

# 第1章

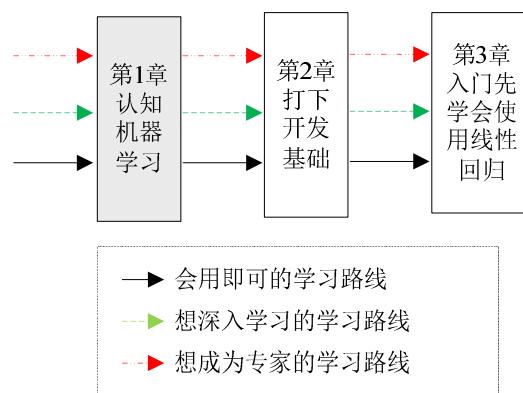


图 1-1 学习路线图

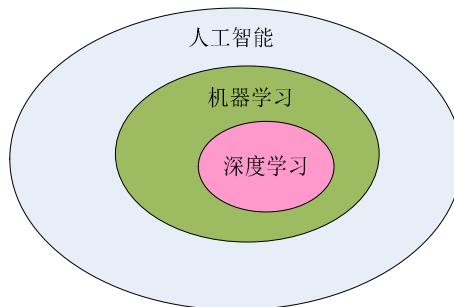


图 1-2 人工智能、机器学习、深度学习三者之间的关系

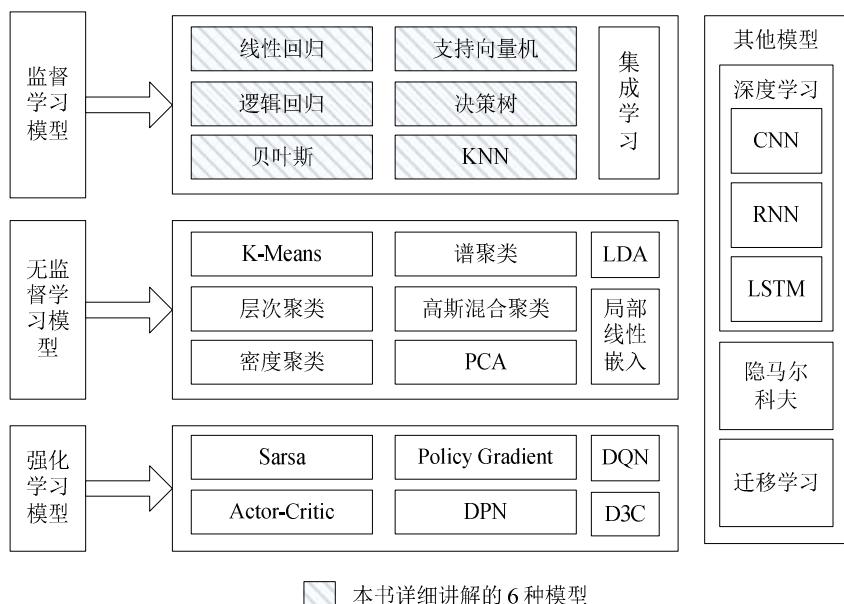


图 1-3 机器学习的模型

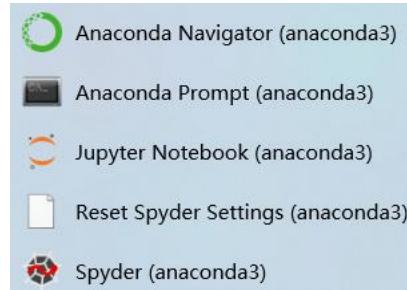


图 1-4 Anaconda 安装的 5 个工具

## 第 2 章

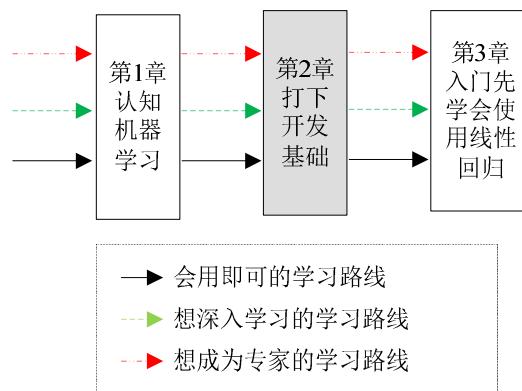


图 2-1 学习路线图

访问列表中的元素示例。在 Jupyter Notebook 的 Console 中输入：studentName[0] 和 studentLessonScore[1:5]，输出结果为：liuXiaoMing 和 [90, 77, 59, 67]。

```
studentName[0]: liuXiaoMing
studentLessonScore[1:5]: [90, 77, 59, 67]
```

图 2-2 访问列表中的元素

更新列表中的元素示例。在 Jupyter Notebook 的 Console 中输入：student[0] = 'LiuBin'，输出结果为：['liuXiaoMing', 'LiuBin']。

```
['liuXiaoMing', 'LiuBin']
```

图 2-3 更新列表中的元素

访问字典的值示例。在 Jupyter Notebook 的 Console 中输入：student['Name'] 和 student['score2']，输出结果为：LiuBin 和 90。

```
student['Name']: LiuBin
student['score2']: 90
```

图 2-4 访问字典的值

访问数组中的元素示例。在 Jupyter Notebook 的 Console 中输入：[[1 2 3], [4 5 6], [7 8 9]]，输出结果为：原数组：[[1 2 3], [4 5 6], [7 8 9]]，取第0行作为一维数组：[1 2 3]，取第0列作为一维数组：[1 4 7]，取第0行至第1行、第0列至第1列作为二维数组：[[1 2], [4 5]]。

```
原数组:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
取第0行作为一维数组: [1 2 3]
取第0列作为一维数组: [1 4 7]
取第0行至第1行、第0列至第1列作为二维数组:
[[1 2]
 [4 5]]
```

图 2-5 访问数组中的元素

```

Console 1/A
沿第0轴堆叠:
[[[ 1  2  3]
 [ 4  5  6]]

[[ 7  8  9]
 [10 11 12]]]
沿第1轴堆叠:
[[[ 1  2  3]
 [ 7  8  9]]

[[ 4  5  6]
 [10 11 12]]]
横向连接:
[[ 1  2  3  7  8  9]
 [ 4  5  6 10 11 12]]
纵向连接:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
In [17]:
```

图 2-6 堆叠或连接数组

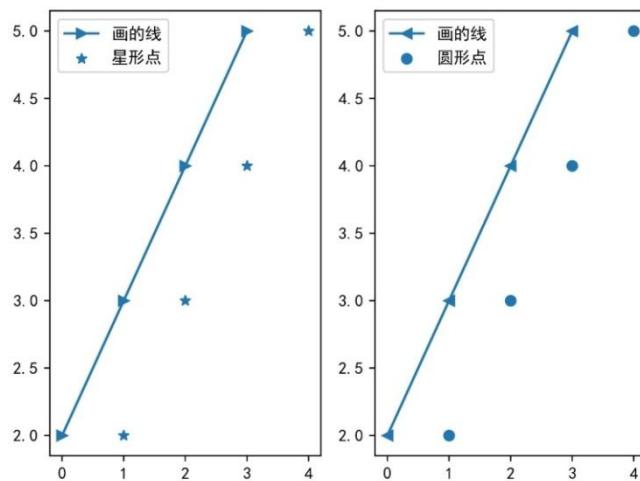


图 2-7 画一个含 2 个子图的图形

# 第 3 章

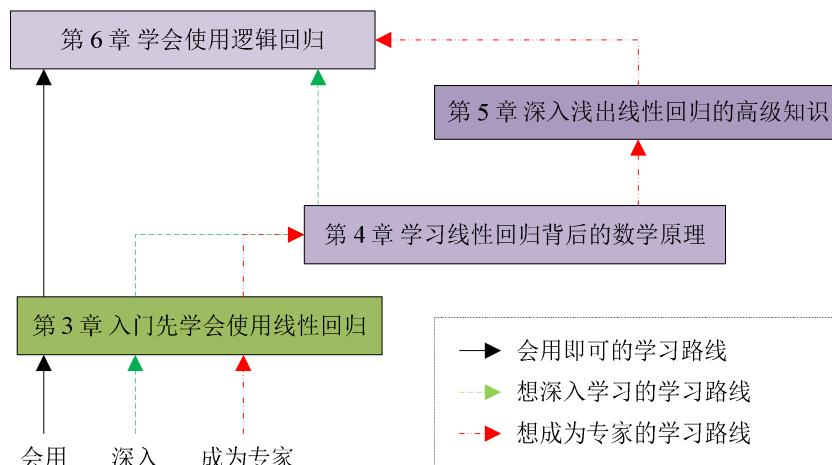


图 3-1 学习路线图

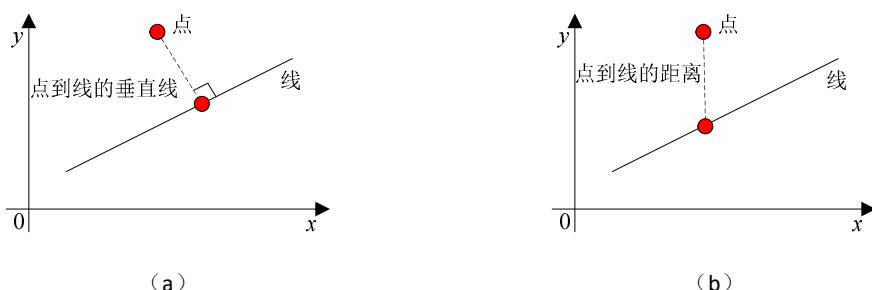


图 3-2 点到线的距离

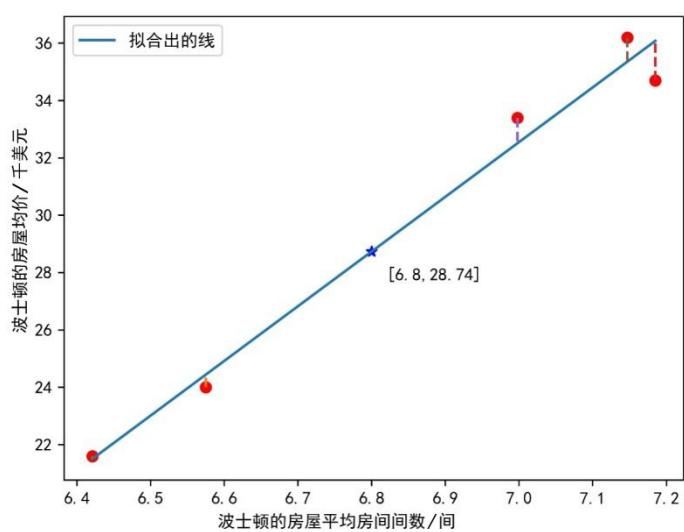


图 3-3 根据已知数据拟合出直线

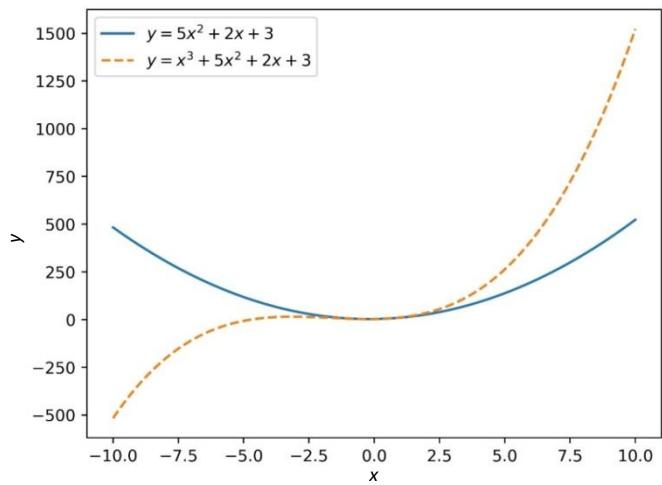


图 3-4 一元二次方程和一元三次方程的图形

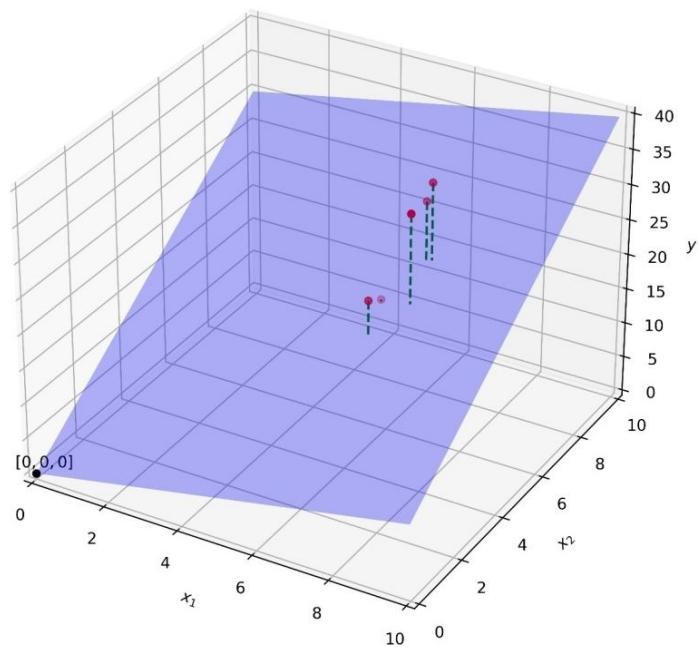


图 3-5  $y = x_1 + 3x_2$  图形

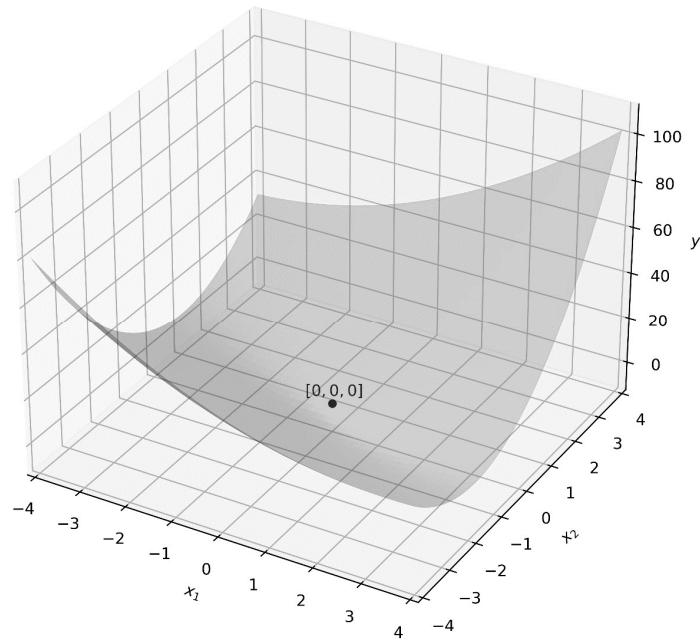


图 3-6  $y = x_1^2 + 2x_1x_2 + 3x_2^2 + x_1 + 2x_2 - 3$  的图形

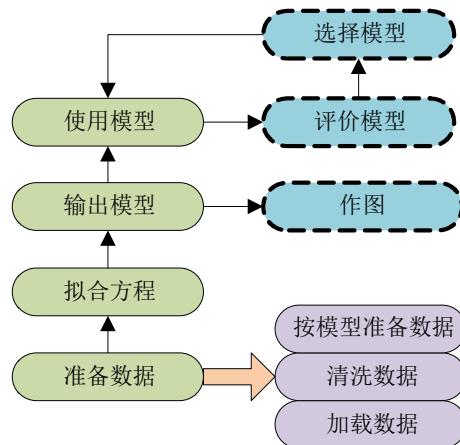


图 3-7 做线性回归的过程

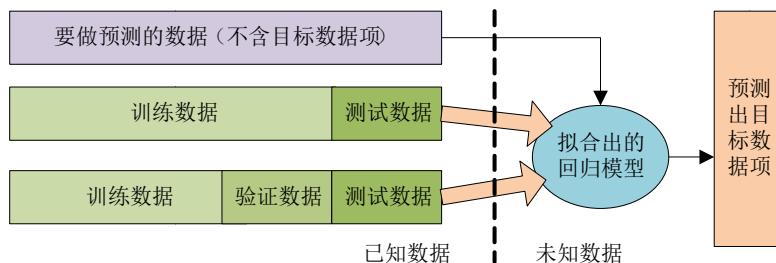
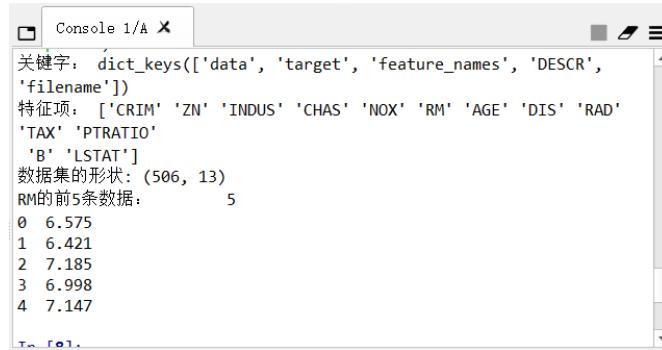


图 3-8 已知数据集的划分



```

关键字: dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
特征项: ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT']
数据集的形状: (506, 13)
RM的前5条数据: 5
0 6.575
1 6.421
2 7.185
3 6.998
4 7.147

```

图 3-9 查看加载的波士顿房价数据集

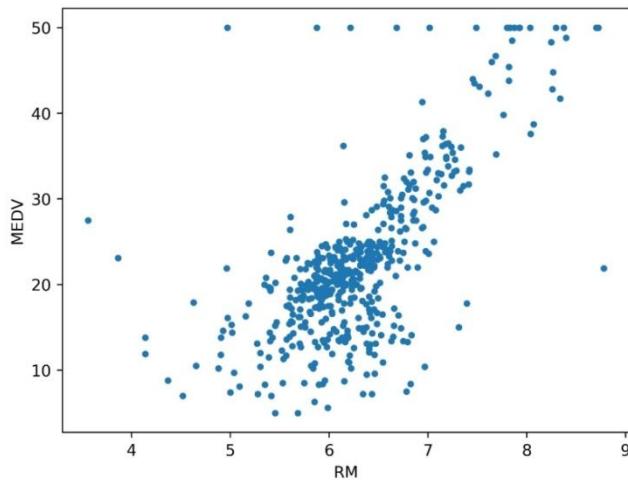
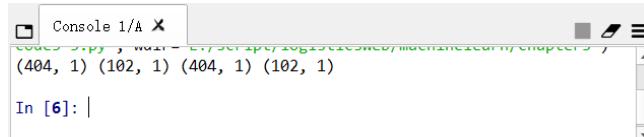


图 3-10 表达特征项 RM 与目标数据项 MEDV 关系的散点图

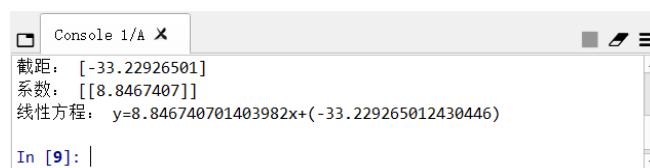


```

(404, 1) (102, 1) (404, 1) (102, 1)
In [6]: |

```

图 3-11 把波士顿房屋价格数据集划分成训练数据和测试数据的结果



```

截距: [-33.22926501]
系数: [[8.8467407]]
线性方程: y=8.846740701403982x+(-33.229265012430446)
In [9]: |

```

图 3-12 某次训练的输出结果



```

预测结果:
[[40.37152813]
[24.04248004]
[32.50438741]
[28.80061326]
[20.92446092]
[18.59946401]
[21.92475029]
[13.67912171]
[19.36545137]
[14.03958634]
[16.463711031]

```

图 3-13 一个预测的示例结果

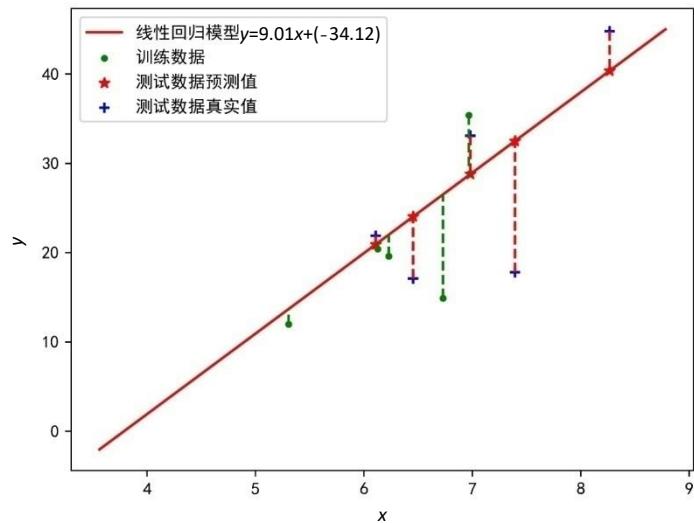


图 3-14 得到的线性回归模型

```

Console 1/A
使用模型对训练数据得到R^2: 0.4614107088025987
使用模型对测试数据得到R^2: 0.5721625318143688
使用metrics和模型对训练数据得到R^2: 0.4614107088025987
使用metrics和模型对测试数据得到R^2: 0.5721625318143688
使用metrics和模型对训练数据得到MAE: 4.473577106360655
使用metrics和模型对测试数据得到MAE: 4.365257521734308
使用metrics和模型对训练数据得到MSE: 45.67497069510493
使用metrics和模型对测试数据得到MSE: 35.40560537877825
使用metrics和模型对训练数据得到RMSE: 6.758326027582934
使用metrics和模型对测试数据得到RMSE: 5.950260950477571
In [3]: |

```

图 3-15 计算模型的 4 个评价指标的值



图 3-16 MinMaxScaler 的数据处理方向

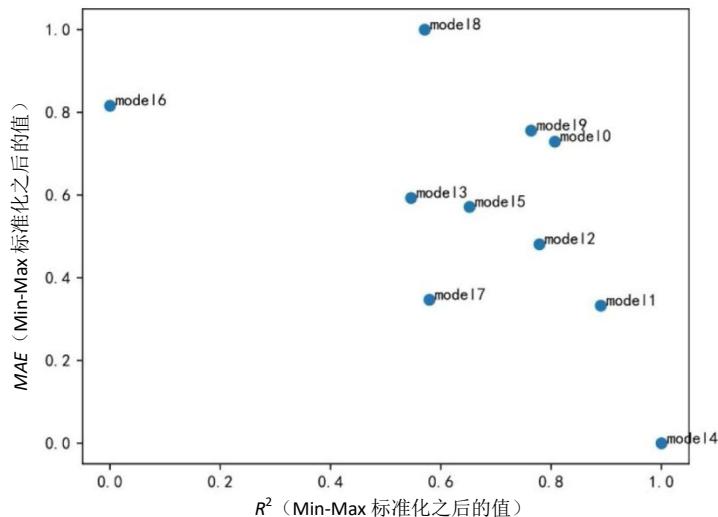


图 3-17 模型评价指标  $R^2$ 、MAE 的散点图

```

Console 1/A ×
codes>ipy, warn=False
第4个模型最优, 综合评价价值为: 1.0
In [18]:

```

图 3-18 自动找到最优模型的结果

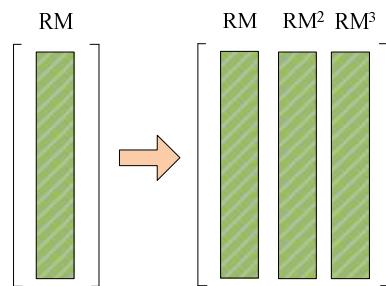


图 3-19 应模型要求数据集应做出的变化

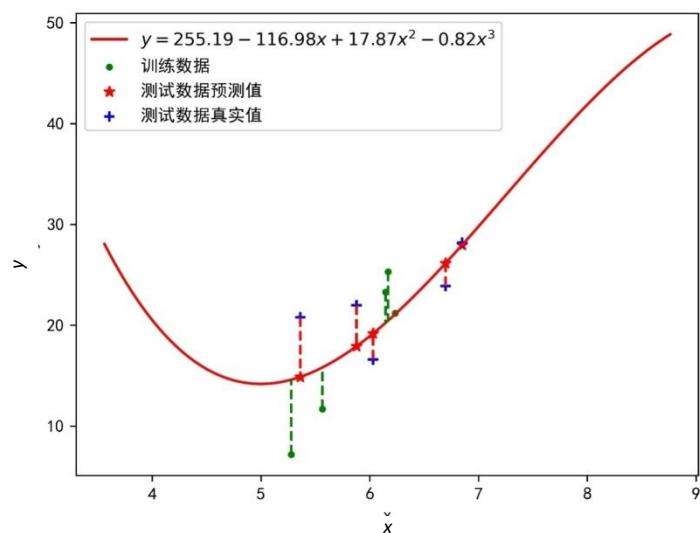


图 3-20 一元三次方程的图形

Console 1/A

```

模型5的方程: y=271.3-124.27x+18.91x^2-0.87x^3
R2: 0.55 MAE: 4.26
=====
模型6的方程: y=194.75-87.2x+12.94x^2-0.55x^3
R2: 0.24 MAE: 4.7
=====
模型7的方程: y=212.37-91.23x+12.97x^2-0.52x^3
R2: 0.4 MAE: 4.37
=====
模型8的方程: y=243.84-112.18x+17.22x^2-0.79x^3
R2: 0.61 MAE: 4.65
=====
模型9的方程: y=265.08-120.8x+18.3x^2-0.84x^3
R2: 0.61 MAE: 4.55
第4个模型最优, 综合评价值为: 1.0

In [6]:

```

图 3-21 对 10 个一元三次方程的综合评价

Console 1/A

```

模型5的方程: y=-34.07+8.74x1+0.47x2
R2: 0.52 MAE: 4.48
=====
模型6的方程: y=-38.75+9.51x1+0.43x2
R2: 0.17 MAE: 4.8
=====
模型7的方程: y=-36.1+9.06x1+0.49x2
R2: 0.48 MAE: 4.11
=====
模型8的方程: y=-34.3+8.8x1+0.41x2
R2: 0.48 MAE: 5.17
=====
模型9的方程: y=-31.71+8.21x1+0.62x2
R2: 0.54 MAE: 4.99
第4个模型最优, 综合评价值为: 1.0

In [8]:

