西南财经大学
SOUTHWESTERN UNIVERSITY OF FINANCE AND ECONOMICS
1925 - 2021
经世济民 孜孜以求

机器学习基础

提升方法

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

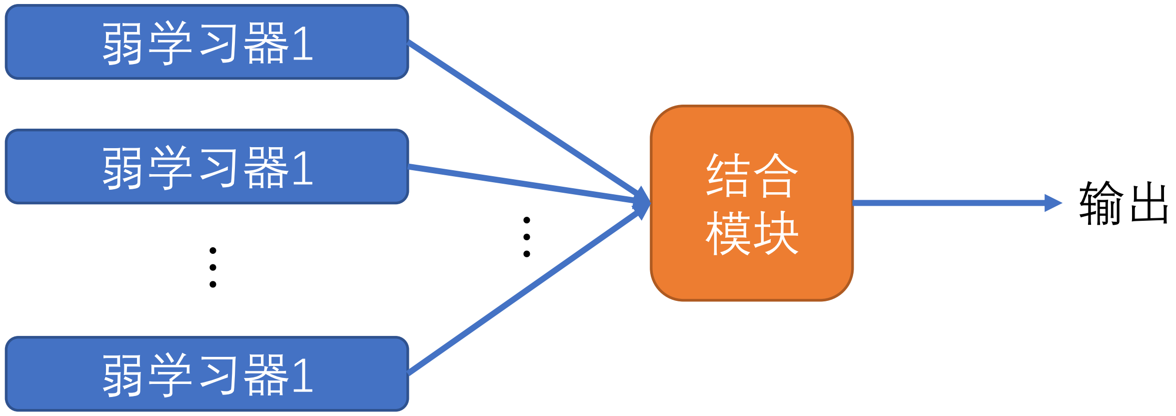
强可学习与弱可学习

- 强可学习：存在一个多项式学习算法能够学习，并且正确率高
- 弱可学习：存在一个多项式学习算法能够学习，正确率仅比随机猜测略好
- 强可学习与弱可学习是等价的
- 弱学习算法是容易发现的，那么如何得到对应的强学习算法?

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

集成学习

- 集成学习：通过构建多个弱学习器（基学习器）来完成学习任务，其一般结构为：



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

集成个体应“好而不同”

- 好：弱学习器要有一定的准确性，不能太坏
- 不同：弱学习器之间有差异，具有一定的多样性

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	×	√
h_3	×	√	√
集成	√	√	√

集成提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

集成不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

集成起负作用

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

集成学习的类型

- 套袋法（bagging）：基学习器是同质弱学习器，且独立并行学习这些弱学习器，再用平均过程组合将它们组合起来
- 提升法（boosting）：基学习器是同质弱学习器，学习器之间有相互依赖关系，再用确定性的策略将它们组合起来
- 堆叠法（stacking）：基学习器是异质弱学习器，并行训练这些学习器，并通过训练一个“元模型”将它们组合起来

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging

- 样本采集：利用自助法从训练样本中随机抽取 n 个样本，共抽取 k 轮，产生 k 组训练集
  - 训练样本是有放回的选取
  - k 组训练集相互之间独立
- 建立弱学习器：对每组训练集独立训练一个基学习器，得到 k 个模型
  - k 个弱学习器是同质的，即都是决策树、感知机等

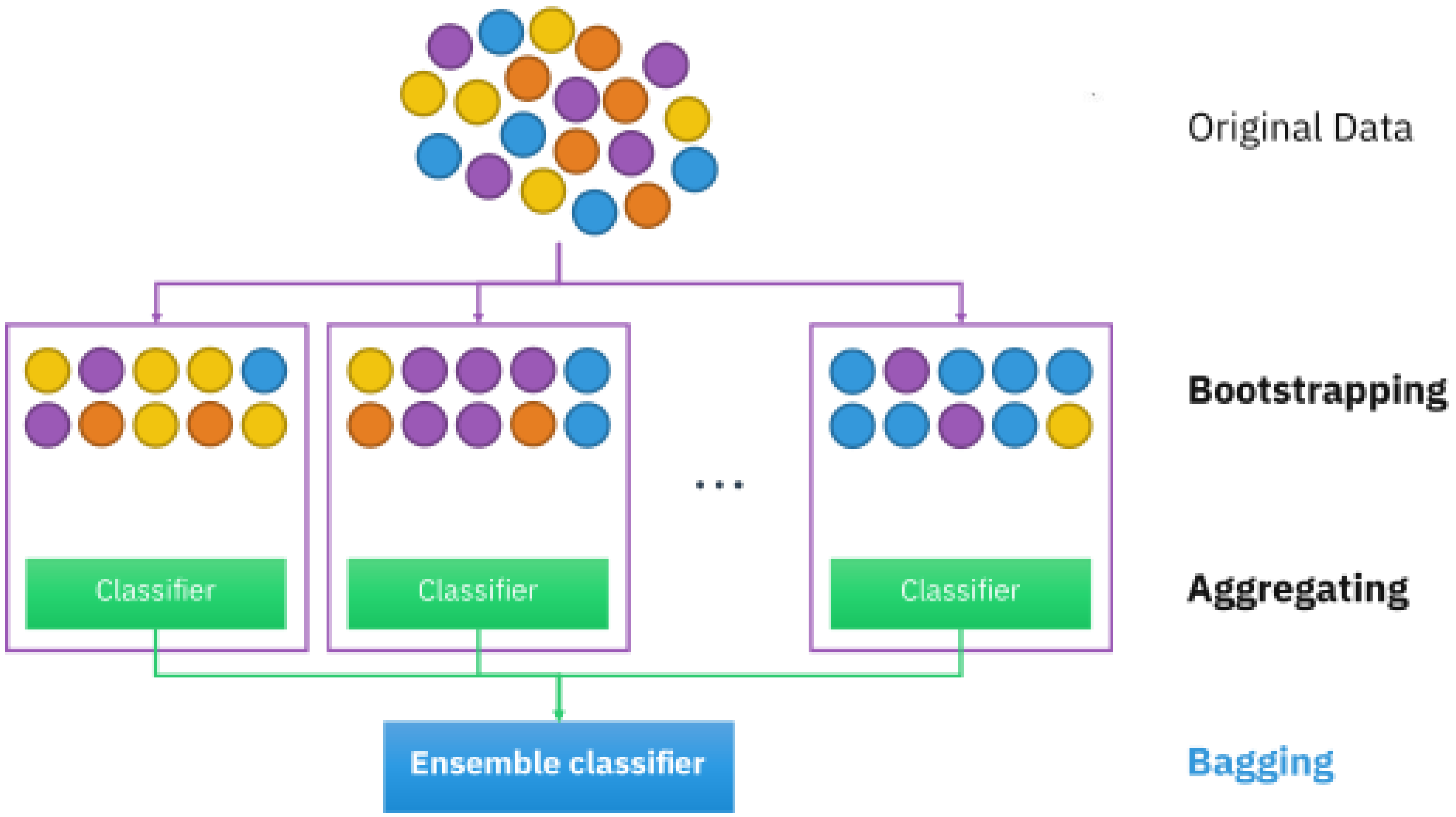
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging

- 学习器组合：
  - 分类问题：将 k 个模型的分类结果采用多数表决原则获得组合后的分类结果
  - 回归问题：将 k 个模型的预测结果的均值作为组合后的预测结果
- 代表算法：随机森林（RF）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging的优缺点

- 优点
  - 组合后的学习器比单个学习器表现好，不容易过拟合
    - 会去除掉高方差、低偏差的学习器
  - 基学习器能并行计算
- 缺点
  - 若基学习器是高偏差的，组合后的学习器也容易欠拟合
  - 模型缺乏解释性
  - 随着样本增多，计算复杂度也会显著增加

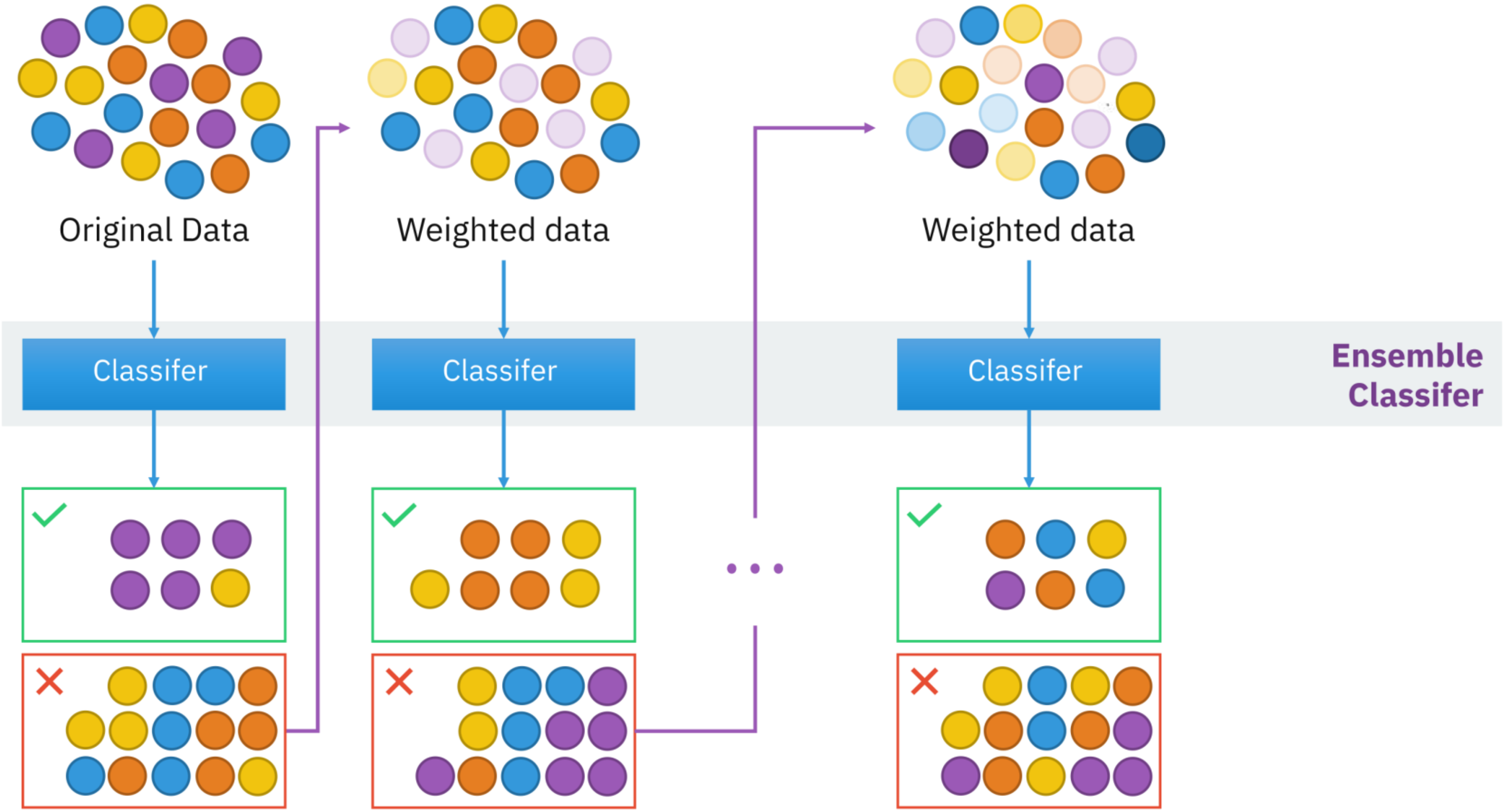
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Boosting

- 建立弱学习器：对所有训练样本进行 k 轮学习，每轮训练出一个弱学习器，一共得到 k 个弱学习器
  - k 个弱学习器是同质的，即都是决策树、感知机等
  - 每一轮训练都要基于前面轮次的弱学习器的学习结果，即弱学习器之间是迭代产生、相互依赖的
- 学习器组合：确定性的组合方式，例如加法模型
- 代表算法：Adaboost，GBDT，XGBT，Catboost

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Boosting



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Boosting的优缺点

- 优点
  - 表现相当出色，例如XGBT几乎横扫所有比赛
  - 模型解释性好
- 缺点
  - 对异常点敏感
    - 新一轮的训练会修补前面基分类器对异常点的错误
  - 训练过程长
    - 弱学习器迭代产生，必须要一个一个建立

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging与Boosting

	Bagging	Boosting
结构	并行	串行
训练集	独立	依赖
基学习器	同质	同质
作用	减小方差	减小偏差

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging不显著减小偏差

- 由于样本随机选取，各基学习器之间有相似的偏差和方差
- 假设X_i是第i个基学习器的预测结果
- 由于

$$\mathbb{E}\left(\frac{\sum X_i}{n}\right)=\frac{\sum \mathbb{E}\left(X_i\right)}{n} \doteq \mathbb{E}\left(X_i\right)$$

- 所以组合后的偏差与基学习器的偏差接近

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging减小方差

- 假设 X_i 相互之间完全独立，则

$$Var\left(\frac{\sum X_i}{n}\right)=\frac{Var\left(\sum X_i\right)}{n^2}=\frac{\sum Var\left(X_i\right)}{n^2}\doteq\frac{Var\left(X_i\right)}{n}$$

- 假设 X_i 相互之间完全相等，则 $Var\left(\frac{\sum X_i}{n}\right)=Var\left(X_i\right)$
- Bagging各基分类器之间有一定的相关性，介于以上两种情况之间因此可以一定程度降低方差

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Bagging的原理

- 尽量减小基分类器的偏差，提高预测能力
  - 组合后的模型与基分类器有近似的偏差
- 减小基分类器的偏差会增加其方差，利用求平均减小组合后模型的方差
- 因此组合模型有较小的偏差和方差

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Boosting显著减小偏差

- 每一轮训练的弱学习器会减小前一轮训练的弱学习器的损失函数，因此偏差会逐步下降
- 弱学习器之间有强相关性，组合后的模型不能显著降低方差
- 弱学习器不需要很强的学习能力，例如深度为1的决策树
  - 弱学习器的偏差大，方差小，组合后的方差相对也小
  - 弱学习器在迭代过程会显著减小偏差

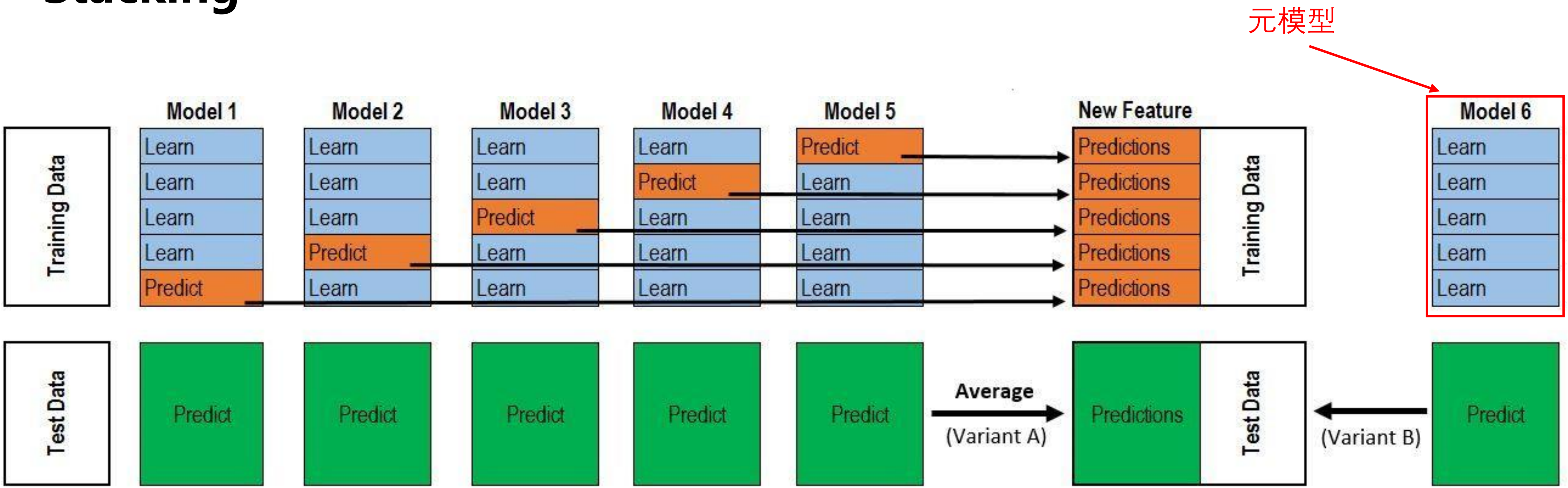
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Stacking的原理

