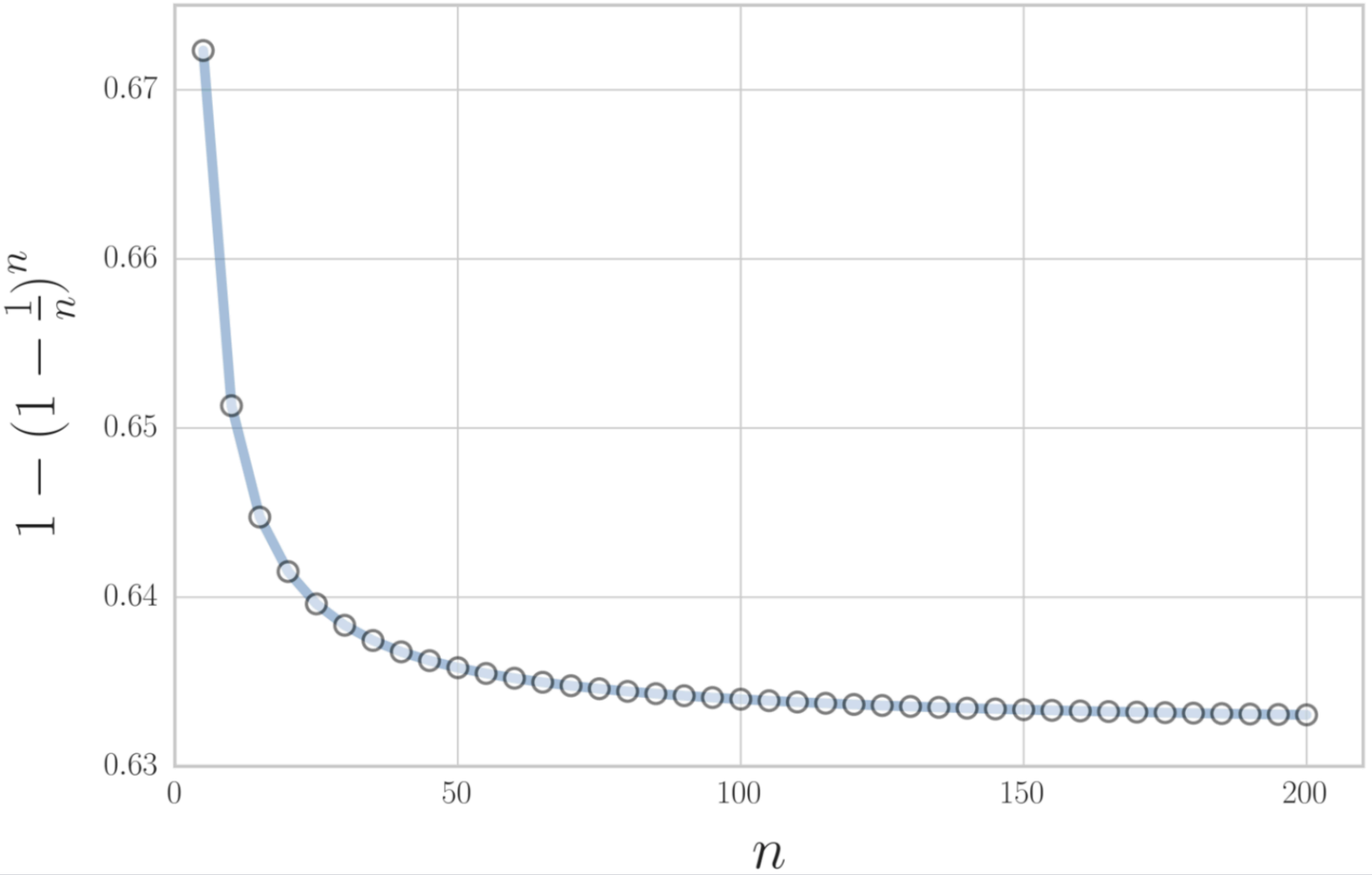
Bootstrap Sampling

Bootstrap Sampling

假设数据是均匀分布，n 个训练数据中的一条数据在 n 次抽样过程都没有被选中的概率:

$$P(\textbf{not chosen}) = \left(1 - \frac{1}{n}\right)^n,$$

$$\frac{1}{e} \approx 0.368, \quad n \to \infty .$$



证明

$$P(\textbf{not chosen}) = \left(1 - \frac{1}{n}\right)^n,$$

$$\frac{1}{e} \approx 0.368, \quad n \to \infty .$$

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

$$\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^x, x = -n$$

证明. 由于 $\lim_{x\to\infty}(1+1/x)^x = \lim_{y\to0}(1+y)^{1/y}$，

考虑极限 $\lim_{y\to0}\ln(1+y)/y$，使用L'hopital法则

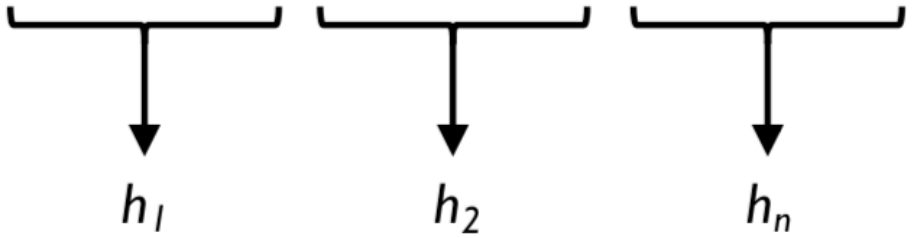
$$\lim_{y\to0}\frac{\ln(1+y)}{y} = \lim_{y\to0}\frac{1}{1+y} = 1$$

在根据指数 $\exp(x)$ 的连续性，可得

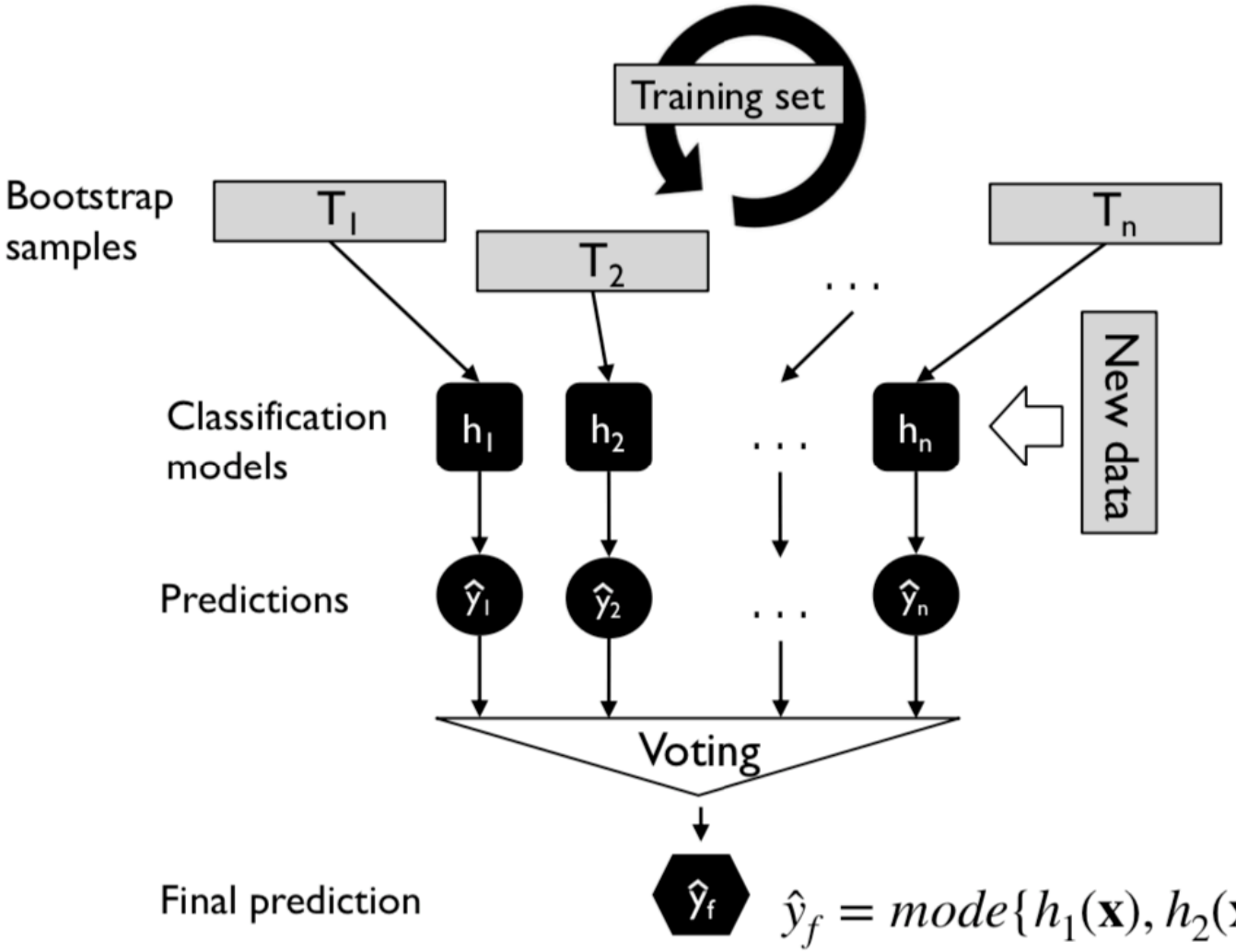
$$\lim_{x\to\infty}(1+1/x)^x = \lim_{y\to0}(1+y)^{1/y} = e^1 = e$$