```

图 3-22 对 10 个二元一次方程的综合评价

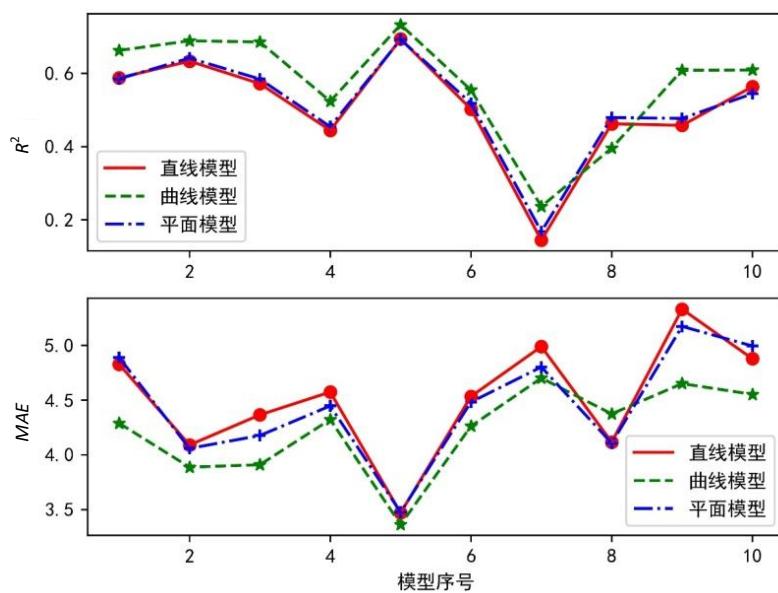


图 3-23 对比 3 种模型的评价指标

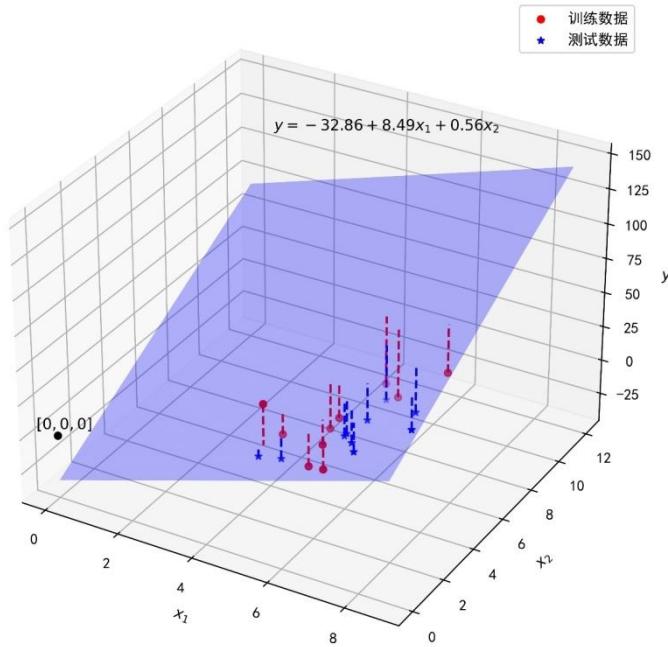


图 3-24 拟合出的平面

```

Console 1/A ×
模型7的方程:
y=182.77-68.56x1-11.67x2+3.81x1x2+7.98x1^2+0.07x2^2-0.14x1^2*x2-0.1
7*x1*x2^2-0.23x1^3+0.04x2^3
R2: 0.43 MAE: 4.14
=====
模型8的方程:
y=223.17-99.36x1+2.26x2-0.24x1x2+14.54x1^2-0.1x2^2+0.13x1^2*x2-0.13
x1*x2^2-0.65x1^3+0.04x2^3
R2: 0.68 MAE: 3.98
=====
模型9的方程:
y=225.18-106.0x1+22.52x2-6.37x1x2+16.43x1^2-0.35x2^2+0.59x1^2*x2-0.
08x1*x2^2-0.8x1^3+0.04x2^3
R2: 0.58 MAE: 4.4
第4个模型最优, 综合评价值为: 1.0
In [26]:

```

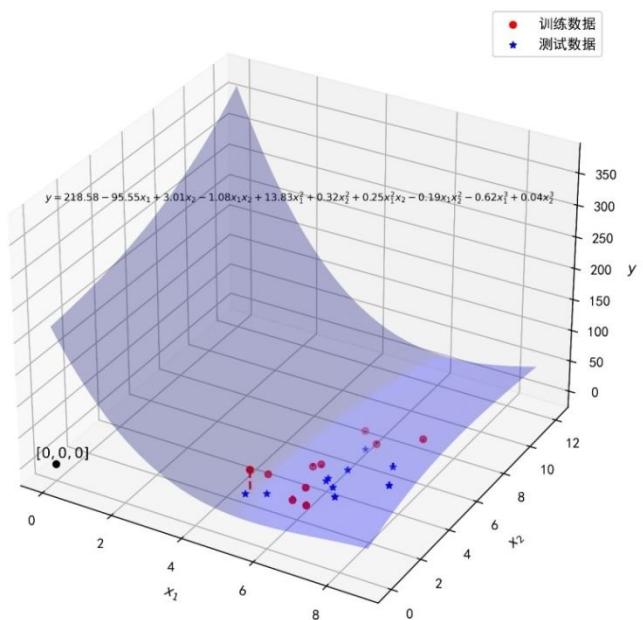
图 3-25 对 10 个二元三次方程的综合评价

```

##获得方程的特征项
dataBos=np.array(bos[[5,7]])
X=np.hstack((dataBos[:,0:1],\ ← x1
.....dataBos[:,1:2],\ ← x2
.....np.multiply(dataBos[:,0:1],dataBos[:,1:2]),\ ← x1x2
.....np.power(dataBos[:,0:1],2),\ ← x1^2
.....np.power(dataBos[:,1:2],2),\ ← x2^2
.....np.multiply(np.power(dataBos[:,0:1],2),dataBos[:,1:2]),\ ← x1^2x2
.....np.multiply(dataBos[:,0:1],np.power(dataBos[:,1:2],2)),\ ← x1x2^2
.....np.power(dataBos[:,0:1],3),\ ← x1^3
.....np.power(dataBos[:,1:2],3)\ ← x2^3
.....))

```

图 3-26 各个二维数组代数的特征项



# 第 4 章

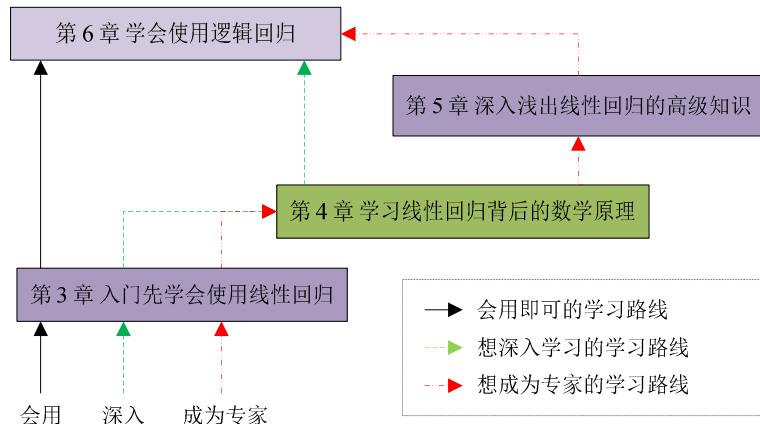


图 4-1 学习路线图

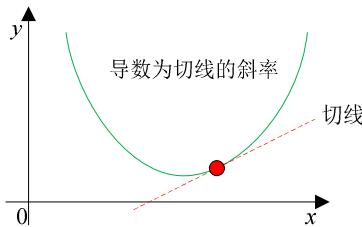


图 4-2 导数的几何意义

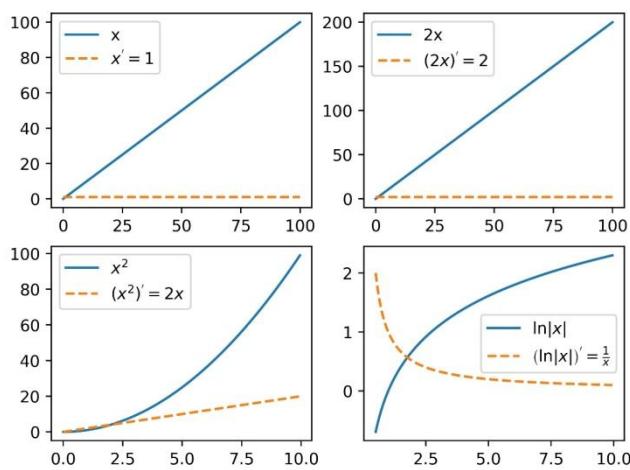


图 4-3 表 4-2 中第 1~4 个求导法则的示例

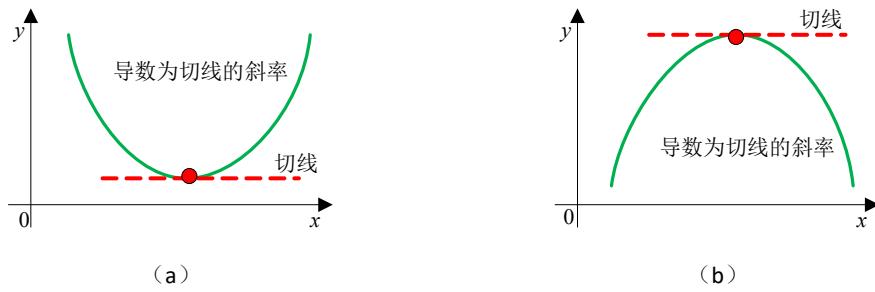


图 4-4 函数在最大值、最小值处的导数

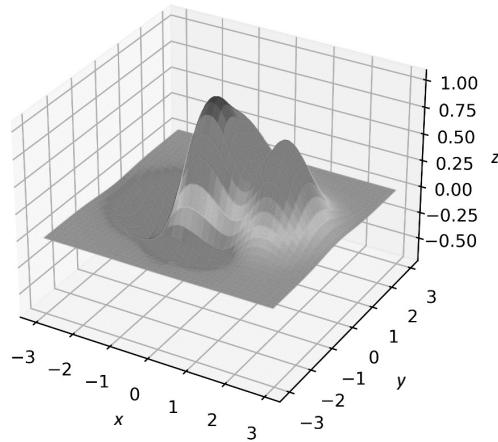


图 4-5 函数有多个极值的示例

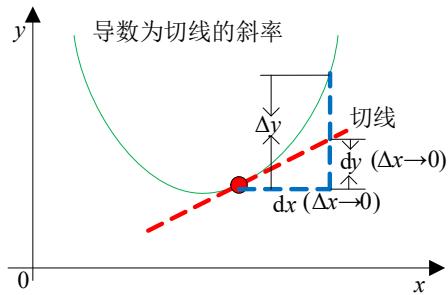


图 4-6 导数的含义示意图

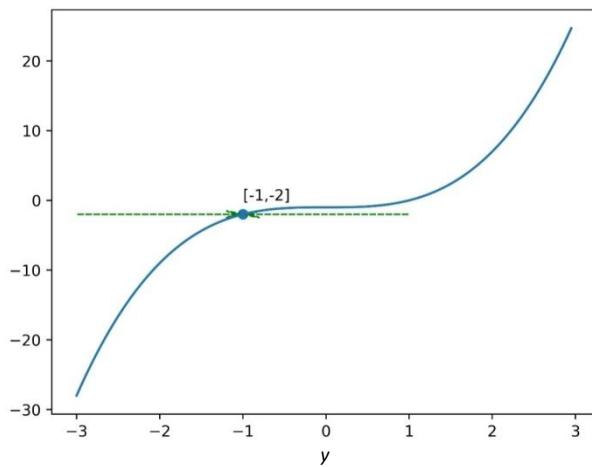


图 4-7 求  $\lim_{x \rightarrow -1} x^3$

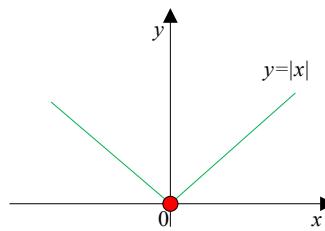


图 4-8 一个连续但不可导的函数示例

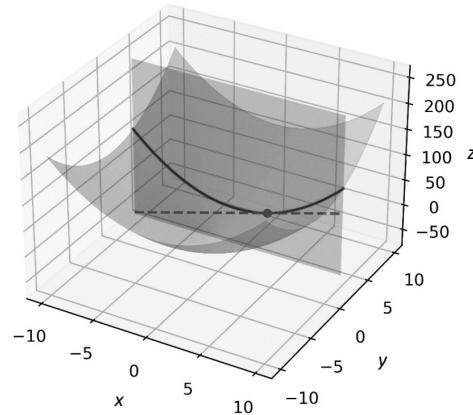


图 4-9 函数偏向  $x$  的导数

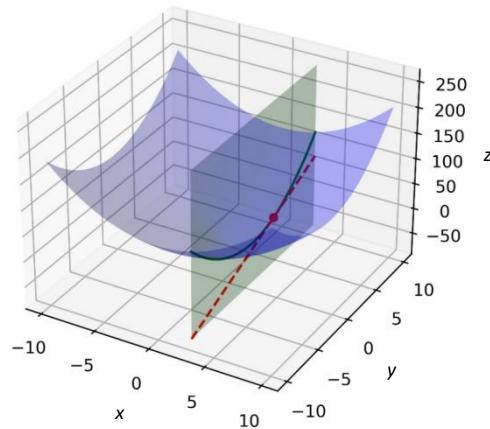


图 4-10 函数偏向  $y$  的导数

```
Console 1/A
det.py , warn= L:/Scrips/LogisticSweu/machinelearn/chapter4/
-3.0000000000000018
In [4]:
```

图 4-11 求行列式的值

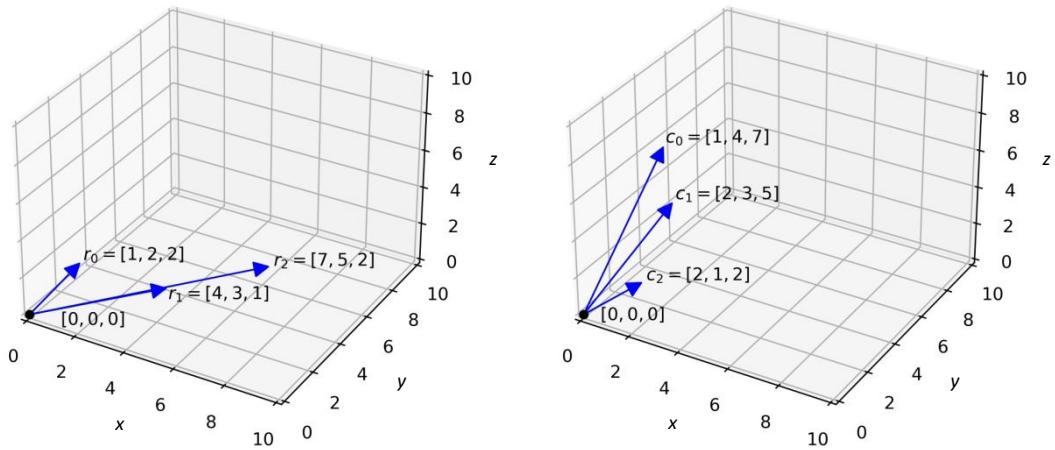


图 4-12 行列式的行向量与列向量

```
Console 1/A
det.py
-9.51619735392994e-16
In [2]:
```

图 4-13 求解行列式的值

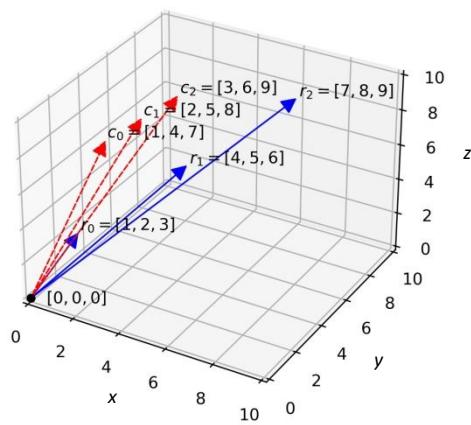
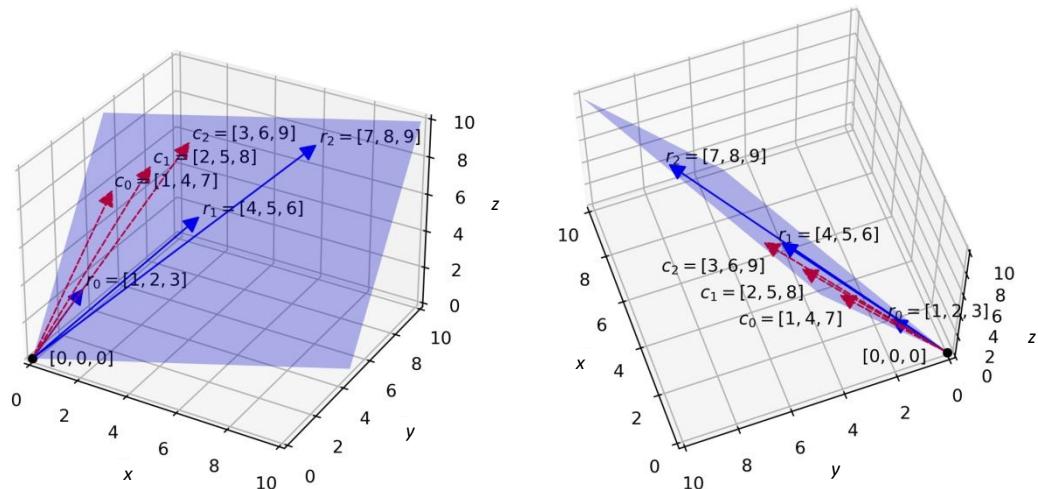


图 4-14 行列式的行向量和列向量



(a) 未调视角时的图形

(b) 调整视角后的图形

图 4-15 行向量和列向量位于同一个平面

图 4-16 求矩阵的秩

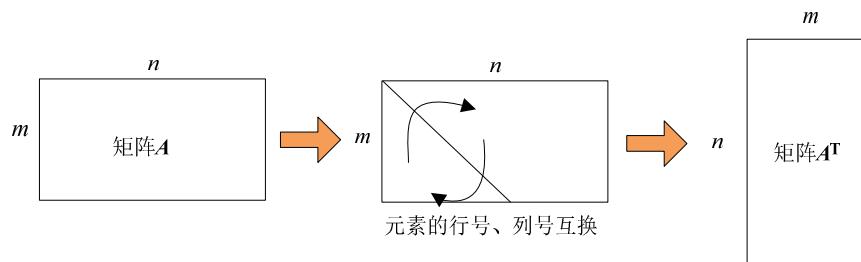
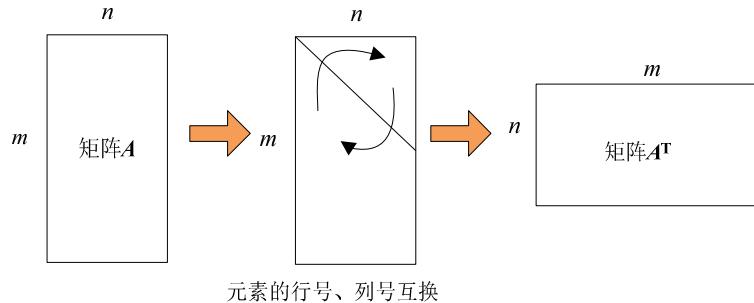
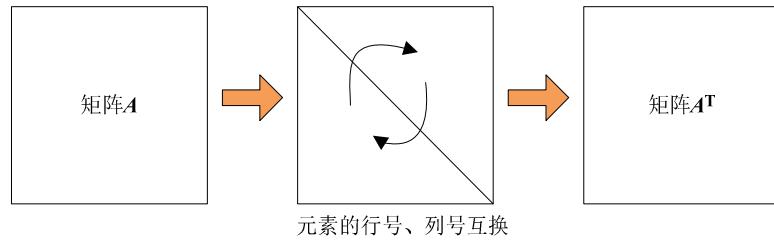


图 4-17 矩阵的转置

```
Console 1/A
矩阵的转置矩阵为:
[[1, 4, 7]
 [2, 5, 8]
 [3, 6, 9]]
In [5]:
```

图 4-18 矩阵的转置运算结果

```
Console 1/A
矩阵matrix1的逆矩阵为:
[[ 3.15251974e+15 -6.30503948e+15 3.15251974e+15]
 [-6.30503948e+15 1.26100790e+16 -6.30503948e+15]
 [ 3.15251974e+15 -6.30503948e+15 3.15251974e+15]]
矩阵matrix2的逆矩阵为:
[[-0.33333333 -2. 1.33333333]
 [ 0.33333333 4. -2.33333333]
 [ 0.33333333 -3. 1.66666667]]
In [10]:
```

图 4-19 求矩阵的逆矩阵

$$B = \begin{bmatrix} -4 & 8 & 1 \\ 5 & 2 & 3 \end{bmatrix}$$

$$A = \begin{bmatrix} 3 & -5 \\ 6 & 9 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

图 4-20 矩阵乘法的计算过程

```
Console 1/A X
[[ -37 14 -12], [ 21 66 33]]
In [2]:
```

图 4-21 矩阵乘法的计算结果

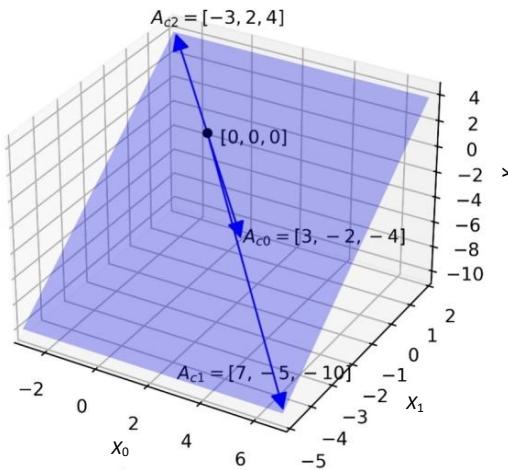


图 4-22  $R(\mathbf{A}) = 2$ 、 $m = n = 3$  时矩阵  $\mathbf{A}$  的列向量及列空间

```
Console 1/A X
模型对训练数据的MAE: 4.39
模型对测试数据的R^2: 0.56
模型对测试数据的MAE: 4.88
根据训练数据对模型的综合评价指标判断, 第6个模型最优, 综合评价值为:
0.93
使用训练数据的模型综合评价结果: [0.4108073512898316,
0.1947454475728576, 0.3526666035109449, 0.5196501452042463, 0.0,
0.45975687250388764, 0.9263653308907502, 0.4048674439652733,
0.7267141647404449, 0.3430684851105521]
根据测试数据对模型的综合评价指标判断, 第4个模型最优, 综合评价值为:
1.0
使用测试数据的模型综合评价结果: [0.5387959074384279,
0.7785541821687656, 0.648764997662773, 0.4764132676051653, 1.0,
0.5402120331145832, 0.09180210863445182, 0.6159258593708536,
0.285441369575901, 0.503583824264562]
In [16]:
```

图 4-23 求解一元一次方程模型的参数

```
Console 1/A X
模型对测试数据的MAE: 4.99
根据训练数据对模型的综合评价指标判断, 第6个模型最优, 综合评价值为:
0.92
使用训练数据的模型综合评价结果: [0.43201725889763276,
0.2088906167721547, 0.32283163714627094, 0.5246986626547803,
-8.881784197001252e-16, 0.4616910145928581, 0.9158782053451375,
0.41586630315526785, 0.7148058439150788, 0.4062906328181948]
根据测试数据对模型的综合评价指标判断, 第4个模型最优, 综合评价值为: 1.0
使用测试数据的模型综合评价结果: [0.4811926273647419,
0.7797524148250357, 0.6904939648233557, 0.48678608126192013, 1.0,
0.537638508047696, 0.10926017817049272, 0.6110396087728327,
0.2945790196201853, 0.4116966675643864]
In [2]:
```

图 4-24 求解多元一次方程模型的参数

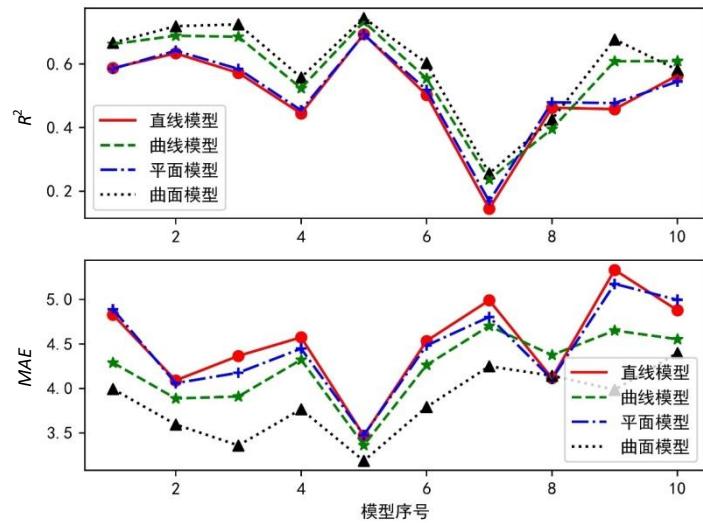


图 4-25 4 种模型的评价指标对比

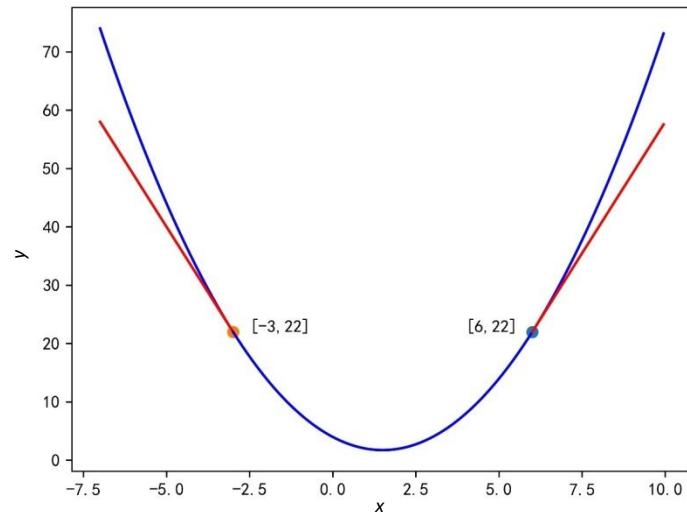


图 4-26 函数的梯度和切线

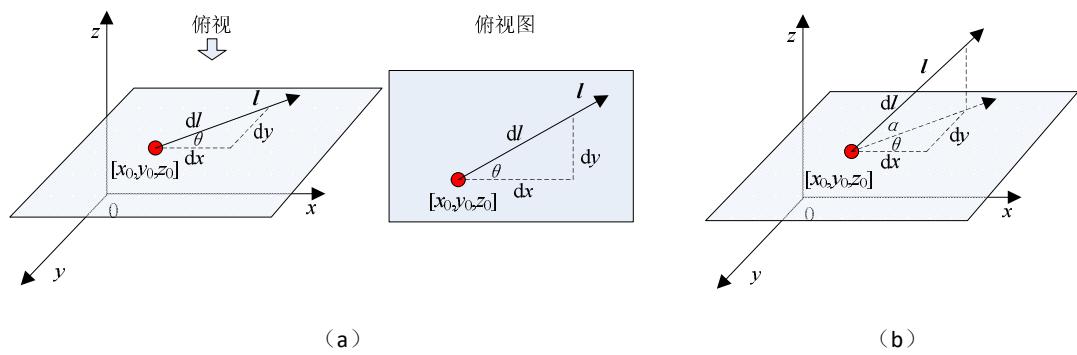


图 4-27 方向导数图示

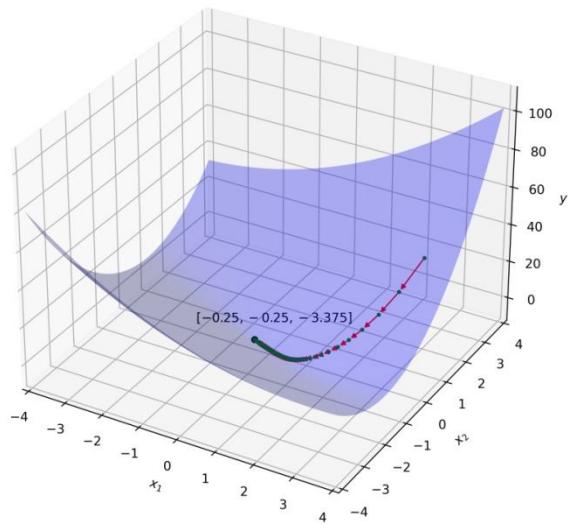


图 4-28 用梯度下降法求极小值

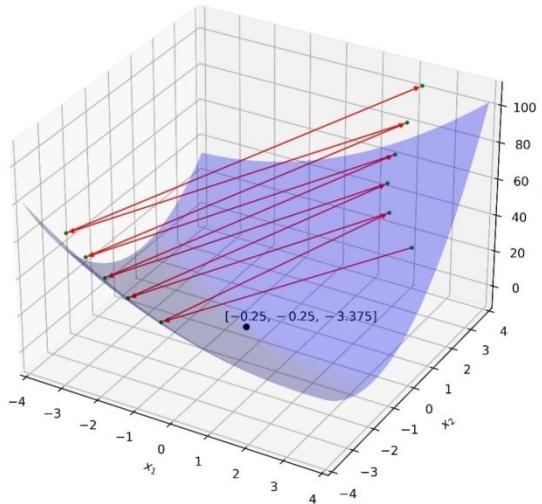


图 4-29 学习率过大时的情形

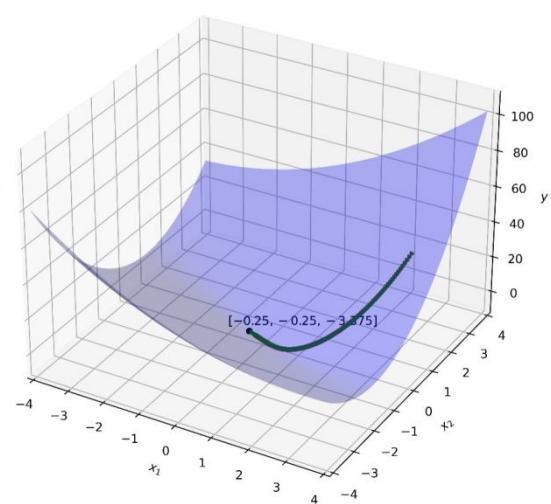


图 4-30 学习率过小时的情形

```

Console 1/A ✘
Reloaded modules: common, common.common
起点时的梯度: [11, 20]
在第219次迭代退出,此时梯度的模:
0.0009919175187914431
极小值: [-0.24921779317965836,
-0.25032400067356625, -3.3749995800942543]

In [12]:

