人工智能的数学基础

华东师范大学 数学科学学院 黎芳(教授) 2019年10月28日

Chapter 8 提升方法

Table of Contents

8.1 提升方法AdaBoost算法

8.1.1 提升方法的基本思路

对于一个复杂任务来说，将多个专家的判断进行适当的综合所得出的判断，要比其中任何一个专家单独的判断好．实际上， 就是“三个臭皮匠顶个诸葛亮” 的道理。

Kearns和Valiaot： 在概率近似正确(probably approximately correct, PAC)的学习的框架中，一个概念（一个类） ， 如果存在一个多项式的学习算法能够学习它， 并且正确率很高，那么就称这个概念是强可学习的； 一个概念，如果存在一个多项式的学习算法能够学习它，学习的正确率仅比随机猜测略好，那么就称这个概念是弱可学习的。

Schapire：证明了强可学习与弱可学习是等价的。The Strength of Weak Learnability.

对提升方法来说，有两个问题需要回答：一是在每一轮如何改变训练数据的权值或概率分布；二是如何将弱分类器组合成一个强分类器．

8.1.2 AdaBoost算法

算法 8.1（AdaBoost）

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，其中 $x_i\in\mathcal{X}\subseteq\mathbf{R}^n$，$y_i\in\mathcal{Y}=\{-1,+1\}$；弱学习算法；

输出：最终分类器 $G(x)$.

（1）初始化训练数据的权值分布

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

（2）对 $m=1,2,\cdots,M$

（a）使用具有权值分布 D_m 的训练数据集学习，得到基本分类器

$$G_m(x):\mathcal{X}\rightarrow\{-1,+1\}$$

（b）计算 $G_m(x)$ 在训练数据集上的分类误差率

$$e_m=P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i) \tag{8.1}$$

（c）计算 $G_m(x)$ 的系数

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m} \tag{8.2}$$

这里的对数是自然对数.

（d）更新训练数据集的权值分布

$$D_{m+1}=(w_{m+1,1},\cdots,w_{m+1,i},\cdots,w_{m+1,N}) \tag{8.3}$$

$$w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)),\quad i=1,2,\cdots,N \tag{8.4}$$

这里，Z_m 是规范化因子

$$Z_m=\sum_{i=1}^{N}w_{mi}\exp(-\alpha_m y_i G_m(x_i)) \tag{8.5}$$

它使 D_{m+1} 成为一个概率分布.

（3）构建基本分类器的线性组合

$$f(x)=\sum_{m=1}^{M}\alpha_m G_m(x) \tag{8.6}$$

得到最终分类器

$$G(x)=\mathrm{sign}(f(x))=\mathrm{sign}\left(\sum_{m=1}^{M}\alpha_m G_m(x)\right) \tag{8.7}$$

8.1.3 AdaBoost的例子

表 8.1 训练数据表

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	–1	–1	–1	1	1	1	–1

求解过程：初始化训练数据的权值分布，$D_1=(w_{11},w_{12},\cdots,w_{110}),w_{1i}=0.1,\quad i=1,2,\cdots,10$

m=1

(a) 无论阈值v取2.5，还是8.5，总得分错3个样本，故可任取其中任意一个如2.5，得到第一个基本分类器为：

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

(b) G_1在训练数据上的误差率 $e_1 = P(G_1(x_i) \neq y_i) = 0.3$

(c) 计算G_1的系数： $\alpha_1 = \frac{1}{2}\log\frac{1-e_1}{e_1} = 0.4236$

(d) 更新权值分布

$$D_2 = (w_{21}, \cdots, w_{2i}, \cdots, w_{210})$$

$$w_{2i} = \frac{w_{1i}}{Z_1}\exp(-\alpha_1 y_i G_1(x_i)), \quad i = 1, 2, \cdots, 10$$

$$D_2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715)$$

$$f_1(x) = 0.4236G_1(x)$$

分类器$\text{sign}[f_1(x)]$在训练数据集上有3个误分类点。

```
x = 0:9;y = [1 1 1 -1 -1 -1 1 1 1 -1];
D = 1/10; v = 2.5;
G = 2*(x<v)-1;
e = sum(D.*double(G~=y));
alpha = 0.5*log((1-e)/e);
Z = sum(D.*exp(-alpha*y.*G));
D = D.*exp(-alpha*y.*G)./Z;format long; D'
```

```
ans = 10×1
   0.071428571428571
   0.071428571428571
   0.071428571428571
   0.071428571428571
   0.071428571428571
   0.071428571428571
   0.166666666666667
   0.166666666666667
   0.166666666666667
   0.071428571428571
```

m=2

(a) 在权值分布为D_i的训练数据上，阈值v是8.5时分类误差率最低， 基本分类器为

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

(b) $G_2(x)$在训练数据上的误差率$e_2 = 0.2143$.