Bootstrap Sampling

Training example indices	Bagging round 1	Bagging round 2	…
1	2	7	…
2	2	3	…
3	1	2	…
4	3	1	…
5	7	1	…
6	2	7	…
7	4	7	…



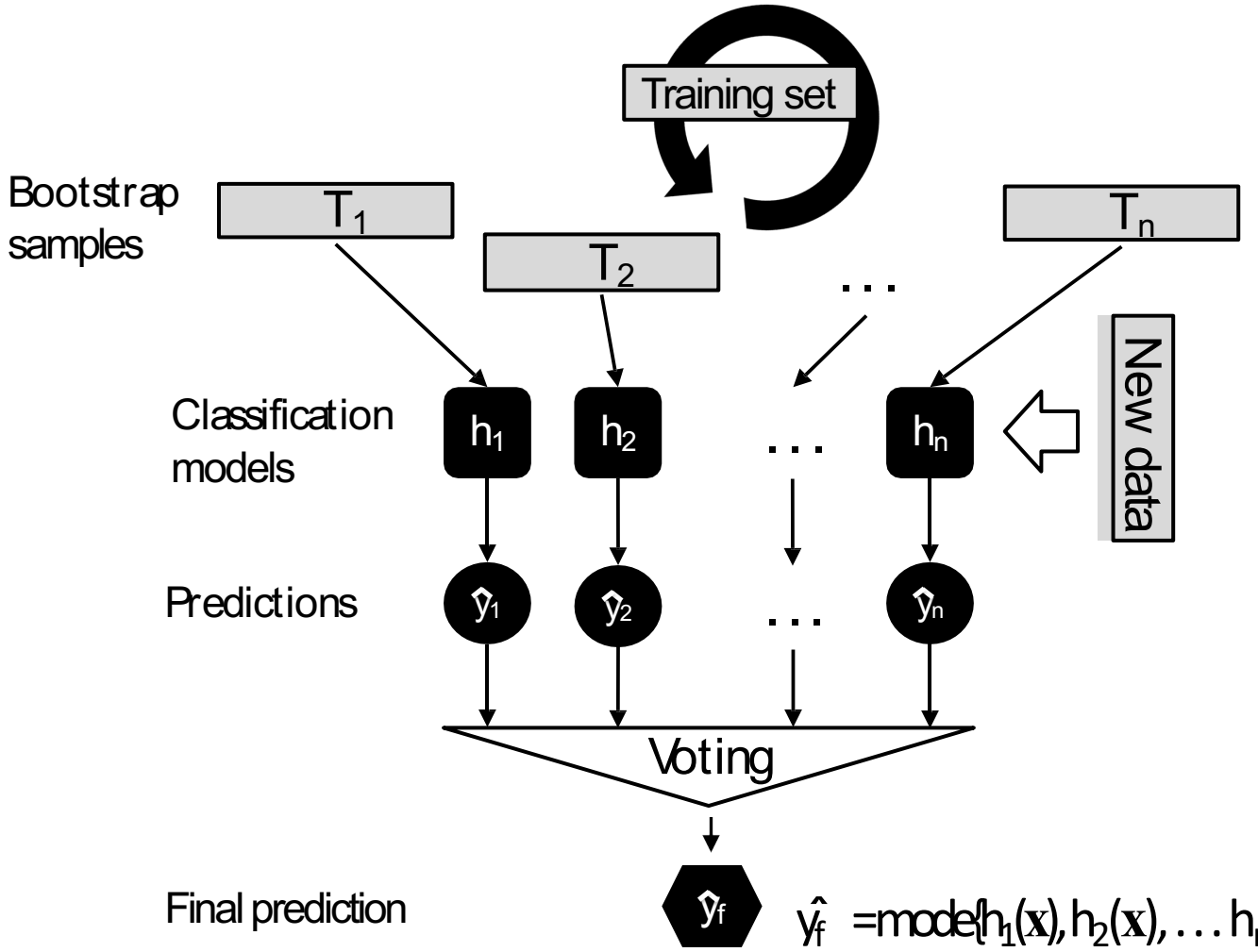
Bagging Classifier



$$\hat{y}_f = mode\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots h_n(\mathbf{x})\}$$