```

图 4-31 设置迭代出口时的运行结果

```
Console 1/A x
logisticswe0/machinelearn/chapter4
在第604次迭代退出,此时两次迭代误差函数值相差:
0.0002986636512609664
此时误差为: 22
方程为: y=-29.78+8.06x1+0.47x2
模型对训练数据的R^2: 0.47
模型对训练数据的MAE: 4.34
模型对测试数据的R^2: 0.58
模型对测试数据的MAE: 4.92
```

图 4-32 线性回归的梯度下降法程序运行的结果

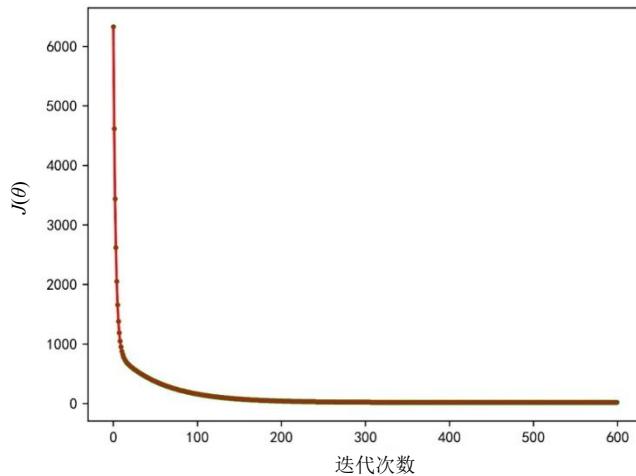


图 4-33 误差值变化的曲线

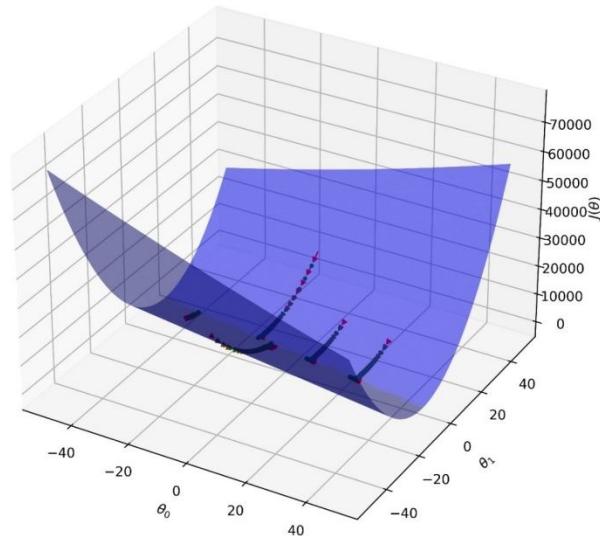


图 4-34 误差函数的图形

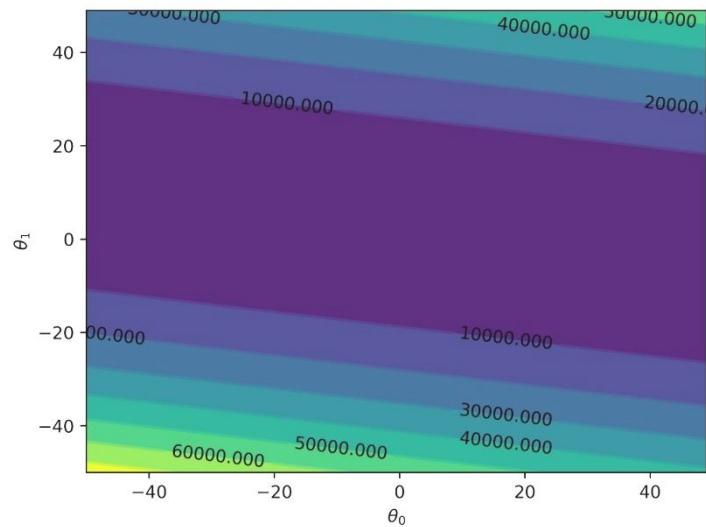


图 4-35 误差函数的等高线

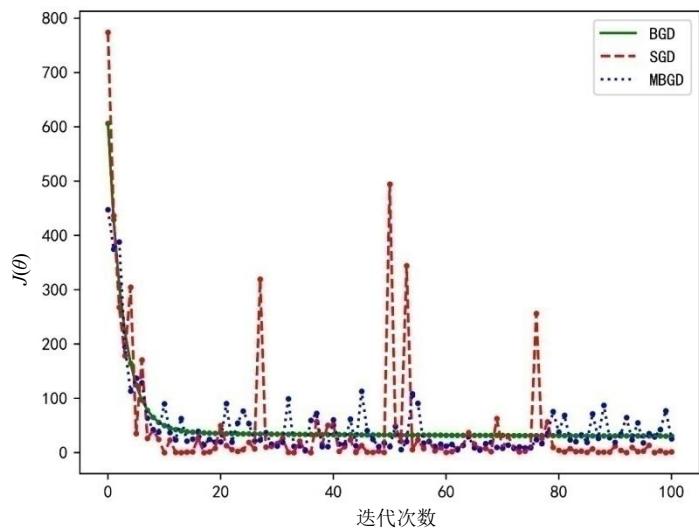


图 4-36 3 种梯度下降法误差下降过程的比较

# 第 5 章

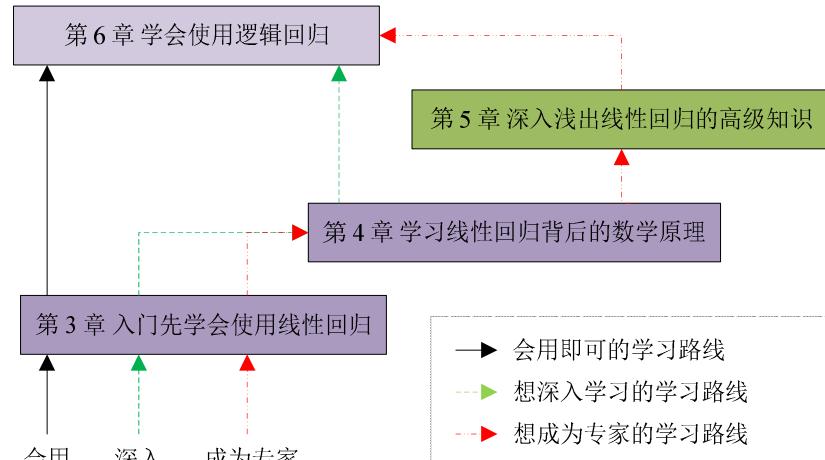


图 5-1 学习路线图

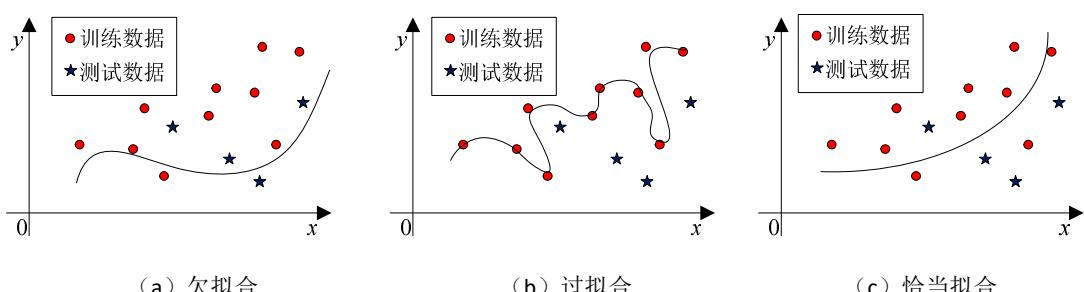


图 5-2 对模型的 3 种定性评价

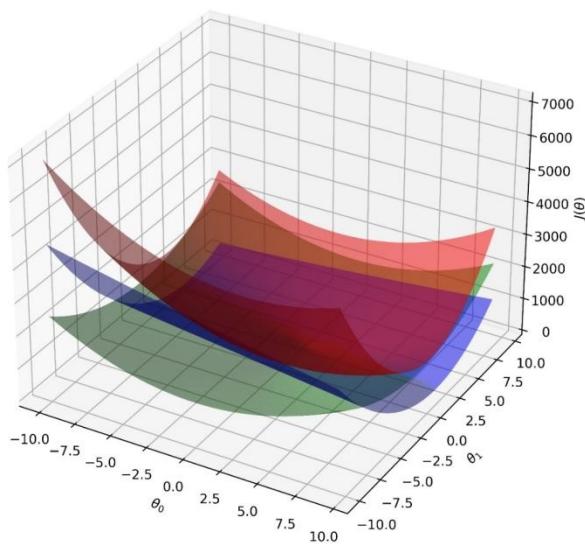
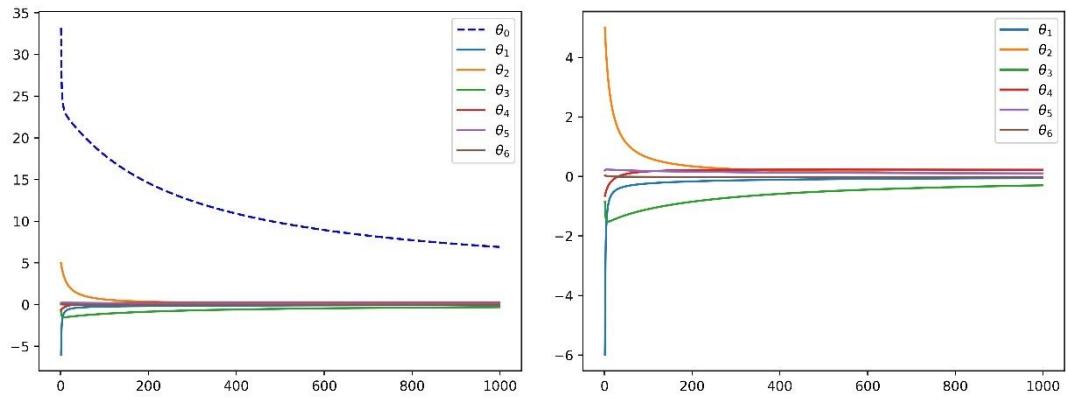


图 5-3 误差函数的图形



(a) 带 $\theta_0$ 的岭迹图

(b) 不带 $\theta_0$ 的岭迹图

图 5-4 岭迹图

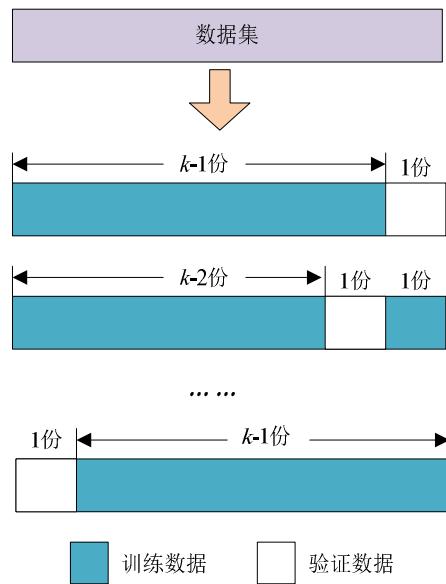


图 5-5 K-Fold 交叉验证法

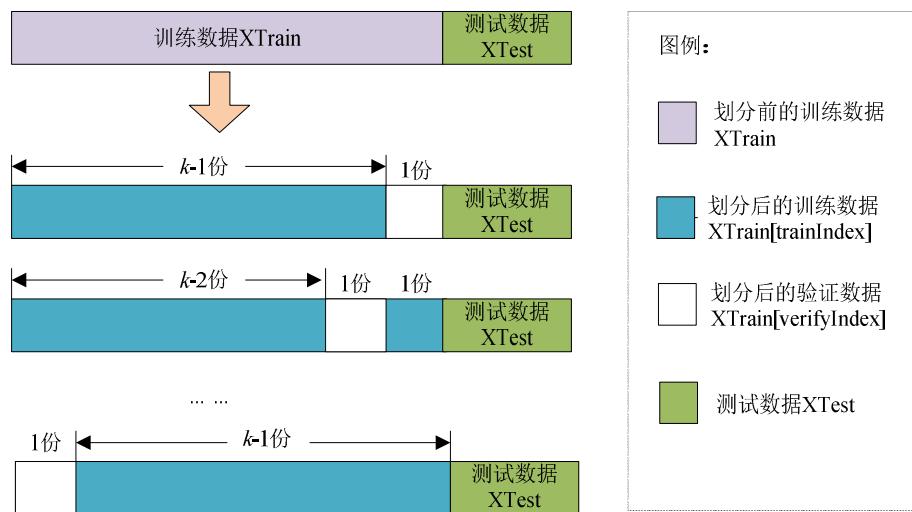


图 5-6 对训练数据使用 K-Fold 再做划分

```

In [8]: runfile('E:/script/logisticsWeb/machinelearn/chapter5/code5-2.py', wdir='E:/script/logisticsWeb/machinelearn/chapter5')
lambda: 0 。MAE: 4.372640307433445 ; R2: 0.40059917989644056
lambda: 1 。MAE: 4.375852660620379 ; R2: 0.4010367182805042
lambda: 2 。MAE: 4.379210867034098 ; R2: 0.4014369438586325
lambda: 3 。MAE: 4.3827459303625265 ; R2: 0.4018011238295614
lambda: 4 。MAE: 4.386252801397256 ; R2: 0.4021304804294747
lambda: 5 。MAE: 4.389722562421435 ; R2: 0.40242619271767255
lambda: 6 。MAE: 4.39324224519703 ; R2: 0.4026893982824086
lambda: 7 。MAE: 4.396929378779491 ; R2: 0.40292119487095535
lambda: 8 。MAE: 4.400675718075966 ; R2: 0.4031226419477038
lambda: 9 。MAE: 4.404475767324109 ; R2: 0.4032947621837045
根据综合评价指标判断, 第5个lambda最优, lambda值为: 5, 综合评价值为: 0.57

```

图 5-7 运行源代码 5-2 后控制台的输出

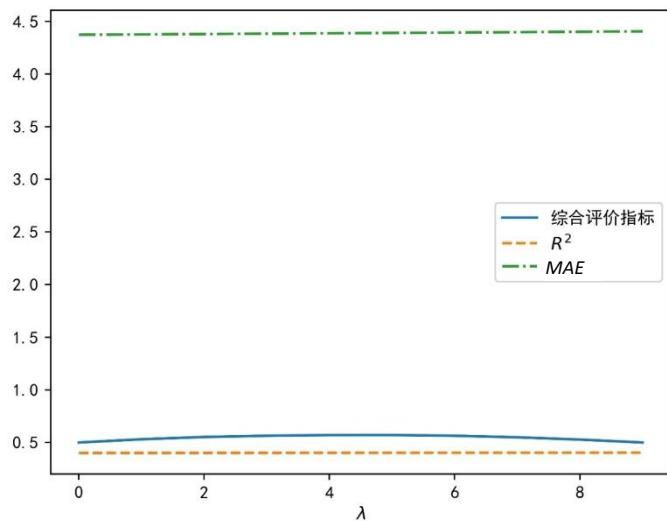


图 5-8 运行源代码 5-2 生成的图

```

In [15]: runfile('E:/script/logisticsWeb/machinelearn/chapter5/code5-3.py', wdir='E:/script/logisticsWeb/machinelearn/chapter5')
lambda: 0 。MAE: 6.135712059501724 ; R2: 0.04597718053265913
lambda: 1 。MAE: 5.986010731196492 ; R2: 0.10154666193714071
lambda: 2 。MAE: 5.26136543813031 ; R2: 0.22073251777952815
lambda: 3 。MAE: 6.000253420052512 ; R2: 0.0615676651687751
lambda: 4 。MAE: 6.000938796183889 ; R2: 0.08233079816862614
lambda: 5 。MAE: 5.533129758484052 ; R2: 0.1979647677017585
lambda: 6 。MAE: 5.801295264723441 ; R2: 0.14190780899920374
lambda: 7 。MAE: 5.085446104902408 ; R2: 0.29477994125635265
lambda: 8 。MAE: 6.101904005254979 ; R2: 0.002342576959671339
lambda: 9 。MAE: 5.89750275410145 ; R2: 0.10041807141975134
根据综合评价指标判断, 第7个lambda最优, lambda值为: 7, 综合评价值为: 1.0

```

图 5-9 运行源代码 5-3 后控制台的输出

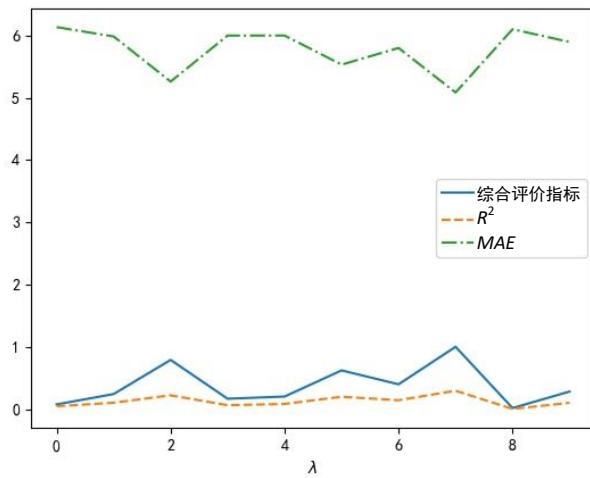


图 5-10 运行源代码 5-3 生成的图

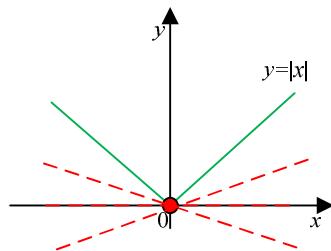
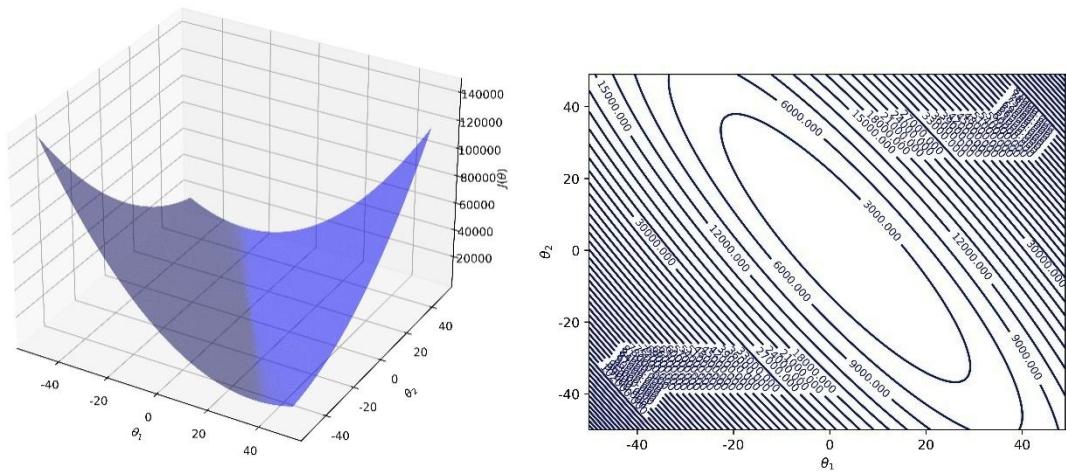


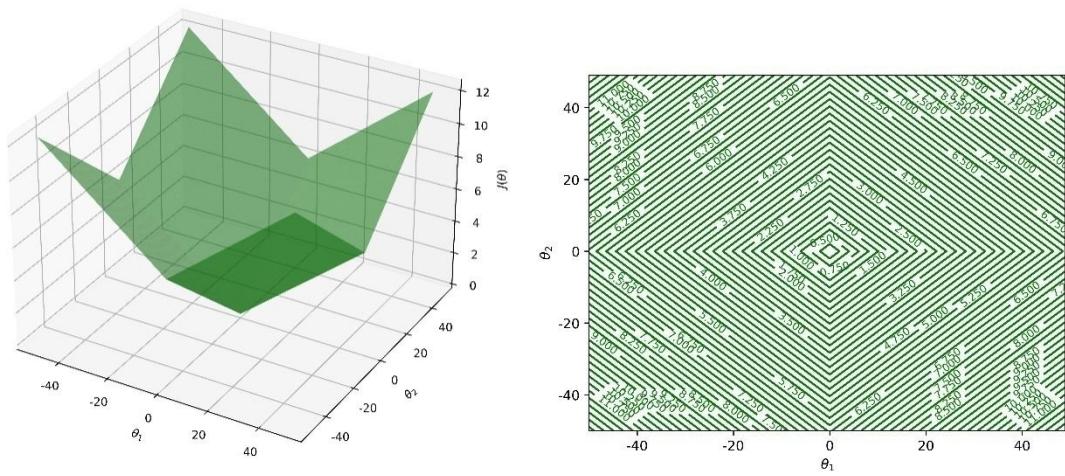
图 5-11  $y = |x|$  的图形



(a) 3D 图

(b) 等高线图

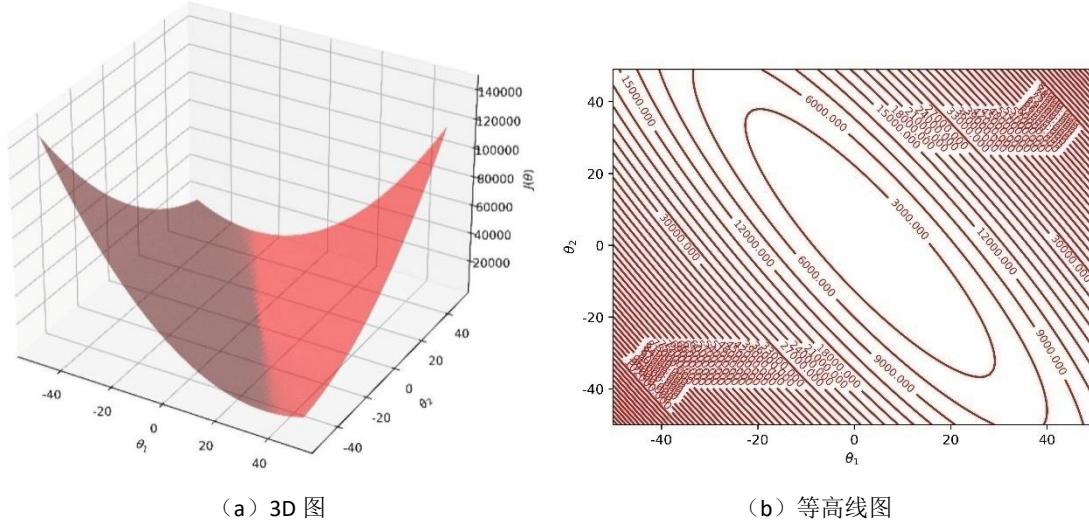
图 5-12 Lasso 回归误差函数前半部分的图形



(a) 3D 图

(b) 等高线图

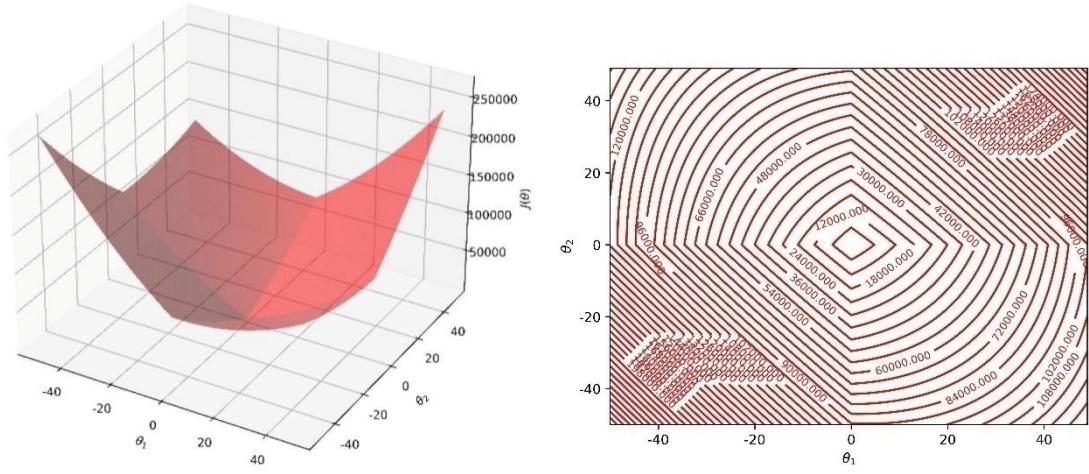
图 5-13 Lasso 回归误差函数后半部分的图形 ( $\lambda = 100$ )



(a) 3D 图

(b) 等高线图

图 5-14 Lasso 回归误差函数的图形 ( $\lambda = 100$ )



(a) 3D 图

(b) 等高线图

图 5-15 Lasso 回归误差函数的图形 ( $\lambda = 1000000$ )