(c) $\alpha_2 = 0.6496$

(d) 更新权值分布

$$D_3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.1667, 0.1060, 0.1060, 0.1060, 0.0455)$$
$$f_2(x) = 0.4236G_1(x) + 0.6496G_2(x)$$

分类器$\mathrm{sign}[f_2(x)]$在训练数据集上有3个误分类点.

```
v = 8.5;
G = 2*(x<v)-1;
e = sum(D.*double(G~=y));
alpha = 0.5*log((1-e)/e);
Z = sum(D.*exp(-alpha*y.*G));
D = D.*exp(-alpha*y.*G)./Z;format long; D'
```

```
ans = 10×1
    0.045454545454545
    0.045454545454545
    0.045454545454545
    0.166666666666667
    0.166666666666667
    0.166666666666667
    0.106060606060606
    0.106060606060606
    0.106060606060606
    0.045454545454545
```

m=3

(a) 在权值分布为D_3的训练数据上，阈值v是5.5时分类误差率最低， 基本分类器为

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

(b) $G_3(x)$在训练样本集上的误差率$e_3 = 0.1820$.

(c) 计算$\alpha_3 = 0.7514$.

(d) 更新训练数据的权值分布:

$$D_4 = (0.125, 0.125, 0.125, 0.102, 0.102, 0.102, 0.065, 0.065, 0.065, 0.125)$$

$$f_3(x) = 0.4236G_1(x) + 0.6496G_2(x) + 0.7514G_3(x)$$

最终分类器为

$$G(x) = \mathrm{sign}[f_3(x)] = \mathrm{sign}[0.4236G_1(x) + 0.6496G_2(x) + 0.7514G_3(x)]$$

分类器$\mathrm{sign}[f_3(x)]$在训练数据集上有0个误分类点.

```
v = 5.5;
G = 2*(x<v)-1;
e = sum(D.*double(G~=y));
```

```
alpha = 0.5*log((1-e)/e);
Z = sum(D.*exp(-alpha*y.*G));
D = D.*exp(-alpha*y.*G)./Z;format long; D'
```

```
ans = 10×1
    0.125000000000000
    0.125000000000000
    0.125000000000000
    0.101851851851852
    0.101851851851852
    0.101851851851852
    0.064814814814815
    0.064814814814815
    0.064814814814815
    0.125000000000000
```

8.2 AdaBoost算法的训练误差分析

定理8.1（AdaBoost的训练误差界）AdaBoost算法最终分类器的训练误差界为

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

其中$G(x) = \mathrm{sign}(f(x))$, $f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$, $Z_m = \sum_{i=1}^{N} w_{mi} \exp\left(-\alpha_m y_i G_m(x_i)\right)$.

证明 前半部分：当$G(x_i) \neq y_i$时，$y_i f(x_i) < 0$，$\exp\left(-y_i f(x_i)\right) \geq 1$，可得。

后半部分：$w_{m,i} \exp\left(-\alpha_m y_i G_m(x_i)\right) = Z_m w_{m+1,i}$

$$\begin{aligned}
&\frac{1}{N}\sum_{i} \exp\left(-y_i f(x_i)\right) \\
&= \frac{1}{N}\sum_{i} \exp\left(-\sum_{m=1}^{M} \alpha_m y_i G_m(x_i)\right) \\
&= \sum_{i} w_{1i} \prod_{m=1}^{M} \exp\left(-\alpha_m y_i G_m(x_i)\right) \\
&= Z_1 \sum_{i} w_{2i} \prod_{m=2}^{M} \exp\left(-\alpha_m y_i G_m(x_i)\right) \\
&= Z_1 Z_2 \sum_{i} w_{3i} \prod_{m=3}^{M} \exp\left(-\alpha_m y_i G_m(x_i)\right) \\
&= \cdots \\
&= Z_1 Z_2 \cdots Z_{M-1} \sum_{i} w_{Mi} \exp\left(-\alpha_M y_i G_M(x_i)\right) \\
&= \prod_{m=1}^{M} Z_m
\end{aligned}$$

这一定理说明，可以在每一轮选取适当的G_m，使Z_m最小，从而使训练误差下降最快。

定理 **8.2** （二类分类问题 **AdaBoost** 的训练误差界）

$$\prod_{m=1}^{M} Z_m = \prod_{m=1}^{M}[2\sqrt{e_m(1-e_m)}] = \prod_{m=1}^{M}\sqrt{(1-4\gamma_m^2)} \le \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)$$