where $h_i(\mathbf{x}) = \hat{y}_i$

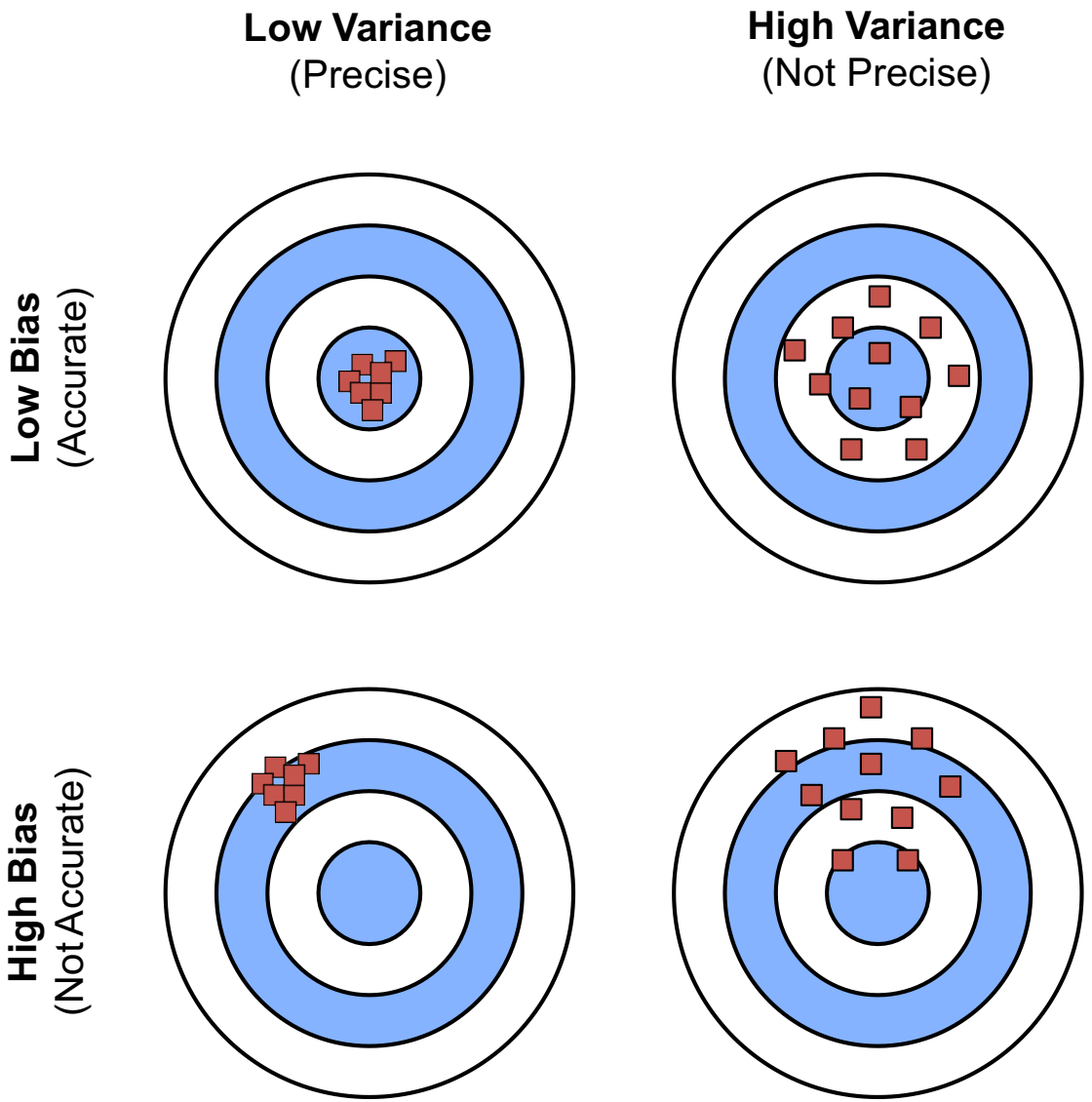
Bagging Classifier



Bias-Variance 分解

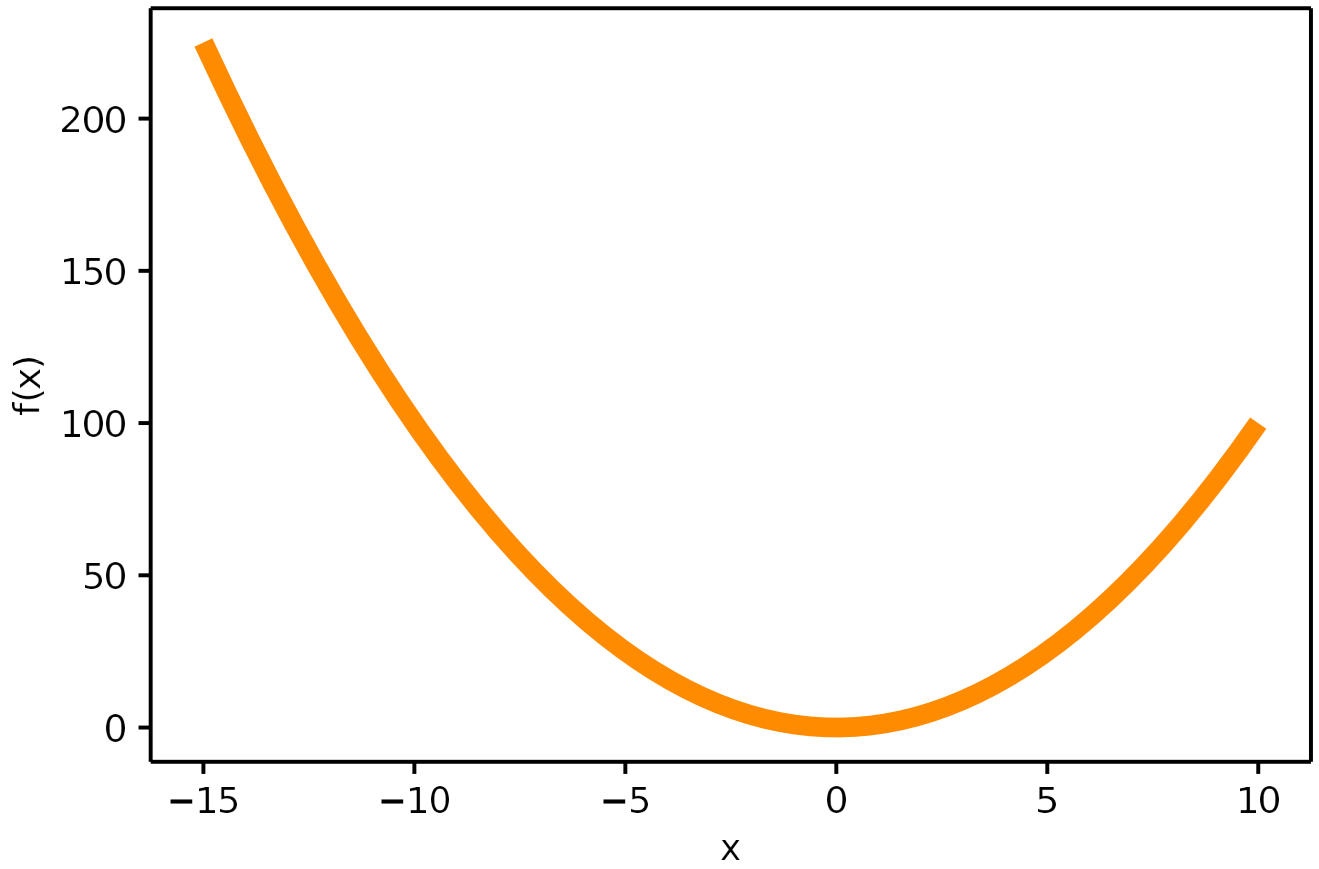
$$\text{Loss} = \text{Bias} + \text{Variance} + \text{Noise}$$