- 建立弱学习器：对训练数据建立 k 个不同的弱学习器，例如弱学习器为kNN、SVM、逻辑斯蒂回归等
- 训练元模型来组合 k 个弱学习器，例如用神经网络作为元模型
  - k 个弱学习器的预测结果作为元模型的输入
  - 元模型的预测结果作为最终预测结果
- 没有成熟的代表算法，甚至Scikit-learn没有直接的Stacking 方法

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

Stacking



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法

- 数据集：$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathbb{R}^n$，$y_i\in\{-1,+1\}$
- 初始化训练样本的权值

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

  - 第一轮训练的样本权重相同，保证在原始数据集上训练

- 第 m 轮训练，$m=1,2,\cdots,M$

  a. 根据当前的样本权值 D_m 训练基分类器 $G_m(x)$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法

- 第m轮训练，$m=1,2,\cdots,M$

b. 计算$G_m(x)$在训练集上的分类错误率 $e_m=\sum_{i=1}^{N} w_{mi}\mathbb{I}(G_m(x_i)\neq y_i)$

- e_m是加权错误率，样本权重大对错误率影响大

c. 计算$G_m(x)$的系数$\alpha_m(x)=\frac{1}{2}\log\frac{1-e_m}{e_m}$

- $\alpha_m(x)$ 表示 $G_m(x)$ 在最终分类器中的重要性
- $e_m\geq\frac{1}{2}$时，$\alpha_m(x)\geq 0$； e_m越小， $\alpha_m(x)$越大， 即$G_m(x)$表现越好，它在最终组合中越重要

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法

- 第m轮训练，$m=1,2,\cdots,M$

  d. 更新训练样本的权值

$$D_{m+1}=(w_{m+1,1},\cdots,w_{m+1,i},\cdots,w_{m+1,N}),$$

$$w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)),\quad i=1,2,\cdots,N$$

其中$Z_m=\sum_{i=1}^{N}w_{mi}\exp(-\alpha_m y_i G_m(x_i))$是规范化因子，保证$D_{m+1}$成为一个概率分布

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法

- 第m轮训练，$m=1,2,\cdots,M$
  - d. 更新训练样本的权值 $w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i))$
    - 当第m轮训练的$G_m(x)$能正确预测x_i时，即$G_m(x_i)=y_i$，有

      $$w_{m+1,i}=\frac{w_{mi}}{Z_m}e^{-\alpha_m}<w_{mi}$$

      即预测正确的样本，权重会下降
    - 否则有$w_{m+1,i}=\frac{w_{mi}}{Z_m}e^{\alpha_m}>w_{mi}$，即预测错误的样本权重会增大，且$G_m(x)$的表现越好，增加的越多

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法

- 将 M 轮训练的所有基分类器组合起来

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

  - α_m 表示 $G_m(x)$ 的重要性；表现越好的$G_m(x)$，α_m 越大

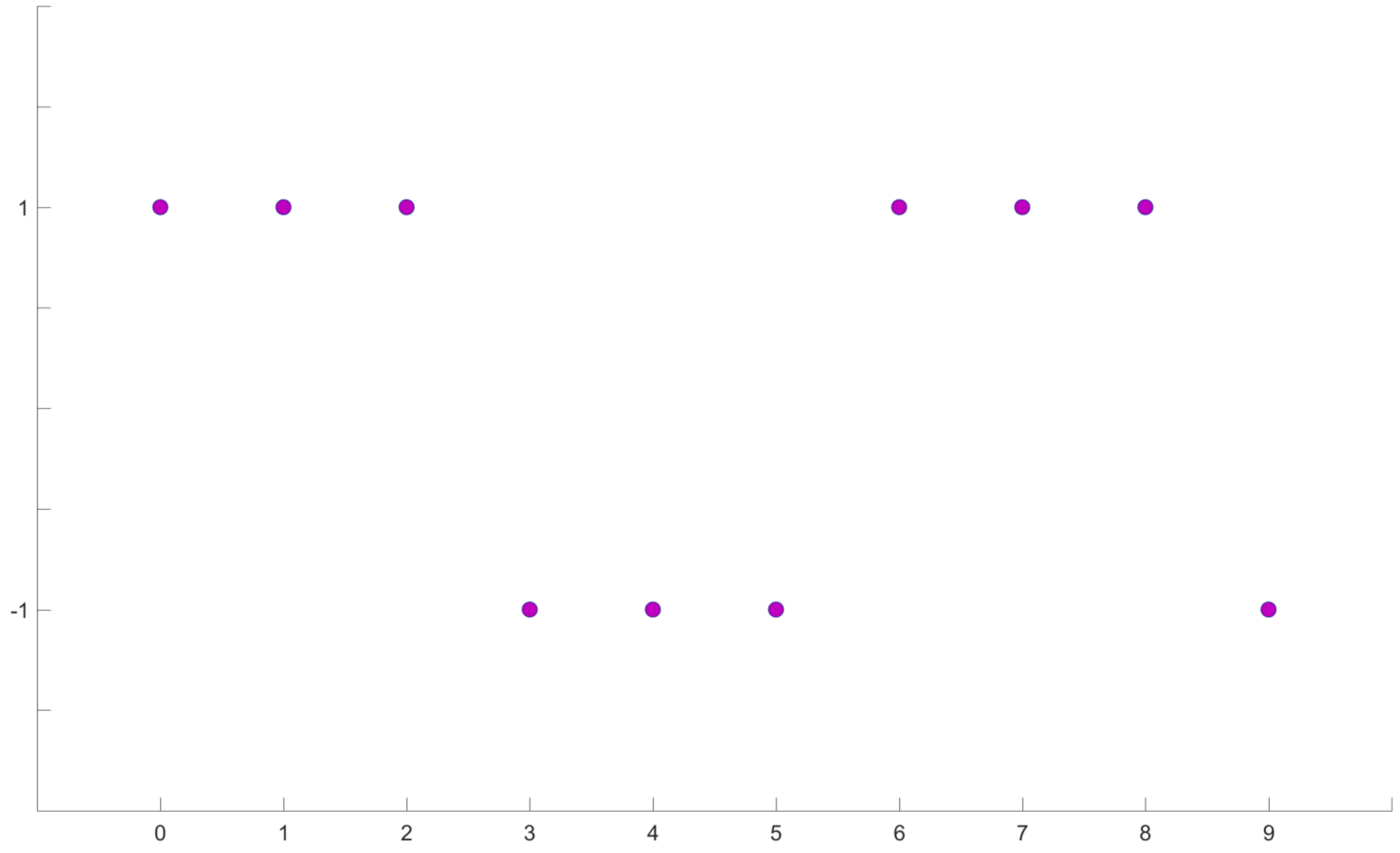
- 最终的分类决策器

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

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.1

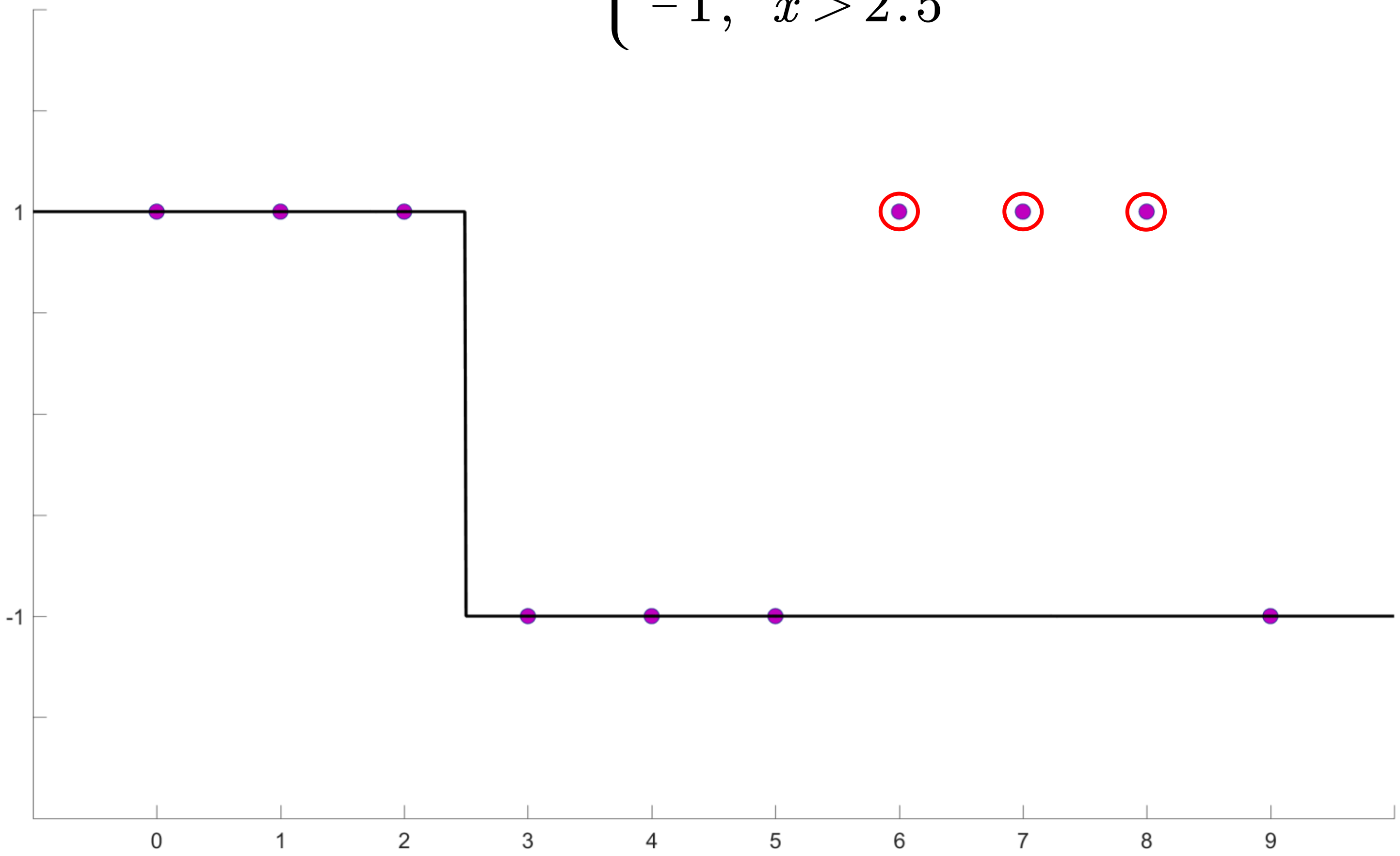
训练数据



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.1

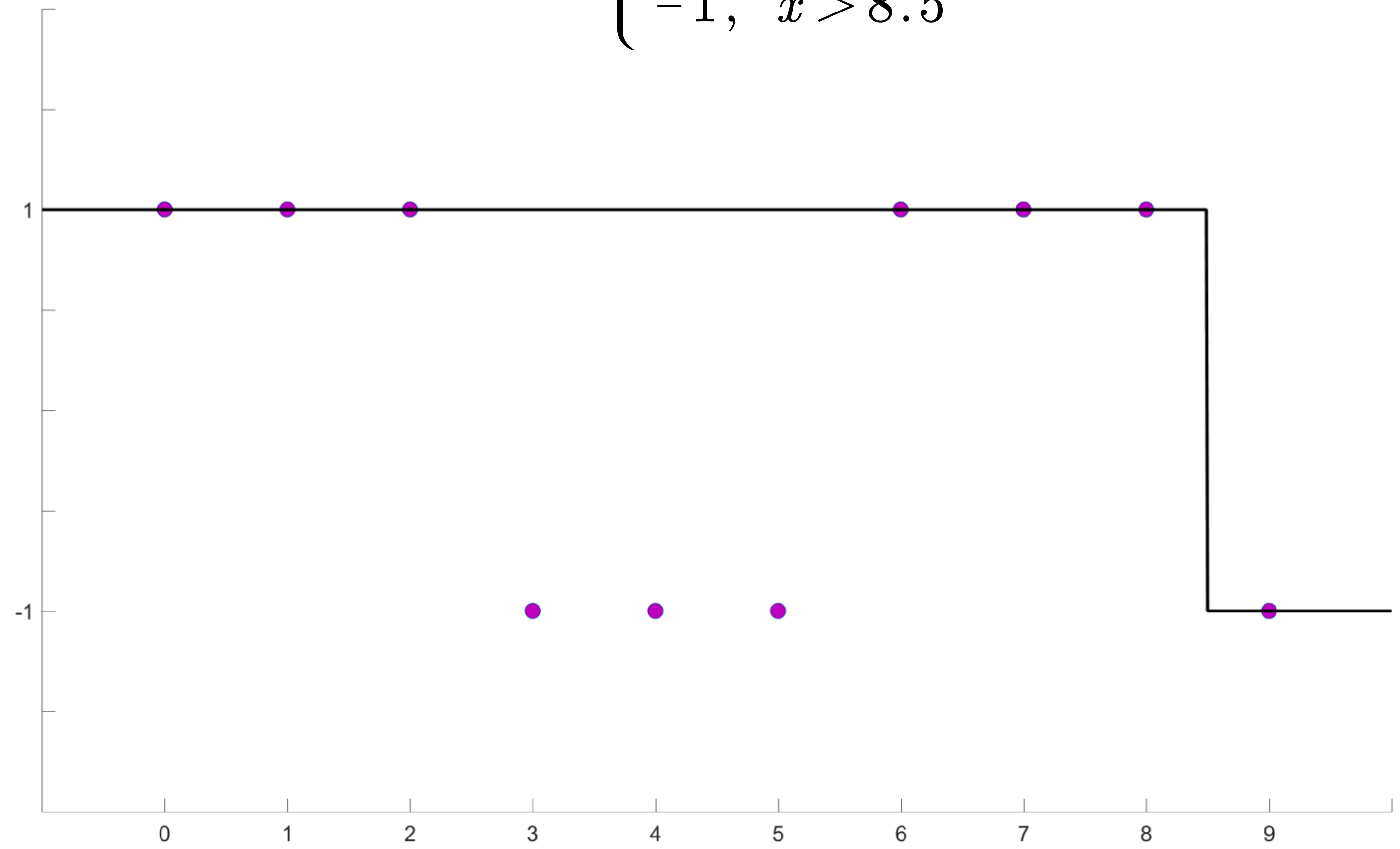
$$G_1(x)=\begin{cases}1, & x<2.5\\ -1, & x>2.5\end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.1