其中$\gamma_m = \frac{1}{2} - e_m$

证明：

$$\begin{aligned} Z_m &= \sum_{i=1}^{N} w_{mi}\exp\left(-\alpha_m y_i G_m(x_i)\right) \\ &= \sum_{y_i=G_m(x_i)} w_{mi}\mathrm{e}^{-\alpha_m} + \sum_{y_i\neq G_m(x_i)} w_{mi}\mathrm{e}^{\alpha_m} \\ &= (1-e_m)\mathrm{e}^{-\alpha_m} + e_m\mathrm{e}^{\alpha_m} \\ &= 2\sqrt{e_m(1-e_m)} = \sqrt{1-4\gamma_m^2} \end{aligned}$$

由泰勒展开式

$$e^{-\frac{x}{2}} = 1 - \frac{x}{2} + \frac{x^2}{8} + o(x^3), \sqrt{1-x} = 1 - \frac{x}{2} - \frac{x^2}{8} + o(x^3)$$

$$\sqrt{1-x} < e^{-\frac{x}{2}} \Rightarrow \sqrt{(1-4\gamma_m^2)} \le \exp\left(-2\gamma_m^2\right)$$

推论 **8.1** 如果存在$\gamma > 0$，对所有m有$\gamma_m \ge \gamma$，则

$$\frac{1}{N}\sum_{i=1}^{N} I(G(x_i) \neq y_i) \le \exp\left(-2M\gamma^2\right)$$

8.3 AdaBoost算法的解释

AdaBoost 算法还有另一个解释， 即可以认为 AdaBoost 算法是模型为加法模型、损失函数为指数函数、学习算法为前向分步算法时的二类分类学习方法。

8.3.1 前向分步算法

考虑加法模型：

$$f(x) = \sum_{m=1}^{M}\beta_m b(x;\gamma_m)$$

$b(x;\gamma_m)$基函数，γ_m基函数的参数。

经验风险极小化：

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

前向分步算法(forward stagewise algorithm)求解这一优化问题的想法是：因为学习的是加法模型，如果能够从前向后，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式, 那么就可以简化优化的复杂度。具体地，每步只需优化如下损失函数；

$$\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, \beta b(x_i;\gamma))$$

算法**8.2**（前向分步算法）

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，损失函数$L(y,f(x))$，基函数集$\{b(x;\gamma)\}$

输出：加法模型$f(x)$

(1)初始化 $f_0(x)=0$

(2) 对$m=1,2,,,,,M$

- (a) 极小化损失函数：$(\beta_m,\gamma_m)=\arg\min_{\beta,\gamma}\sum_{i=1}^{N} L(y_i,f_{m-1}(x_i)+\beta b(x_i;\gamma))$，得到参数$\beta_m,\gamma_m$.
- (b)更新 $f_m(x)=f_{m-1}(x)+\beta_m b(x;\gamma_m)$

(3) 得到加法模型 $f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b(x;\gamma_m)$

8.3.2 前向分步算法与AdaBoost

定理**8.3** AdaBoost算法是前向分步加法算法的特例．这时模型是由基本分类器组成的加法模型，损失函数是指数函数。

证明：前向分步算法学习的是加法模型，当基函数为基本分类器时，该加法模型等价于AdaBoost的最终分类器

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

假设经过m-1轮迭代前向分步算法已经得到$f_{m-1}(x)$，

$$\begin{aligned} f_{m-1}(x) &= f_{m-2}(x)+\alpha_{m-1}G_{m-1}(x) \\ &= \alpha_1 G_1(x)+\cdots+\alpha_{m-1}G_{m-1}(x) \end{aligned}$$

在第m轮迭代得到$\alpha_m, G_m(x)$和$f_m(x)$，

$$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$$

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

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

$$\bar{w}_{mi} = \exp\left[-y_i f_{m-1}(x_i)\right]$$

$$\begin{aligned}
&\sum_{i=1}^{N} \bar{w}_{mi} \exp\left[-y_i \alpha G(x_i)\right] \\
&= \sum_{y_i = G_m(x_i)} \bar{w}_{mi} \mathrm{e}^{-\alpha} + \sum_{y_i \neq G_m(x_i)} \bar{w}_{mi} \mathrm{e}^{\alpha} \\
&= (\mathrm{e}^{\alpha} - \mathrm{e}^{-\alpha}) \sum_{i=1}^{N} \bar{w}_{mi} I(y_i \neq G_m(x_i)) + \mathrm{e}^{-\alpha} \sum_{i=1}^{N} \bar{w}_{mi}
\end{aligned}$$