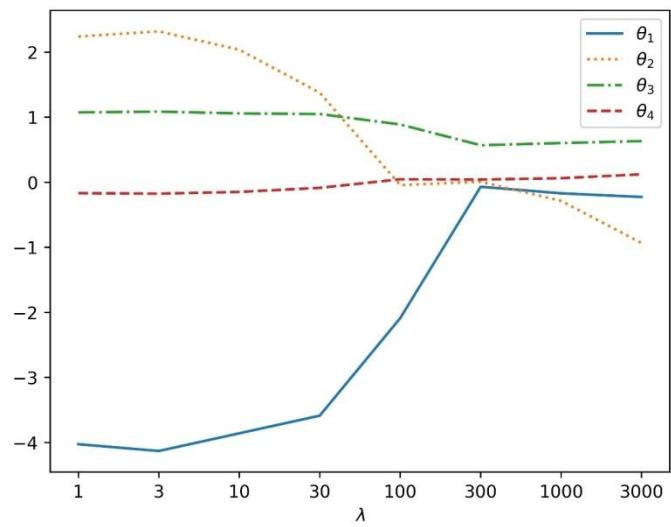


图 5-16 用坐标轴下降法做 Lasso 回归找到合适的 $\lambda$ 值

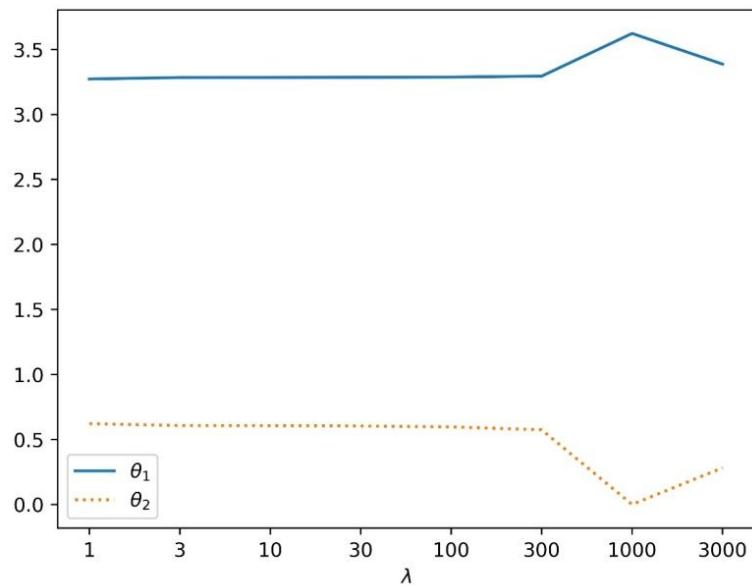


图 5-17 源代码 5-5 的运行结果

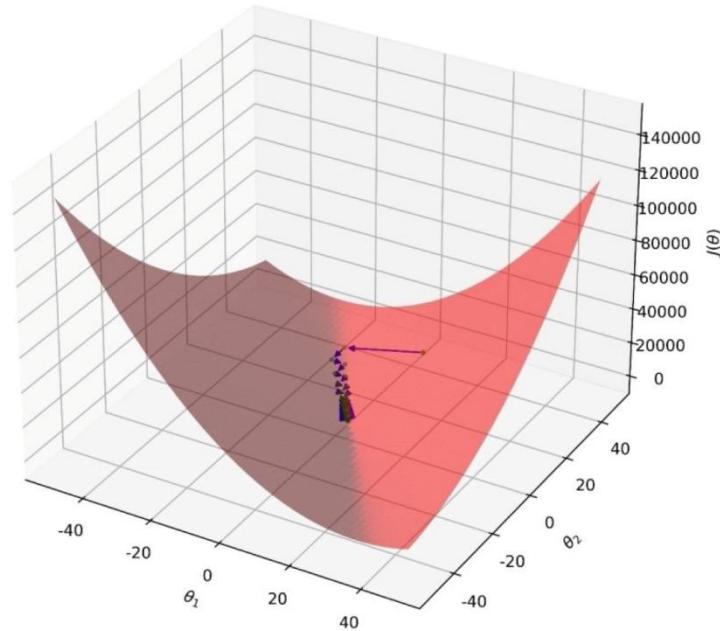


图 5-18 绘制带惩罚项的误差函数的图形及迭代的过程

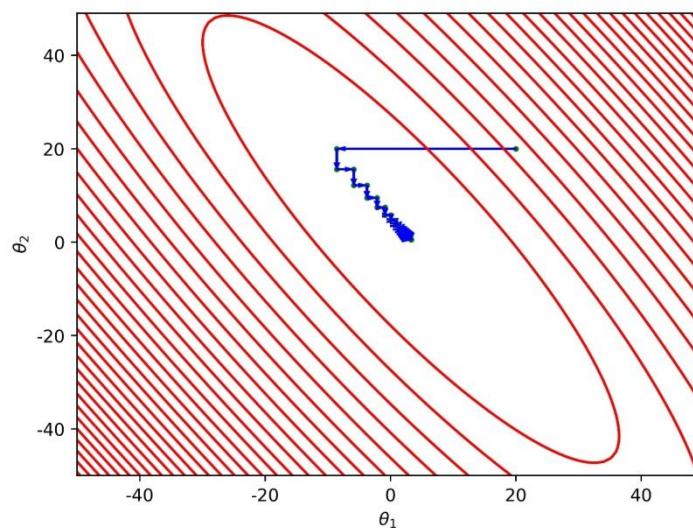


图 5-19 用等高线绘制带惩罚项的误差函数的图形及迭代的过程

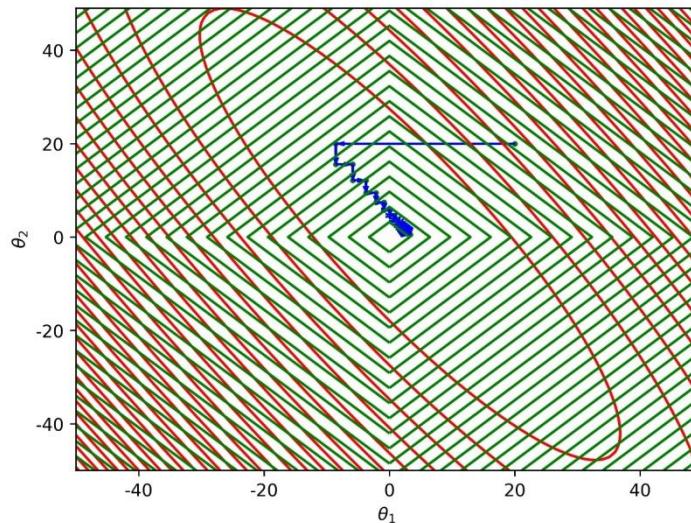


图 5-20 源代码 5-8 的运行结果



图 5-21 使用 LinearRegression 类的流程

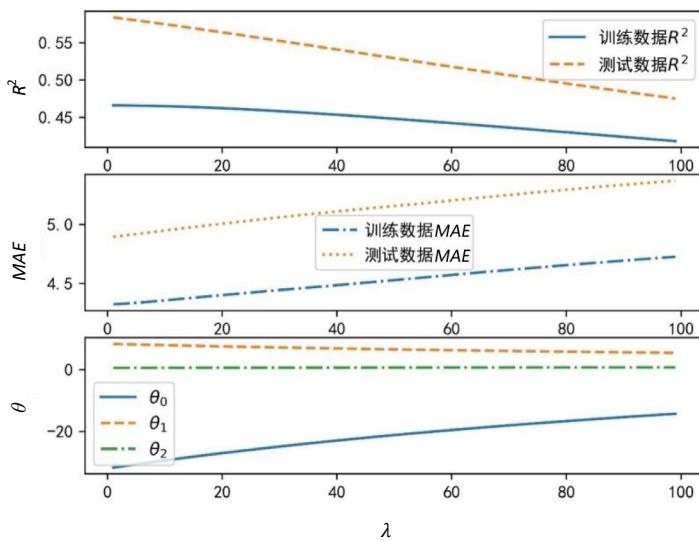


图 5-22 使用 Ridge 类看变换 $\lambda$ 值后  $MAE$ 、 $R^2$  的变化情况

```

最好的模型的R2值: 0.4034385428811724
最好的模型的lambda值: 10.0
最好的模型的系数: [[7.926732 0.5820548]]
最好的模型的截距: [-29.3957636]
最好的模型对测试数据的拟合度: 0.5750374607582032

```

图 5-23 使用 RidgeCV 类寻找具有最好的拟合度的线性模型

```
Console 1/A
最好的模型的R2值: 0.4037645998814273
最好的模型的lambda值: 15
最好的模型的系数: [[7.72565556 0.5956479]]
最好的模型的截距: [-28.18726595]
最好的模型对测试数据的拟合度: 0.5696566167566552

In [3]: |
```

图 5-24 寻找更好的 $\lambda$ 值

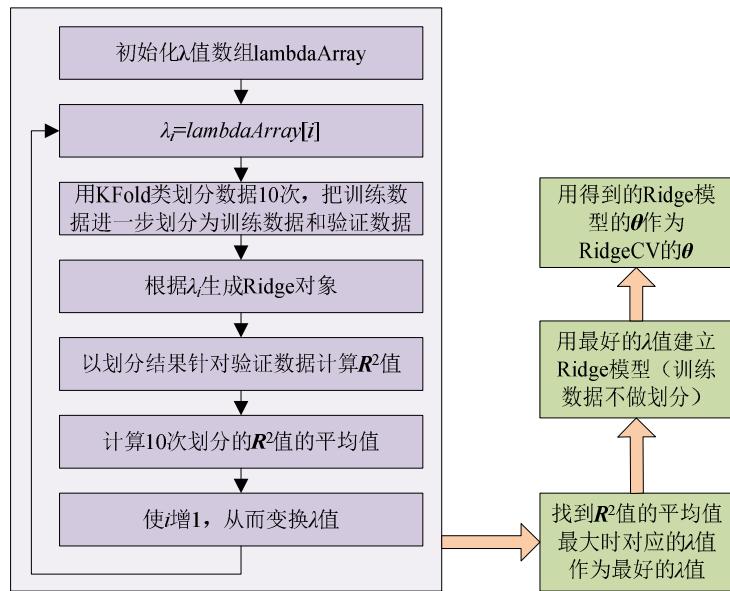


图 5-25 RidgeCV 构建最好的模型的过程

```
Console 1/A
最好的模型的R2值: 0.5596211916137239
最好的模型的lambda值: 0.01
最好的模型的系数: [-28.38150369 2.8497701 2.85426679
-0.2112797]
最好的模型的截距: 80.05875613092094
最好的模型对测试数据的拟合度: 0.6860025943775283

In [12]: |
```

图 5-26 使用 LassoCV 类寻找具有最好的拟合度的线性模型

```
Console 1/A
=====RidgeCV=====
最好的模型的R2值: 0.4651415699201855
最好的模型的lambda值: 10.0
最好的模型的系数: [7.926732 0.5820548]
最好的模型的截距: -29.395763595061116
最好的模型对测试数据的拟合度: 0.5750374607582032
=====LassoCV=====
最好的模型的R2值: 0.46619175946061486
最好的模型的lambda值: 0.01
最好的模型的系数: [8.34177186 0.55143399]
最好的模型的截距: -31.880403588849905
最好的模型对测试数据的拟合度: 0.584712014498803
=====ElasticNetCV=====
最好的模型的R2值: 0.46609452248491035
最好的模型的l1_ratio值: 0.3
最好的模型的lambda值: 0.01
最好的模型的系数: [8.22826739 0.56076188]
最好的模型的截距: -31.20455718169214
最好的模型对测试数据的拟合度: 0.582249445910006

In [14]: |
```

图 5-27 对比 RidgeCV、LassoCV、ElasticNetCV

# 第 6 章

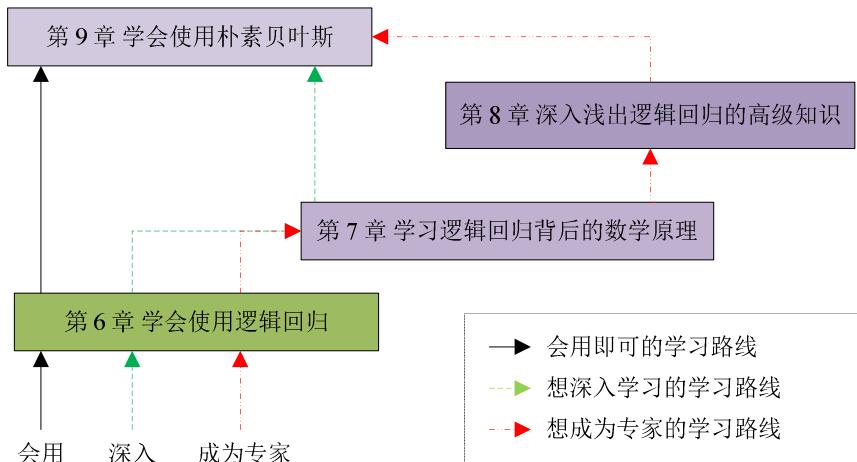


图 6-1 学习路线图

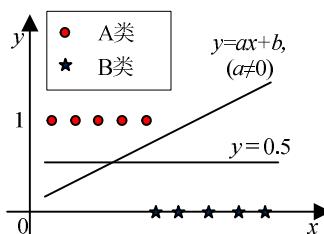


图 6-2 用线性模型来做二分法分类

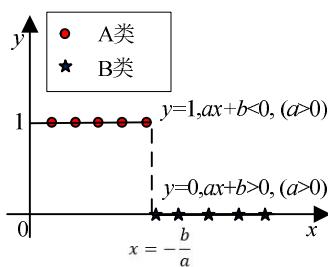


图 6-3 真正的分类函数的图示

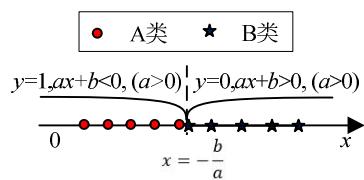


图 6-4 用一维来表示二分法分类

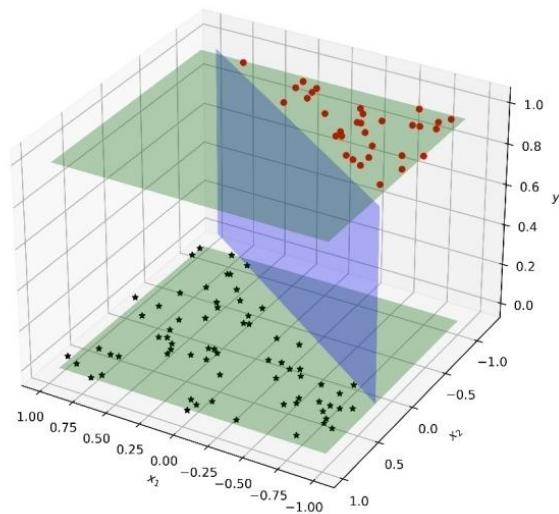


图 6-5 用三维空间表示分类函数

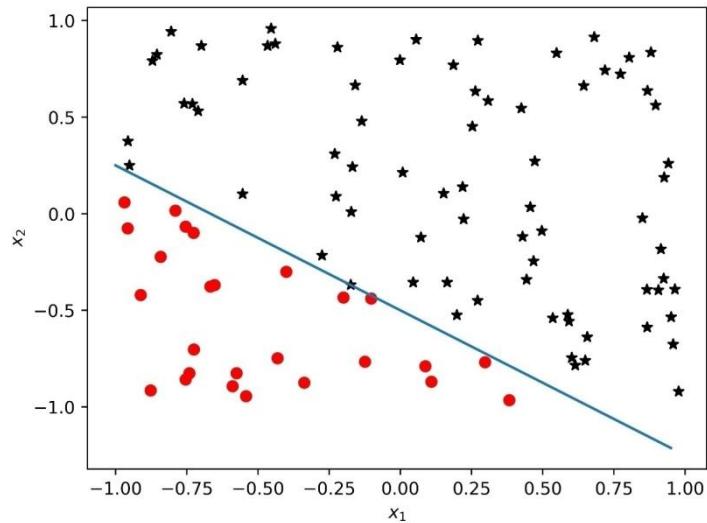


图 6-6 用二维空间表示分类函数

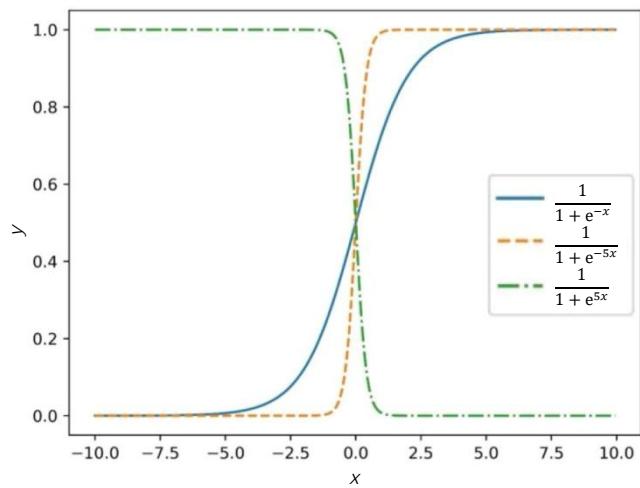


图 6-7 sigmoid 函数的图形

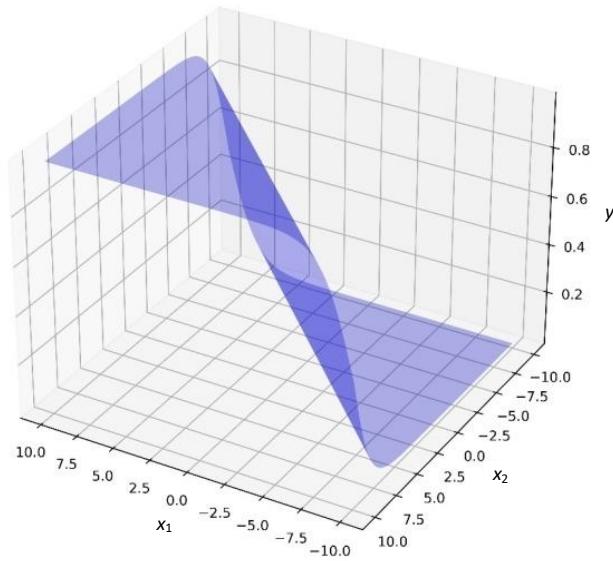


图 6-8 2 个自变量时的 sigmoid 函数的图形

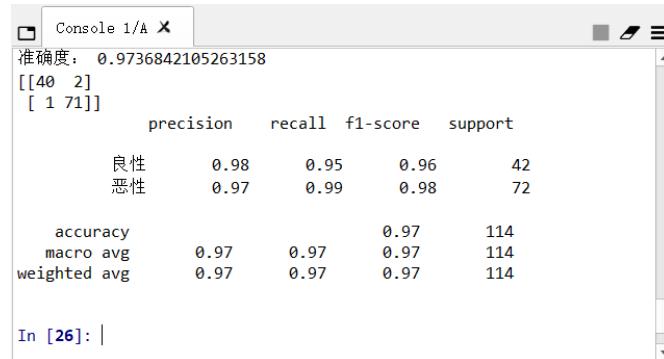


图 6-9 用逻辑回归预测乳腺癌



图 6-10 计算标准差

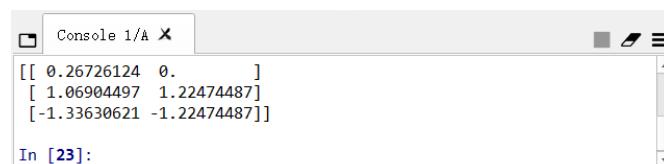


图 6-11 用 StandardScaler 类做标准化处理

In [27]:

图 6-12 得到逻辑回归模型的参数

In [44]:

图 6-13 得到分类的可能性值

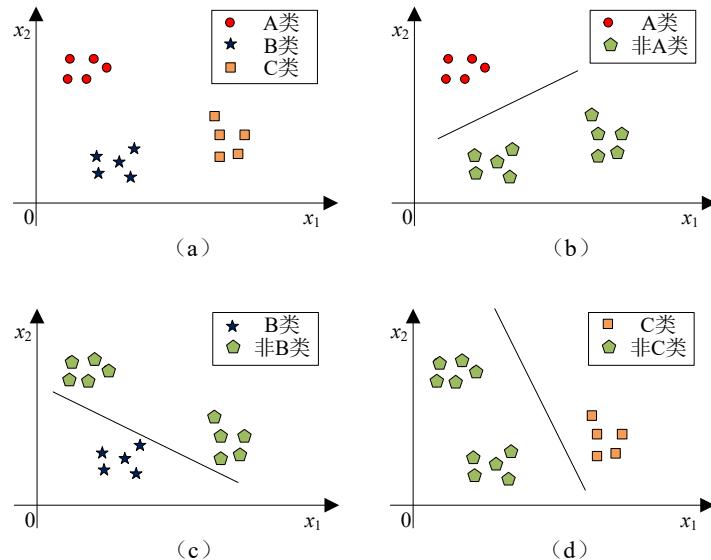


图 6-14 用 One-Vs-All 解决多分类问题

```

Console 1/A X
code-2.py, wdir = E:/scripts/logisticswedo/machinelearn/chapter6/
第1次分类的准确度: 1.0
第2次分类的准确度: 0.5333333333333333
第3次分类的准确度: 0.9666666666666667
测试样本的目标数据值:
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2]
One-Vs-All的预测值:
[0 1 1 0 2 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2]
预测正确的数量: 28
一共 30 个测试样本
One-Vs-All的准确度: 0.9333333333333333

```

图 6-15 用 One-Vs-All 做鸢尾花分类

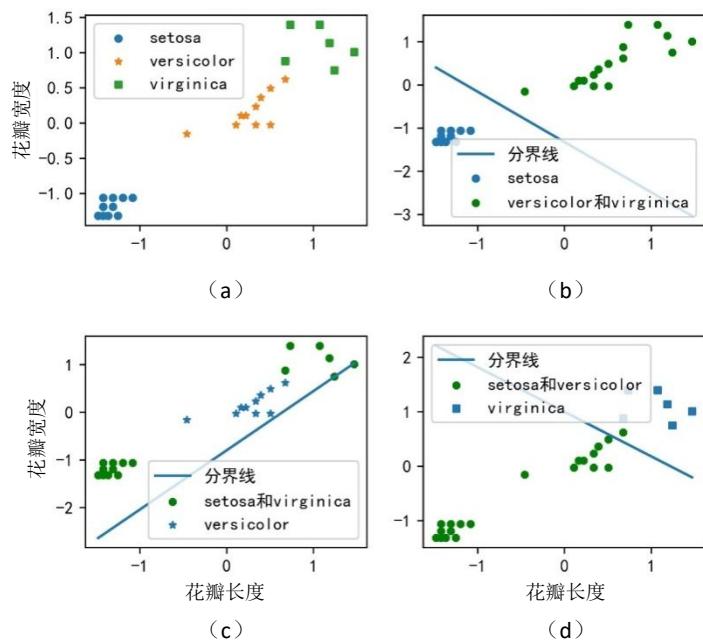


图 6-16 用 One-Vs-All 做鸢尾花分类的效果图

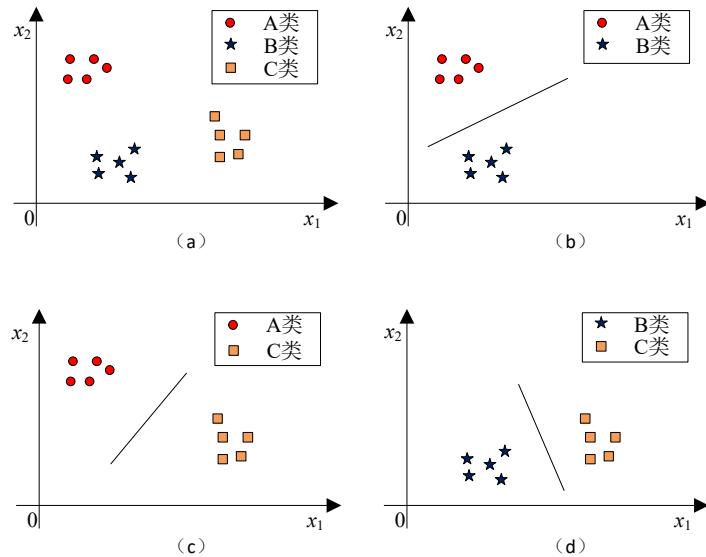
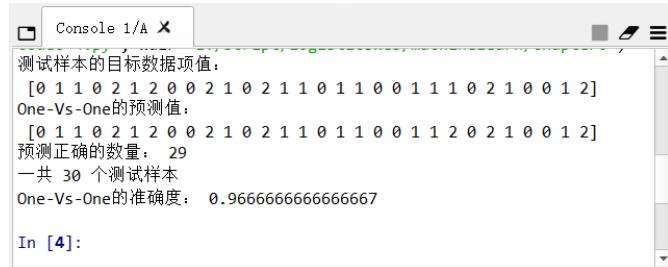


图 6-17 用 One-Vs-One 解决多分类问题



```
Console 1/A
测试样本的目标数据项值:
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2]
One-Vs-One的预测值:
[0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 2 0 2 1 0 0 1 2]
预测正确的数量: 29
一共 30 个测试样本
One-Vs-One的准确度: 0.9666666666666667

In [4]:
```

图 6-18 用 One-Vs-One 做鸢尾花分类

# 第 7 章

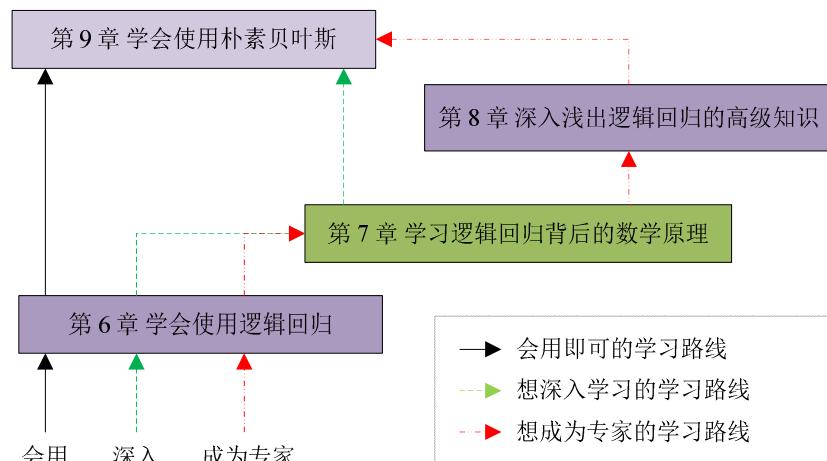


图 7-1 学习路线图

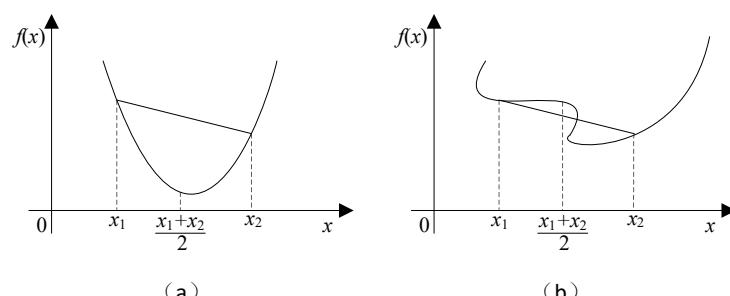


图 7-2 凸函数和非凸函数

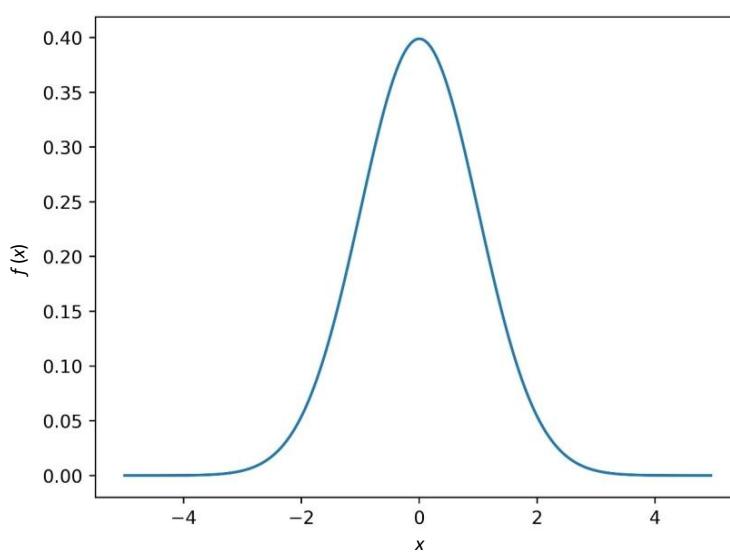


图 7-3 标准正态分布的图形

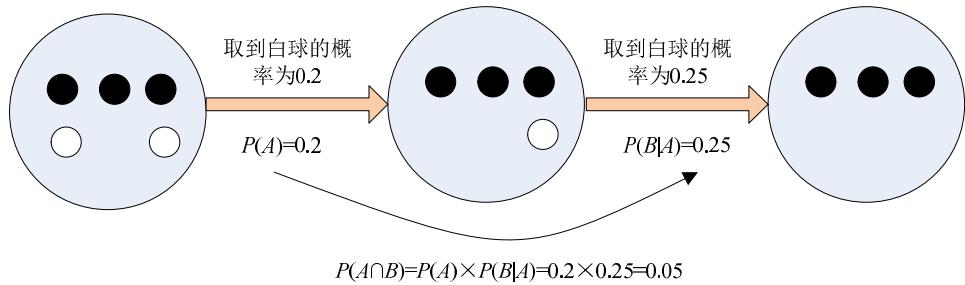


图 7-4 理解条件概率

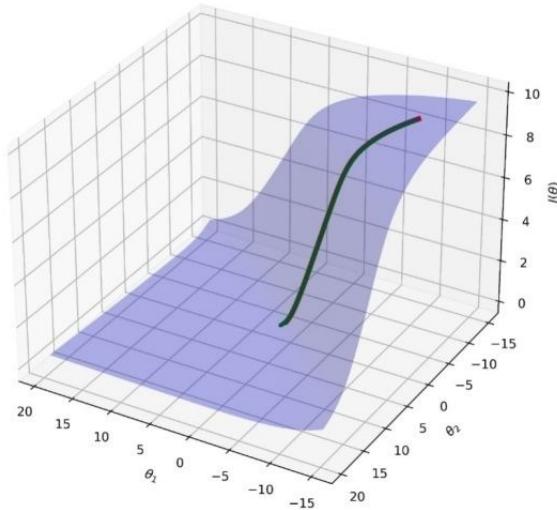


图 7-5 误差函数及梯度下降的过程

```

Console 1/A ×
第 517 次迭代, 误差值为 0.29779521156906513 , 两次迭代差为
0.001038289476240073
第 518 次迭代, 误差值为 0.2967713384721463 , 两次迭代差为
0.0010238730969188148
第 519 次迭代, 误差值为 0.2957615883967735 , 两次迭代差为
0.001009750075372795
第 520 次迭代, 误差值为 0.2947656756590386 , 两次迭代差为
0.0009959127377349164
参数值: [[0.92939968 0.61073482]]
真实值: [0 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 1 0 1 1 0 0 1
1]
预测值: [0 0 1 0 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1
1]
precision      recall      f1-score      support
setosa       0.92      1.00      0.96       11
非setosa     1.00      0.95      0.97       19
accuracy      0.96      0.97      0.96       30
macro avg    0.96      0.97      0.96       30
weighted avg 0.97      0.97      0.97       30