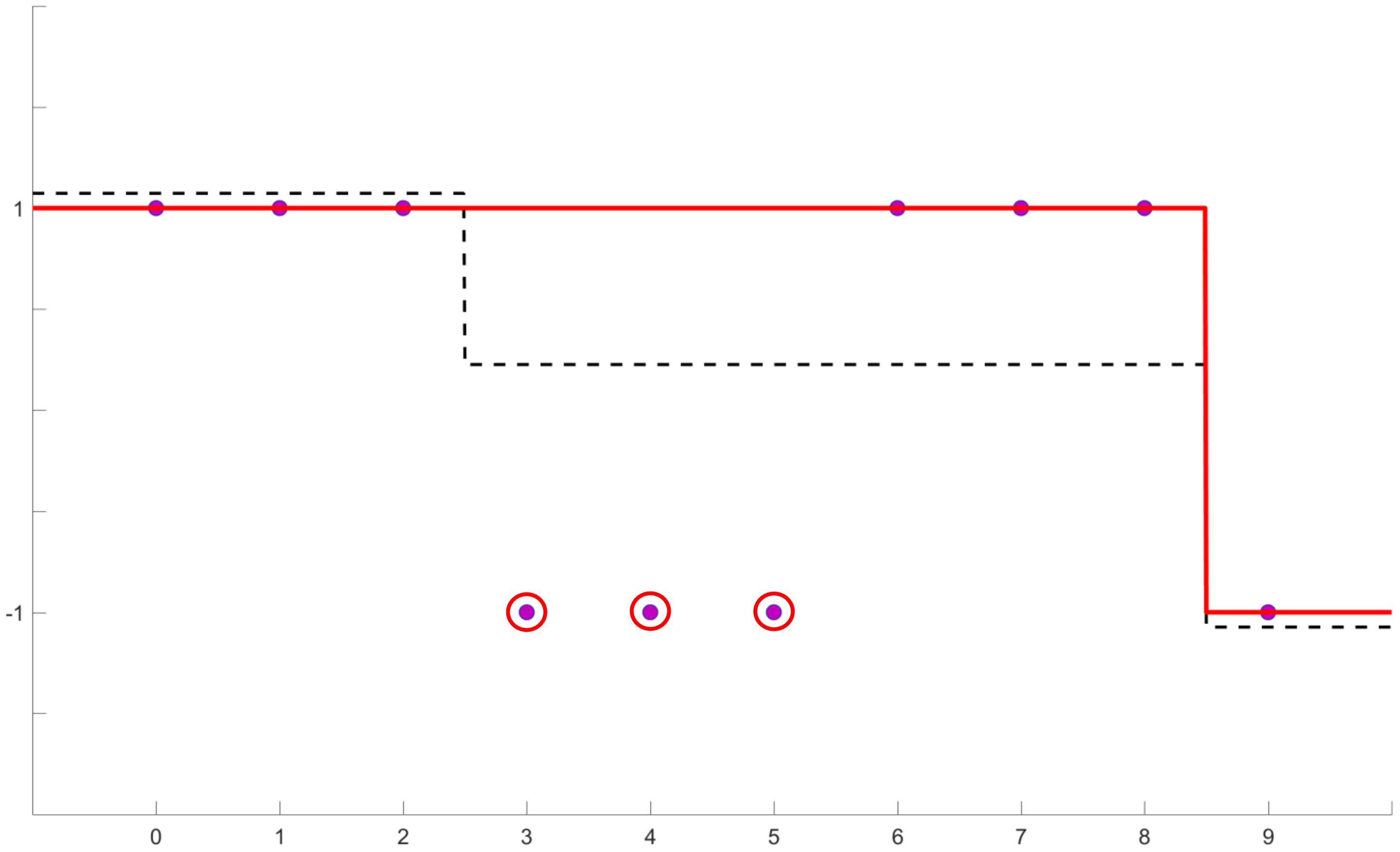
$$G_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x > 8.5 \end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.1

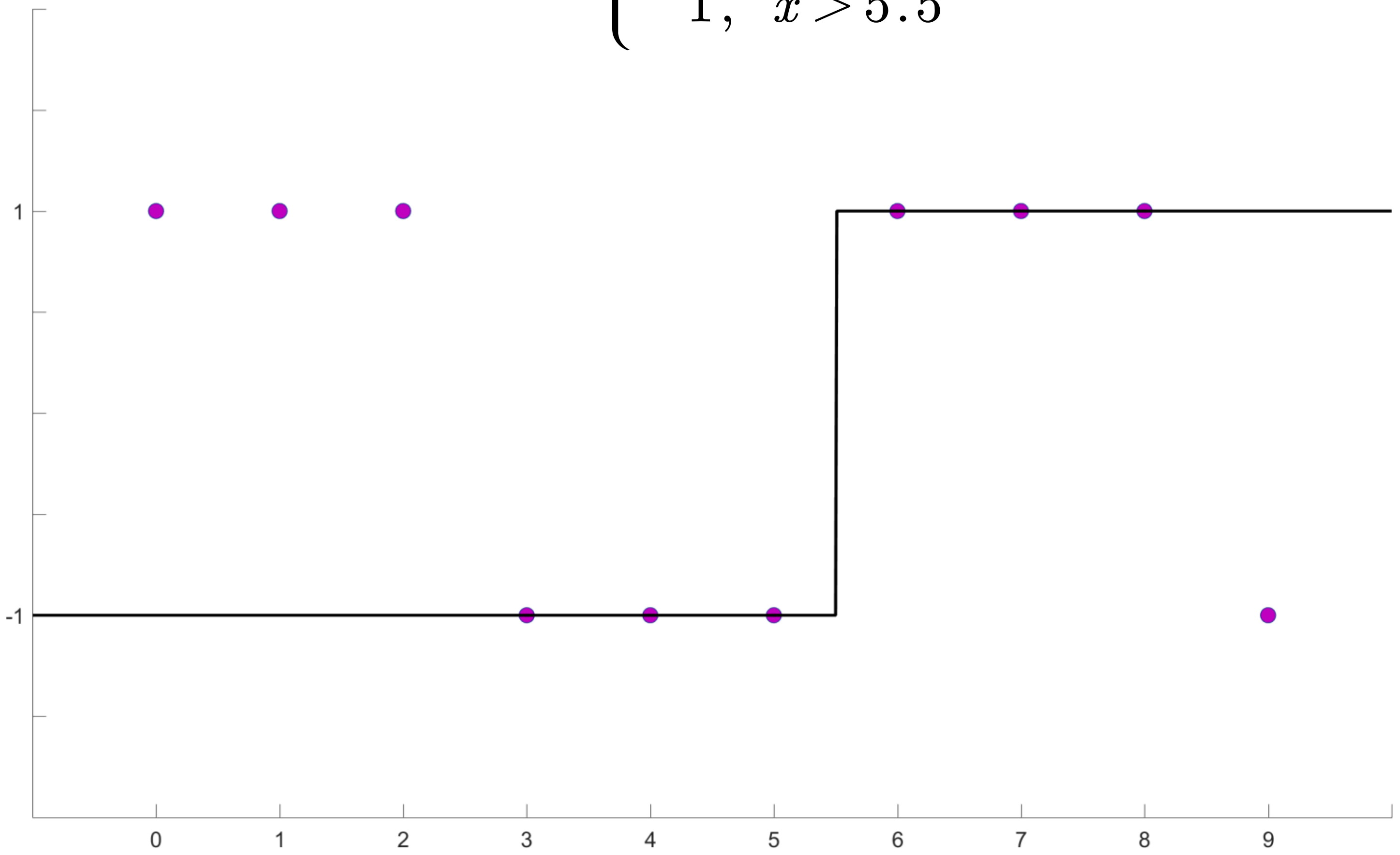
$$f_2(x) = \mathrm{sign}\left(\alpha_1 G_1(x) + \alpha_2 G_2(x)\right)$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.1

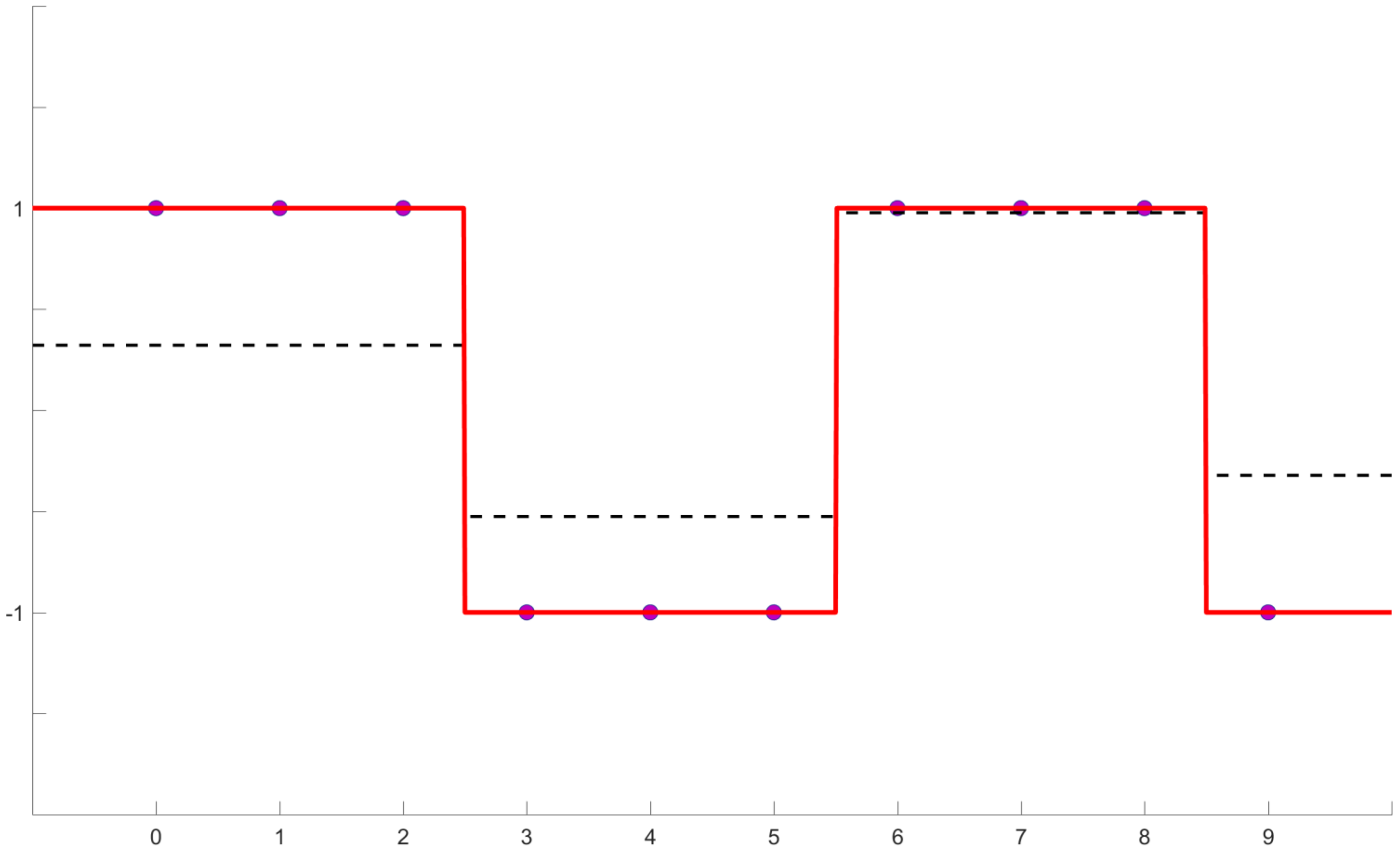
$$G_3(x)=\begin{cases}-1, & x<5.5\\ 1, & x>5.5\end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

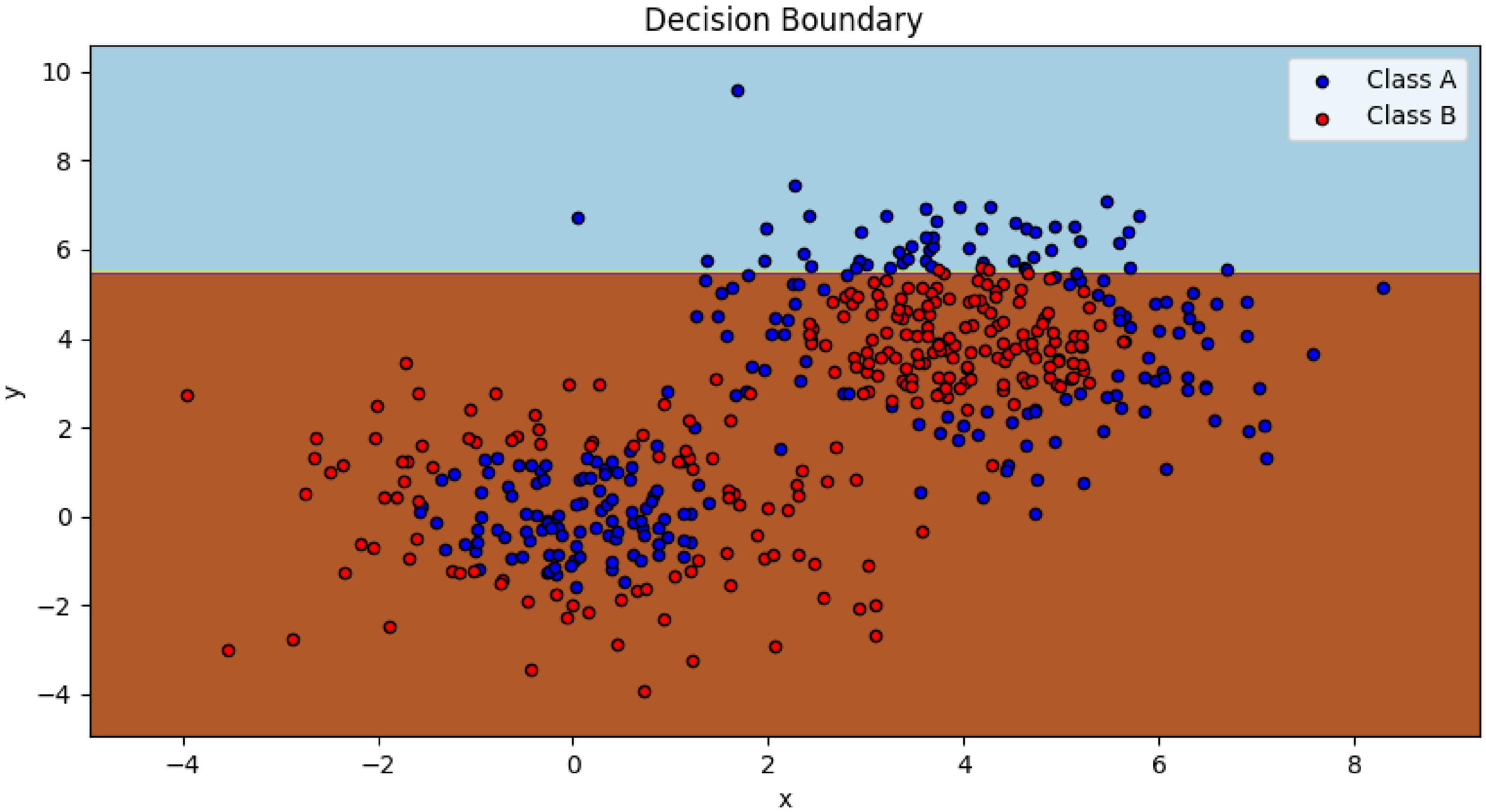
例 8.1

$$f_3(x) = \mathrm{sign}\left(\alpha_1 G_1(x) + \alpha_2 G_2(x) + \alpha_3 G_3(x)\right)$$