求解上述问题分两步:

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

此分类器$G_m^*(x)$即为 AdaBoost 算法的基本分类器$G_m(x)$，因为它是使第m 轮加权训练数据分类误差率最小的基本分类器.

$$\Rightarrow \alpha_m^* = \frac{1}{2}\log\frac{1-e_m}{e_m}, \qquad e_m = \frac{\sum_{i=1}^{N} \bar{w}_{mi} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \bar{w}_{mi}} = \sum_{i=1}^{N} \bar{w}_{mi} I(y_i \neq G_m(x_i))$$

其中

这里的α_m^*与 AdaBoost 算法第 2(c) 步的α_m完全一致。

最后来看每一轮样本权值的更新. 由$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$， $\bar{w}_{mi} = \exp\left[-y_i f_{m-1}(x_i)\right]$

$\Rightarrow \bar{w}_{m+1,i} = \bar{w}_{m,i} \exp\left[-y_i \alpha_m G_m(x)\right]$

这与 AdaBoost 算法第 2(d) 步的样本权值的更新只相差规范化因子， 因而等价。

8.4 提升树

提升树是以分类树或回归树为基本分类器的提升方法. 提升树被认为是统计学习中性能最好的方法之一.

8.4.1 提升树模型

提升树模型可以表示为决策树的加法模型;

$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

其中， $T(x;\Theta_m)$表示决策树； Θ_m为决策树的参数； M为树的个数.

8.4.2 提升树算法

提升树算法采用前向分步算法． 首先确定初始提升树 $f_0(x)=0$，第m步的模型是

$$f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$$

$$\hat{\Theta}_m=\arg\min_{\Theta_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T(x_i;\Theta_m))$$

对于二类分类问题，提升树算法只需将AdaBoost 算法 8.1 中的基本分类器限制为二类分类树即可。

回归问题的提升树

如果将输入空间$\mathcal{X}$划分为J个互不相交的区域$R_1,R_2,\cdots,R_J$，并且在每个区域上确定输出的常量c_j，那么树可表示为 $T(x;\Theta)=\sum_{j=1}^{J}c_jI(x\in R_j)$

$\Theta=\{(R_1,c_1),(R_2,c_2),\cdots,(R_J,c_J)\}$.

回归问题提升树使用以下前向分步算法:

$$\begin{aligned}&f_0(x)=0\\&f_m(x)=f_{m-1}(x)+T(x;\Theta_m),\quad m=1,2,\cdots,M\\&f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)\end{aligned}$$

在前向分步算法的第m步，给定当前模型$f_{m-1}(x)$，需求解

$$\hat{\Theta}_m=\arg\min_{\Theta_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T(x_i;\Theta_m))$$

当采用平方误差损失函数时， $L(y,f(x))=(y-f(x))^2$

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

这里, $r=y-f_{m-1}(x)$.

算法**8.3** （回归问题的提升树算法）

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\},x_i\in\mathcal{X}\subseteq\mathbf{R}^n,y_i\in\mathcal{Y}\subseteq\mathbf{R}$

输出：提升树$f_M(x)$

(I) 初始化 $f_0(x)=0$

(2) 对$m=1,2,\cdots,M$

(a) 计算残差 $$r_{mi}=y_i-f_{m-1}(x_i),\quad i=1,2,\cdots,N$$

(b) 拟合残差r_{mi}学习一个回归树，得到$T(x;\Theta_m)$

(c) 更新 $$f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$$

(3) 得到回归问题提升树

$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

例**8.2** 已知如表8.2所示的训练数据，x的取值范围为区间[0.5,10.5], y的取值范围为区间[5.0,10.0], 学习这个回归问题的提升树模型，考虑只用树桩作为基函数.

表 8.2 训练数据表

x_i	1	2	3	4	5	6	7	8	9	10
y_i	5.56	5.70	5.91	6.40	6.80	7.05	8.90	8.70	9.00	9.05

求解优化问题： $$\min_{s}\left[\min_{c_1}\sum_{x_i\in R_1}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2}(y_i-c_2)^2\right]$$