Bias-Variance 直觉



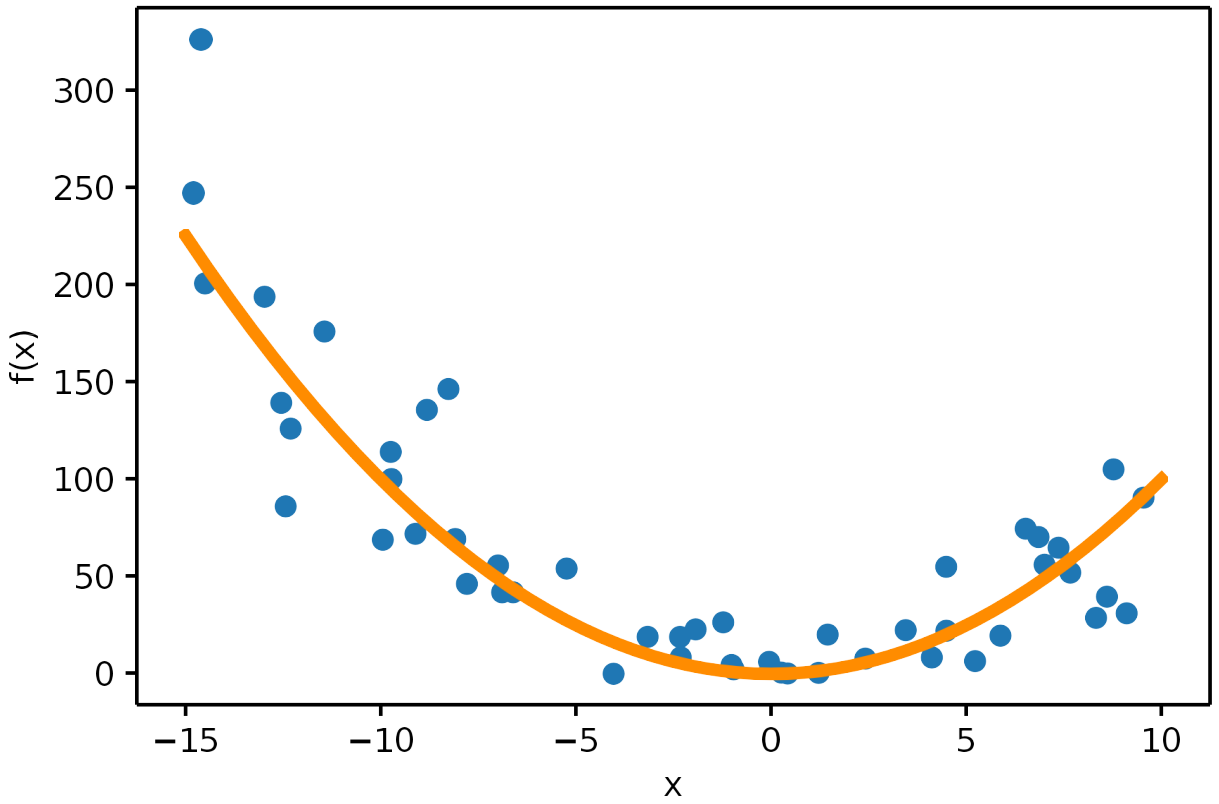
Bias and Variance

Example



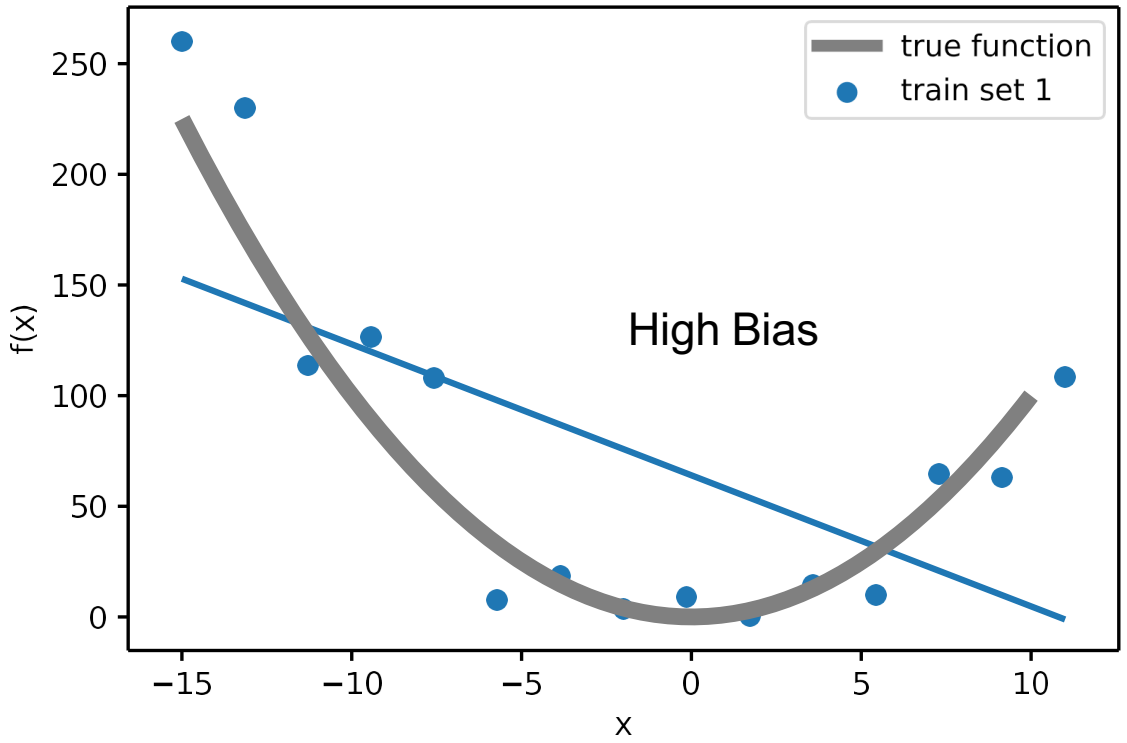
f(x) 为真实目标函数

Bias and Variance Example



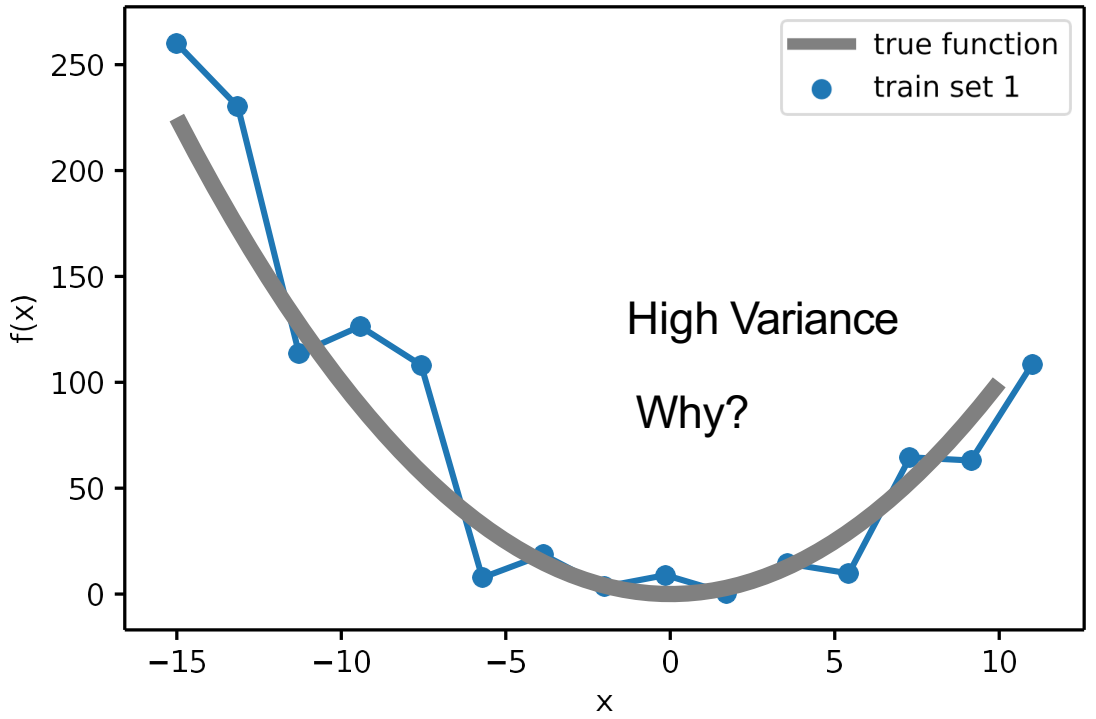
蓝色的数据为训练数据(包含高斯噪声)