```

In [9]:

图 7-6 做鸢尾花分类的控制台输出

```
Console 1/A X
-0.75285052 -0.745044 -1.29405984 -1.87002309 -0.12185411
0.30308441]
真实值: [1 0 1 0 0 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 0 0
0 0 1 0 0 1 1 0
1 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1
1 1 0 0
1 0 0 0 1 0 1 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1
1 0 0 0
1 1 1]
预测值: [1 0 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0 0
0 0 1 0 0 1 1 0
1 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 0 1 1
1 1 1 0
1 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0
1 1 1]
precision    recall    f1-score    support
良性          0.97      0.93      0.95      42
恶性          0.96      0.99      0.97      72
accuracy          0.96      0.96      0.96      114
macro avg       0.97      0.96      0.96      114
weighted avg    0.97      0.96      0.96      114
In [14]:
```

图 7-7 预测乳腺癌的控制台输出

# 第 8 章

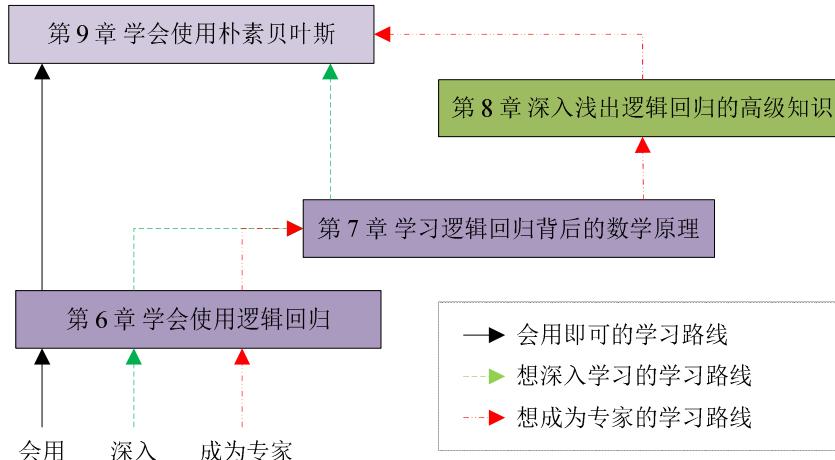


图 8-1 学习路线图

Console 1/A

	precision	recall	f1-score	support
良性	0.97	0.93	0.95	42
恶性	0.96	0.99	0.97	72
accuracy			0.96	114
macro avg	0.97	0.96	0.96	114
weighted avg	0.97	0.96	0.96	114

In [4]:

图 8-2 用逻辑回归的 L2 正则化预测乳腺癌

Console 1/A

	precision	recall	f1-score	support
良性	1.00	0.93	0.96	42
恶性	0.96	1.00	0.98	72
accuracy			0.97	114
macro avg	0.98	0.96	0.97	114
weighted avg	0.97	0.97	0.97	114

In [8]:

图 8-3 用逻辑回归的 L1 正则化来预测乳腺癌

```

theta0: [-13.56570372]
theta1~thetan: [[ 0.          0.          0.          0.          0.
-18.88836989
 25.04322771 -16.4833904 -23.6215185  23.17834918 -1.47752881
-56.63132669  0.         -41.80975637 -42.28182493 -10.80010431
 7.33224931  20.4523717 -35.41500115  10.80141276  58.55897567
-36.03108094 -38.77667759 -32.54973903 -32.10138701 -1.97088715
-8.73725303 -32.78732039 -28.53294351 -28.53959576 -29.90074278]]
迭代次数: 41
precision    recall    f1-score   support
良性          1.00      0.95      0.98      42
恶性          0.97      1.00      0.99      72

accuracy          0.98      114
macro avg       0.99      0.98      0.98      114
weighted avg    0.98      0.98      0.98      114

```

图 8-4 用 SGDClassifier 类的 L1 正则化来预测乳腺癌

```

theta0: [0.66713511]
theta1~thetan: [[ 0.          -0.01536528  0.          0.          0.
-0.25964199 -0.50061344  0.          0.         -0.38459871  0.
 0.          0.         -0.0036297  0.          0.          0.
 0.          0.         -3.42228454 -1.03457294 -0.43561417  0.
-0.45901456  0.         -0.085997 -1.20001901 -0.30731831  0.]]
迭代次数: [[[ 1     1     4   832 1311 1809 3884 1374 197   23]
[ 1     1     7 1245 1055 2716 3945 1765 505   72]
[ 1     1     6 562 2148 1544 3159 1247 167   19]
[ 1     1     5 821 2105 2209 2982 1612 233   34]
[ 1     1     3 1068 1400 2050 4455 1420 121   20]
[ 1     1     3 613 1903 2289 3869 1246 175   23]
[ 1     1     3 684 1590 1676 3893 927 247   29]
[ 1     1     6 569 2245 1915 3834 1822 271   32]
[ 1     1     4 671 1798 2422 2826 1277 251   30]
[ 1     1     4 1081 1975 1188 2457 1221 201   19]]]
precision    recall    f1-score   support
良性          1.00      0.90      0.95      42
恶性          0.95      1.00      0.97      72

accuracy          0.96      114
macro avg       0.97      0.95      0.96      114
weighted avg    0.97      0.96      0.96      114

```

图 8-5 用 LogisticRegressionCV 类的 L1 正则化来预测乳腺癌

```

theta0: [-0.79593261  2.58998706 -1.79405444]
theta1~thetan: [[-5.51046284  0.          ]
[ 0.          0.          ]
[ 3.43640999  4.55982758]]
迭代次数: [154]
precision    recall    f1-score   support
setosa        1.00      1.00      1.00      11
versicolor    1.00      0.92      0.96      13
virginica     0.86      1.00      0.92       6

accuracy          0.97      30
macro avg       0.95      0.97      0.96      30
weighted avg    0.97      0.97      0.97      30

```

图 8-6 用 LogisticRegression 类对鸢尾花做三分类

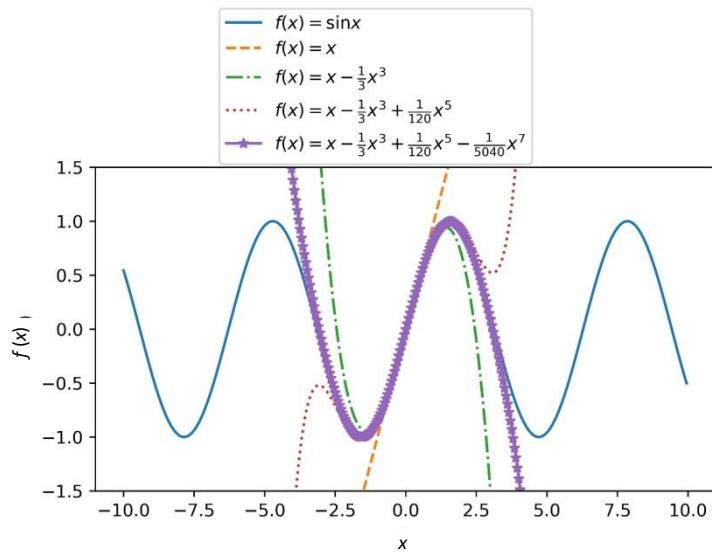


图 8-7  $f(x) = \sin x$  的近似表达

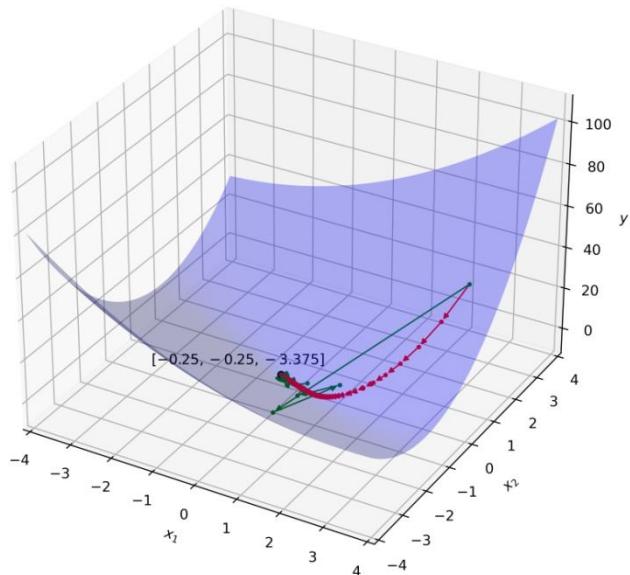


图 8-8 对比普通的梯度下降法和回溯法的下降过程

```

Console 1/A ×
Reloaded modules: common, common.common
====使用普通的梯度下降法（与源代码4-3相同）====
起点时的梯度: [11, 20]
在第139次迭代退出,此时两次迭代函数值相差: 9.542879446655661e-06
极小值: [-0.2363089680763057, -0.25567101110562496, -3.3748713585321575]
====使用回溯法====
起点时的梯度: [11, 20]
在第20次迭代退出,此时两次迭代函数值相差: 3.6978156727762723e-06
极小值: [-0.24776895652544315, -0.25029102593395375, -3.3749960669397536]
In [4]: |

```

图 8-9 对比普通的梯度下降法和回溯法的结果

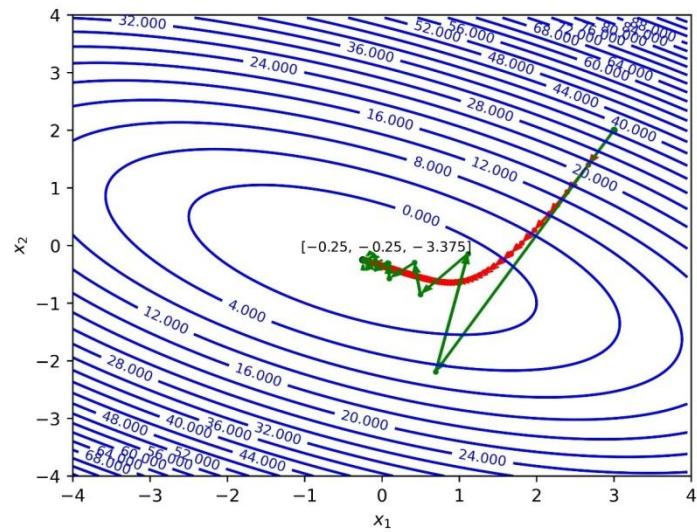


图 8-10 图 8-8 对应的二维图

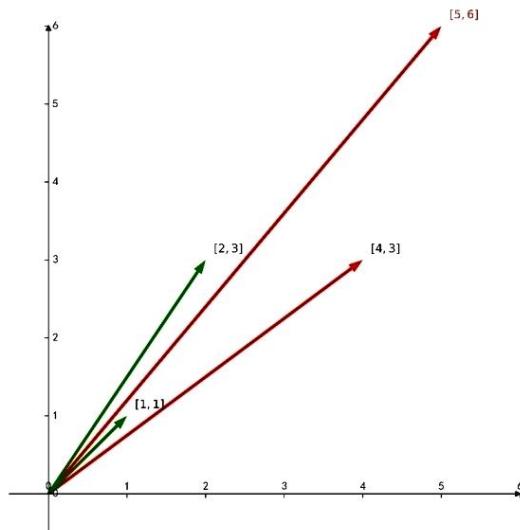


图 8-11 向量的图示

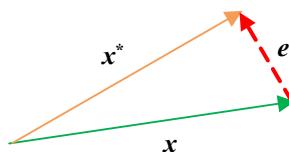


图 8-12 解与中间向量的误差

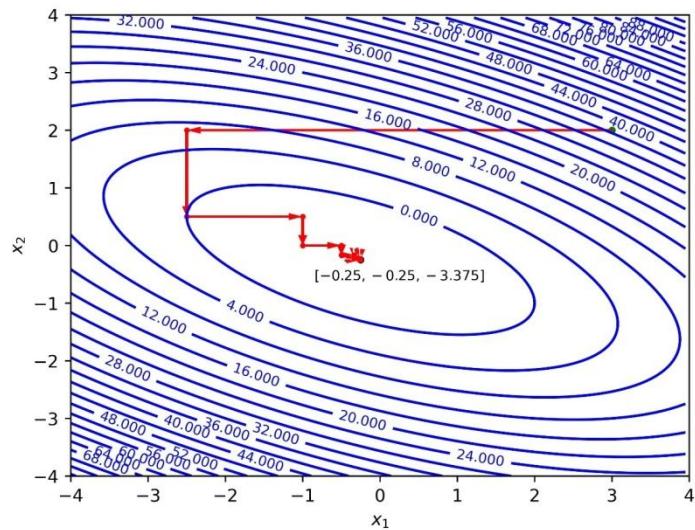


图 8-13 使用坐标下降法来求极小值

```

Console 1/A ✘
=====
=====sag算法=====
经过100次迭代, 误差值为 1.8686422895016084
经过200次迭代, 误差值为 0.5061821950124185
=====saga算法=====
经过100次迭代, 误差值为 3.1198903100098176
经过200次迭代, 误差值为 0.4474746581951357
=====SGD算法=====
经过100次迭代, 误差值为 2.5732200803430993
经过200次迭代, 误差值为 0.7144083133597467

```

图 8-14 200 次迭代后控制台的输出

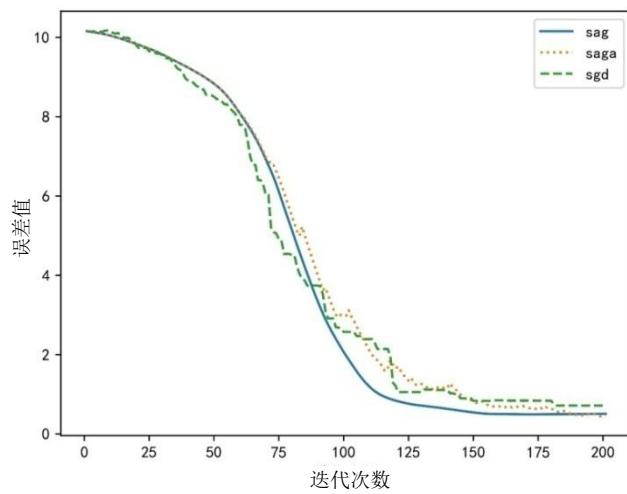


图 8-15 3 种算法的误差值变化曲线

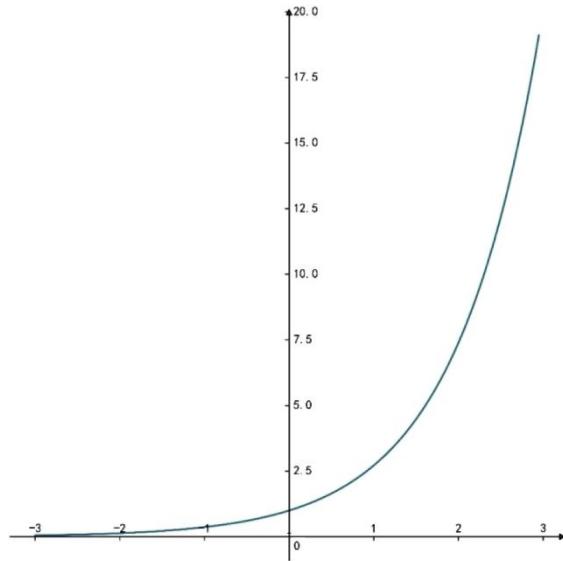


图 8-16 指数函数的图形



图 8-17 softmax 的计算效果

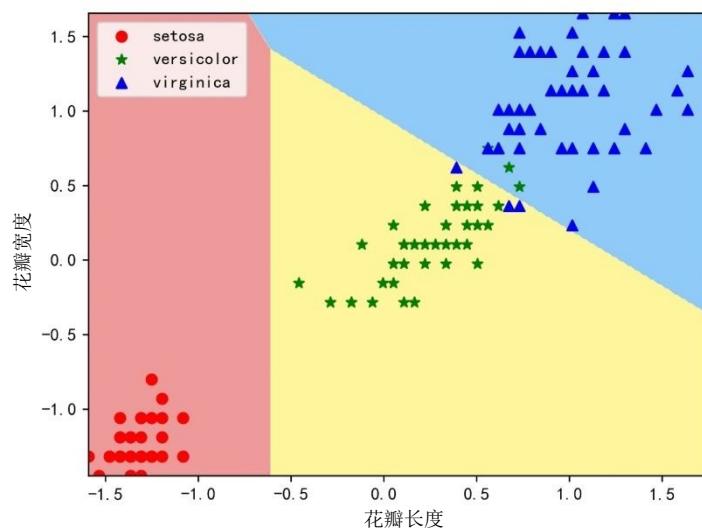


图 8-18 对鸢尾花做三分类的结果图示

# 第 9 章

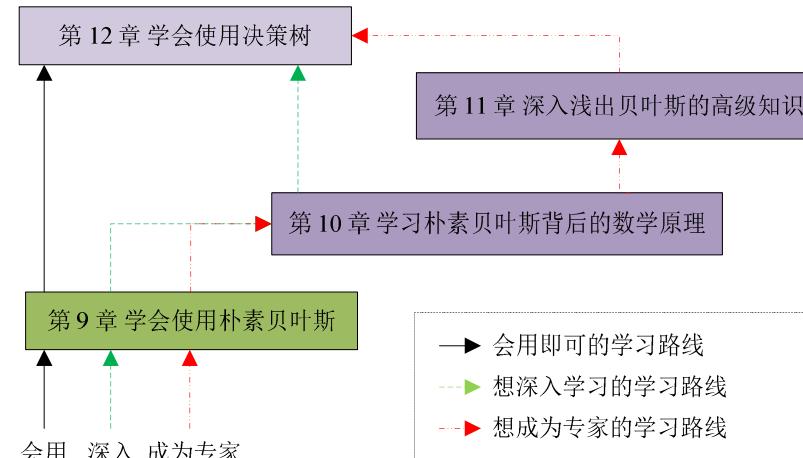


图 9-1 学习路线图

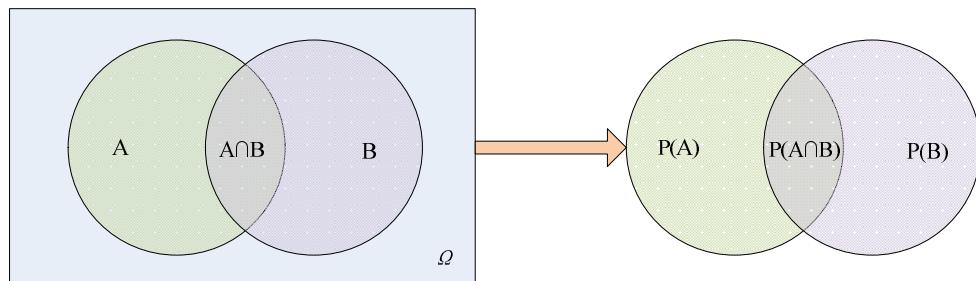


图 9-2 联合概率和条件概率

月收入 $x_{re0}$	相熟度 $x_{cl}$	结交分类 $Y$
1: 高	1: 不认识	1: 不发展
2: 中	2: 一般	2: 可尝试
3: 低	3: 感情很好	3: 应确定

图 9-3 相亲的应用场景描述

The screenshot shows a Jupyter Notebook cell with the following content:

```
Console 1/A ×
[3.08426338e-149 1.30779471e-001 8.69220529e-001]
[1.00000000e+000 1.83267245e-015 2.64881038e-022]
[1.24123122e-224 8.04117153e-005 9.99919588e-001]
[4.76778248e-082 9.99808297e-001 9.19702900e-004]
[1.00000000e+000 2.63115919e-016 6.04633949e-023]
[1.00000000e+000 1.53373081e-014 2.39137210e-021]
[5.86218548e-108 9.93960093e-001 6.03990658e-003]
[7.34237102e-162 6.95294528e-003 9.93047055e-001]
precision      recall      f1-score      support
setosa       1.00       1.00       1.00        11
versicolor    1.00       0.92       0.96        13
virginica     0.86       1.00       0.92         6
accuracy      0.95       0.97       0.96        30
macro avg     0.95       0.97       0.96        30
weighted avg  0.97       0.97       0.97        30
```

图 9-4 使用 GuassianNB 类对鸢尾花做三分类

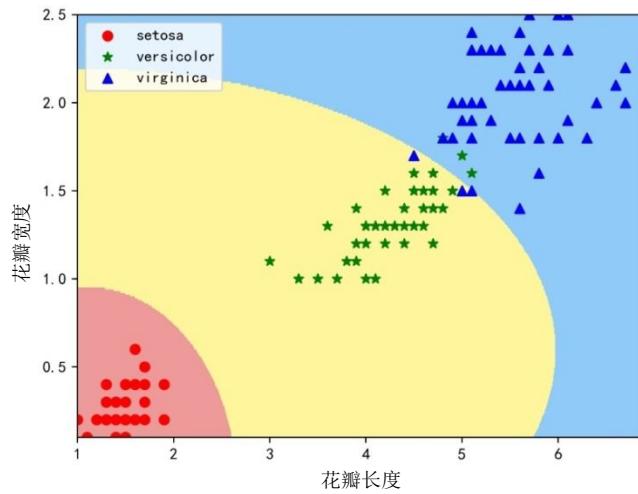


图 9-5 GuassianNB 类对鳶尾花做三分类的效果图

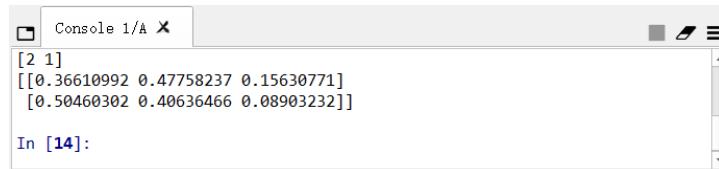


图 9-6 使用 MultinomialNB 类辅助某姑娘做相亲后的决策

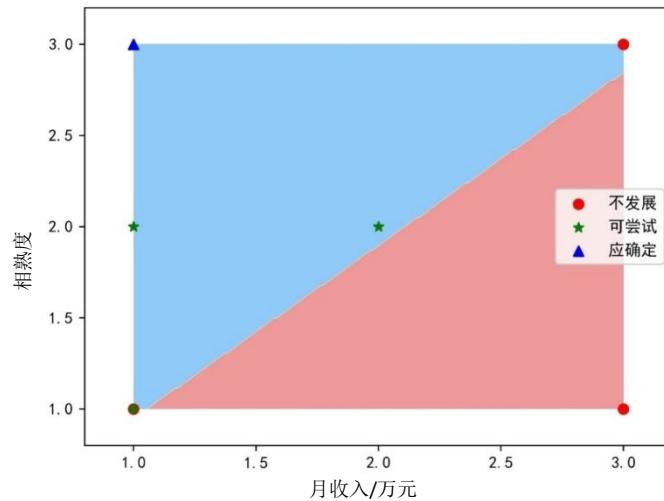


图 9-7 MultinomialNB 类辅助某姑娘做相亲后的决策效果图

# 第 10 章

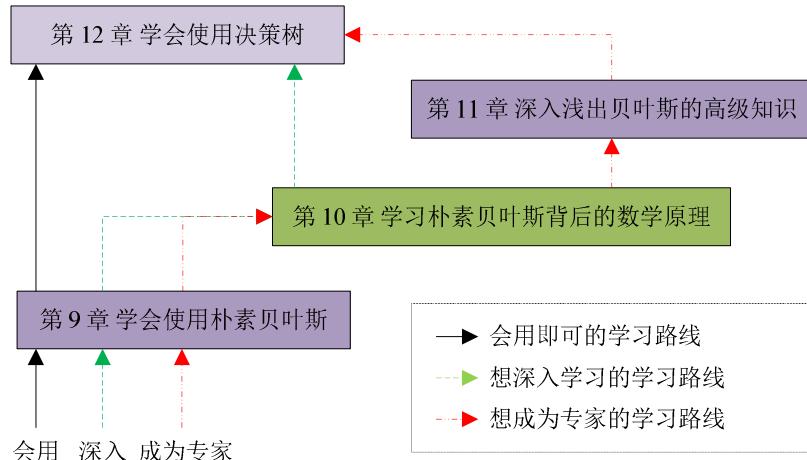


图 10-1 学习路线图

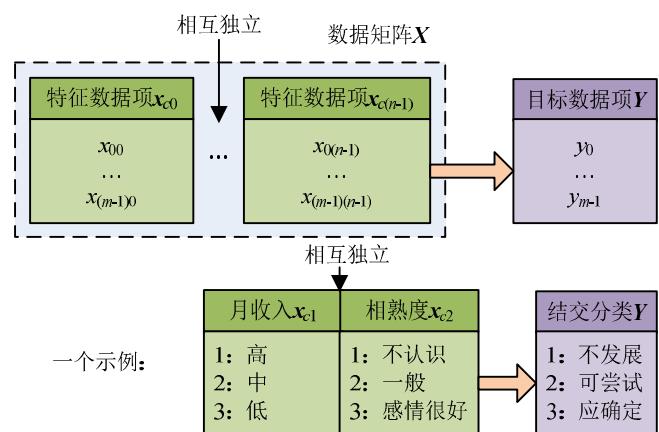


图 10-2 机器学习的应用场景

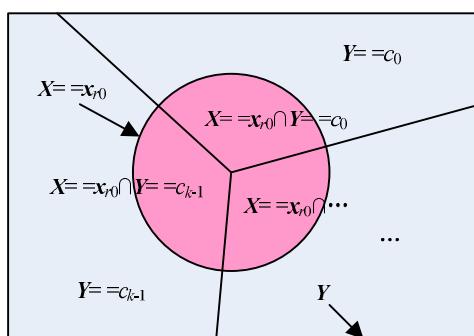


图 10-3 全概率公式的图示

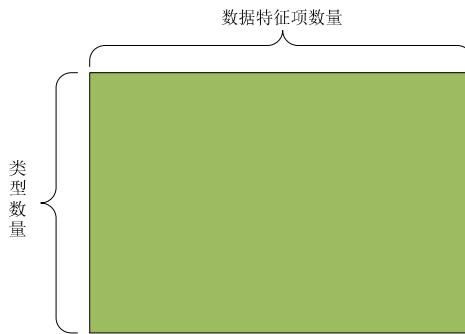


图 10-4 GaussianNB.theta\_ 和 GaussianNB.sigma\_ 的形状

```
Console 1/A ×
各个类别的先验概率: [0.325 0.30833333 0.36666667]
各个类别的训练样本数: [39. 37. 44.]
特征数据项在各个类别的平均值:
[[1.46666667 0.23333333]
 [4.22972973 1.30540541]
 [5.53409091 2.02045455]]
特征数据项在各个类别的标准差,
[[0.02529915 0.01094017]
 [0.21344047 0.03943024]
 [0.30315599 0.0788998 ]]
模型预测值: [[1.00000000e+000 6.31125192e-17 1.38220366e-23]]
计算出来的预测值: [1.00000000e+000 1.65954733e-190 1.87517920e-136]
In [17]:
```