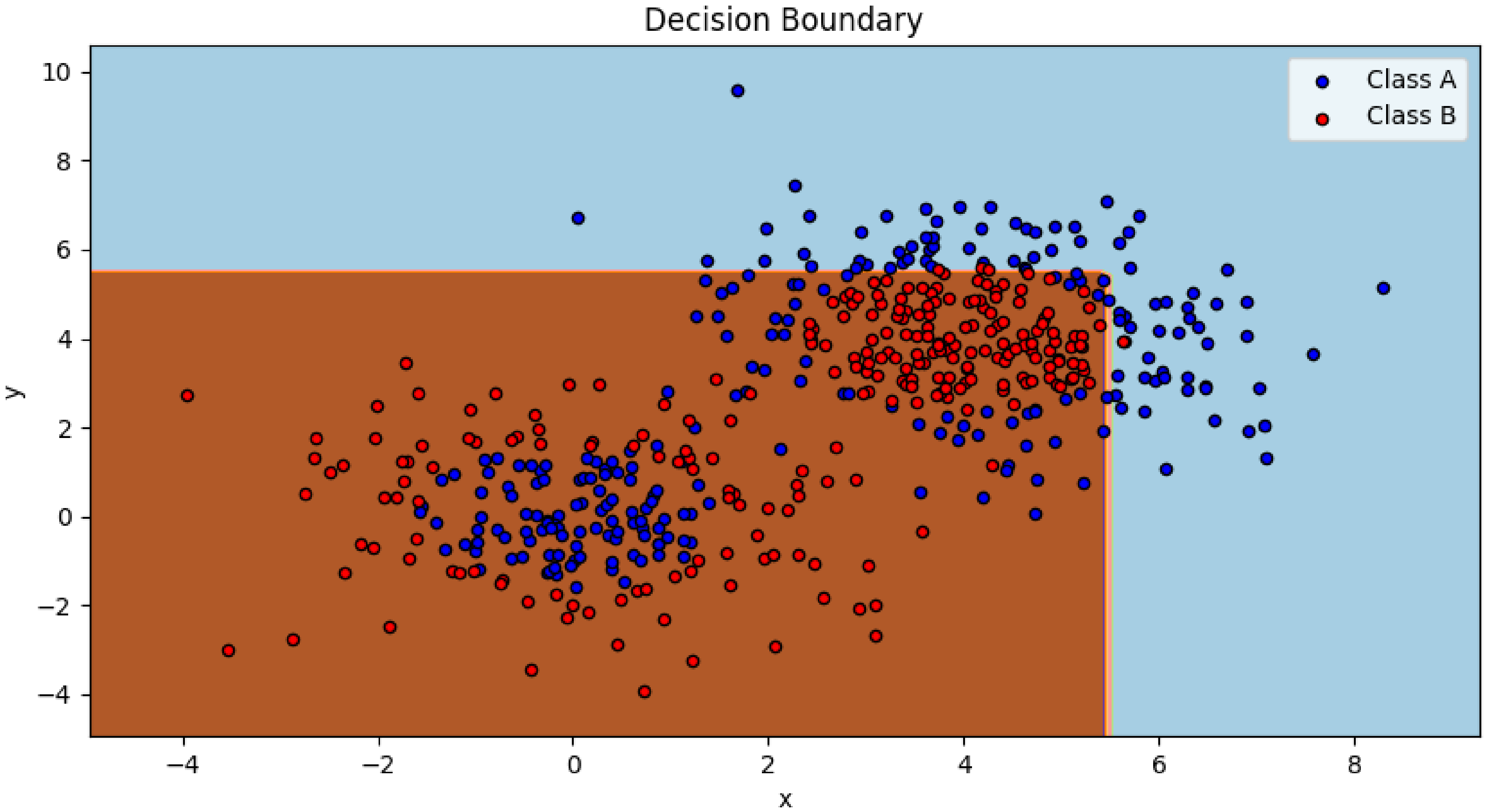
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 1$



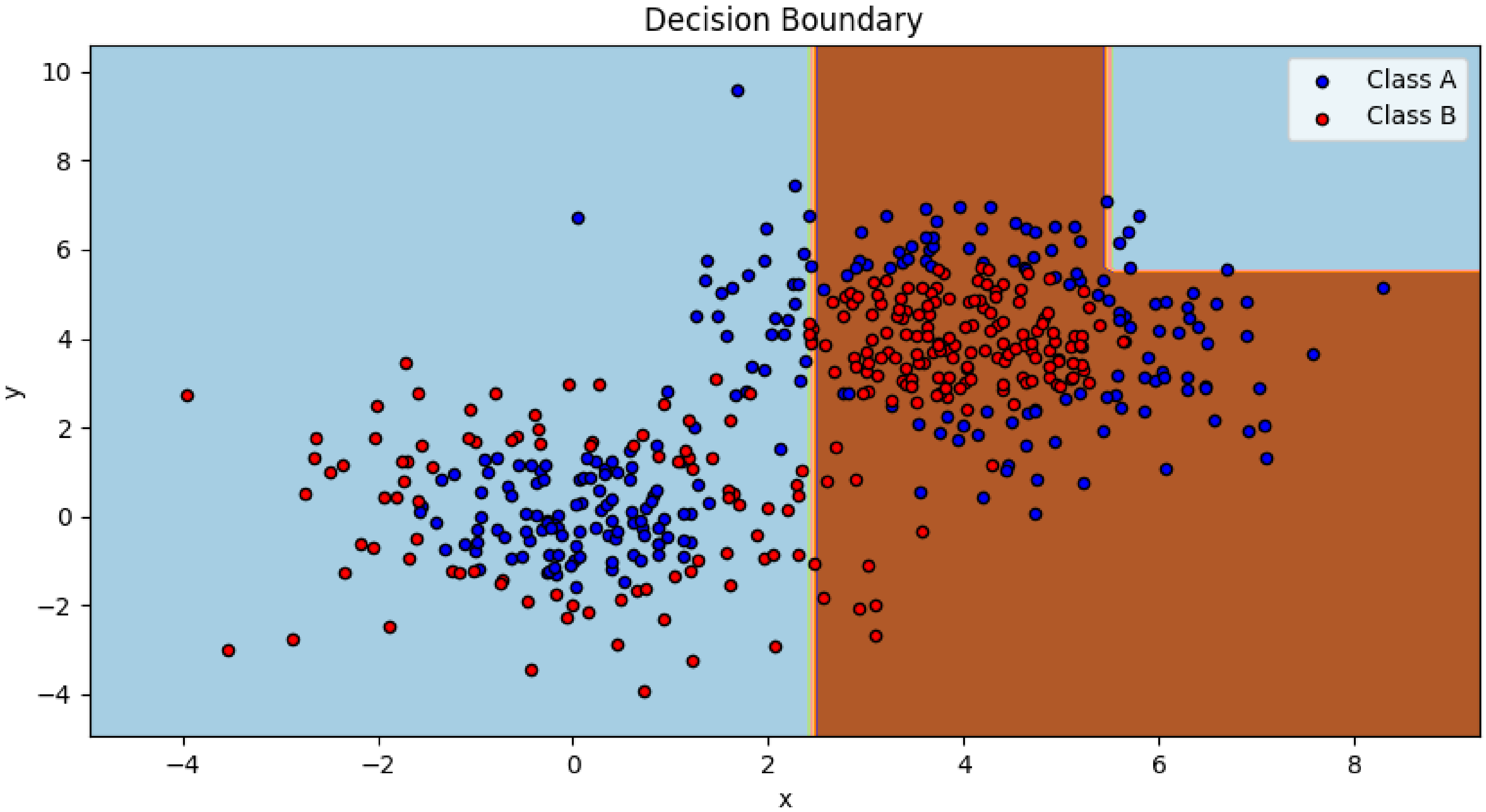
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 3$



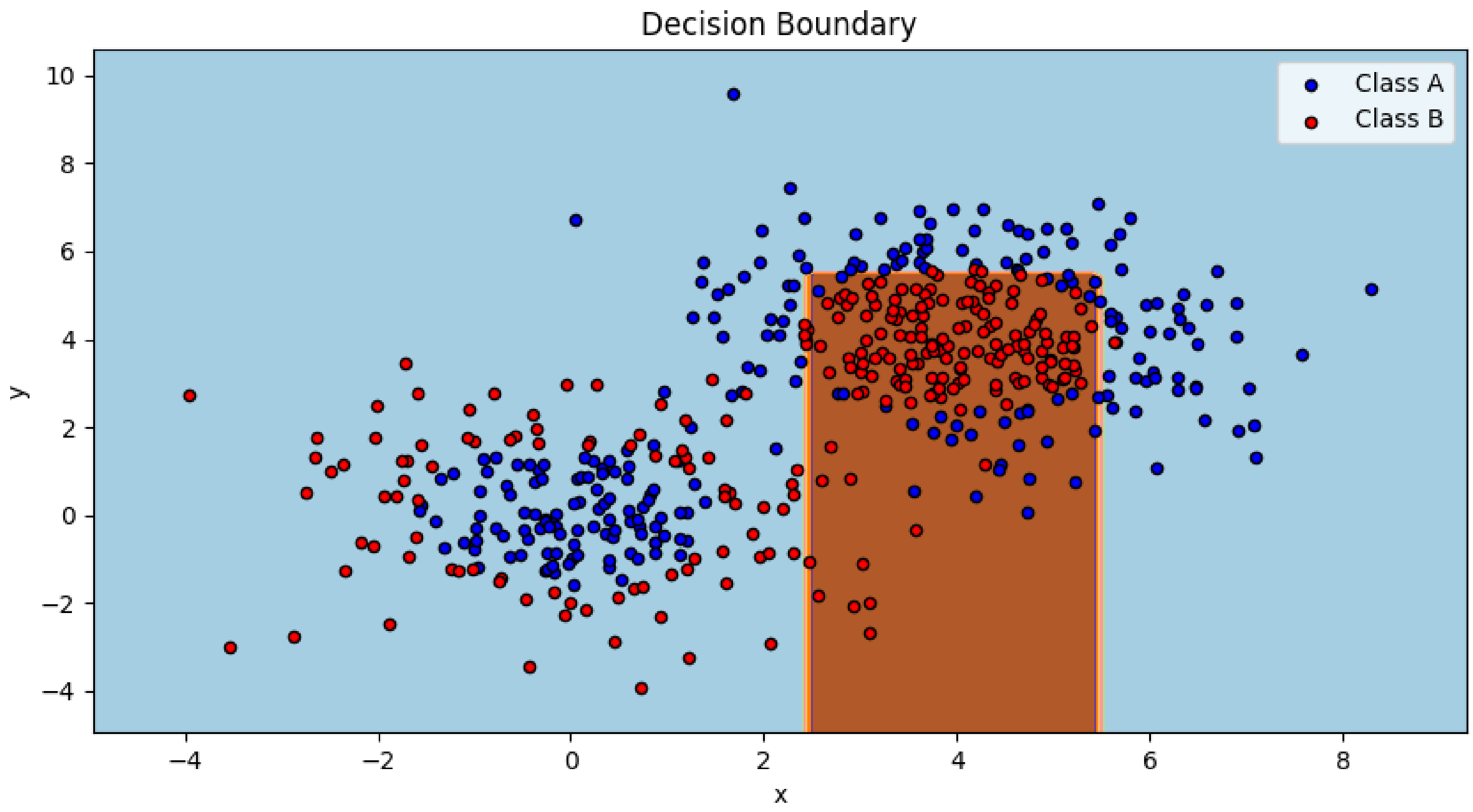
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 4$



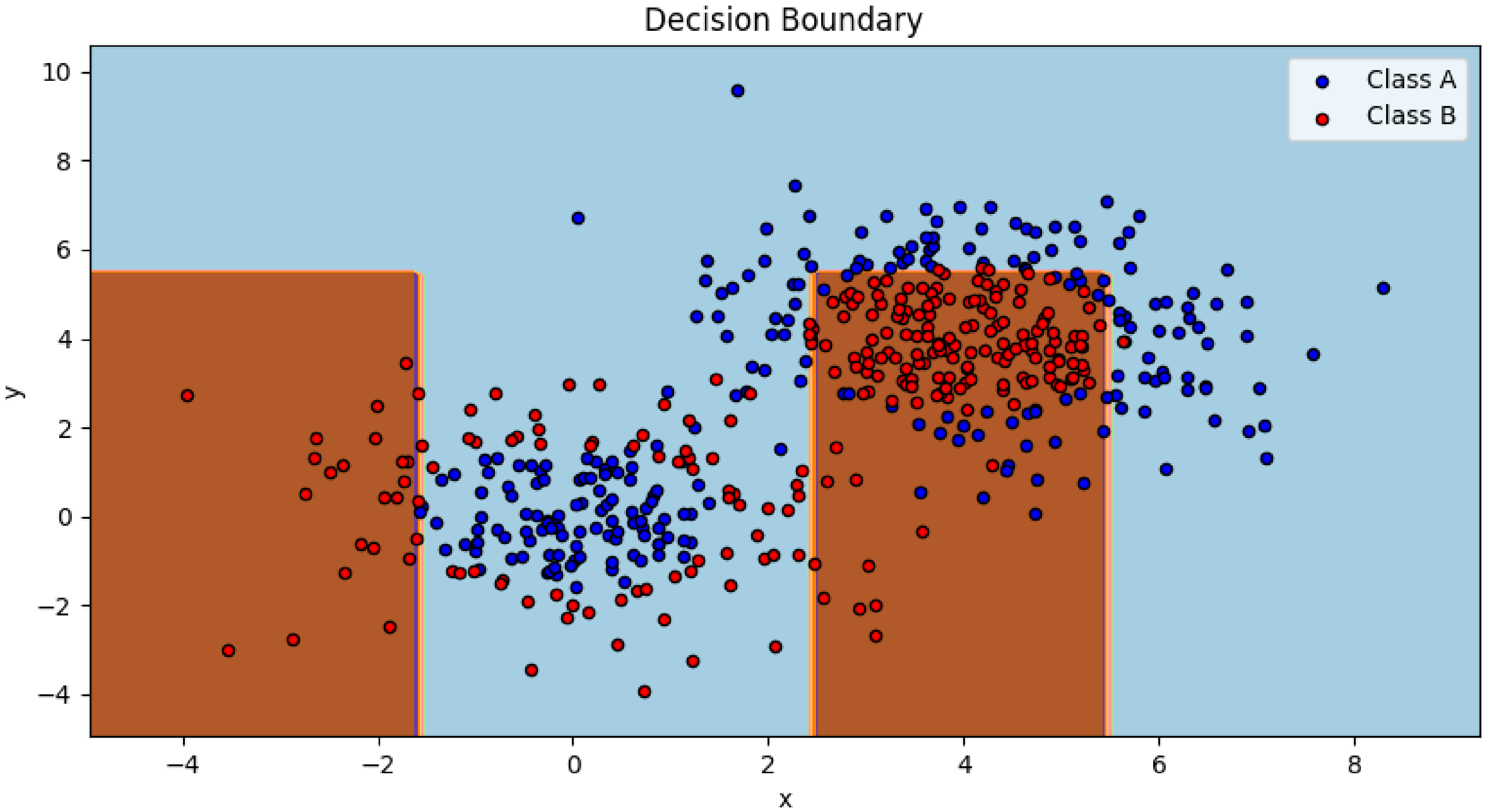
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 5$