Bias and Variance Example



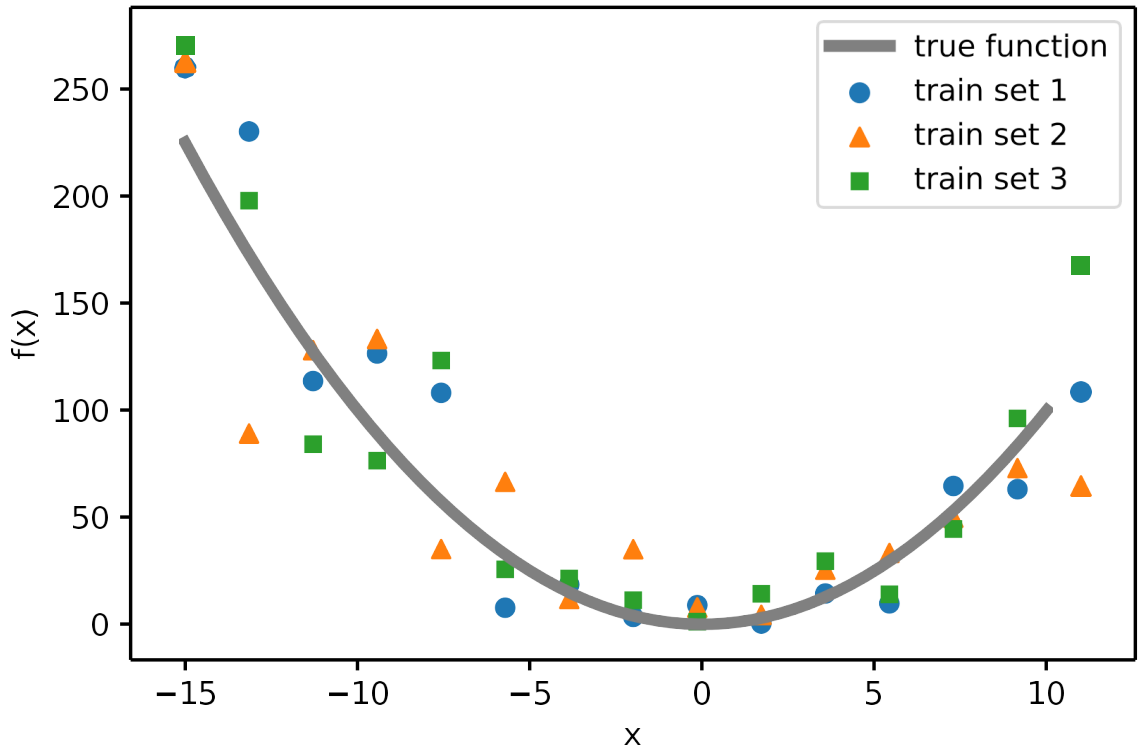
如果使用简单模型(线性回归)来拟合数据,
会造成较大Bias

Bias and Variance Example



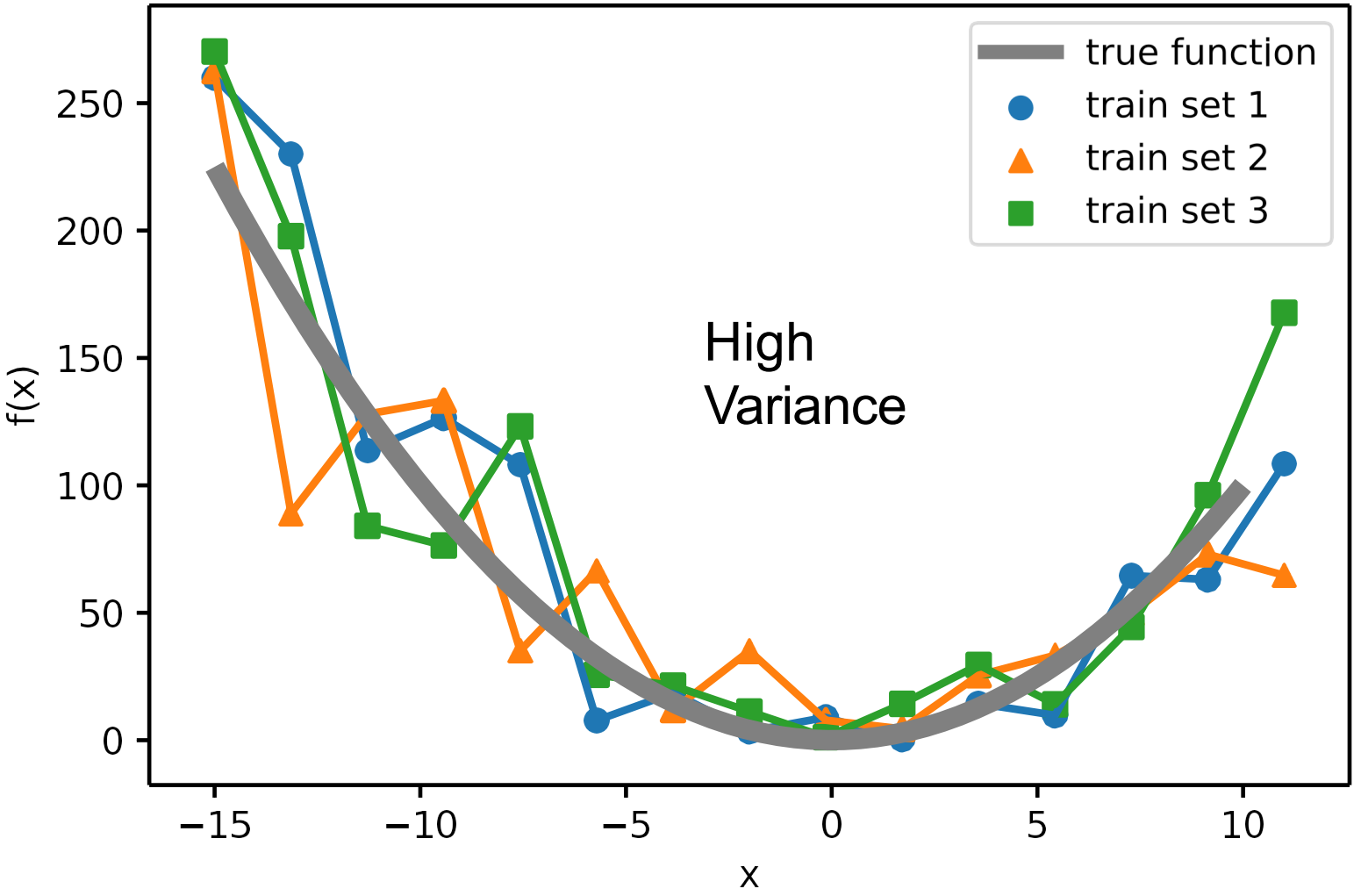
如果使用复杂模型(决策树)来拟合数据, 会造成较大Variance

Bias and Variance Example

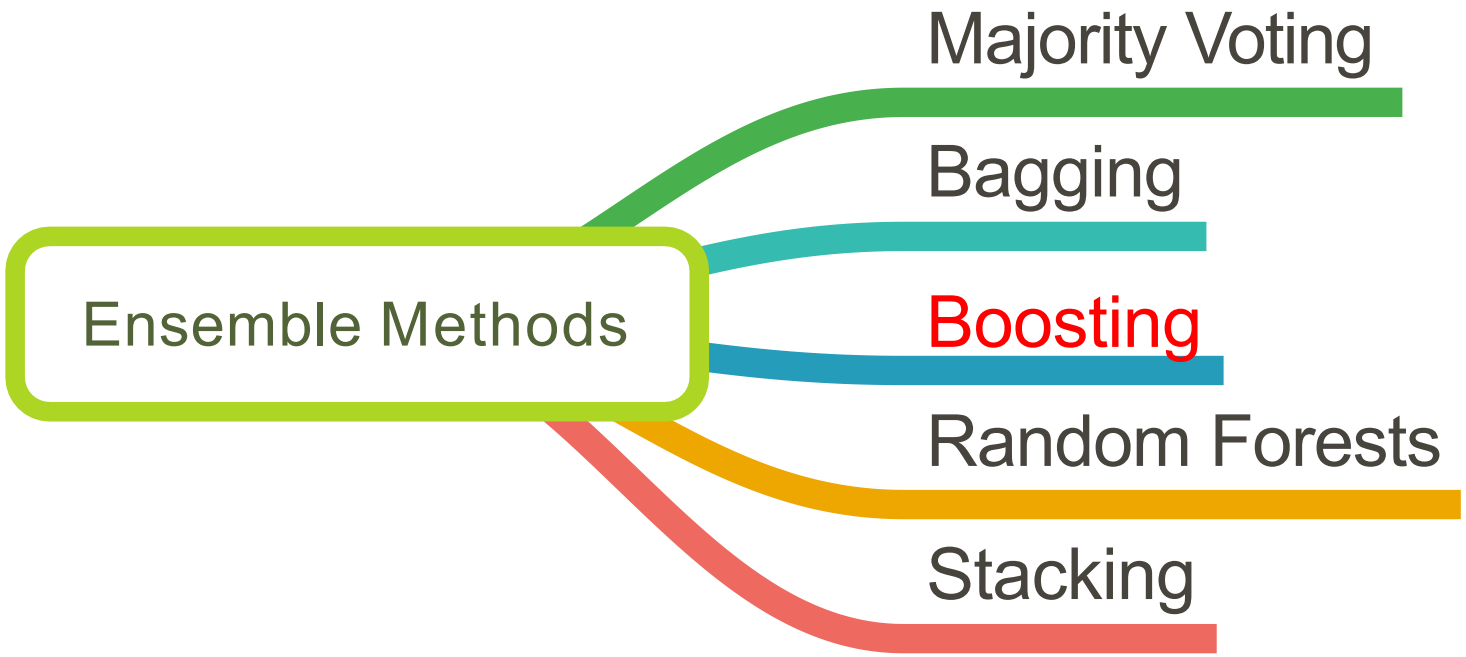


假设我们有多组不相关的训练数据

Bias and Variance Example



假设我们有多组不相关的训练数据. 为每一组训练数据训练一个复杂模型, 集成多个模型的结果, 相互抵消, 可以减小 Variance.


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

Boosting

Adaptive Boosting

e.g., AdaBoost

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting.
Journal of computer and system sciences, *55*(1), 119-139.

Gradient Boosting

e.g., LightGBM, XGBoost, scikit-learn's GradientBoostingClassifier

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine.
Annals of statistics, 1189-1232.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 785-794). ACM.

Adaptive Boosting V.S. Gradient Boosting

e.g., AdaBoost

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, *55*(1), 119-139.

主要区别在于
- 如何训练每一个模型
- 如何集成模型

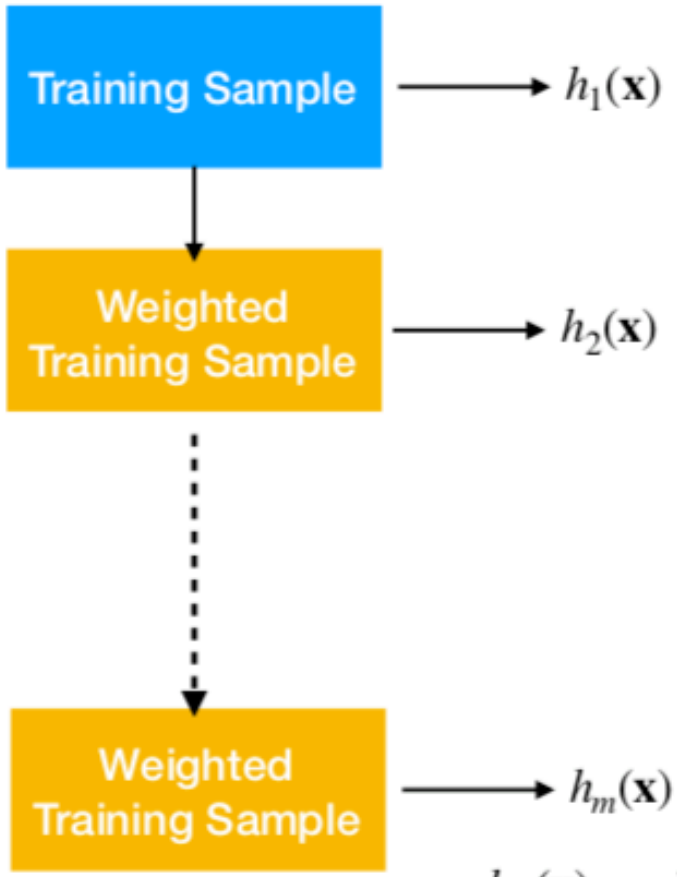
Gradient Boosting

e.g., LightGBM, XGBoost, scikit-learn's GradientBoostingClassifier

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189-1232.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 785-794). ACM.

Boosting



$$h_m(\mathbf{x}) = sign\left(\sum_{j=1}^{m} w_j\, h_j(\mathbf{x})\right) \quad \text{for} \quad h(\mathbf{x}) \in \{-1,1\}$$

$$\text{or} \quad h_m(\mathbf{x}) = \arg\max_i \left(\sum_{j=1}^{m} w_j\, \mathbf{1}[h_j(\mathbf{x}) = i]\right) \text{ for } \quad h(\mathbf{x}) = i, \qquad i \in \{1,\ldots,n\}$$

Boosting

- 初始化一个权重向量, 里面的值都相等
- 循环:
  - 使用带权重的训练数据训练一个模型
  - 为预测错误的的训练数据增大权重
- 使用带权重的 majority voting

Adaboost

Given: $(x_1, y_1), \ldots, (x_N, y_N)$ where $x_i \in X$, $y_i \in \{-1, +1\}$
Initialize $D_1(i) = 1/N$.
For $t = 1, \ldots, T$:

- Train weak learner using training data weighted according to distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$.
- Measure "goodness" of h_t by its weighted error with respect to D_t:

$$\epsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] = \sum_{i : h_t(x_i) \neq y_i} D_t(i).$$

- Let $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \tag{1}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final classifier:

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

Adaboost 证明

- Adaboost训练的模型 H 在训练数据集上面的误差上限为

$$\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

$$\epsilon_t = \frac{1}{2} - \gamma_t$$

Adaboost 证明（1）

- 证明 $D_{T+1}(i) = \frac{1}{N} \cdot \frac{\exp\left(-y_i f(x_i)\right)}{\prod_t Z_t}$
- 其中 $f(x) = \sum_t \alpha_t h_t(x)$
- 证明过程: 因为 y_i和$h_t(x_i)$值都为$\{-1, +1\}$, 可以改写公式(1)为如下形式
- $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases} \qquad D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$
- 按照递归 $$D_{T+1}(i) = D_1(i) \cdot \frac{\exp\left(-\alpha_1 y_i h_1(x_i)\right)}{Z_1} \cdot \ldots \cdot \frac{\exp\left(-\alpha_T y_i h_T(x_i)\right)}{Z_T}$$

$$= \frac{1}{N} \cdot \frac{\exp\left(-y_i \sum_t \alpha_t h_t(x_i)\right)}{\prod_t Z_t}$$

$$= \frac{1}{N} \cdot \frac{\exp\left(-y_i f(x_i)\right)}{\prod_t Z_t}.$$

Adaboost 证明（2）

- 证明最终 H 的训练误差为 $\prod_{t=1}^{T} Z_t$

$$
\begin{aligned}
\text{training error}(H) &= \frac{1}{N}\sum_i \begin{cases} 1 & \text{if } y_i \neq H(x_i) \\ 0 & \text{else} \end{cases} \\
&= \frac{1}{N}\sum_i \begin{cases} 1 & \text{if } y_i f(x_i) \leq 0 \\ 0 & \text{else} \end{cases} \\
&\leq \frac{1}{N}\sum_i \exp(-y_i f(x_i)) \qquad \text{根据 } e^{-z} \geq 1 \text{ if } z \leq 0 \\
&= \sum_i D_{T+1}(i) \prod_t Z_t \qquad \text{根据上一步的证明} \\
&= \prod_t Z_t
\end{aligned}
$$

$$
D_{T+1}(i) = \frac{1}{N} \cdot \frac{\exp\left(-y_i f(x_i)\right)}{\prod_t Z_t}
$$

Adaboost 证明（3）

- 计算 Z_t

$$
\begin{aligned}
Z_t &= \sum_i D_t(i) \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\
&= \sum_{i:h_t(x_i)=y_i} D_t(i)e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} D_t(i)e^{\alpha_t} \\
&= e^{-\alpha_t} \sum_{i:h_t(x_i)=y_i} D_t(i) + e^{\alpha_t} \sum_{i:h_t(x_i)\neq y_i} D_t(i) \\
&= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \qquad \text{根据 } \epsilon_t \text{的定义} \\
&= 2\sqrt{\epsilon_t(1-\epsilon_t)} \qquad \text{根据 } \alpha_t \text{选择的标准,是使得此表达式的值最小} \\
&= \sqrt{1-4\gamma_t^2} \qquad \text{根据 } \epsilon_t = \tfrac{1}{2} - \gamma_t \\
&\leq e^{-2\gamma_t^2}. \qquad \text{根据 } 1+x \leq e^x
\end{aligned}
$$