图 10-5 用 GuassianNB 的属性来做朴素贝叶斯分类

```
Console 1/A ×
====高斯模型====
precision recall f1-score support
非垃圾邮件 0.95 0.74 0.83 564
垃圾邮件 0.69 0.94 0.80 357

accuracy 0.82 0.82 0.82 921
macro avg 0.82 0.82 0.82 921
weighted avg 0.85 0.82 0.82 921

====多项式模型====
precision recall f1-score support
非垃圾邮件 0.82 0.82 0.82 564
垃圾邮件 0.72 0.71 0.71 357

accuracy 0.78 0.78 0.78 921
macro avg 0.77 0.77 0.77 921
weighted avg 0.78 0.78 0.78 921

====伯努利模型====
precision recall f1-score support
非垃圾邮件 0.89 0.94 0.92 564
垃圾邮件 0.90 0.82 0.86 357

accuracy 0.89 0.89 0.89 921
macro avg 0.90 0.88 0.89 921
weighted avg 0.89 0.89 0.89 921
```

图 10-6 用 3 种朴素贝叶斯模型区分垃圾邮件和非垃圾邮件

# 第 11 章

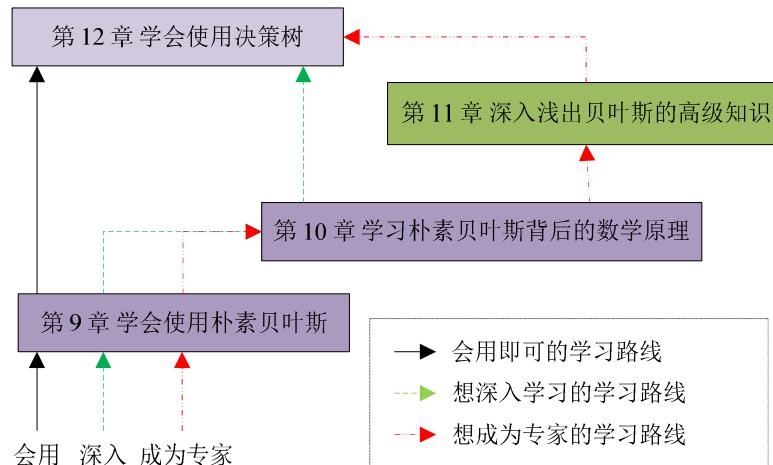


图 11-1 学习路线图

$$P(A, B) = P(A \cap B) = P(A) \times P(B)$$

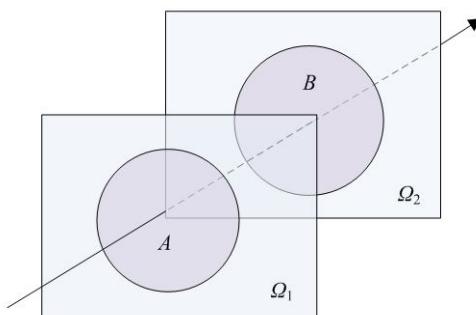


图 11-2 两个独立的事件

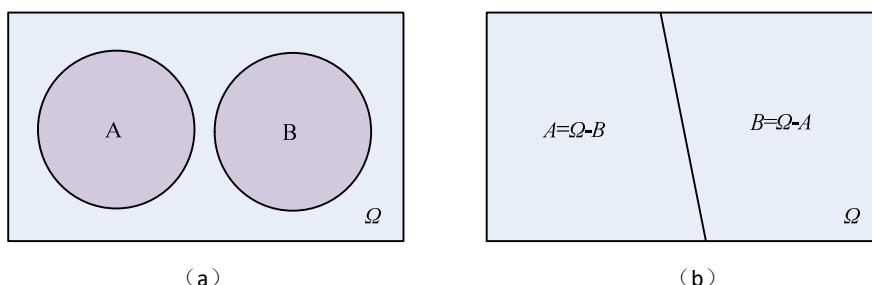


图 11-3 两个互斥的事件

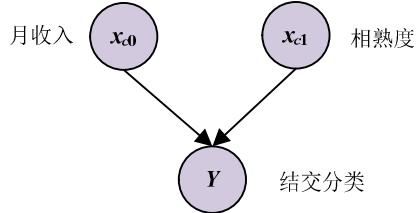


图 11-4 “某姑娘相亲后决定结交分类”的贝叶斯网络

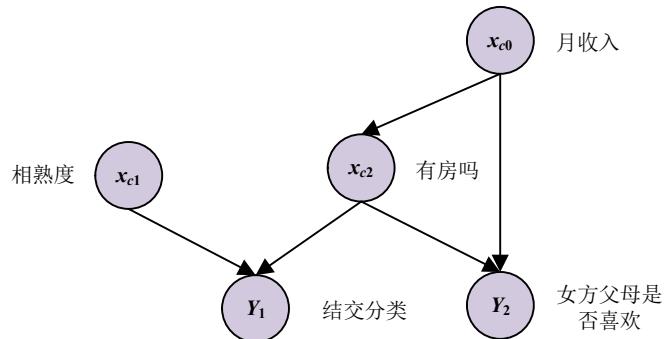


图 11-5 更复杂一点的贝叶斯网络

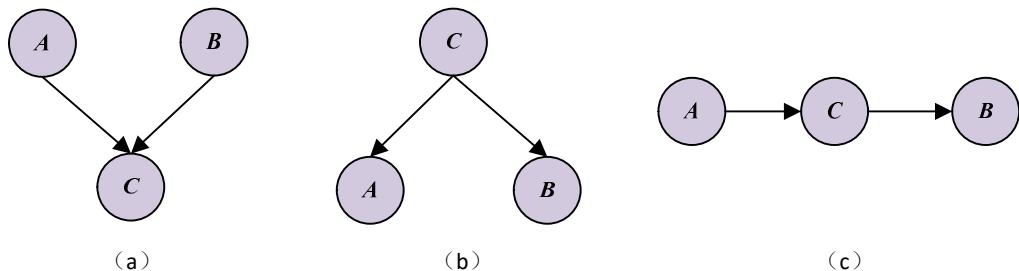


图 11-6 贝叶斯网络的 3 种基本结构

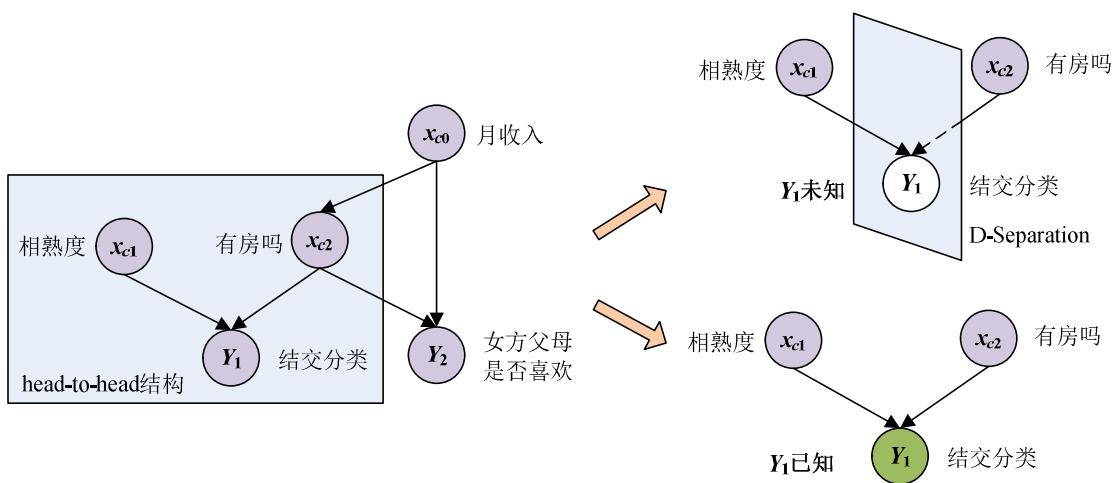


图 11-7 head-to-head 结构的示例

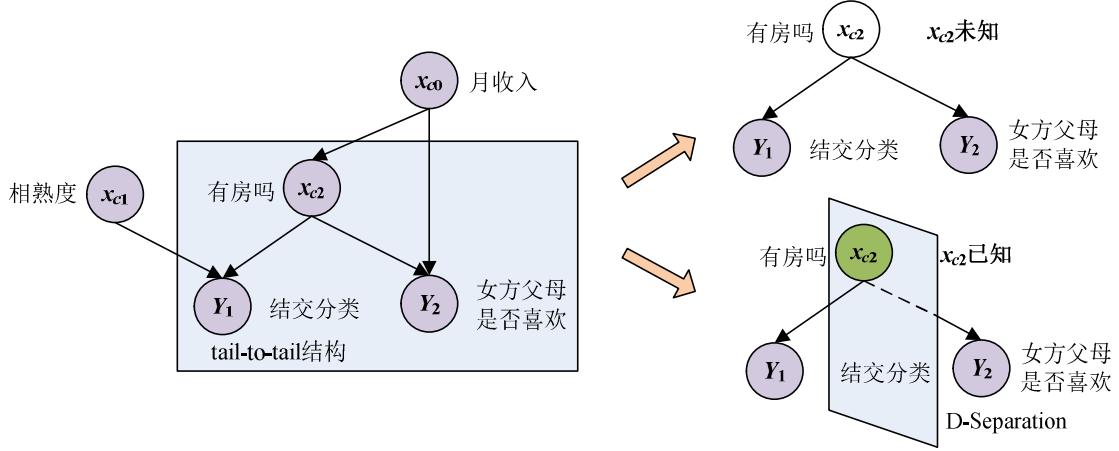


图 11-8 tail-to-tail 结构的示例

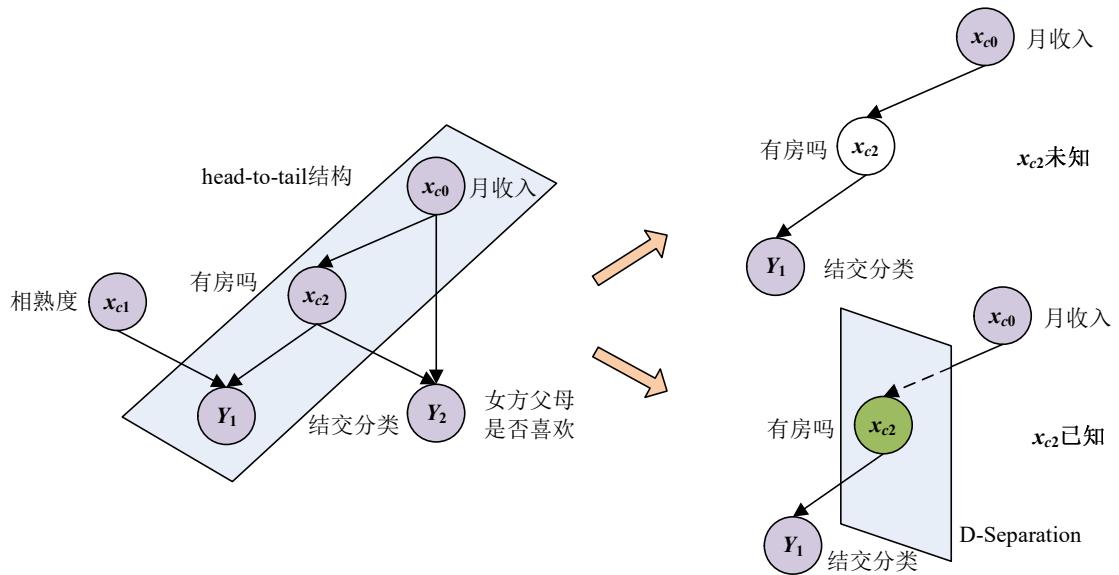


图 11-9 head-to-tail 结构的示例

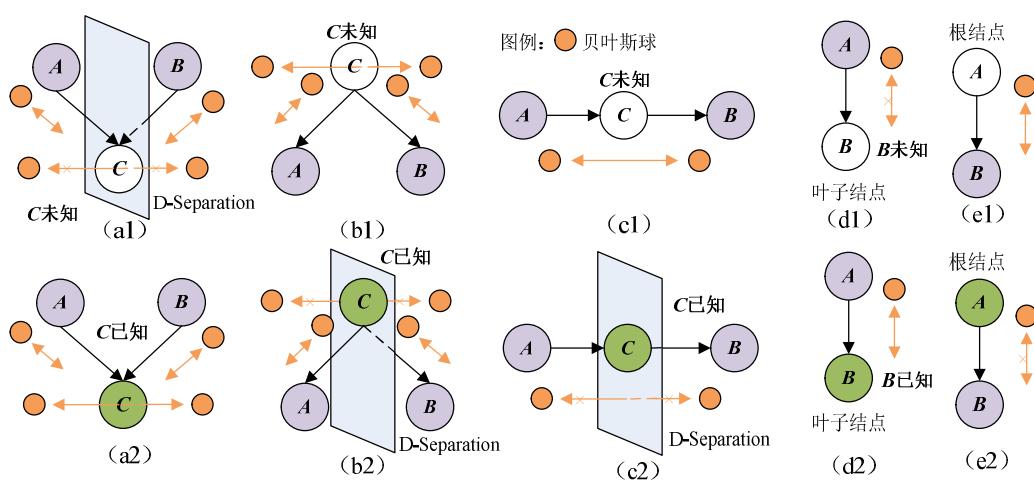


图 11-10 3 种基本结构对应的 6 条规则

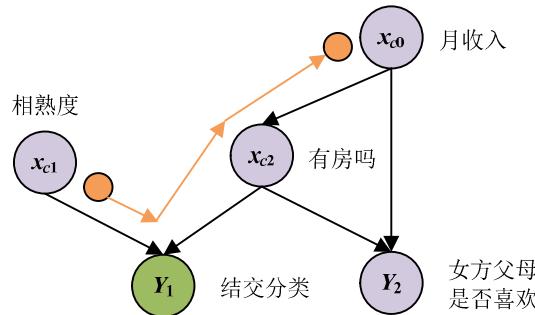


图 11-11 打贝叶斯球的过程

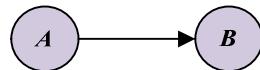


图 11-12 创建的简单的贝叶斯网络

```

Console 1/A ✘
True
(Association ⊥ Income, Parentslike | House)
(Association ⊥ Income, Parentslike | Familiarity, House)
(Association ⊥ Parentslike | Income, House)
(Association ⊥ Income | Parentslike, House)
(Association ⊥ Parentslike | Familiarity, Income, House)
(Association ⊥ Income | Familiarity, Parentslike, House)
(Familiarity ⊥ Income, Parentslike, House)
(Familiarity ⊥ Income, Parentslike | House)
(Familiarity ⊥ Parentslike, House | Income)
(Familiarity ⊥ Income, House | Parentslike)
(Familiarity ⊥ Parentslike | Income, House)
(Familiarity ⊥ Income | Parentslike, House)
(Familiarity ⊥ Income, Parentslike | Association, House)
(Familiarity ⊥ House | Income, Parentslike)
(Familiarity ⊥ Parentslike | Association, Income, House)
(Familiarity ⊥ Income | Association, Parentslike, House)
(Income ⊥ Familiarity)
(Income ⊥ Familiarity, Association | House)
(Income ⊥ Familiarity | Parentslike)
(Income | Association | Familiarity House)

```

图 11-13 构建的贝叶斯网络的依赖关系

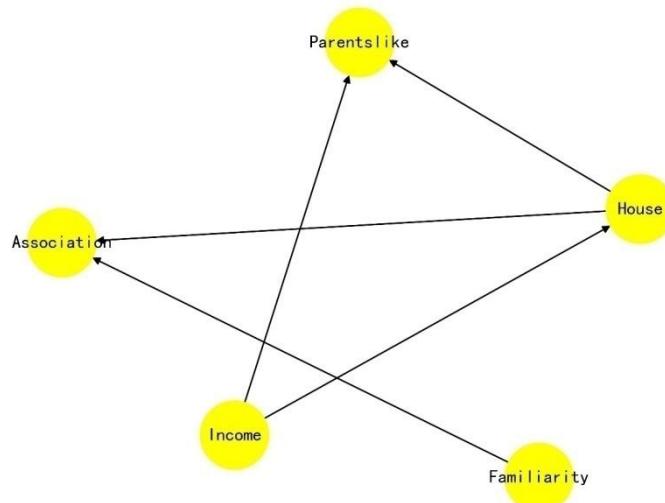


图 11-14 构建的贝叶斯网络的结构

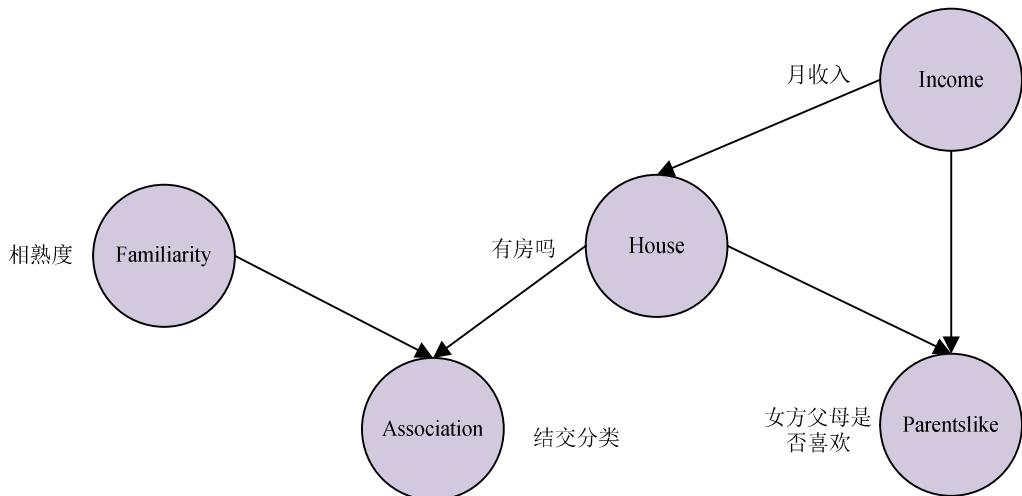


图 11-15 调整后等价的贝叶斯网络

Console 1/A	
+-----+-----+	
Familiarity(0)   0.2	
+-----+-----+	
Familiarity(1)   0.3	
+-----+-----+	
Familiarity(2)   0.5	
+-----+-----+	

图 11-16 结点 Familiarity 的概率分布表

Console 1/A	
+-----+-----+-----+-----+	
Income   Income(0)   Income(1)   Income(2)	
+-----+-----+-----+-----+	
House(0)   0.8   0.5   0.1	
+-----+-----+-----+-----+	
House(1)   0.2   0.5   0.9	
+-----+-----+-----+-----+	

图 11-17 “House” 结点的条件概率表

Console 1/A	
+-----+-----+-----+-----+	
Familiarity   Familiarity(0)   ...   Familiarity(2)   Familiarity(2)	
+-----+-----+-----+-----+	
House   House(0)   ...   House(0)   House(1)	
+-----+-----+-----+-----+	
Association(0)   0.1   ...   0.4   0.4	
+-----+-----+-----+-----+	
Association(1)   0.3   ...   0.5   0.5	
+-----+-----+-----+-----+	
Association(2)   0.6   ...   0.1   0.1	
+-----+-----+-----+-----+	

图 11-18 “Association” 结点的条件概率表

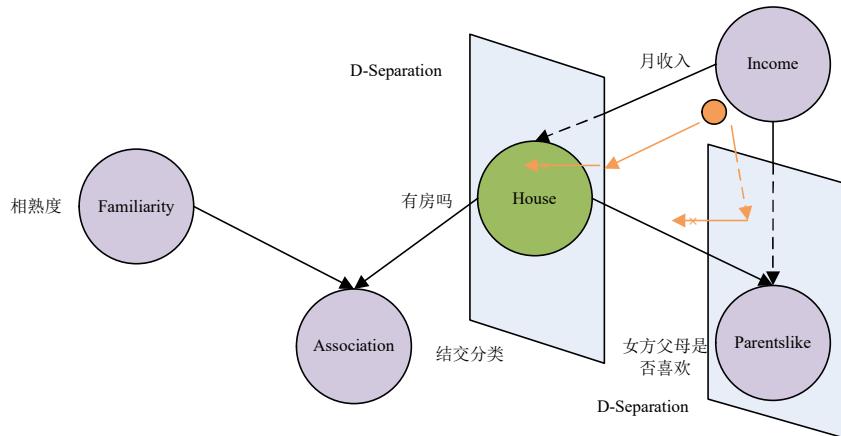


图 11-19 已知“House”事件时“Association”事件与“Income”事件的独立性

```

Console 1/A ×
所有结点: ['Familiarity', 'Association', 'Income', 'House', 'Parentslike']
叶子结点: ['Association', 'Parentslike']
0%| 0/2 [00:00<?, ?it/s]
Parentslike Association
0 1 1
1 0 2
Parentslike_0 Parentslike_1 Association_0 Association_1 Association_2
0 0.1 0.9 0.3 0.4 0.3
1 0.7 0.3 0.1 0.3 0.6
In [40]: |

```

图 11-20 用贝叶斯网络预测的结果

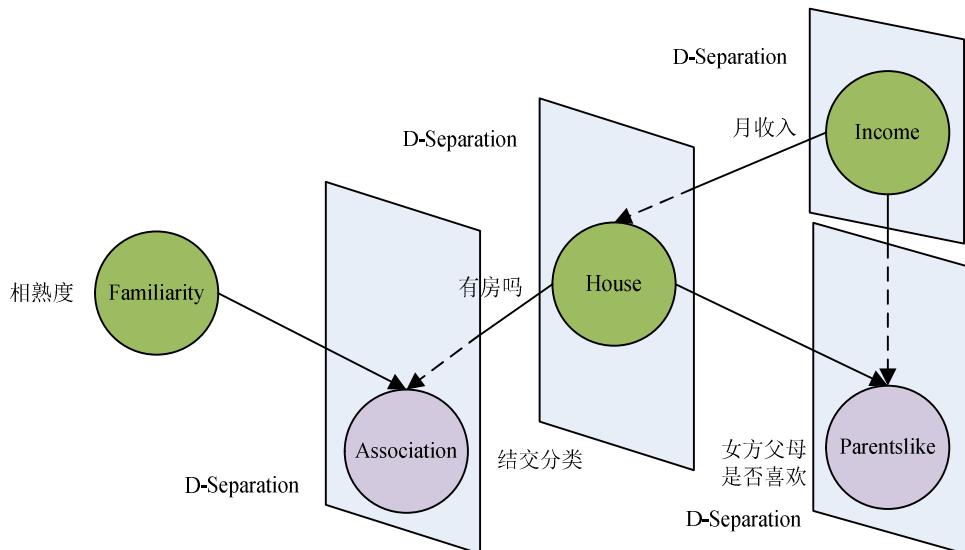


图 11-21 加上 D-Separation 的贝叶斯网络

```
Console 1/A ×
CPD of Parentslike:
+-----+-----+-----+-----+
| House | House(0) | ... | House(1) | House(1) |
+-----+-----+-----+-----+
| Income | Income(0) | ... | Income(1) | Income(2) |
+-----+-----+-----+-----+
| Parentslike(0) | 1.0 | ... | 0.0 | 0.0 |
+-----+-----+-----+-----+
| Parentslike(1) | 0.0 | ... | 1.0 | 1.0 |
+-----+-----+-----+-----+
所有结点: ['Familiarity', 'Association', 'Income', 'House', 'Parentslike']
叶子结点: ['Association', 'Parentslike']
 0% | 0/2 [00:00<?, ?it/s]
    Association Parentslike
0      0      1
1      0      0
  Association_0  Association_1  Association_2  Parentslike_0  Parentslike_1
0      1.000000      0.000000      0.0      0.0      1.0
1      0.666667      0.333333      0.0      1.0      0.0
In [7]: |
```

图 11-22 自动学习贝叶斯网络的参数

# 第 12 章

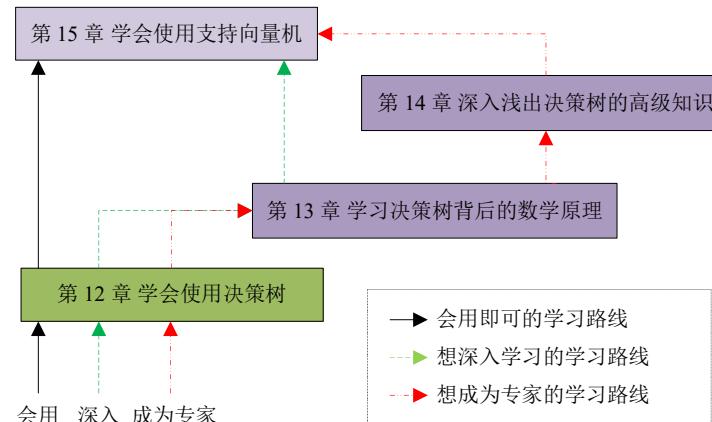


图 12-1 学习路线图

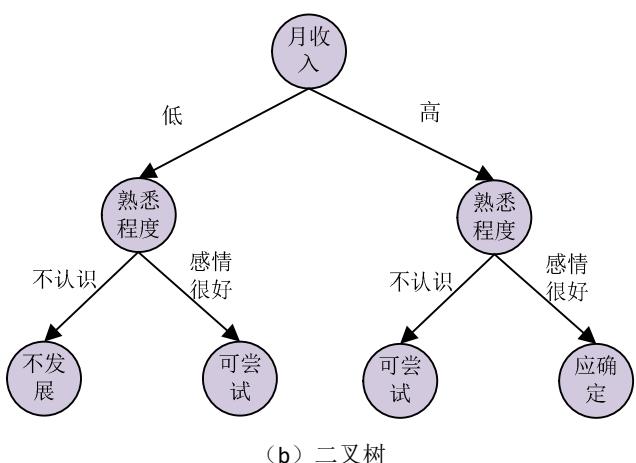
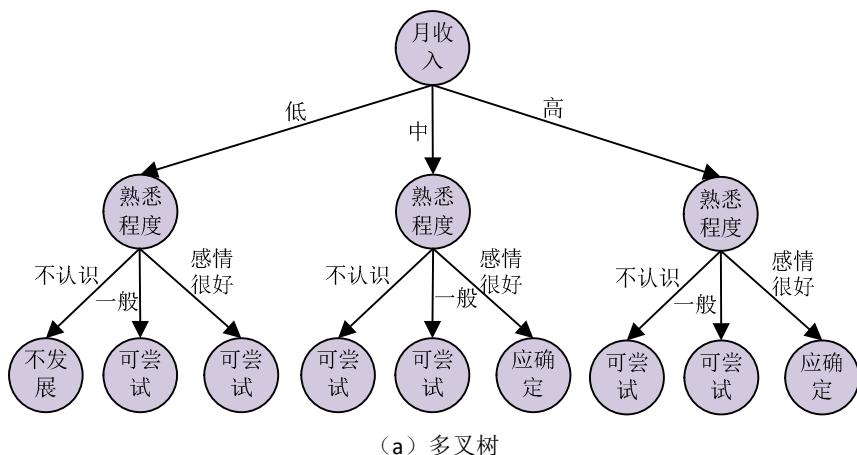