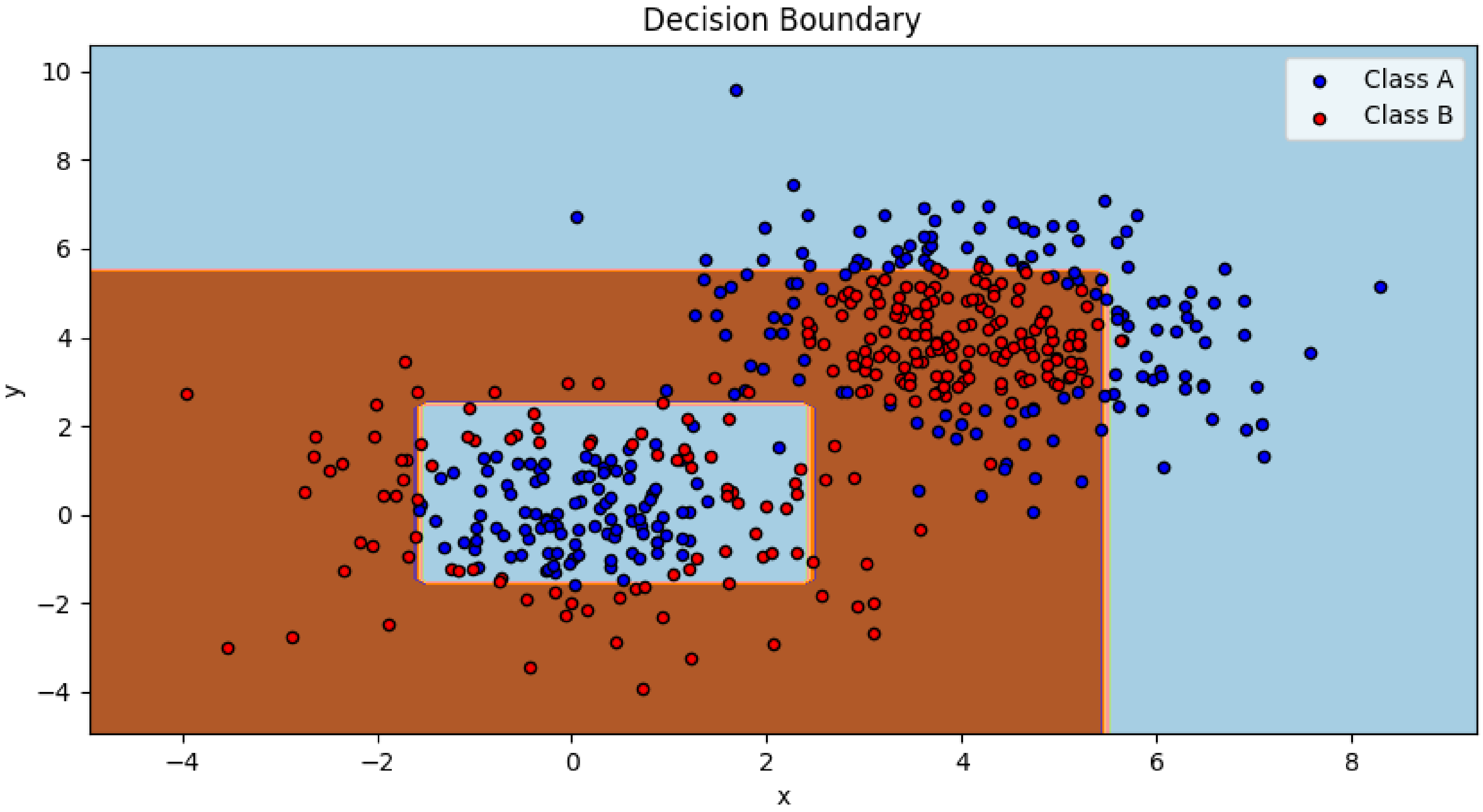
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 10$



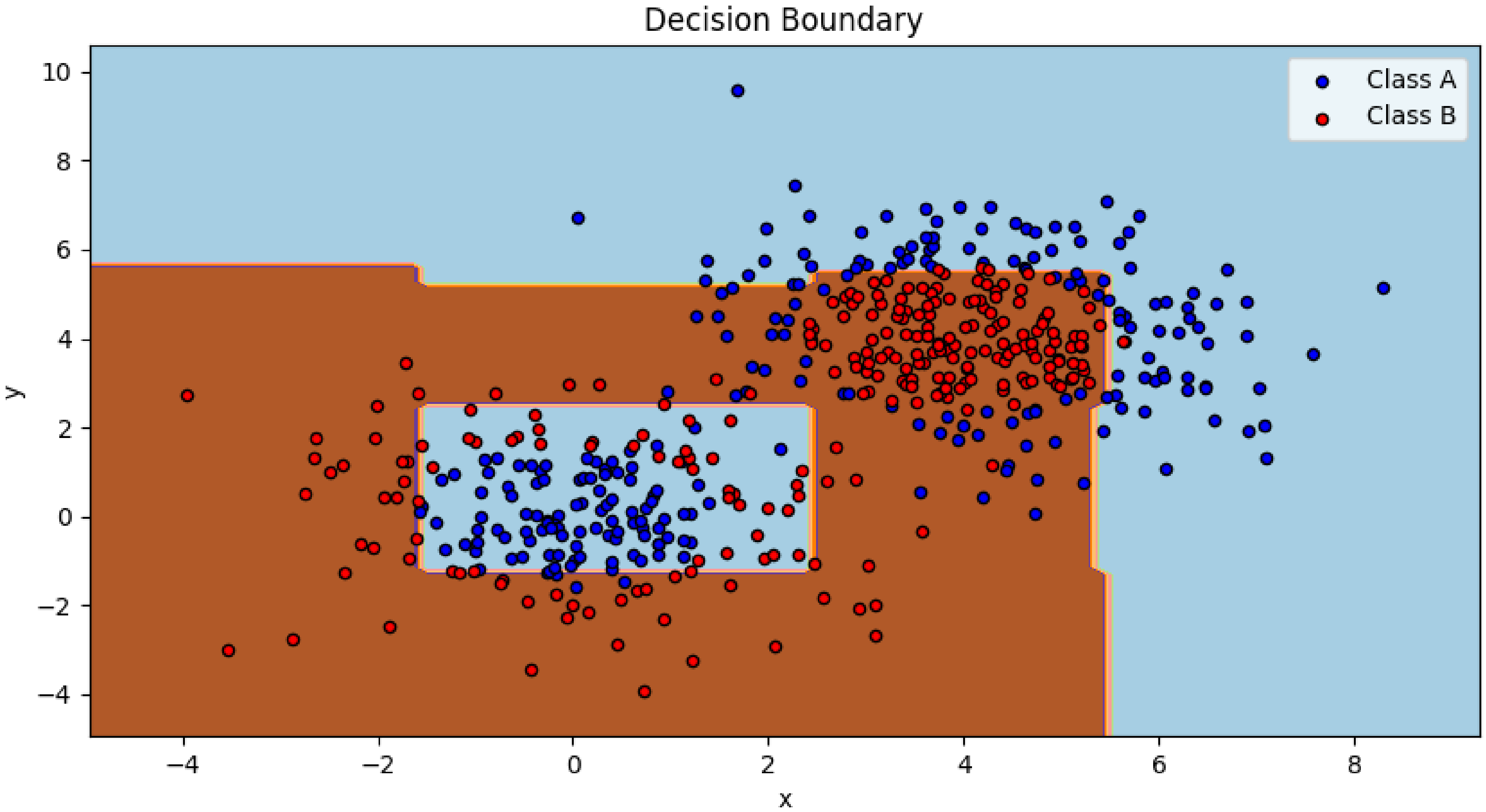
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 15$



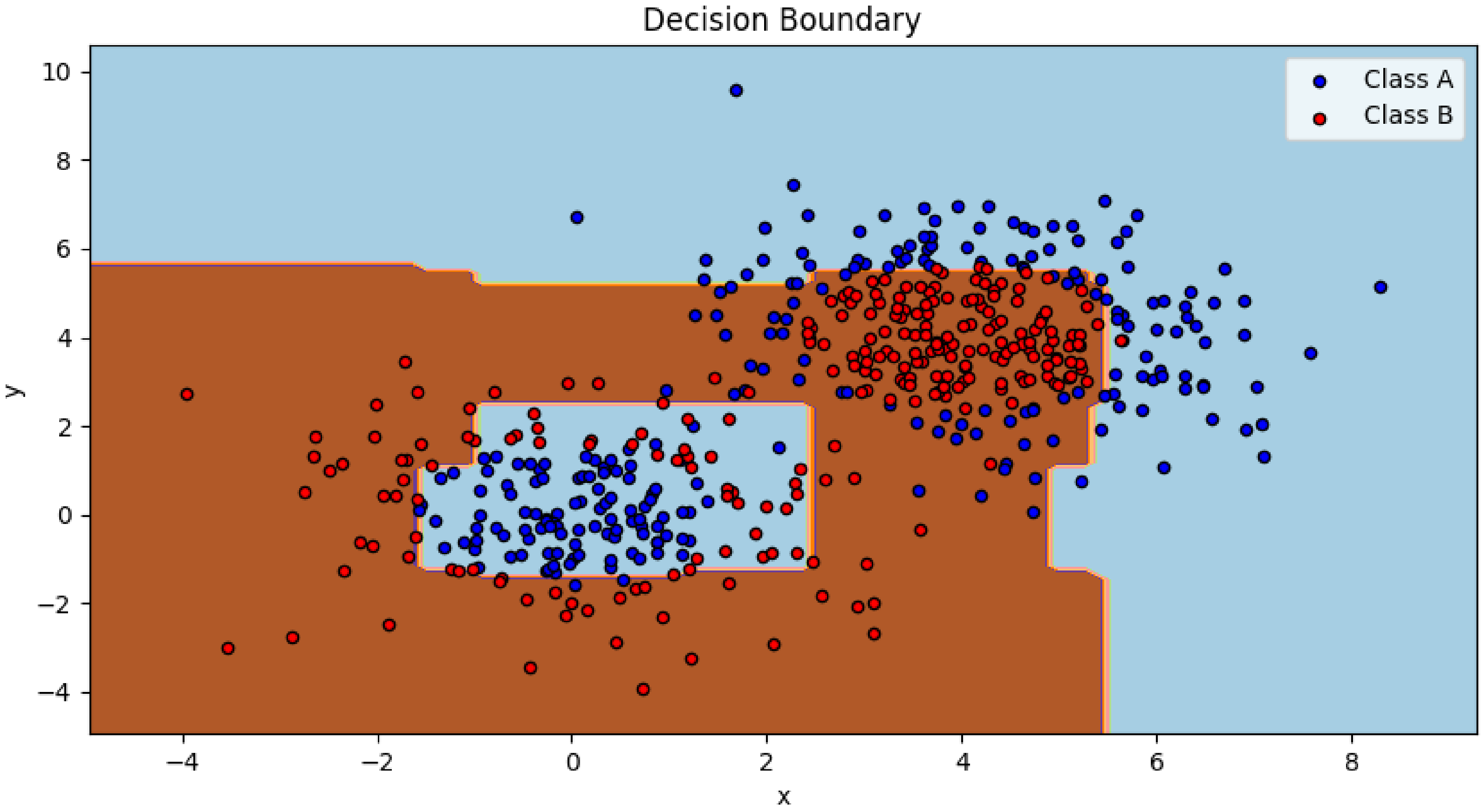
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 20$



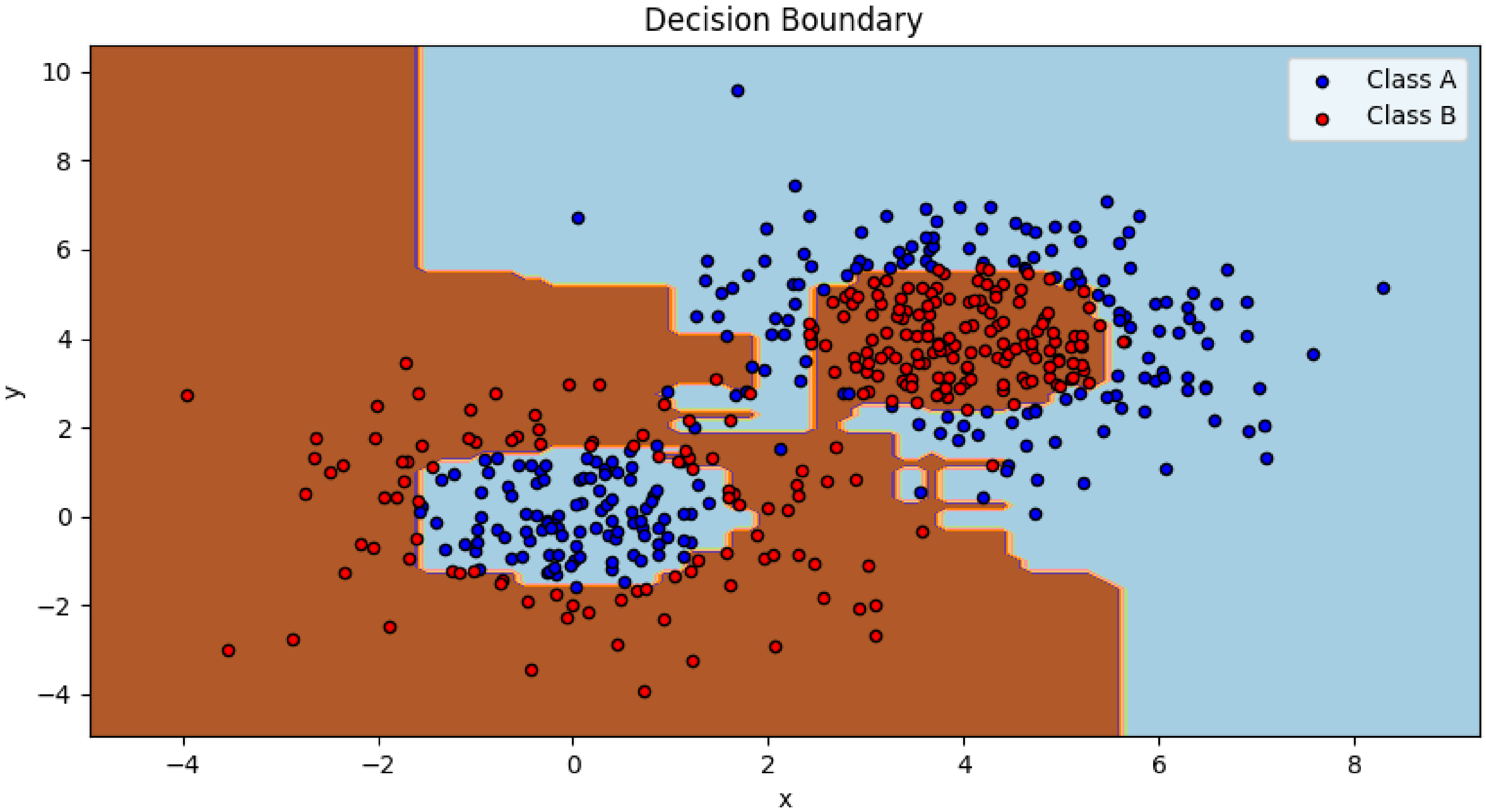
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 40$



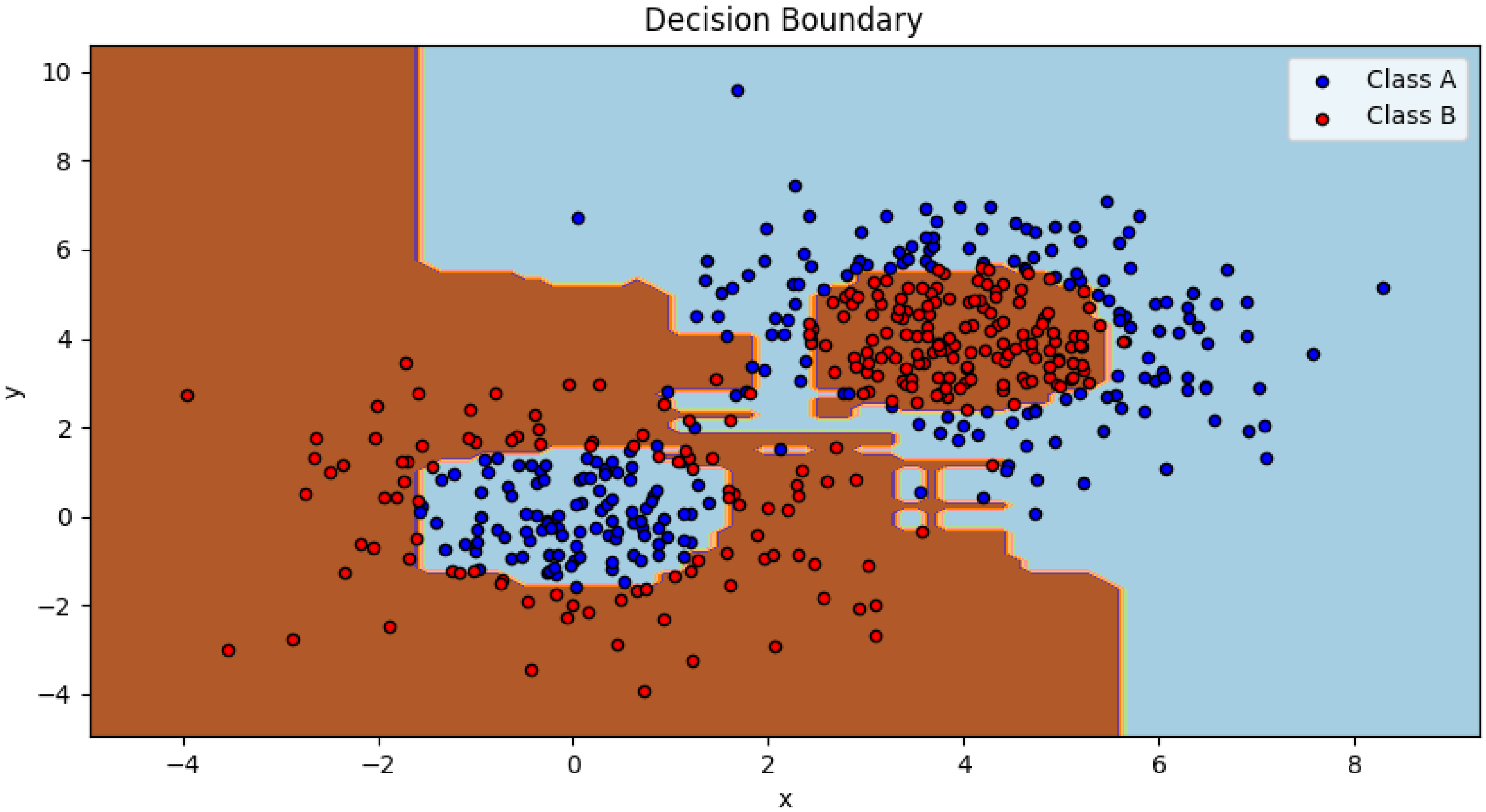
黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 100$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