结合第2步的证明结果可得误差上限: $\exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$

$h_r: X \rightarrow \{-1, +1\}$

AdaBoost

$$\mathbf{1}\left(h_r(i) \neq y_i\right) = \begin{cases} 0 & \text{if } h_r(i) = y_i \\ 1 & \text{if } h_r(i) \neq y_i \end{cases}$$

Algorithm 1 AdaBoost

1: Initialize k: the number of AdaBoost rounds
2: Initialize $\mathcal{D}$: the training dataset, $\mathcal{D} = \{\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, ..., \mathbf{x}^{[n]}, y^{[n]} \rangle\}$
3: Initialize $w_1(i) = 1/n, \quad i = 1, ..., n, \ \mathbf{w}_1 \in \mathbb{R}^n$
4:
5: **for** r=1 to k **do**
6: For all i : $\mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$ [normalize weights]
7: $h_r := FitWeakLearner(\mathcal{D}, \mathbf{w}_r)$
8: $\epsilon_r := \sum_i w_r(i)\ \mathbf{1}(h_r(i) \neq y_i)$ [compute error]
9: if $\epsilon_r > 1/2$ then stop
10: $\alpha_r := \frac{1}{2} \log[(1 - \epsilon_r)/\epsilon_r]$ [small if error is large and vice versa]
11: $w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$
12: Predict: $h_{ens}(\mathbf{x}) = \arg\max_j \sum_r^k \alpha_r \mathbf{1}[h_r(\mathbf{x}) = j]$
13:

不同模型的权重

训练数据的权重

Decision Tree Stumps

decision tree stump 为一个二元分类的弱分类模型

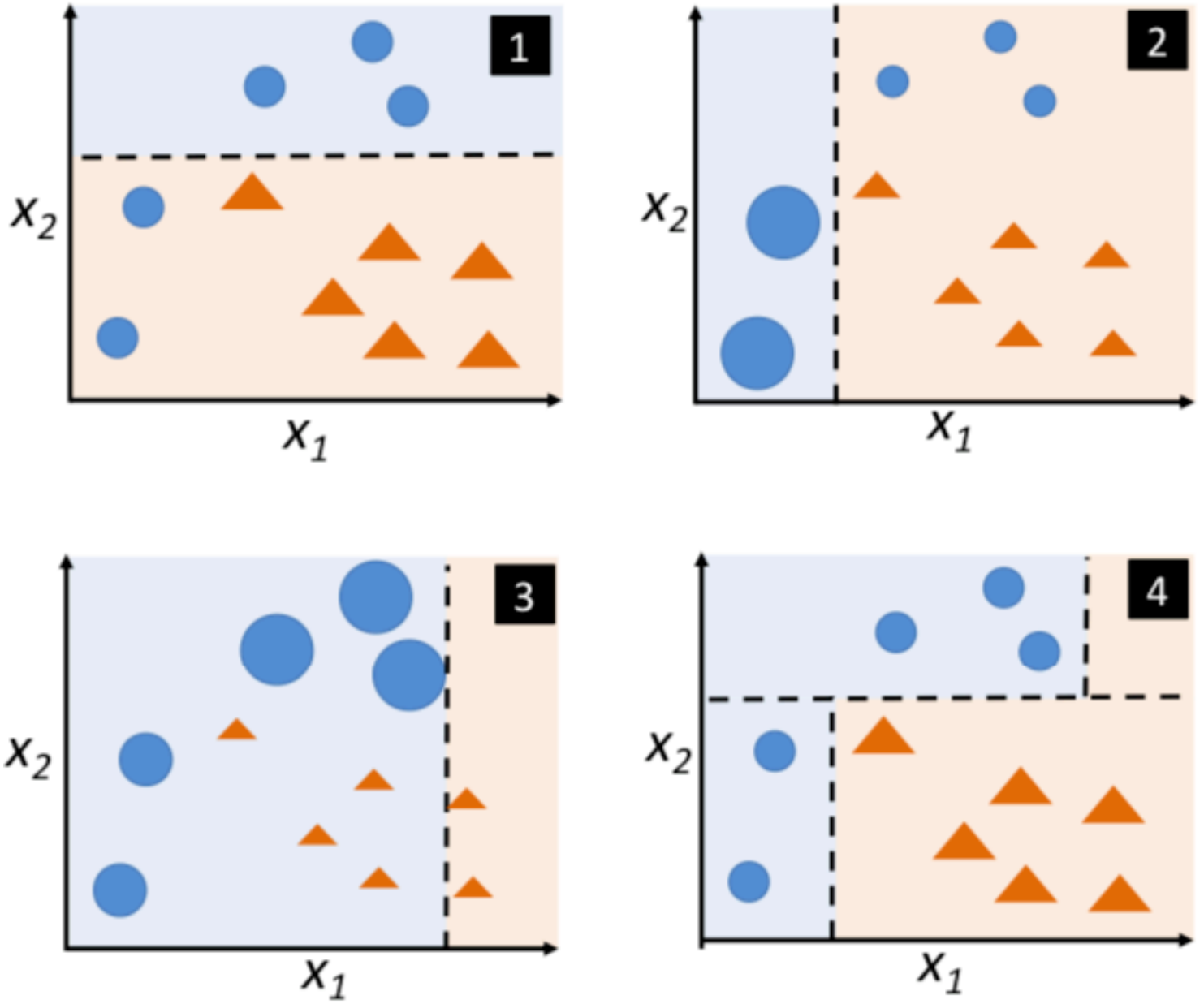
$$h(\mathbf{x}) = s\big(\mathbf{1}(x_k \geq t)\big)$$

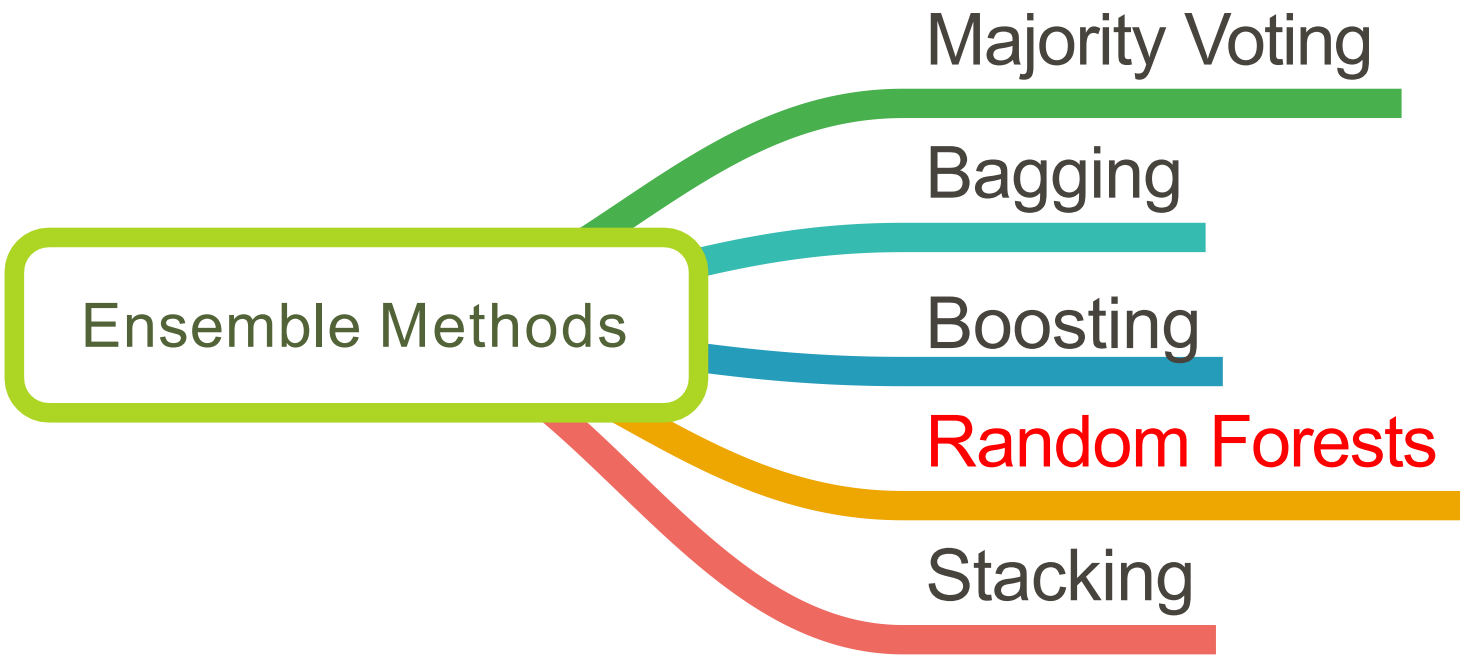
where

$s(x) \in \{-1, 1\}$

$k \in \{1, \ldots, K\}$ (K is the number of features)

AdaBoost + Decision Tree Stumps




Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

Random Forests

Random Forests

= Bagging w. trees + random feature subsets

为每一颗树，还是每一个树的节点随机选择特征

Tin Kam Ho used the "**random subspace method,**" where each tree got a random subset of features. 为每一颗树

"*Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space …*"

- Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

"Trademark" random forest:

"… random forest with random features is formed by selecting at random, at each node, a small group of input variables to split on." 为每一个树的节点

- **Breiman**, Leo. "Random Forests" Machine learning 45.1 (2001): 5-32.