图 12-2 多叉树和二叉树

```
Console 1/A X
[3 1]
[[0. 0. 1. ]
 [0.5 0.5 0. ]]
In [4]:
```

图 12-3 用决策树辅助某姑娘做相亲后的决策

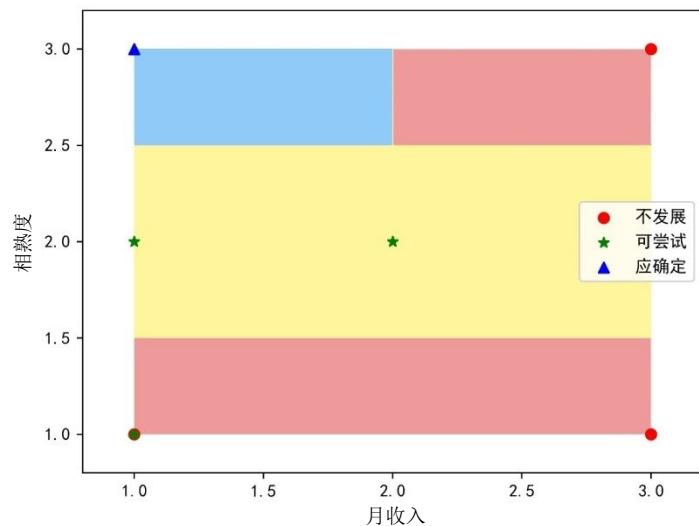


图 12-4 DecisionTreeClassifier 类辅助某姑娘做相亲后的决策效果图

```
Console 1/A X
[0. 1. 0.]
[1. 0. 0.]
[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]
precision    recall    f1-score   support
  setosa      1.00      1.00      1.00       11
versicolor   1.00      0.92      0.96       13
 virginica   0.86      1.00      0.92        6
accuracy           0.95      0.97      0.96       30
macro avg      0.95      0.97      0.96       30
weighted avg   0.97      0.97      0.97       30
In [6]:
```

图 12-5 用决策树做鸢尾花分类

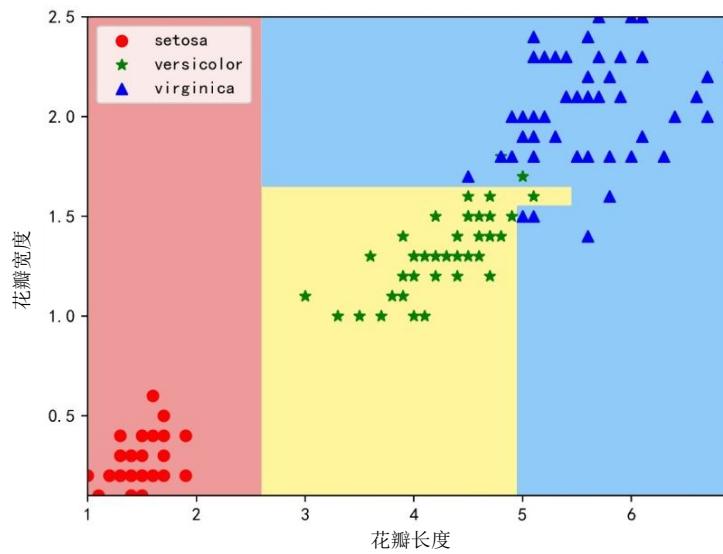


图 12-6 画出决策树对鸢尾花分类的分界线

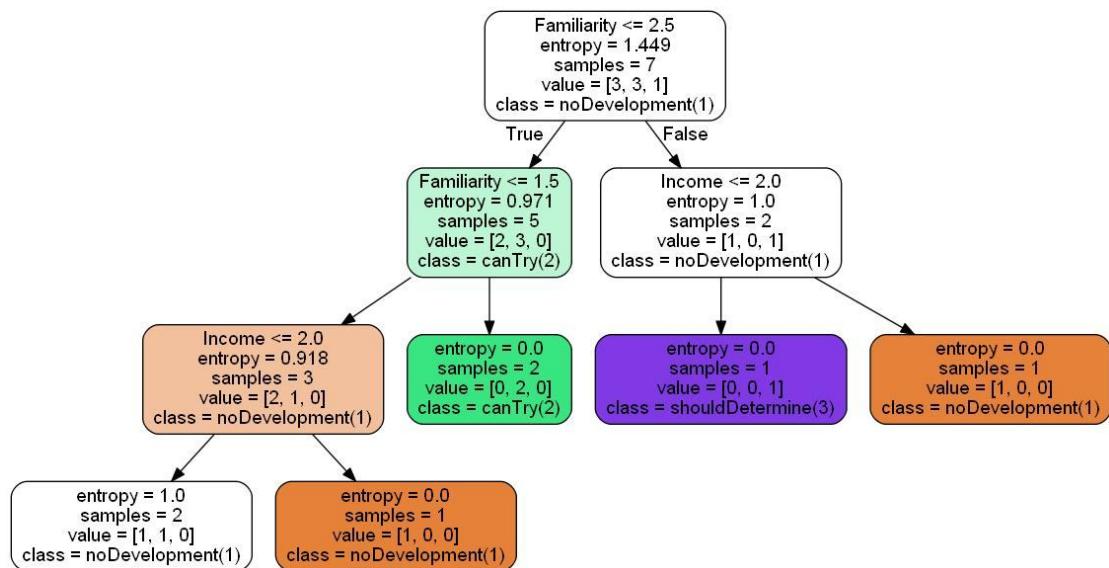


图 12-7 某姑娘相亲后决策的决策树

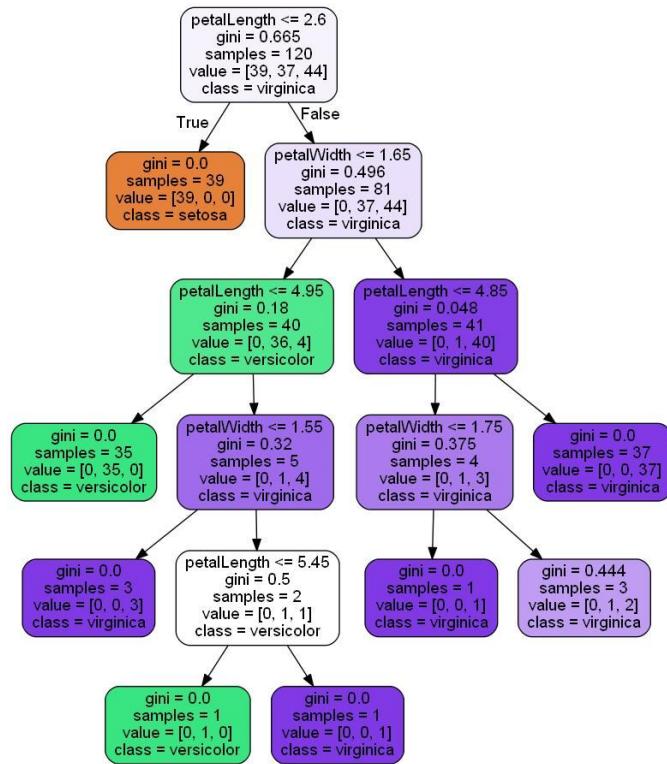


图 12-8 鸢尾花分类的决策树

# 第 13 章

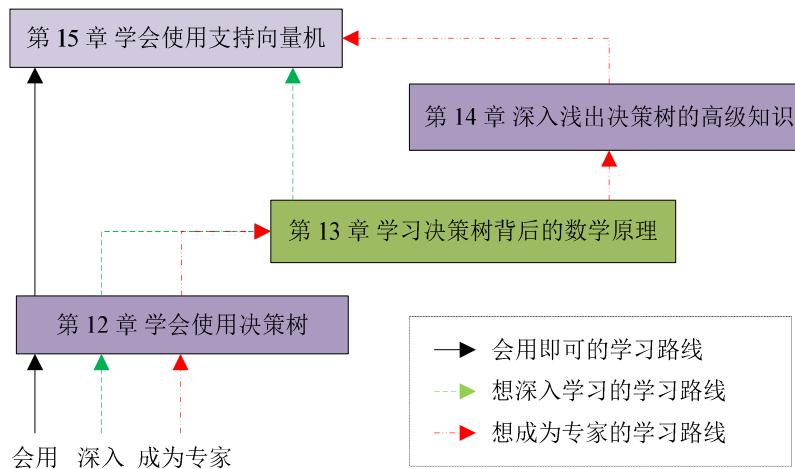


图 13-1 学习路线图

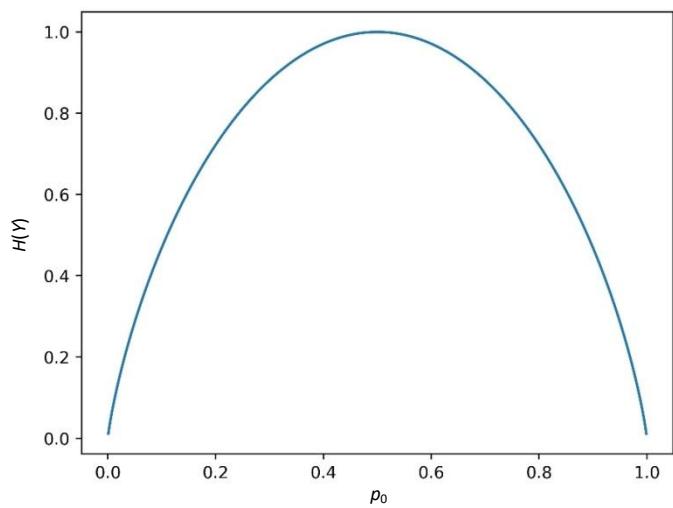


图 13-2 数据项只有 2 种取值时式 (13-1) 的图形

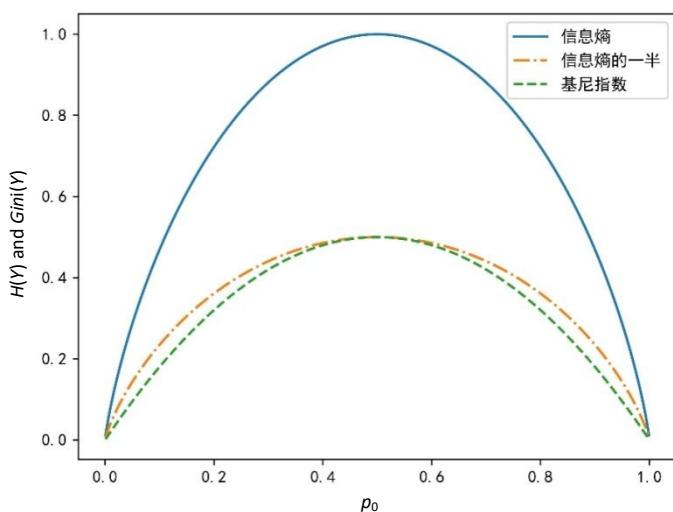


图 13-3 基尼指数的图形

训练数据集 $D$	特征数据项 $x_{c0}$	...	特征数据项 $x_{c(n-1)}$	目标数据项 $Y$
样本 $x_{i0}$	$x_{i0c0}$	...	$x_{i0c(n-1)}$	$y_0$
...	...	...	...	...
样本 $x_{i(n-1)}$	$x_{i(n-1)c0}$	...	$x_{i(n-1)c(n-1)}$	$y_{n-1}$

目标数据项  $Y$  的取值有  $k$  种, 这些取值的集合为  $C = \{c_0, \dots, c_{k-1}\}$

图 13-4 当前训练数据集的图示

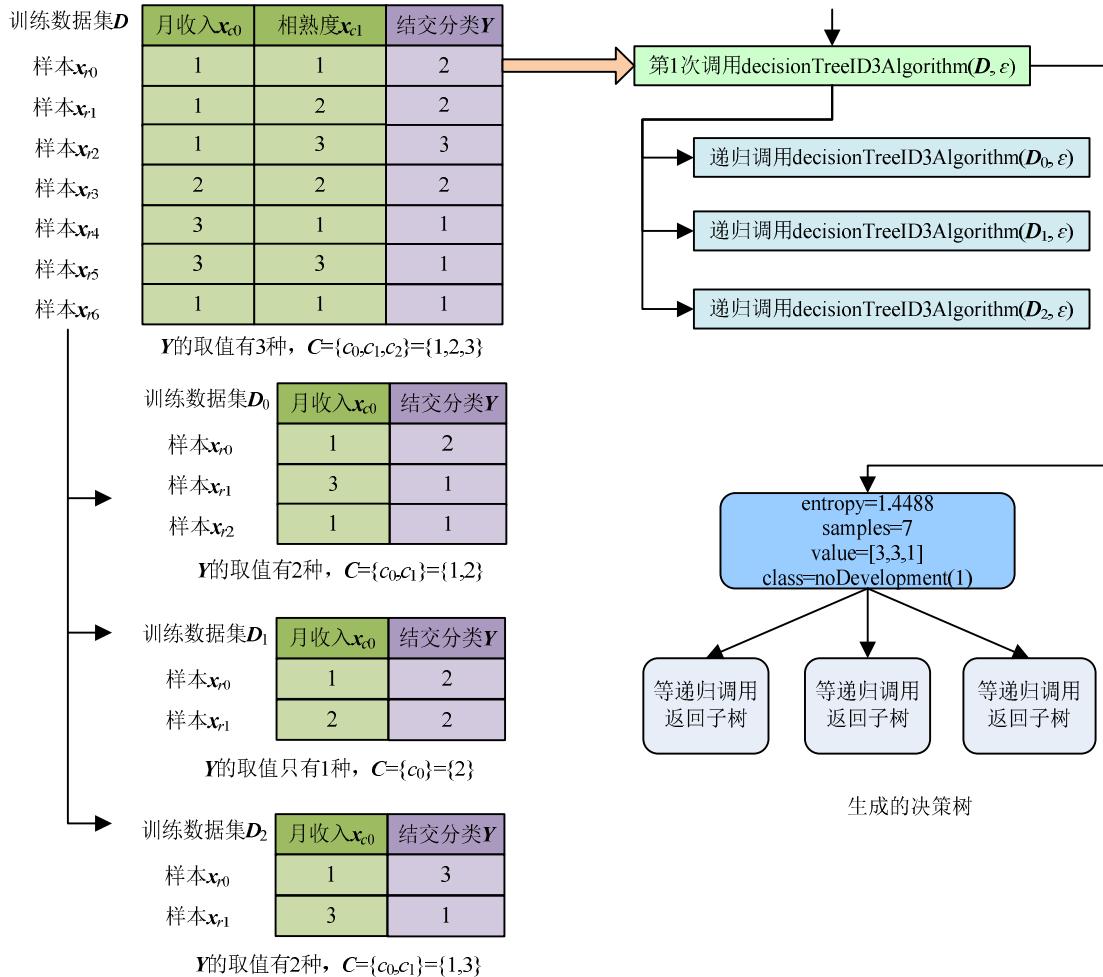


图 13-5 第 1 次调用  $\text{decisionTreeID3Algorithm}(D, \varepsilon)$

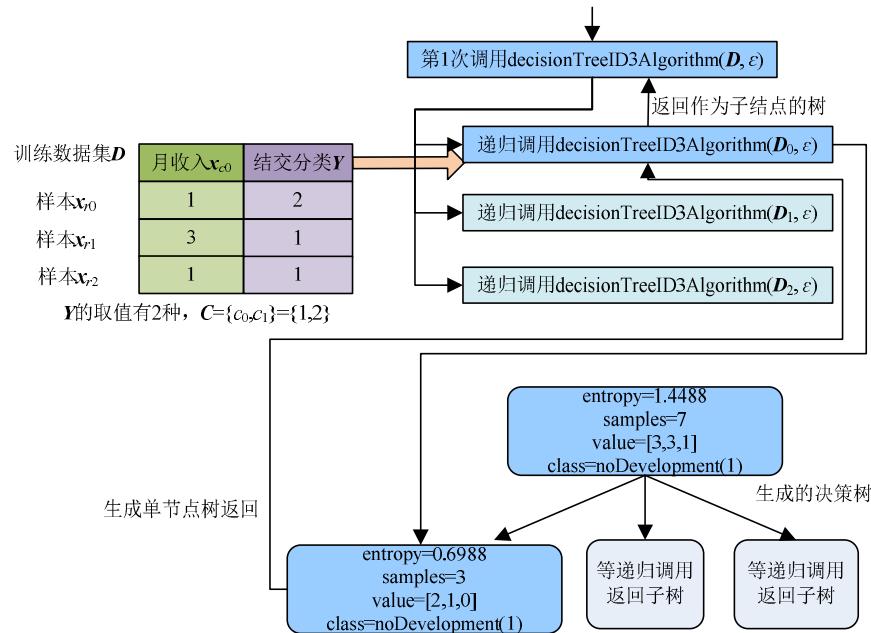


图 13-6 递归调用  $\text{decisionTreeID3Algorithm}(D_0, \epsilon)$

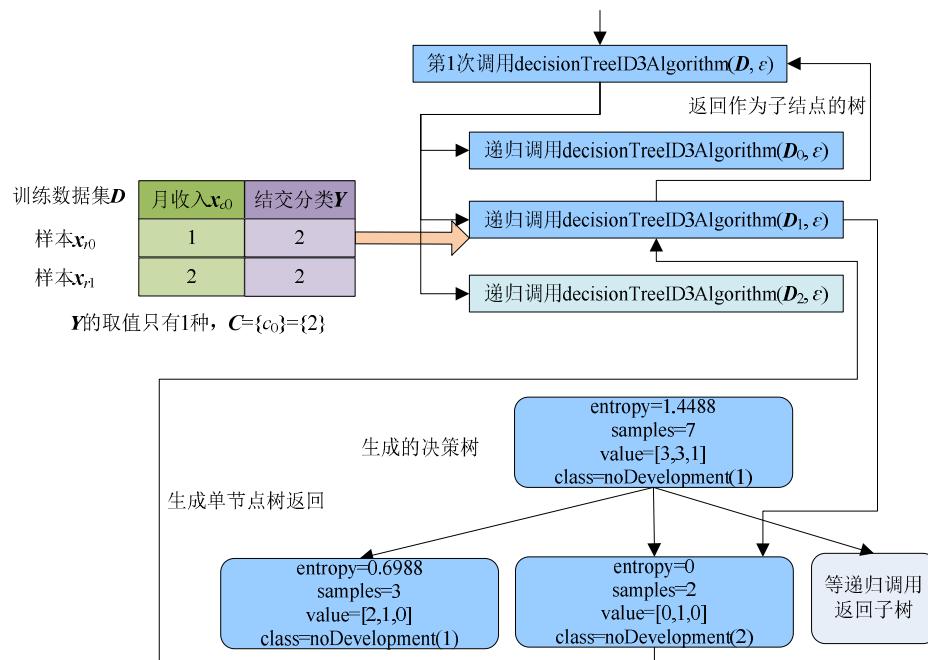


图 13-7 递归调用  $\text{decisionTreeID3Algorithm}(D_1, \epsilon)$

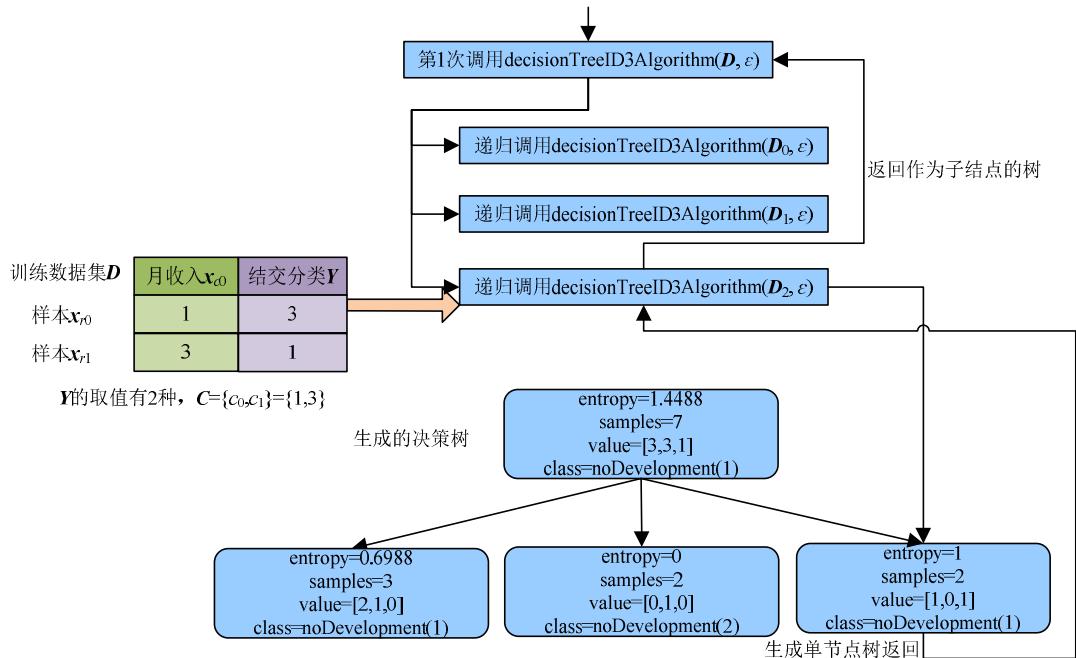


图 13-8 递归调用  $\text{decisionTreeID3Algorithm}(D_2, \varepsilon)$

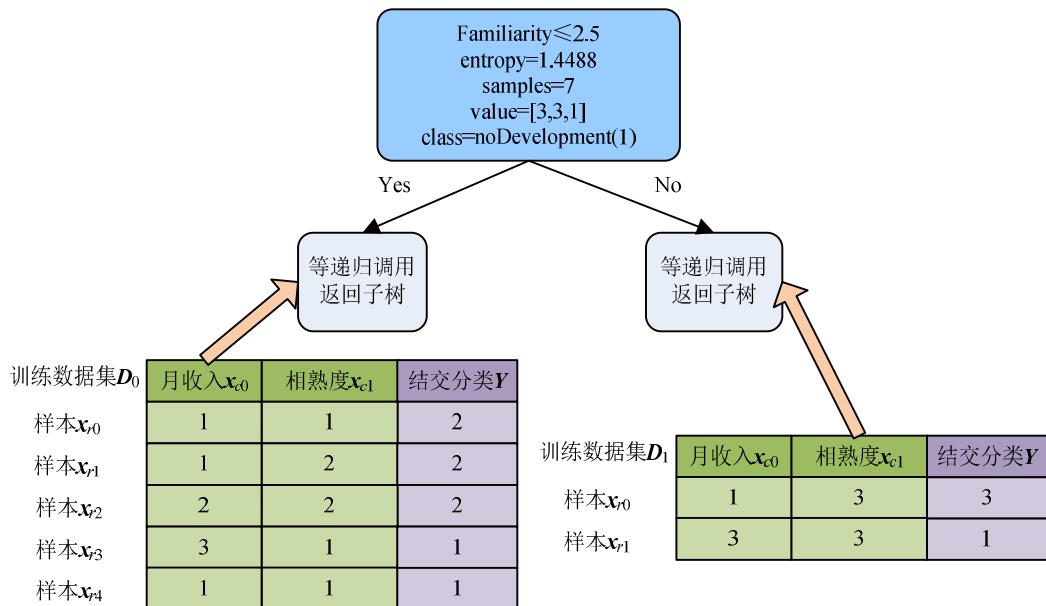
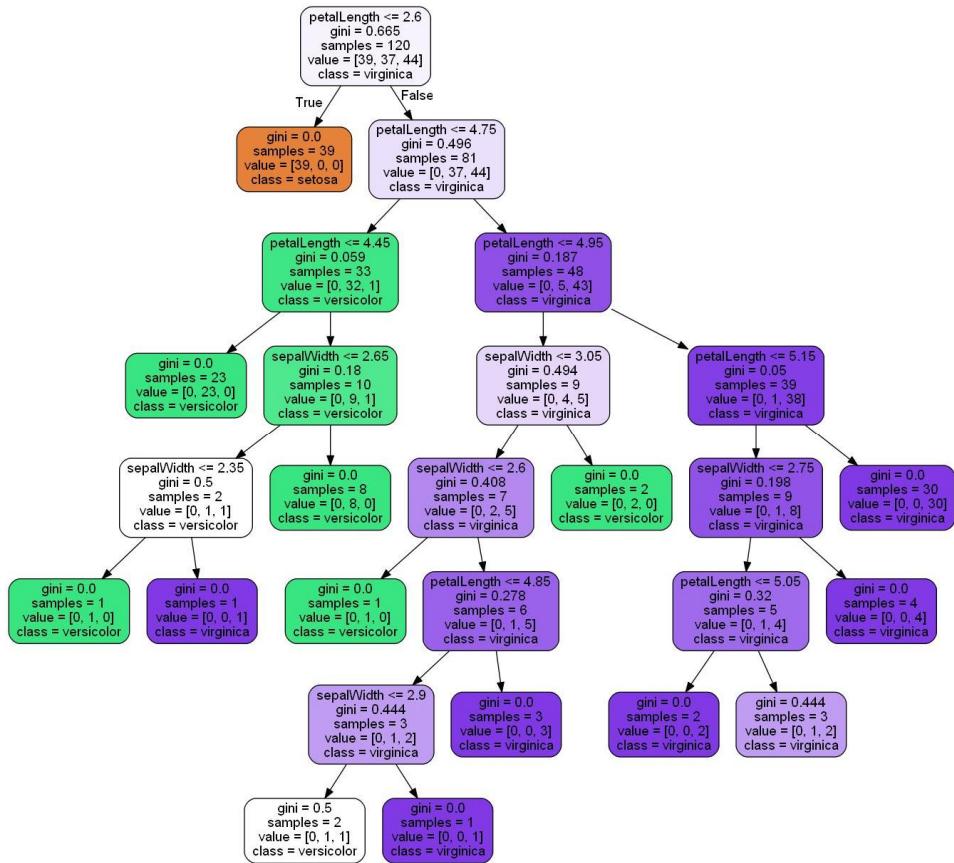
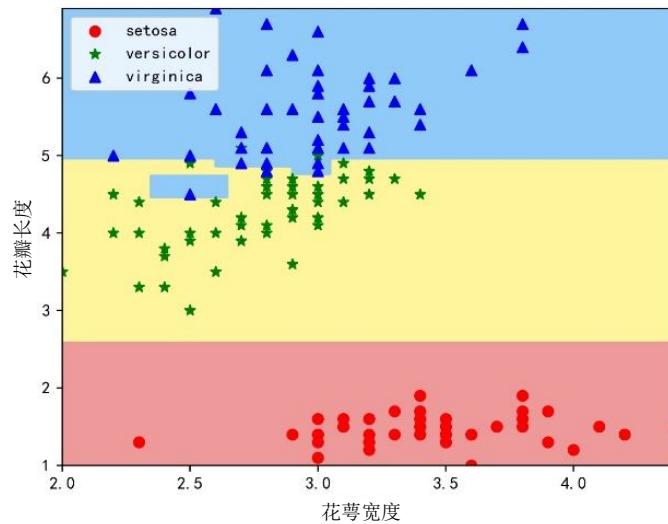