另一个例子：$m = 200$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法的误差分析

- 定理8.1：AdaBoost算法最终分类器的训练误差界为

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

- 定理8.2：$\prod_{m}Z_m\leq\exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)$，其中$\gamma_m=\frac{1}{2}-e_m$
- 推论8.1：如果$\gamma=\min_m\{\gamma_m\}>0$，则$\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(G(x_i)\neq y_i)\leq\exp(-2M\gamma^2)$
  - AdaBoost的训练误差随着训练总轮次M呈指数速率下降

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

加法模型（additive model）

- 加法模型 $f(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$
  - 其中 $b(x;\gamma_m)$ 是基函数，γ_m 是基函数的参数，β_m是基函数的系数
  - AdaBoost显然是个加法模型
- 假设损失函数为 $L(y,f(x))$，学习加法模型是极小化以下问题

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

  - 要同时优化所有的 β_m，γ_m 是一个困难问题

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

前向分布算法

- 原理：从前到后，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，即每一步优化损失函数 $\min_{\beta,\gamma}\sum_{n=1}^{N}L(y_i,\beta b(x_i;\gamma))$
- 初始化 $f_0(x)=0$
- 极小化损失函数 $(\beta,\gamma)=arg\min_{\beta,\gamma}\sum_{n=1}^{N}L(y_i,f_{m-1}(x_i)+\beta b(x;\gamma))$
- 更新 $f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$
- 最终加法模型 $f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

前向分布算法与AdaBoost

- 定理8.3：AdaBoost是前向分步加法算法，其损失函数是指数函数
- 证明：AdaBoost的分类器$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$显然是加法算法

  每一轮只训练该轮对应的基分类器

  假设已经训练得到了$f_{m-1}(x)=\alpha_1 G_1(x)+\cdots+\alpha_{m-1}G_{m-1}(x)$

  令损失函数是指数函数，则第 m 的目标是优化以下问题

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

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

前向分布算法与AdaBoost

- 证明：令 $\bar{w}_{mi}=\exp[-y_i f_{m-1}(x_i)]$ （确定值），则优化问题变为

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

对任意 $\alpha > 0$，$\bar{w}_{mi}\exp[-y_i\alpha G(x_i)] = \begin{cases} \bar{w}_{mi}e^{-\alpha}, & \text{if } y_i = G(x_i) \\ \bar{w}_{mi}e^{\alpha}, & \text{otherwise} \end{cases}$

优化问题等价于

$$G_m^*(x)) = arg\min_{G} \sum_{y_i = G(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G(x_i)} \bar{w}_{mi} e^{\alpha}$$

$$= arg\min_{G} \sum_{i=1}^{N} \bar{w}_{mi} \mathbb{I}(y_i \neq G(x_i))$$

使第 m 轮加权数据分类错误率最小的基分类器

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

前向分布算法与AdaBoost

- 证明：同样的，对于优化问题

$$G_m^*(x)) = arg\min_G \sum_{y_i = G(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G(x_i)} \bar{w}_{mi} e^{\alpha}$$

$$= arg\min_G \sum_{y_i = G(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G(x_i)} \bar{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G(x_i)} \bar{w}_{mi} e^{\alpha} - \sum_{y_i \neq G(x_i)} \bar{w}_{mi} e^{-\alpha}$$

$$= arg\min_G e^{-\alpha} \sum_{i=1}^{N} \bar{w}_{mi} + (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} \bar{w}_{mi} \mathbb{I}(y_i \neq G(x_i))$$

对 α 求导等于0有，$\alpha_m^* = \frac{1}{2} \log \frac{1 - e_m}{e_m}$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法的优点

- AdaBoost算法提供的是框架，可以使用各种算法作为弱分类器
- 弱分类器的构造极其简单，不用对特征进行筛选
- 不需要预先知道弱分类器的错误率上限，算法可以根据弱分类器的反馈，自适应地调整假定的错误率，能显著的提高学习精度
- 不容易发生过拟合

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

AdaBoost算法的缺点

- 对异常值很敏感，容易受噪音干扰
- 算法的迭代次数也就是弱分类器数目不好设定，可以使用交叉验证来进行确定
- 算法依赖于弱分类器，而弱分类器的训练时间往往很长，所以比较耗时
- 数据不平衡导致分类精度下降

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

提升树模型

- 采用加法模型与前向分步算法
- 弱分类器采用决策树算法
  - 对分类和回归问题，决策树都是二叉树
- 模型是 $f_M(x)=\sum_{m=1}^{M}T(x;\theta_m)$
  - 其中 $T(x;\theta_m)$ 表示决策树，θ_m 是参数，M 是树的个数

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

提升树算法

- 弱分类器决策树表示为 $T(x;\theta)=\sum_{j=1}^{J}c_j\mathbb{I}(x\in R_j)$
  - 其中 J 是叶节点个数
  - $R_1,R_2,\cdots,R_J$是特征空间中不相交的划分，c_j 是对应输出
- 利用前向分步算法训练模型

$$\begin{cases} f_m(x)=f_{m-1}(x)+T(x;\theta_m) \\ \hat{\theta}_m=arg\min_{\theta_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T(x_i;\theta_m)) \end{cases}$$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

提升树算法

- 损失函数的选取:
  - 回归问题：平方损失函数

$$L(y,f(x))=(y-f(x))^2$$

  - 分类问题：指数损失函数

$$L(y,f(x))=\exp(-y\cdot f(x))$$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