为每一颗树，还是每一个树的节点随机选择特征

Tin Kam Ho used the "**random subspace method,**" where each tree got a random subset of features. 为每一颗树

"*Our method relies on an autonomous, pseudo-random procedure to select a small number of dimensions from a given feature space …*"

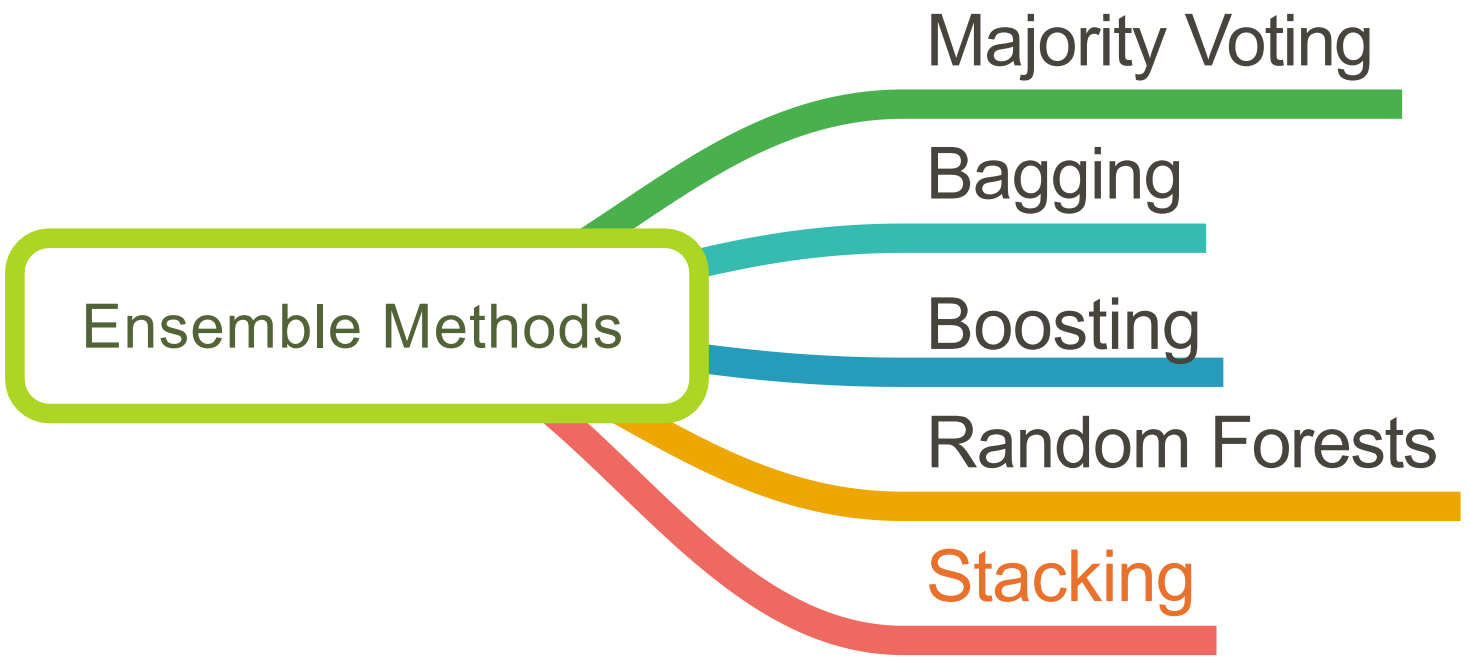
- Ho, Tin Kam. "The random subspace method for constructing decision forests." IEEE transactions on pattern analysis and machine intelligence 20.8 (1998): 832-844.

"Trademark" random forest:

"… random forest with random features is formed by selecting at random, at each node, a small group of input variables to split on." 为每一个树的节点

- **Breiman**, Leo. "Random Forests" Machine learning 45.1 (2001): 5-32.

Overview


Ensemble Methods
Majority Voting
Bagging
Boosting
Random Forests
Stacking

Stacking

Stacking Algorithm

Wolpert, David H. "Stacked generalization." Neural networks 5.2 (1992): 241-259.

Algorithm 19.7 Stacking

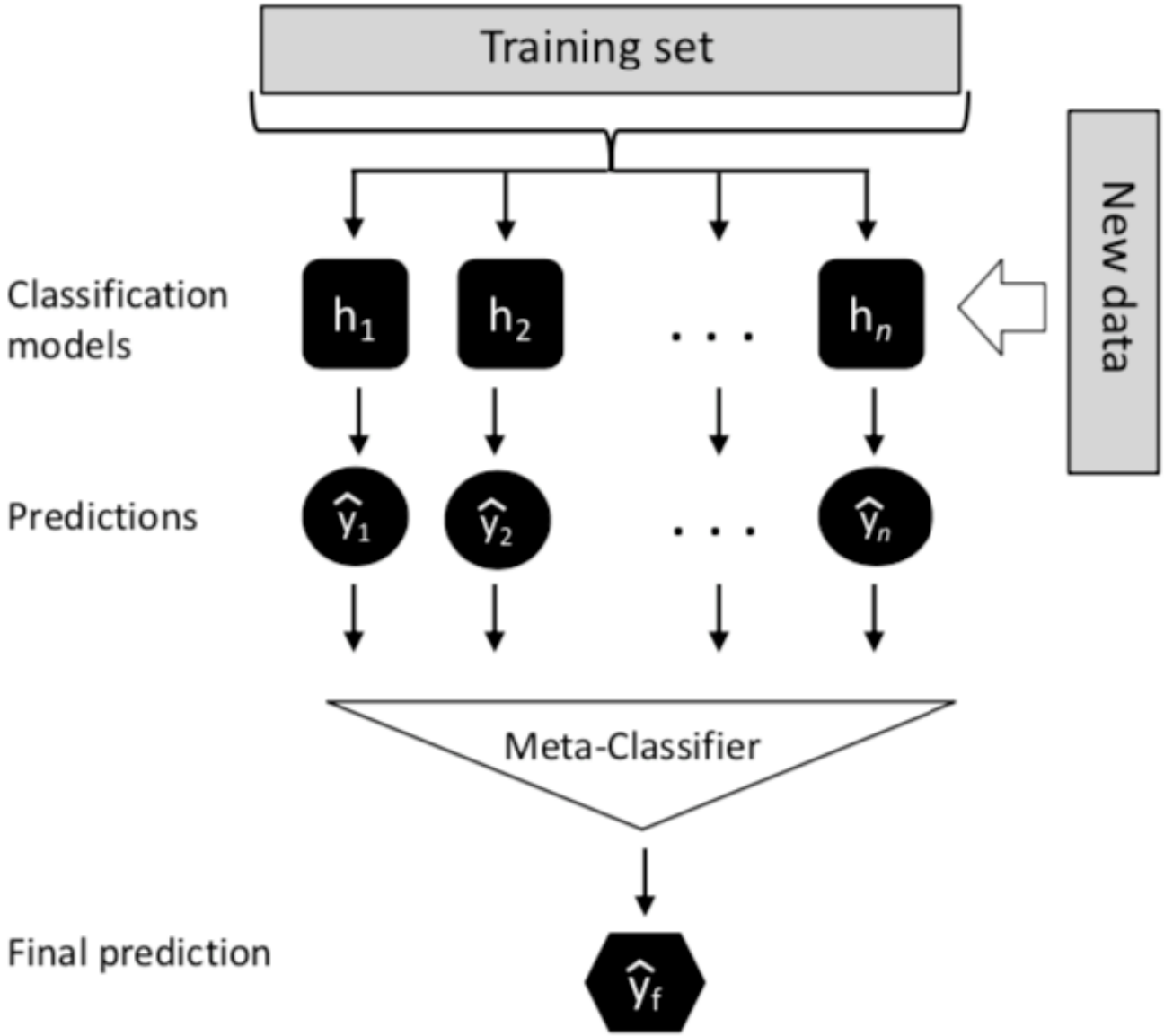
Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ $(\mathbf{x}_i \in \mathbb{R}^n, y_i \in \mathcal{Y})$

Output: An ensemble classifier H

1: Step 1: Learn first-level classifiers
2: **for** $t \leftarrow 1$ to T **do**
3: Learn a base classifier h_t based on $\mathcal{D}$
4: **end for**
5: Step 2: Construct new data sets from $\mathcal{D}$
6: **for** $i \leftarrow 1$ to m **do**
7: Construct a new data set that contains $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \ldots, h_T(\mathbf{x}_i)\}$
8: **end for**
9: Step 3: Learn a second-level classifier
10: Learn a new classifier h' based on the newly constructed data set
11: **return** $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