求切分点s: $$R_1=\{x|x\le s\},\quad R_2=\{x|x>s\}$$

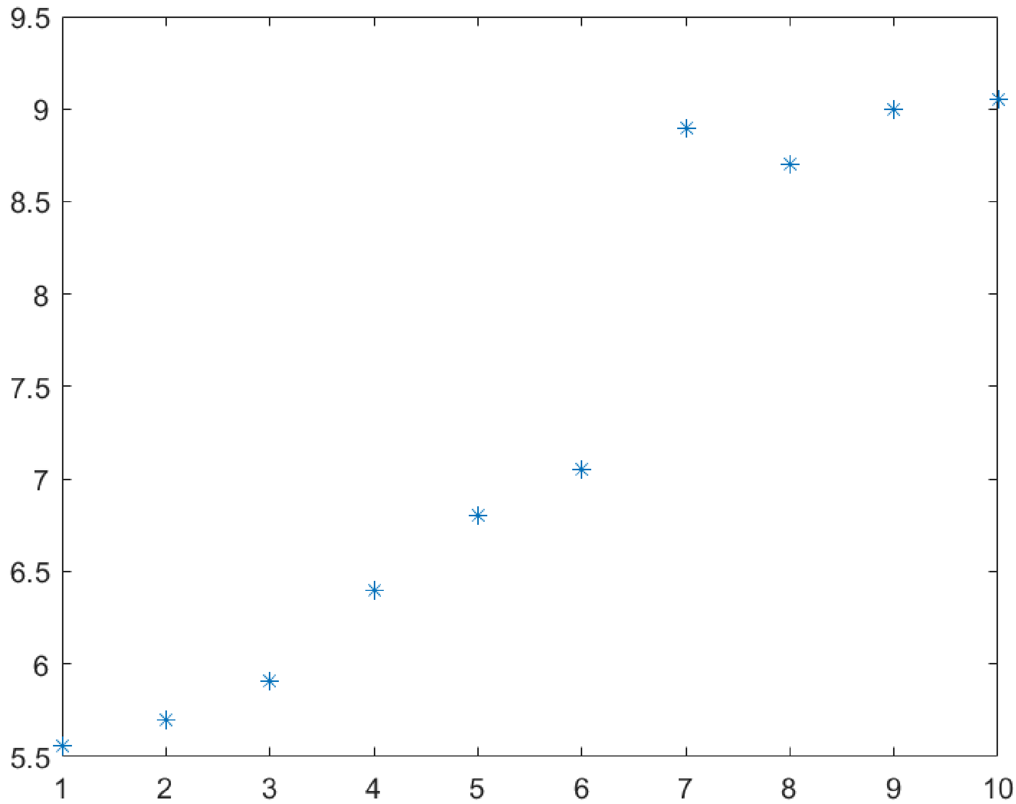
$$c_1=\frac{1}{N_1}\sum_{x_i\in R_1}y_i,\quad c_2=\frac{1}{N_2}\sum_{x_i\in R_2}y_i$$

$$m(s)=\min_{a}\sum_{x_i\in R_1}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2}(y_i-c_2)^2$$

表 8.3 计算数据表

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
$m(s)$	15.72	12.07	8.36	5.78	3.91	1.93	8.01	11.73	15.74