回归问题的提升树算法

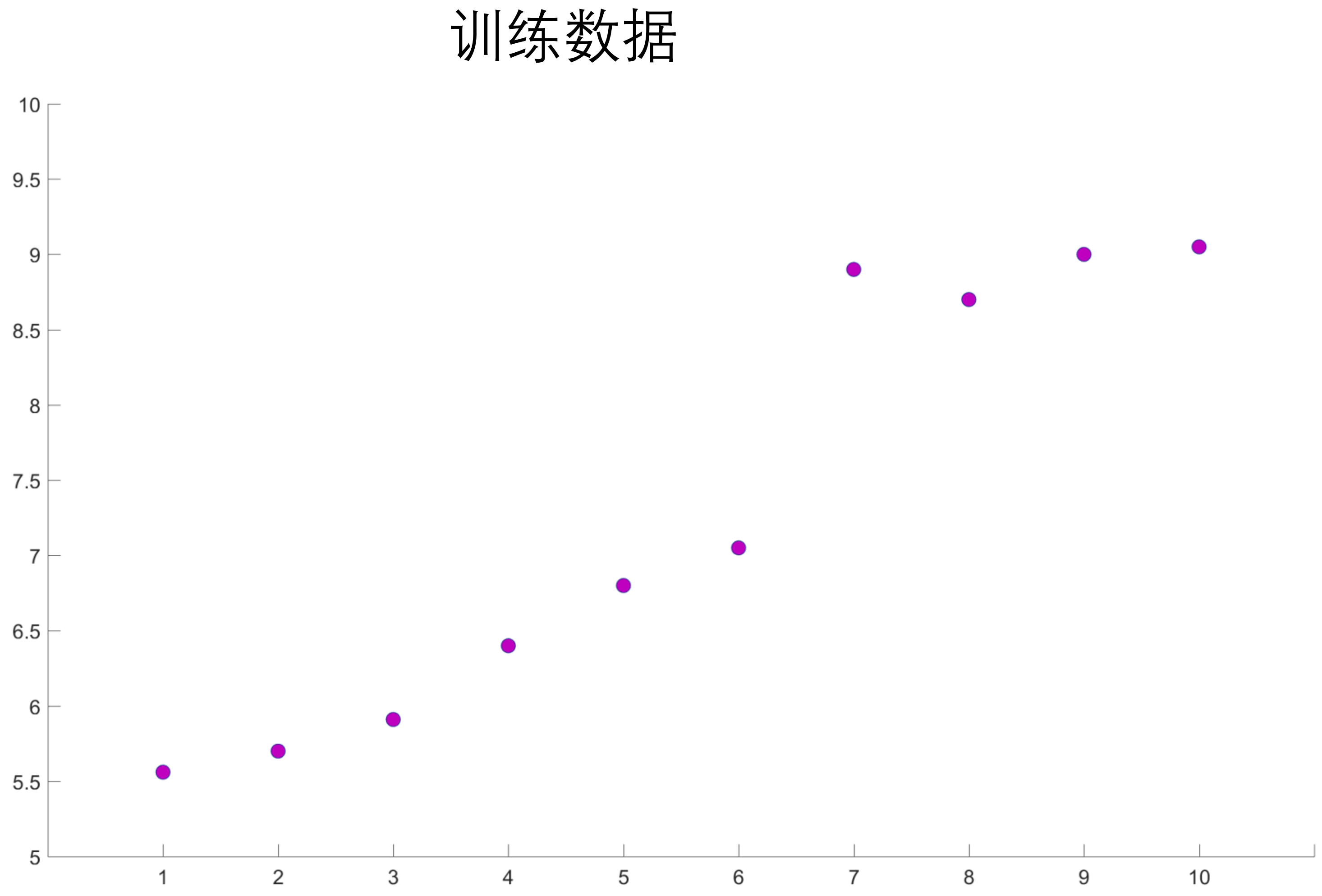
- 建立平方损失误差函数，有

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

- 其中$r = y - f_{m-1}(x)$表示当前模型拟合数据的残差
- 第m轮训练的决策树实际上是对残差进行拟合

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

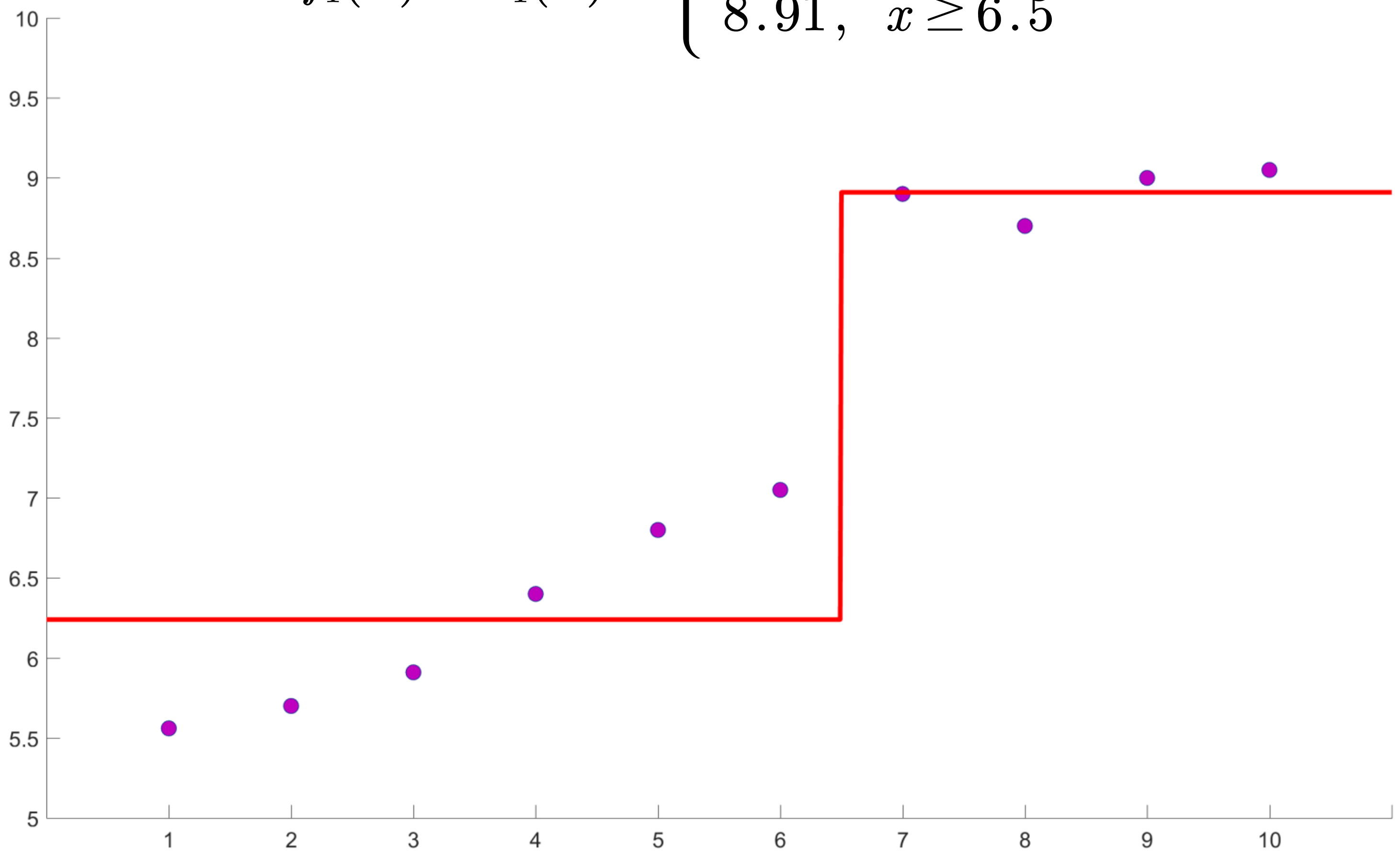
例 8.2



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

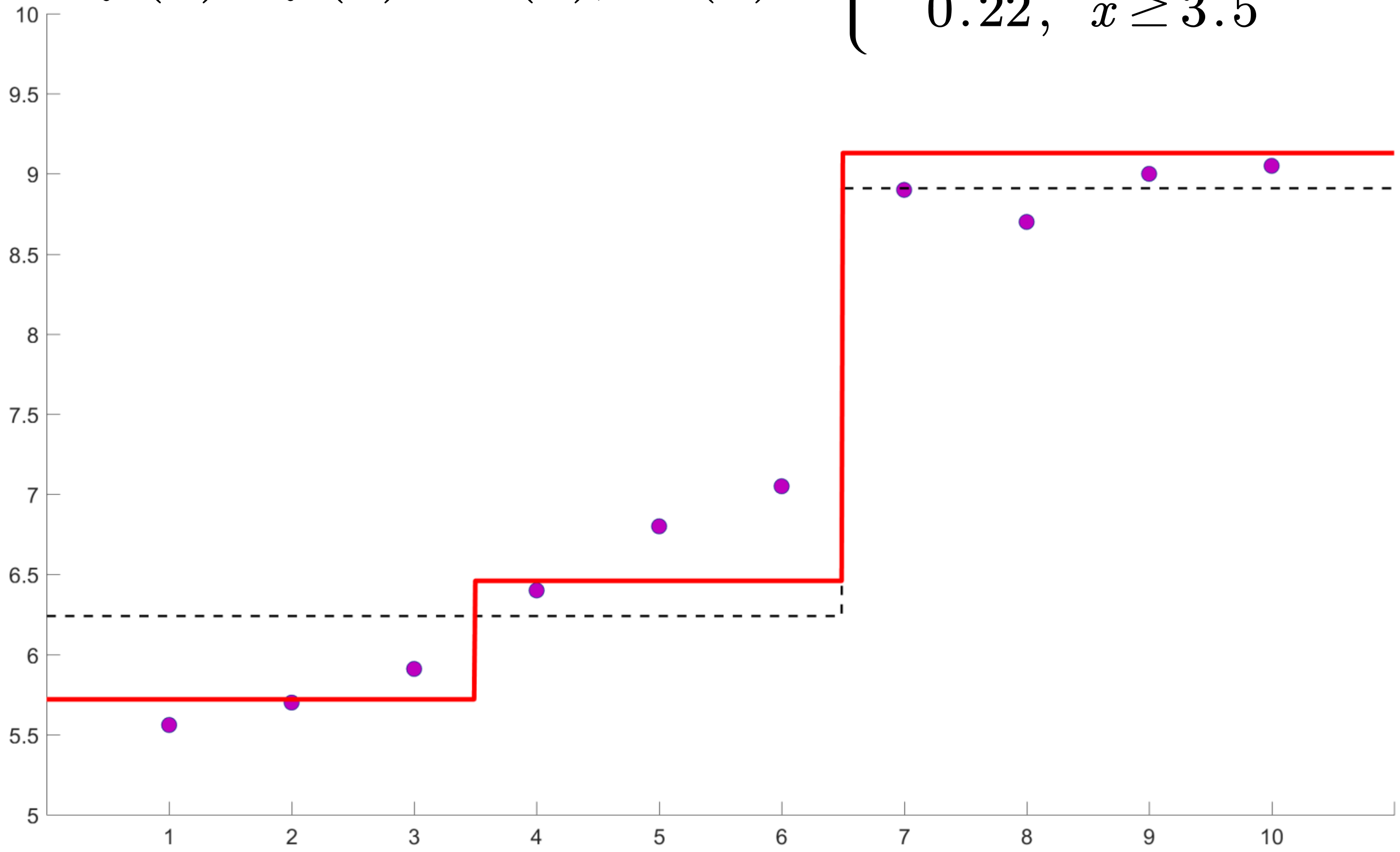
$$f_1(x) = T_1(x) = \begin{cases} 6.24, & x < 6.5 \\ 8.91, & x \geq 6.5 \end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

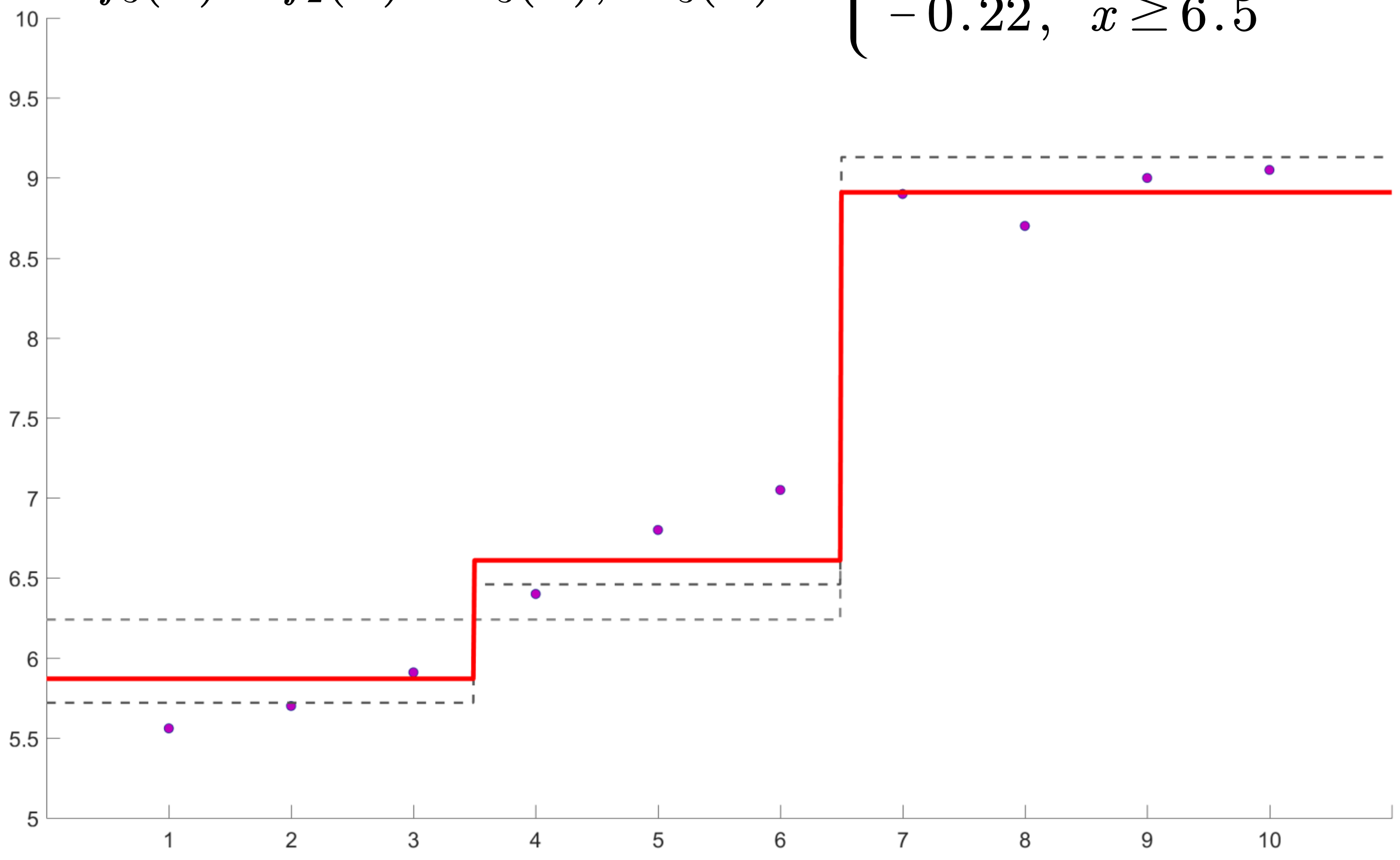
$$f_2(x)=f_1(x)+T_2(x),\quad T_2(x)=\begin{cases}-0.52, & x<3.5\\ 0.22, & x\geq 3.5\end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

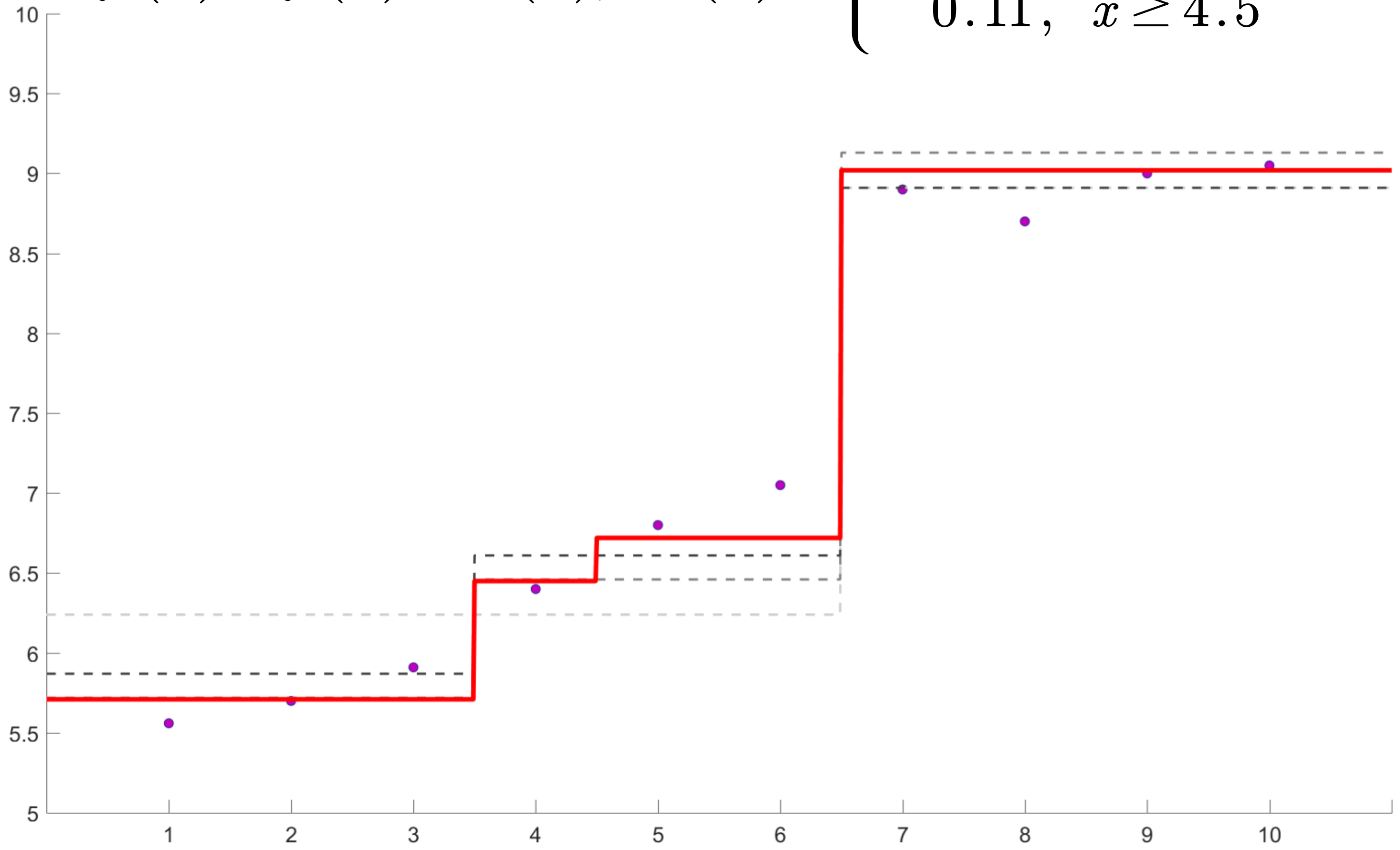
$$f_3(x)=f_2(x)+T_3(x),\quad T_3(x)=\begin{cases} 0.15, & x<6.5 \\ -0.22, & x\geq 6.5 \end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

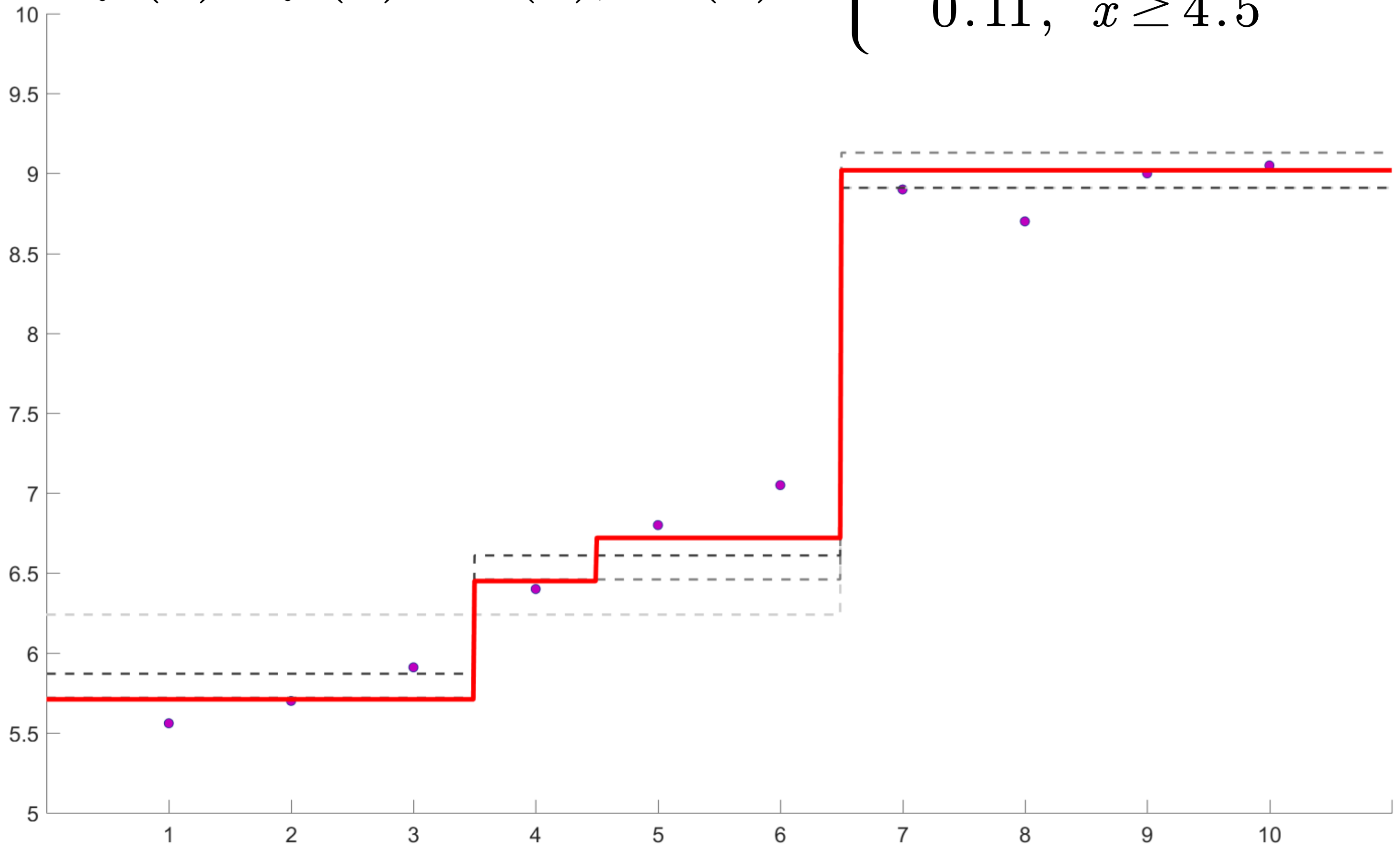
$$f_4(x)=f_3(x)+T_4(x),\quad T_4(x)=\begin{cases}-0.16, & x<4.5\\ 0.11, & x\geq 4.5\end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