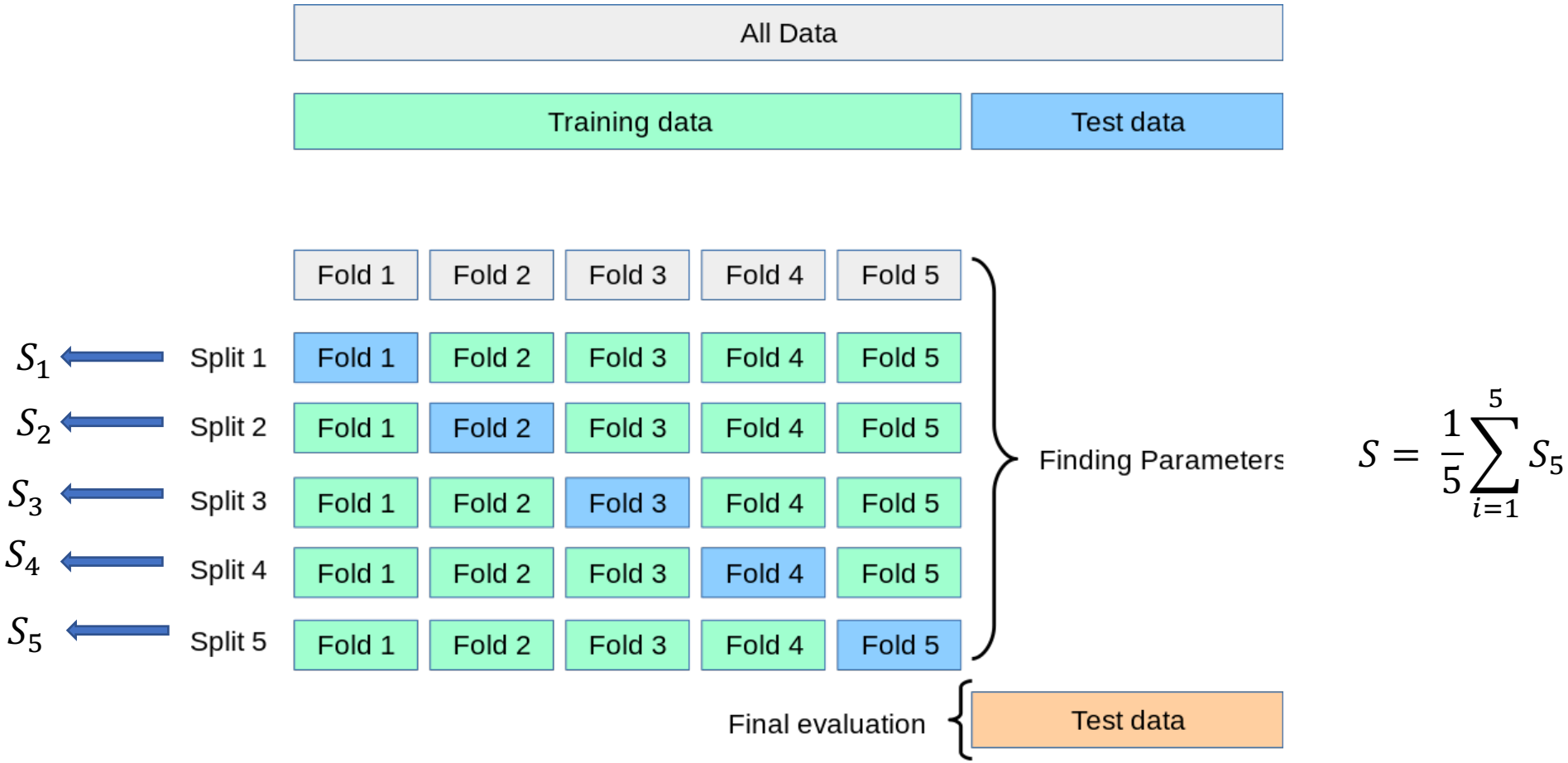
Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.

Stacking Algorithm



缺点: 容易过拟合
如果第一层的模型已经过拟合了，
第二层的模型基于过拟合的数据

使用交叉验证（Cross Validation）



$$S = \frac{1}{5}\sum_{i=1}^{5} S_5$$

Stacking Algorithm with Cross-Validation

Wolpert, David H. "Stacked generalization." Neural networks 5.2 (1992): 241-259.

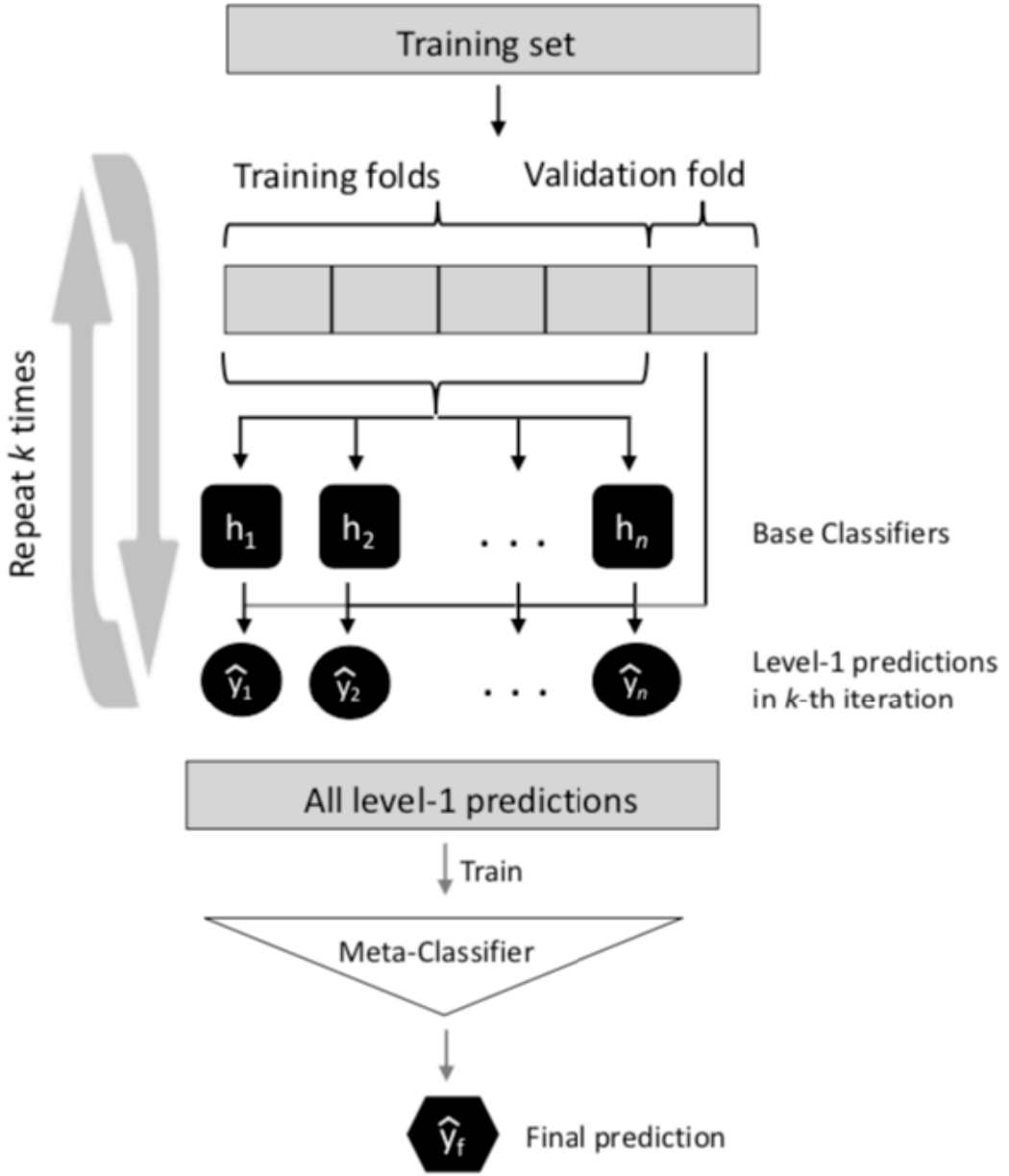
Algorithm 19.8 Stacking with K-fold Cross Validation

Input: Training data $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^{m}$ ($\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$)
Output: An ensemble classifier H

Step 1: Adopt cross validation approach in preparing a training set for second-level classifier
Randomly split $\mathcal{D}$ into K equal-size subsets: $\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K\}$
for $k \leftarrow 1$ to K **do**
    Step 1.1: Learn first-level classifiers
    for $t \leftarrow 1$ to T **do**
        Learn a classifier h_{kt} from $\mathcal{D} \setminus \mathcal{D}_k$
    end for
    Step 1.2: Construct a training set for second-level classifier
    for $\mathbf{x}_i \in \mathcal{D}_k$ **do**
        Get a record $\{\mathbf{x}'_i, y_i\}$, where $\mathbf{x}'_i = \{h_{k1}(\mathbf{x}_i), h_{k2}(\mathbf{x}_i), \ldots, h_{kT}(\mathbf{x}_i)\}$
    end for
end for
Step 2: Learn a second-level classifier
Learn a new classifier h' from the collection of $\{\mathbf{x}'_i, y_i\}$
Step 3: Re-learn first-level classifiers
for $t \leftarrow 1$ to T **do**
    Learn a classifier h_t based on $\mathcal{D}$
end for
return $H(\mathbf{x}) = h'(h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

Tang, J., S. Alelyani, and H. Liu. "Data Classification: Algorithms and Applications." Data Mining and Knowledge Discovery Series, CRC Press (2015): pp. 498-500.

Stacking Algorithm with Cross-Validation



贪心科技 GREEDY TECHNOLOGY

贪心科技　　让每个人享受个性化教育服务

THANKS

贪心学院讲师：袁源