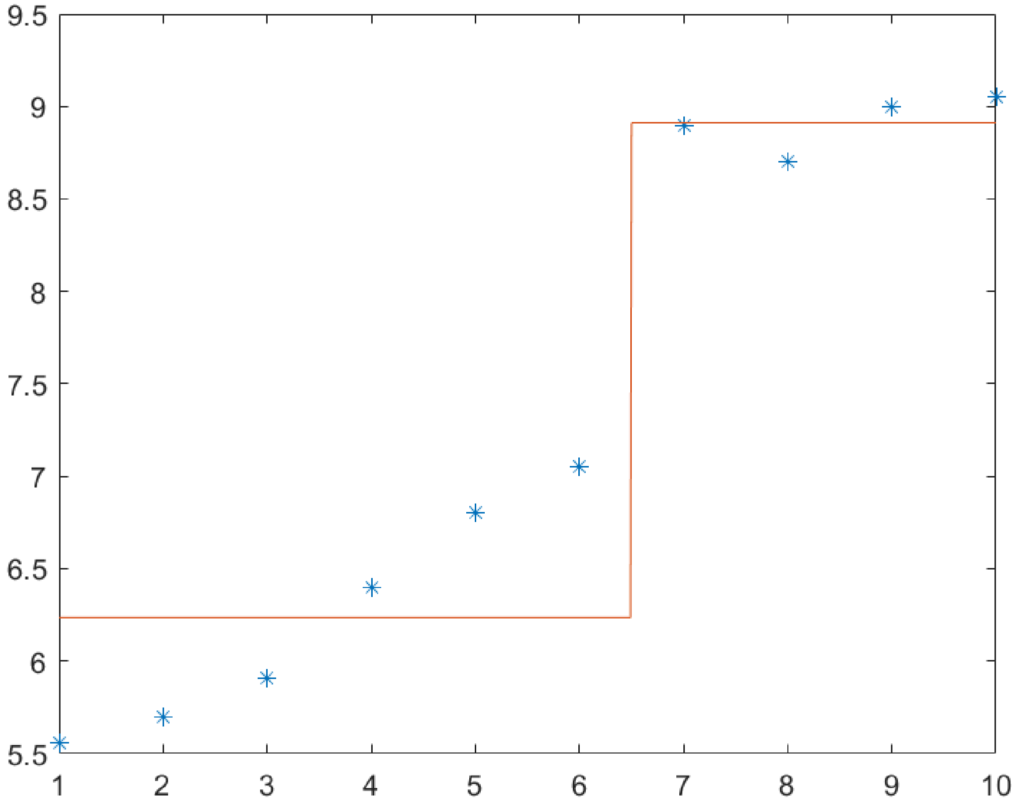
```
close all
x = 1:10;
xx = 1:0.01:10;
y = [5.56 5.70 5.91 6.40 6.80 7.05 8.90 8.70 9.00 9.05]; y0 = y;
plot(y,'*')
```




```
s = [1.5 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5];%切分点
[ms,c1,c2] = boosting_tree_regression(x,y,s);
[~,opt_index] = min(ms);
format short
ms,opt_s = s(opt_index), opt_c1 = c1(opt_index), opt_c2 = c2(opt_index)
```

```
ms = 1×9
   15.7231   12.0834    8.3656    5.7755    3.9113    1.9300    8.0098   11.7354 ⋯
opt_s = 6.5000
opt_c1 = 6.2367
opt_c2 = 8.9125
```

```
T1 = @(x) opt_c1*(x<opt_s)+opt_c2*(x>=opt_s);
figure,plot(y,'*');hold on; plot(xx,T1(xx));hold off
```



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

$$f_1(x) = T_1(x)$$

$$r_{2i} = y_i - f_1(x_i), \quad i = 1, 2, \cdots, 10$$

```
% T1 = f1(x)拟合数据的误差
r2 = y-T1(x)
```

```
r2 = 1×10
   -0.6767   -0.5367   -0.3267    0.1633    0.5633    0.8133   -0.0125   -0.2125 ···
```

```
loss = sum(r2.^2)
```

```
loss = 1.9300
```

第二步 求$T_2(x)$

表 8.4　残差表

x_i	1	2	3	4	5	6	7	8	9	10
r_{2i}	−0.68	−0.54	−0.33	0.16	0.56	0.81	−0.01	−0.21	0.09	0.14

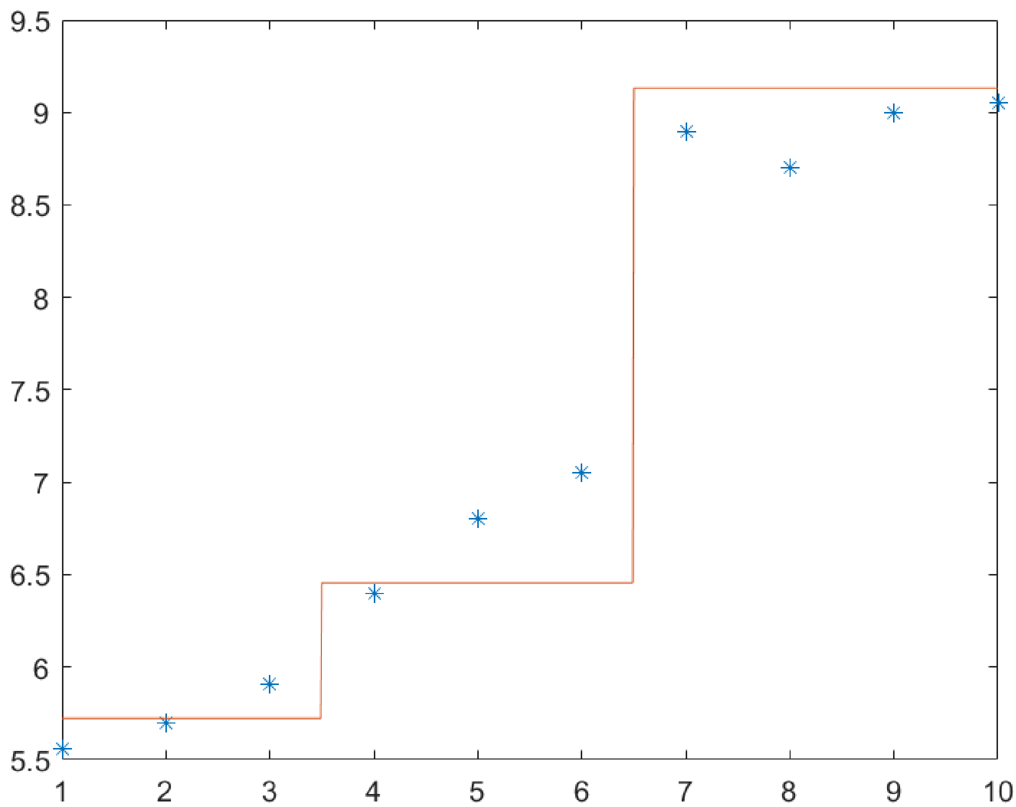
```
y = r2;
[ms,c1,c2] = boosting_tree_regression(x,y,s);
[~,opt_index] = min(ms);
ms,opt_s = s(opt_index), opt_c1 = c1(opt_index), opt_c2 = c2(opt_index)
```

```
ms = 1×9
    1.4213    1.0099    0.8007    1.1403    1.6654    1.9300    1.9299    1.8984 ···
opt_s = 3.5000
opt_c1 = -0.5133
opt_c2 = 0.2200
```

```
T2 = @(x) opt_c1*(x<opt_s)+opt_c2*(x>=opt_s);
loss = sum((y-T2(x)).^2)
```

```
loss = 0.8007
```

```
figure,plot(y0,'*');hold on; plot(xx,T1(xx)+T2(xx));hold off
```