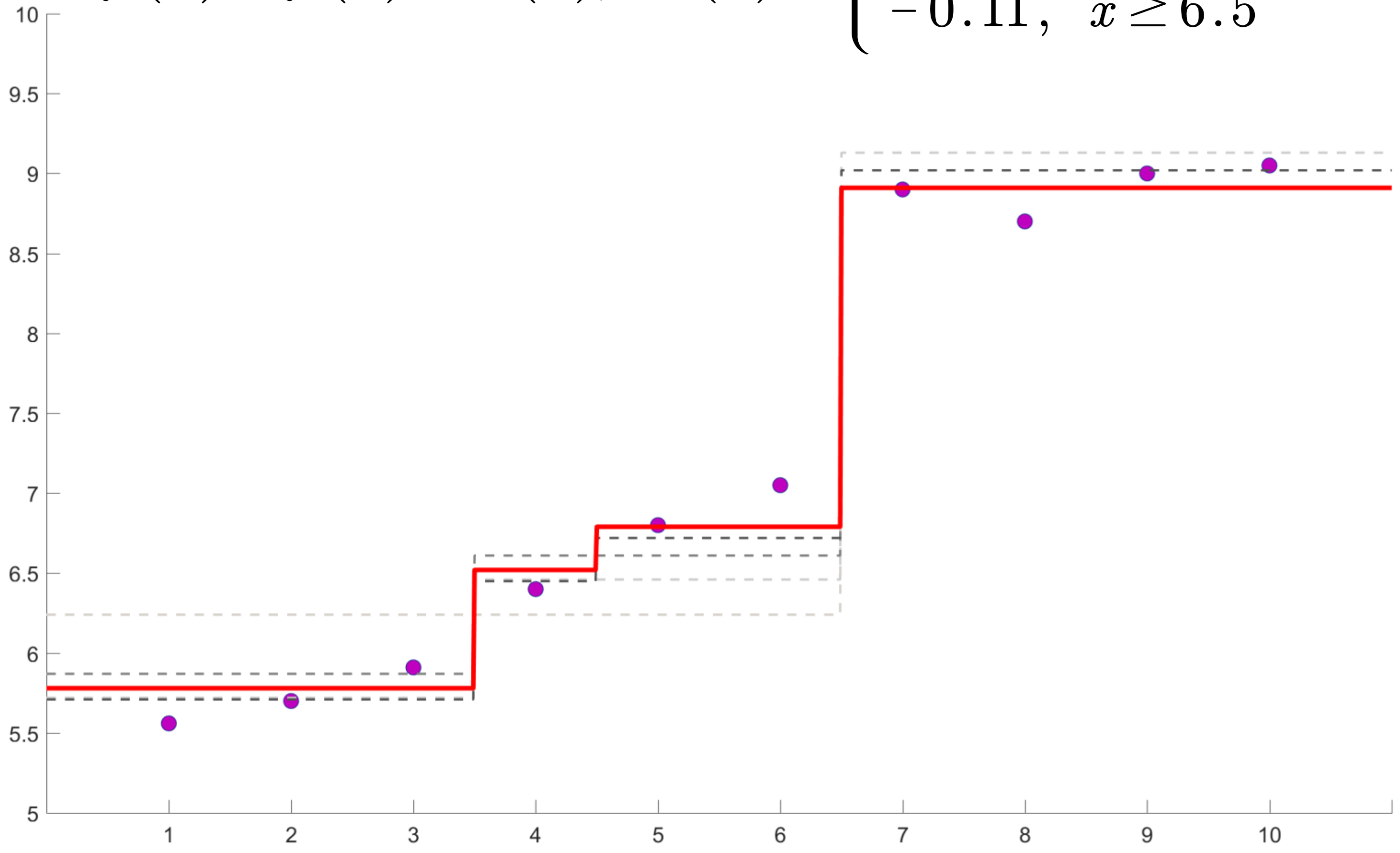
$$f_4(x) = f_3(x) + T_4(x),\quad T_4(x) = \begin{cases} -0.16, & x < 4.5 \\ 0.11, & x \geq 4.5 \end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

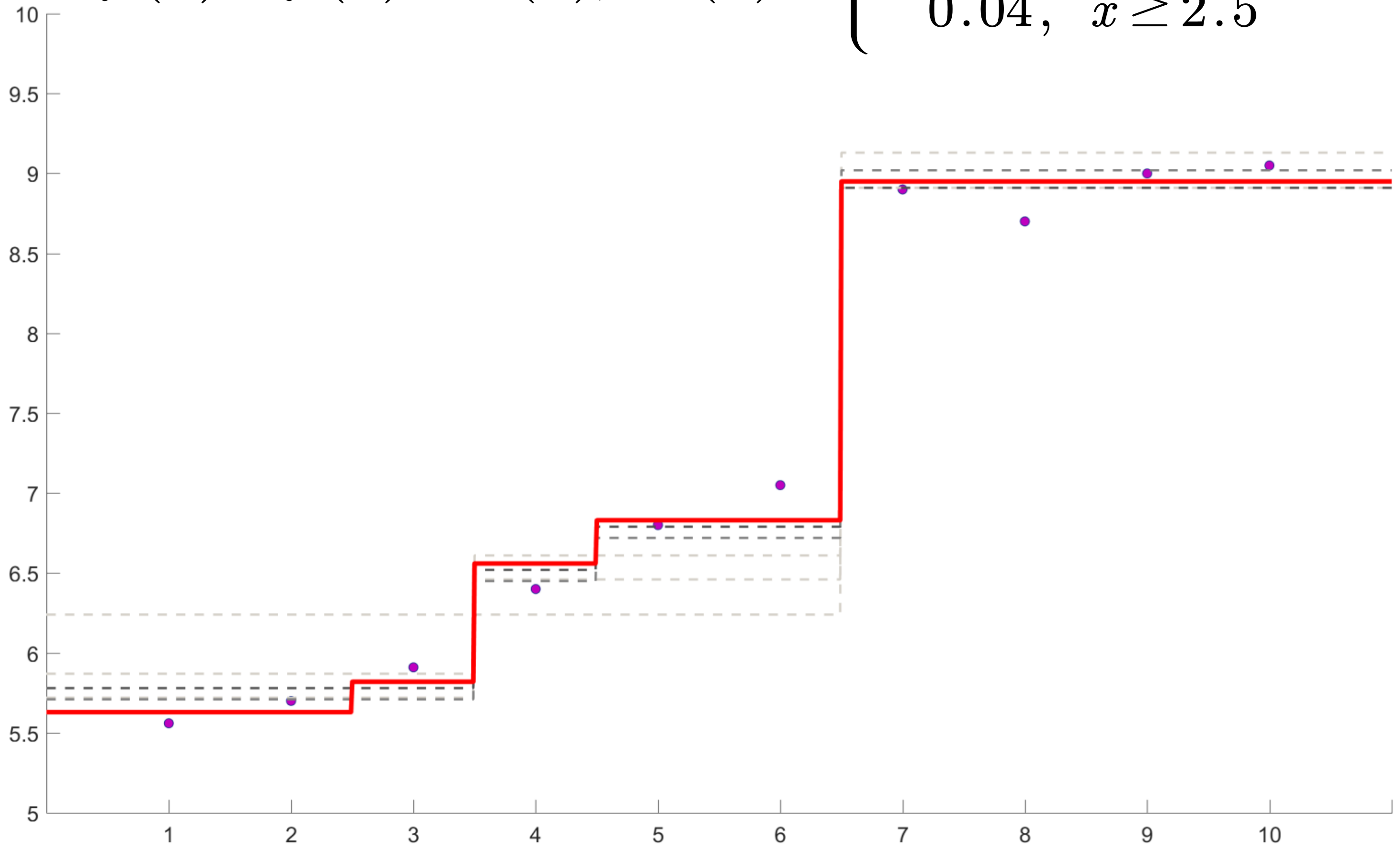
$$f_5(x) = f_4(x) + T_5(x),\quad T_5(x) = \begin{cases} 0.07, & x < 6.5 \\ -0.11, & x \geq 6.5 \end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

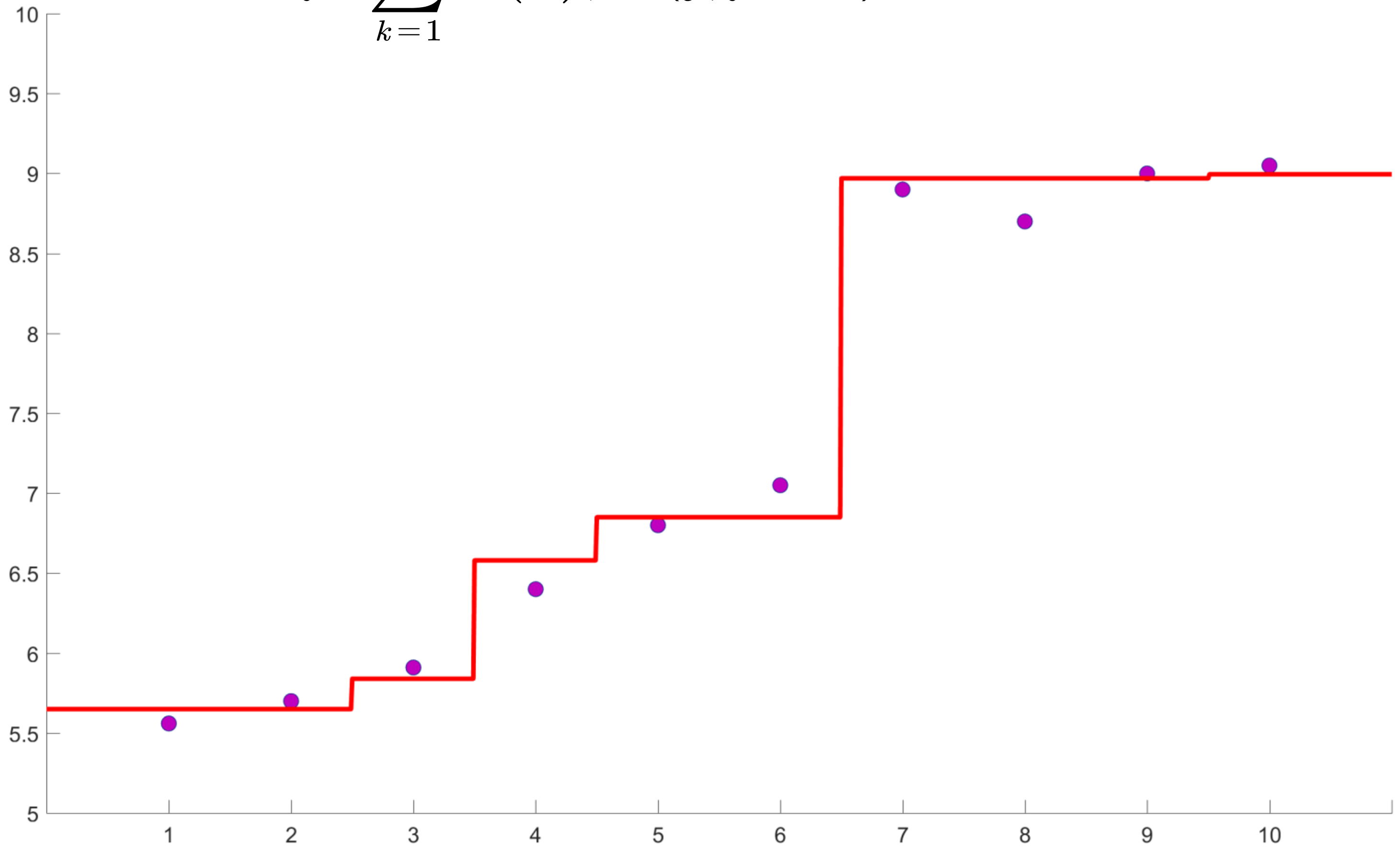
$$f_6(x) = f_5(x) + T_6(x),\quad T_6(x) = \begin{cases} -0.15, & x < 2.5 \\ 0.04, & x \geq 2.5 \end{cases}$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

例 8.2

$$f_{16}\sum_{k=1}^{16}T_k(x),\ L(y,f_{16}(x))=0.141$$



黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

平方损失函数的缺点

- 平方损失函数在数学上容易处理，但对异常值缺乏鲁棒性
- 例如

真实值 y_i	0.5	1.2	2	5
预测值 $f(x_i)$	0.6	1.4	1.5	1.7
$L=(y_i-f(x_i))^2/2$	0.005	0.02	0.125	5.445

- 损失函数在异常值处的值太多，优化时会过度关注异常点，影响回归效果

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

其他损失函数

- 绝对值损失函数

$$L(y,f(x)) = |y - f(x)|$$

- Huber损失函数

$$L(y,f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2, & |y - f(x)| \le \delta \\ \delta\left(|y - f(x)| - \frac{\delta}{2}\right), & |y - f(x)| > \delta \end{cases}$$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

其他损失函数

真实值 y_i	0.5	1.2	2	5
预测值 $f(x_i)$	0.6	1.4	1.5	1.7
$L=(y_i-f(x_i))^2/2$	0.005	0.02	0.125	5.445
绝对值损失函数	0.1	0.2	0.5	3.3
Huber损失函数 $\delta=0.5$	0.005	0.02	0.125	1.525

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

- 集成算法
- AdaBoost算法
  - 算法建立
  - 算法的误差分析
  - 算法的另一解释
  - 总结
- 提升树
  - 梯度提升树（GBDT）

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

梯度提升树（GBDT）

- 对于一般的损失函数，计算损失函数的负梯度：

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

平方损失函数时，
负梯度正好是残差

- 新建立的树是对 $\{r_{m1}, r_{m2}, \cdots, r_{mN}\}$ 进行拟合
- 原因是将损失函数在 f_{m-1} 处进行一阶泰勒展开，有

$$L(y, f(x)) \doteq L(y, f_{m-1}(x)) + \left.\frac{\partial L}{\partial f}\right|_{f_{m-1}} \cdot (f(x) - f_{m-1}(x))$$

- 当 $T_m(x) \doteq f(x) - f_{m-1}(x) = -\left.\frac{\partial L}{\partial f}\right|_{f_{m-1}}$ 时，损失函数下降最快

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

其他损失函数的负梯度

- 绝对值损失函数

$$-\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}=\mathrm{sign}(y_i-f(x_i))$$

- Huber损失函数

$$-\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}=\begin{cases} y_i-f(x_i), & |y_i-f(x_i)|\le\delta \\ \delta\,\mathrm{sign}(y_i-f(x_i)), & |y_i-f(x_i)|>\delta \end{cases}$$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

梯度提升树算法

- 初始化 $f_0(x) = arg\min_c \sum_{i=1}^{N} L(y_i, c)$
- 对第 m 轮训练，计算 $r_{mi} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$
- 对 r_{mi} 拟合一个回归树，得到第 m 棵树的叶节点区域 R_{mj}, $j=1,\cdots,J$
- 计算 $c_{mi} = arg\min_c \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c)$
- 更新 $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J} c_{mj}\mathbb{I}(x \in R_{mj})$
- 最终回归树为 $\hat{f}(x) = f_M(x) = \sum_{m=1}^{M}\sum_{j=1}^{J} c_{mj}\mathbb{I}(x \in R_{mj})$

极限梯度提升树（XGBT）

- GBDT的第m轮训练的损失函数可以表示为

$$\min_{T_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T_m(x_i))$$

- XGBT在其中加入正则项

$$\min_{T_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T_m(x_i))+\gamma J+\frac{\lambda}{2}\sum_{j=1}^{J}c_{mj}^2$$

- 其中J是叶节点个数，c_{mj}是叶节点的最优值，与GBDT相同

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

极限梯度提升树（XGBT）

- 将损失函数在 f_{m-1} 进行二阶泰勒展开有

$$\sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T_m(x_i)) + \gamma J + \frac{\lambda}{2}\sum_{j=1}^{J} c_{mj}^2$$

$$\doteq \sum_{i=1}^{N}\left\{L\left(y_i, f_{m-1}(x_i)\right) + \frac{\partial L\left(y_i, f_{m-1}(x_i)\right)}{\partial f\left(x_i\right)} T_m(x_i) + \frac{1}{2}\frac{\partial^2 L\left(y_i, f_{m-1}(x_i)\right)}{\partial f_{m-1}^2(x_i)} T_m^2(x_i)\right\} + \gamma J + \frac{\lambda}{2}\sum_{j=1}^{J} c_{mj}^2$$

- 第一项是常数对优化无影响，所以最终优化问题变为

$$\min_{T_m} \sum_{i=1}^{N}\left\{\frac{\partial L\left(y_i, f_{m-1}(x_i)\right)}{\partial f\left(x_i\right)} T_m(x_i) + \frac{1}{2}\frac{\partial^2 L\left(y_i, f_{m-1}(x_i)\right)}{\partial f_{m-1}^2(x_i)} T_m^2(x_i)\right\} + \gamma J + \frac{\lambda}{2}\sum_{j=1}^{J} c_{mj}^2$$

黄迟 计算机与人工智能学院
huangchi@swufe.edu.cn

XGBT与GBDT比较

- GBDT以传统CART作为基分类器，而XGBT支持线性分类器
- GBDT在优化时只用到一阶导数， XGBT对代价函数做了二阶泰勒展开，收敛速度更快
- XGBT的代价函数引入正则化项，控制了模型的复杂度，正则化项包含全部叶子节点的个数，每个叶子节点输出值的L2正则化，防止模型过拟合
- 优化过程的空间复杂度很高