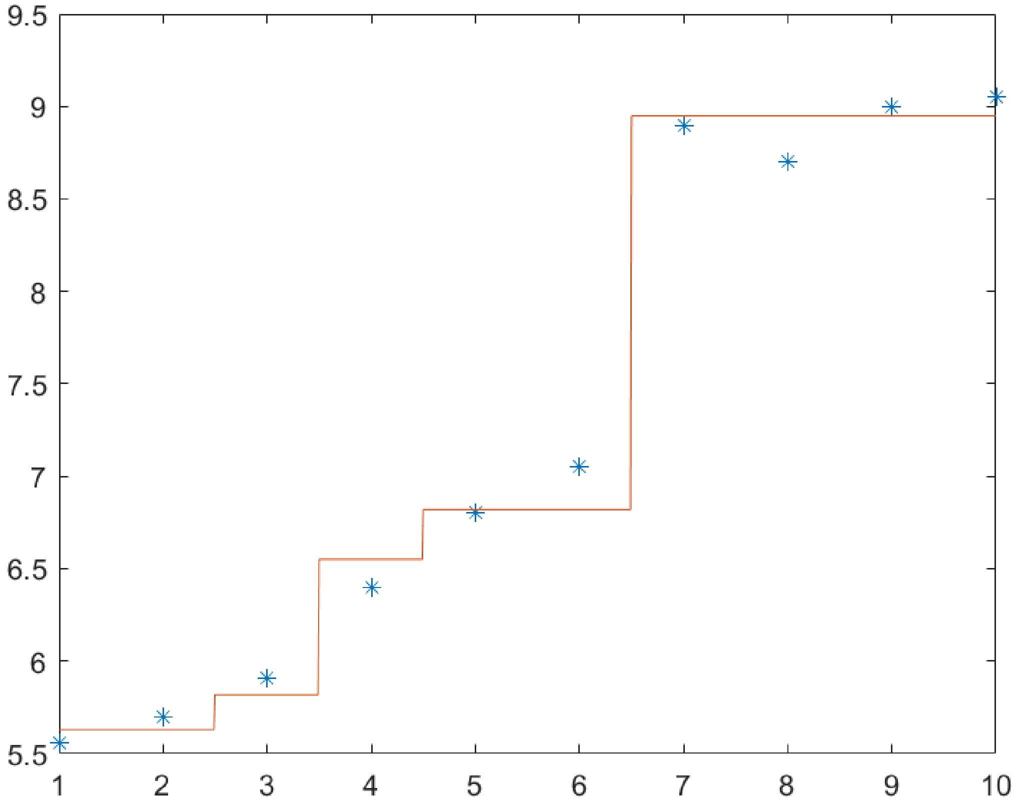



```
y = r2;
T = T2;
f = T1(xx)+T2(xx);
for kk = 3:6
    y = y-T(x)
    [ms,c1,c2] = boosting_tree_regression(x,y,s);
    [~,opt_index] = min(ms);
```

```
    m = kk, opt_s = s(opt_index), opt_c1 = c1(opt_index), opt_c2 = c2(opt_index)
    T = @(x) opt_c1*(x<opt_s)+opt_c2*(x>=opt_s);
    loss = sum((y-T(x)).^2)
    f = f + T(xx);
end
```

```
y = 1×10
   -0.1633   -0.0233    0.1867   -0.0567    0.3433    0.5933   -0.2325   -0.4325 ⋯
m = 3
opt_s = 6.5000
opt_c1 = 0.1467
opt_c2 = -0.2200
loss = 0.4780
y = 1×10
   -0.3100   -0.1700    0.0400   -0.2033    0.1967    0.4467   -0.0125   -0.2125 ⋯
m = 4
opt_s = 4.5000
opt_c1 = -0.1608
opt_c2 = 0.1072
loss = 0.3056
y = 1×10
   -0.1492   -0.0092    0.2008   -0.0425    0.0894    0.3394   -0.1197   -0.3197 ⋯
m = 5
opt_s = 6.5000
opt_c1 = 0.0715
opt_c2 = -0.1072
loss = 0.2289
y = 1×10
   -0.2206   -0.0806    0.1294   -0.1140    0.0180    0.2680   -0.0125   -0.2125 ⋯
m = 6
opt_s = 2.5000
opt_c1 = -0.1506
opt_c2 = 0.0377
loss = 0.1722
```

```
figure,plot(y0,'*');
hold on; plot(xx,f)
```



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

$$f_2(x) = f_1(x) + T_2(x) = \begin{cases} 5.72, & x < 3.5 \\ 6.46, & 3.5 \leq x < 6.5 \\ 9.13, & x \geq 6.5 \end{cases}$$

$$T_3(x) = \begin{cases} 0.15, & x < 6.5 \\ -0.22, & x \geq 6.5 \end{cases} \quad L(y, f_3(x)) = 0.47$$

$$T_4(x) = \begin{cases} -0.16, & x < 4.5 \\ 0.11, & x \geq 4.5 \end{cases} \quad L(y, f_4(x)) = 0.30$$

$$T_5(x) = \begin{cases} 0.07, & x < 6.5 \\ -0.11, & x \geq 6.5 \end{cases} \quad L(y, f_5(x)) = 0.23$$

$$T_6(x) = \begin{cases} -0.15, & x < 2.5 \\ 0.04, & x \geq 2.5 \end{cases} \quad L(y, f_6(x)) = 0.17$$

$$f_6(x) = f_5(x) + T_6(x) = T_1(x) + \cdots + T_5(x) + T_6(x)$$

$$=\begin{cases} 5.63, & x < 2.5 \\ 5.82, & 2.5 \le x < 3.5 \\ 6.56, & 3.5 \le< 4.5 \\ 6.83, & 4.5 \le x < 6.5 \\ 8.95, & x \ge 6.5 \end{cases}$$

8.4.3 梯度提升

算法8.4（梯度提升算法）

输入：训练数据集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}, x_i \in \mathcal{X} \subseteq \mathbf{R}^n, y_i \in \mathcal{Y} \subseteq \mathbf{R}$，损失函数$L(y,f(x))$

输出：回归树$\hat{f}(x)$

(1) 初始化 $$f_0(x)=\arg\min_c \sum_{i=1}^{N} L(y_i,c)$$

(2) 对$m=1,2,...,M$

(a) 对$i=1,2,\cdots,N$，计算

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

(b) 对r_{mi}拟合一个回归树，得到第m棵树的叶结点区域$R_{mj}, j=1,2,\cdots,J$

(c) 对$j=1,2,\cdots,J$，计算

$$c_{mj}=\arg\min_c \sum_{x_i \in R_{mj}} L(y_i,f_{m-1}(x_i)+c)$$

(d) 更新$f_m(x)=f_{m-1}(x)+\sum_{i=1}^{j} c_{mj} I(x \in R_{mj})$

(3) 得到回归树

$$\hat{f}(x)=f_M(x)=\sum_{m=1}^{M}\sum_{j=1}^{J} c_{mj} I(x \in R_{mj})$$

作业

习题8.1

```
function [ms,c1,c2] = boosting_tree_regression(x,y,s)
for i = 1:length(s)
    R1 = x<=s(i);
    R2 = x>s(i);
    c1(i) = mean(y(R1));
    c2(i) = mean(y(R2));
    ms(i) = sum((y(R1)-c1(i)).^2)+sum((y(R2)-c2(i)).^2);
end
end
```