图 13-9 第 1 次分叉及切分方案

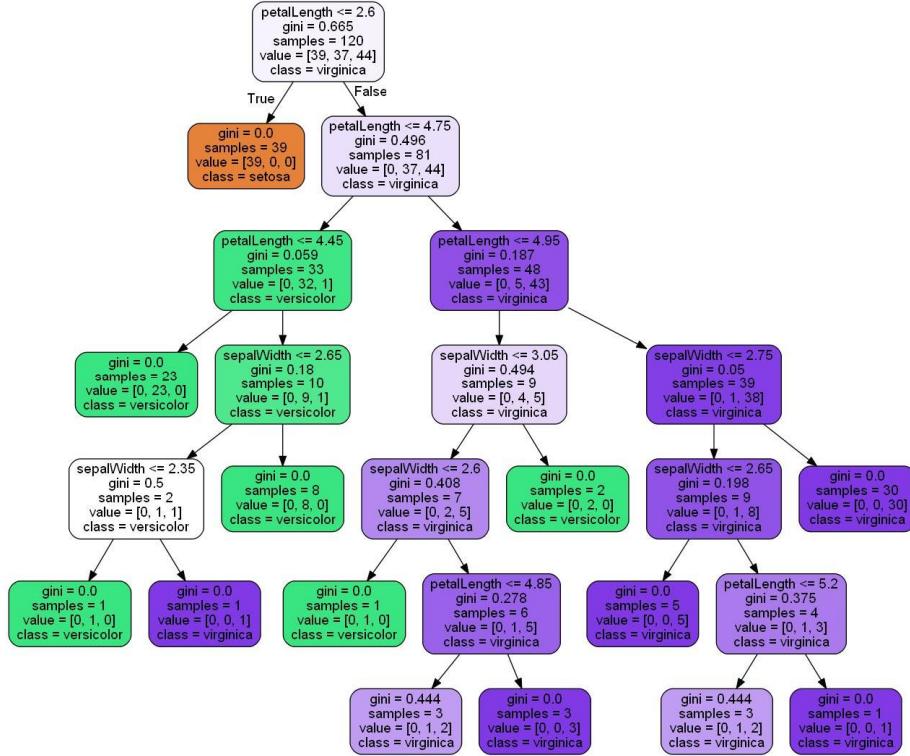


(a) 不限制树的深度时的决策树

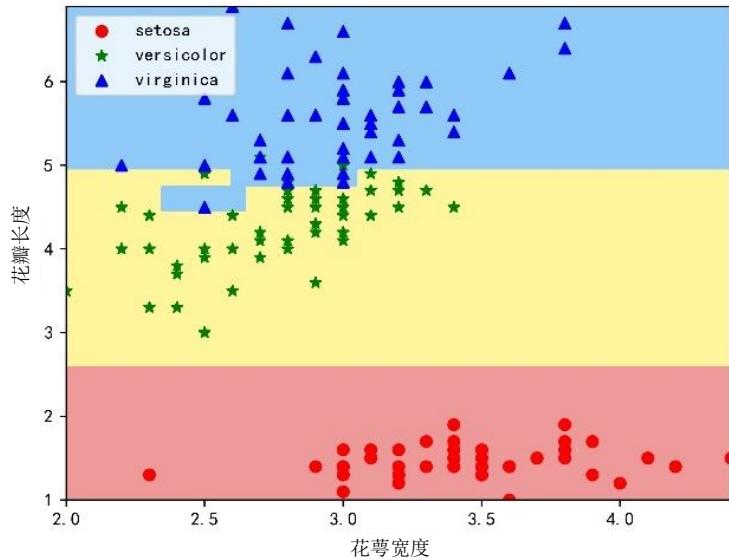


(b) 不限制树的深度时的分类效果

图 13-10 鸢尾花分类的决策树及分类效果

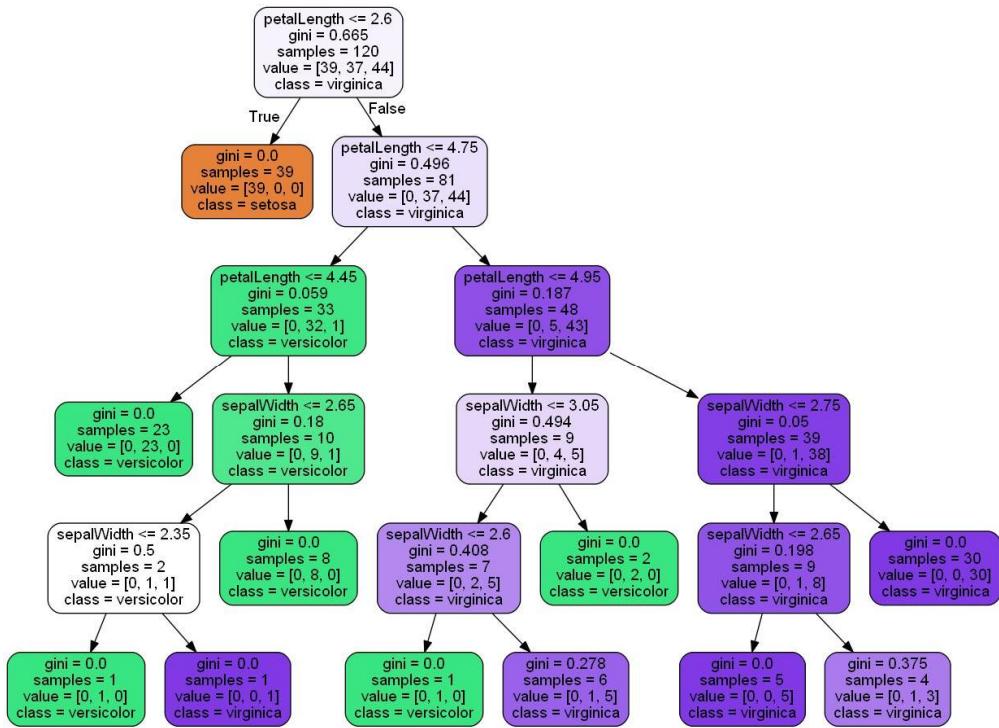


(c) 树的深度最大为 6 时的决策树

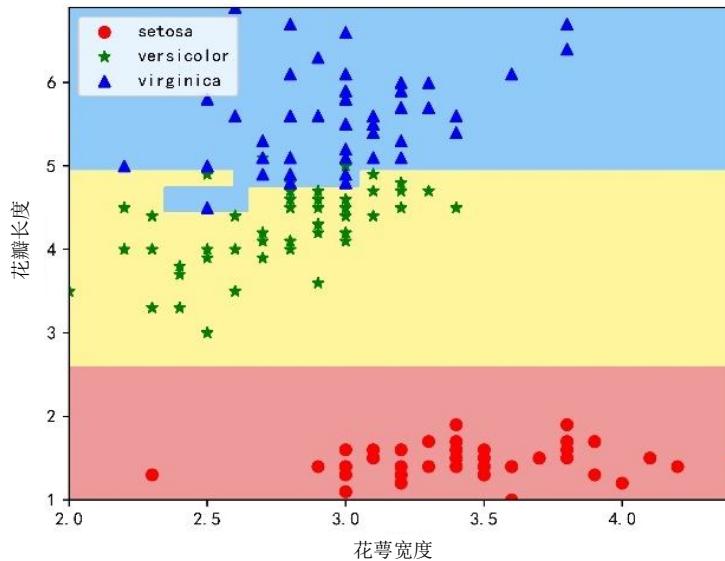


(d) 树的深度最大为 6 时的分类效果

图 13-10 鸢尾花分类的决策树及分类效果 (续图)

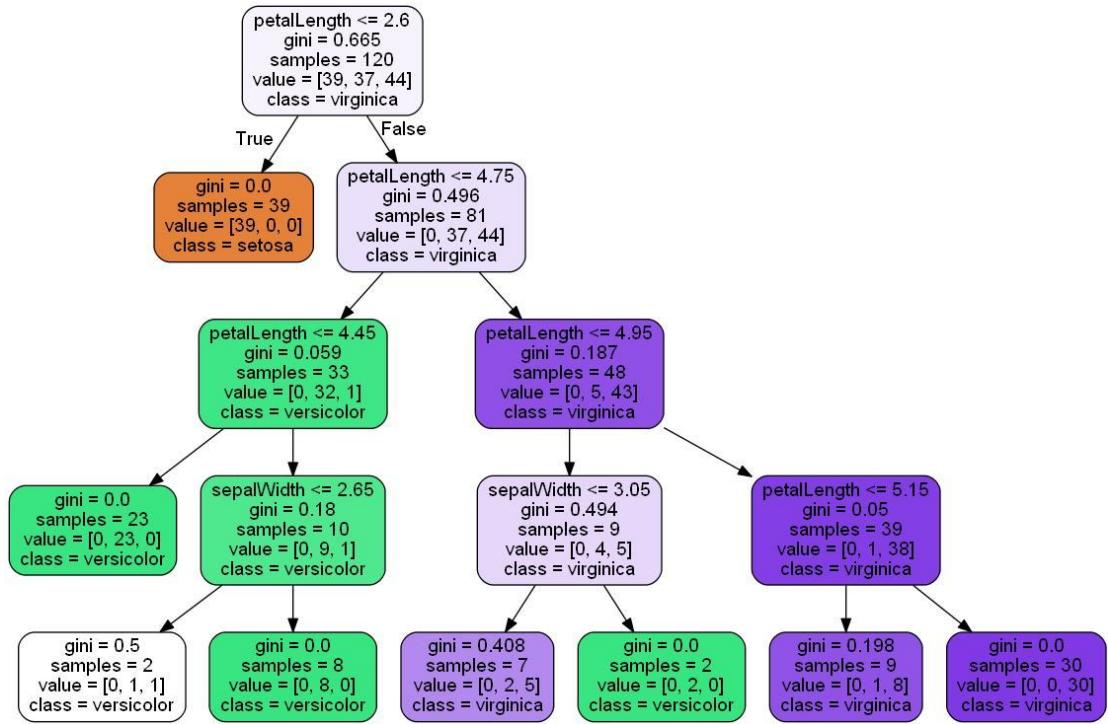


(e) 树的深度最大为 5 时的决策树

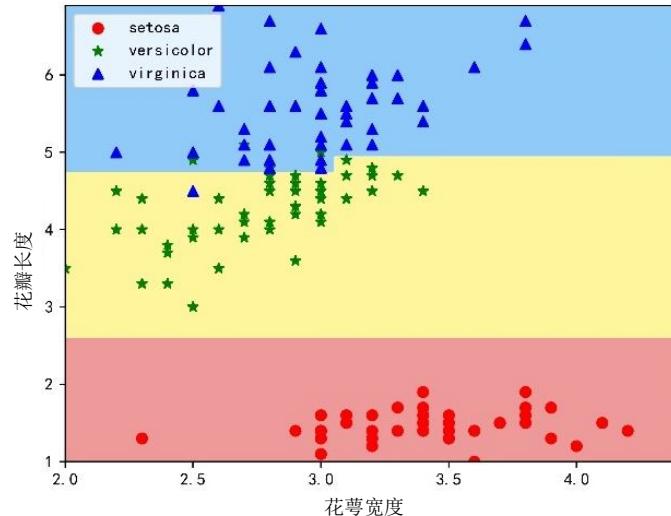


(f) 树的深度最大为 5 时的分类效果

图 13-10 翩尾花分类的决策树及分类效果 (续图)

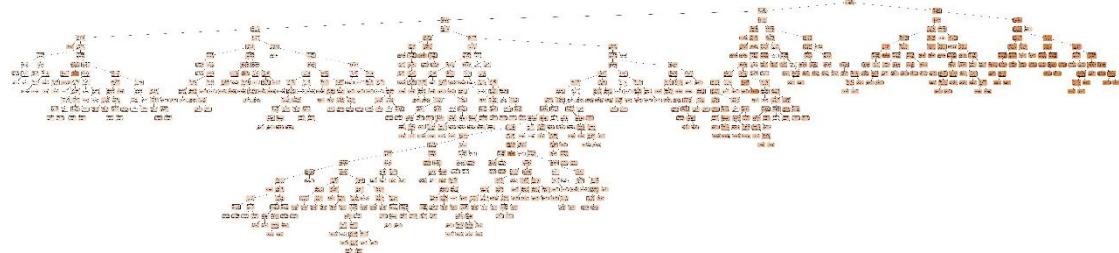


(g) 树的深度最大为 4 时的决策树



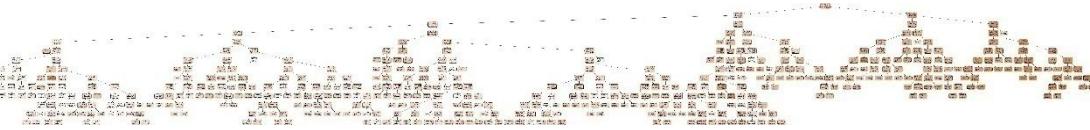
(h) 树的深度最大为 4 时的分类效果

图 13-10 鸢尾花分类的决策树及分类效果 (续图)

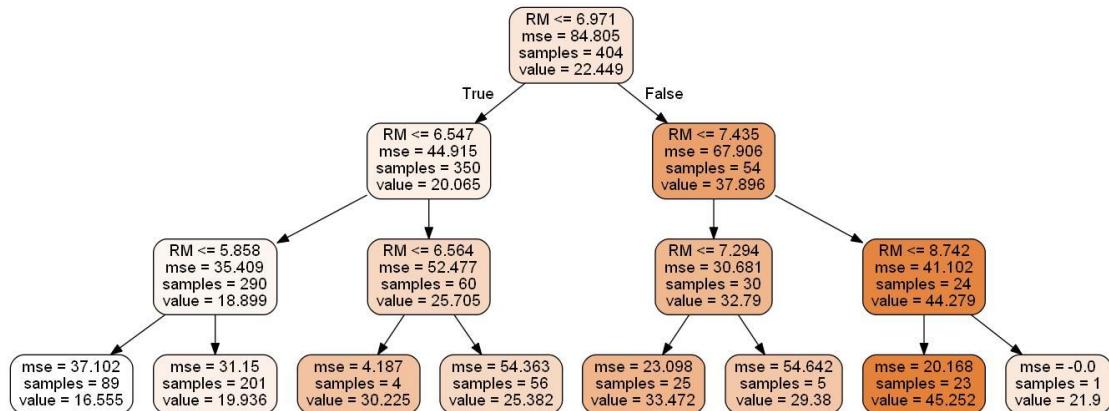


(a) 不限制树的深度时的决策树

图 13-11 房价回归分析的决策树



(b) 树的深度最大为 13 时的决策树



(c) 树的深度最大为 3 时的决策树

图 13-11 房价回归分析的决策树 (续图)

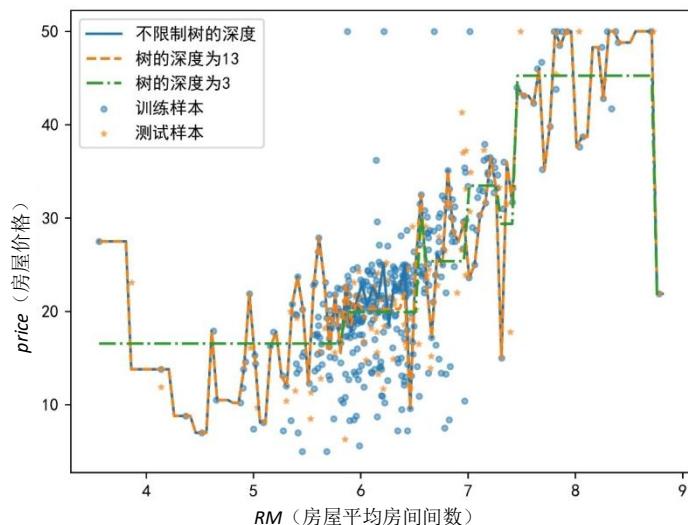


图 13-12 房价回归分析的决策树的拟合情况

# 第 14 章

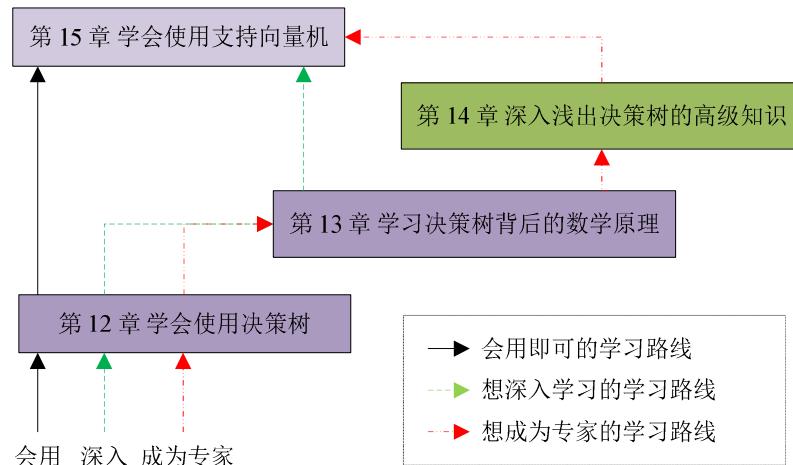


图 14-1 学习路线图

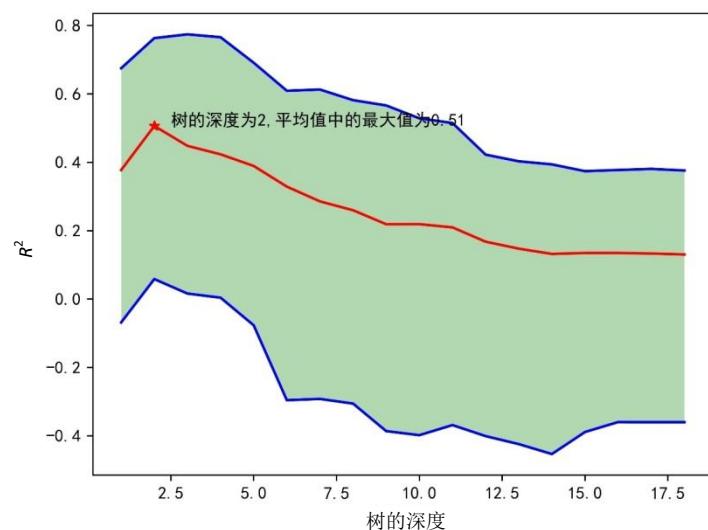


图 14-2 不同深度的决策树的  $R^2$

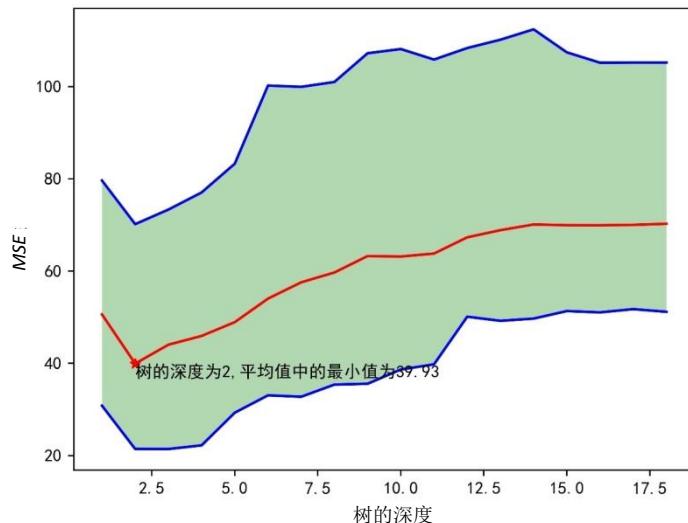


图 14-3 不同深度的决策树的  $MSE$

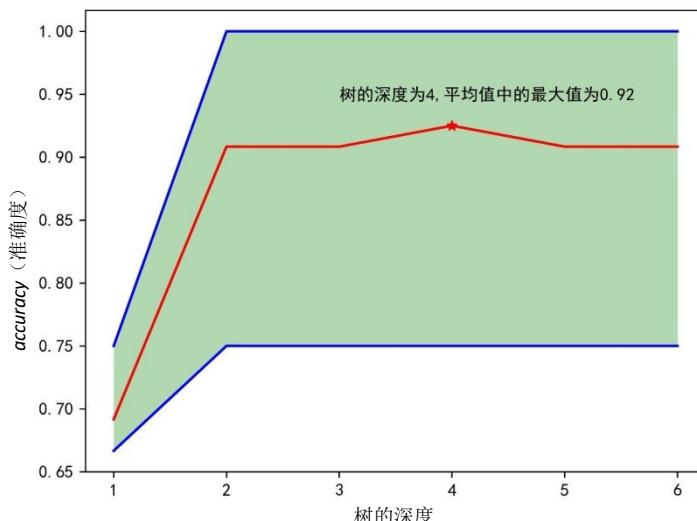


图 14-4 不同深度的决策树的准确度

```
Console 1/1 x
[1]: from sklearn import DecisionTreeRegressor, cross_val_score
最优的模型是: DecisionTreeRegressor(max_depth=17, min_samples_leaf=2)
最好的成绩(R2)是: 0.29807590258487915
最优的参数是: {'max_depth': 17, 'min_samples_leaf': 2, 'min_samples_split': 2}
最好成绩的模型索引是: 2
执行的是几折交叉验证: 10
结果列表: ['mean_fit_time', 'mean_score_time', 'mean_test_score', 'mean_train_score', 'param_max_depth', 'param_min_samples_leaf', 'param_min_samples_split', 'params', 'rank_test_score', 'split0_test_score', 'split0_train_score', 'split1_test_score', 'split1_train_score', 'split2_test_score', 'split2_train_score', 'split3_test_score', 'split3_train_score', 'split4_test_score', 'split4_train_score', 'split5_test_score', 'split5_train_score', 'split6_test_score', 'split6_train_score', 'split7_test_score', 'split7_train_score', 'split8_test_score', 'split8_train_score', 'split9_test_score', 'split9_train_score', 'std_fit_time', 'std_score_time', 'std_test_score', 'std_train_score']
结果:
{'mean_fit_time': array([0.00129333, 0.00089388, 0.00089674, 0.00079069, 0.00110073, 0.00092015, 0.00079398, 0.00099697]), 'std_fit_time': array([0.00045044, 0.00029816, 0.00053713, 0.0003956, 0.00029826, 0.00030095, 0.00039716, 0.00077177]), 'mean_score_time': array([0.00030222, 0.00049853, 0.00059905, 0.00049837, 0.00049911, 0.00029519, 0.00069759, 0.00059855]), 'std_score_time': array([0.00046173, 0.00049853, 0.00048912, 0.00049837, 0.00049943, 0.00045103, 0.00063845, 0.00048871]), 'param_max_depth': masked_array(data=[17, 17, 17, 17, 18, 18, 18, 18], mask=[False, False, False, False, False, False, False, False], fill_value='?'), 'param_min_samples_leaf': masked_array(data=[1, 1, 2, 2, 1, 1, 2, 2], mask=[False, False, False, False, False, False, False, False], fill_value='?'), 'params': {'max_depth': 17, 'min_samples_leaf': 2, 'min_samples_split': 2}, 'rank_test_score': array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]), 'split0_train_score': 0.29807590258487915, 'split0_test_score': 0.29807590258487915, 'split1_train_score': 0.29807590258487915, 'split1_test_score': 0.29807590258487915, 'split2_train_score': 0.29807590258487915, 'split2_test_score': 0.29807590258487915, 'split3_train_score': 0.29807590258487915, 'split3_test_score': 0.29807590258487915, 'split4_train_score': 0.29807590258487915, 'split4_test_score': 0.29807590258487915, 'split5_train_score': 0.29807590258487915, 'split5_test_score': 0.29807590258487915, 'split6_train_score': 0.29807590258487915, 'split6_test_score': 0.29807590258487915, 'split7_train_score': 0.29807590258487915, 'split7_test_score': 0.29807590258487915, 'split8_train_score': 0.29807590258487915, 'split8_test_score': 0.29807590258487915, 'split9_train_score': 0.29807590258487915, 'split9_test_score': 0.29807590258487915}
```

图 14-5 用 GridSearchCV 调节决策树模型的参数

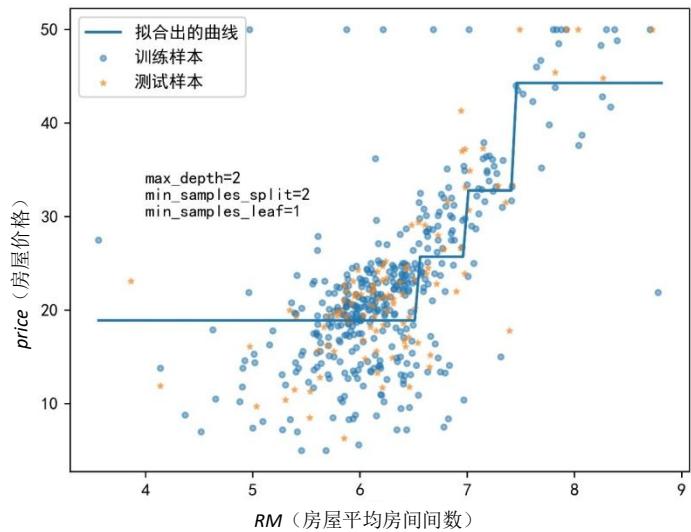


图 14-6 用 GridSearchCV 调节决策树模型的参数后得到的拟合曲线

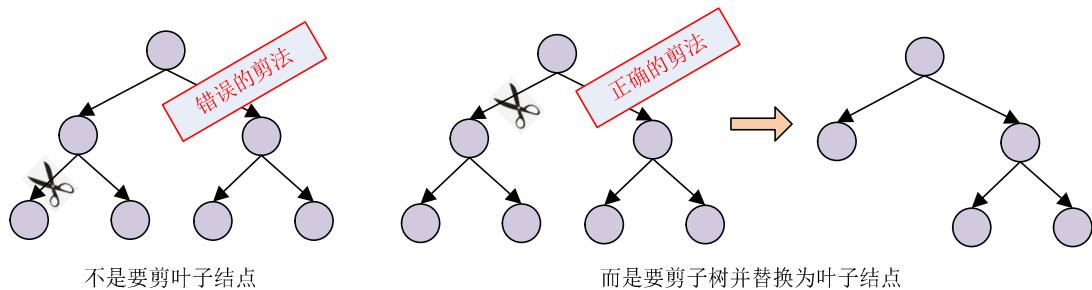


图 14-7 剪枝就是要把子树替换成叶子结点

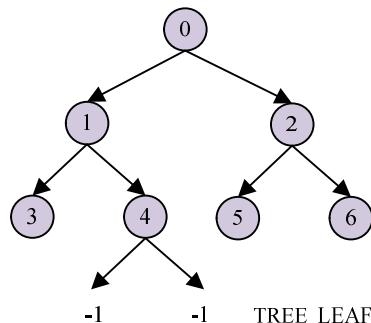


图 14-8 叶子结点的图示

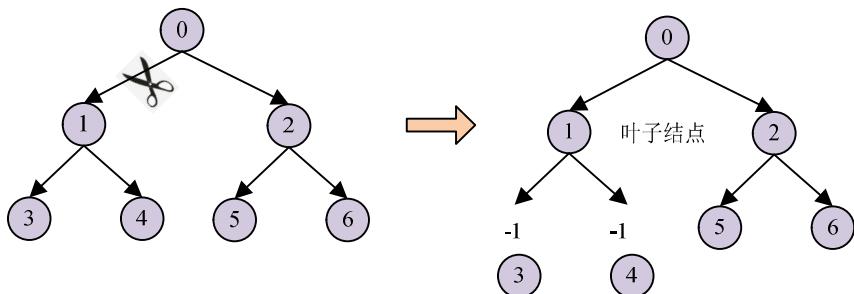


图 14-9 剪枝的做法

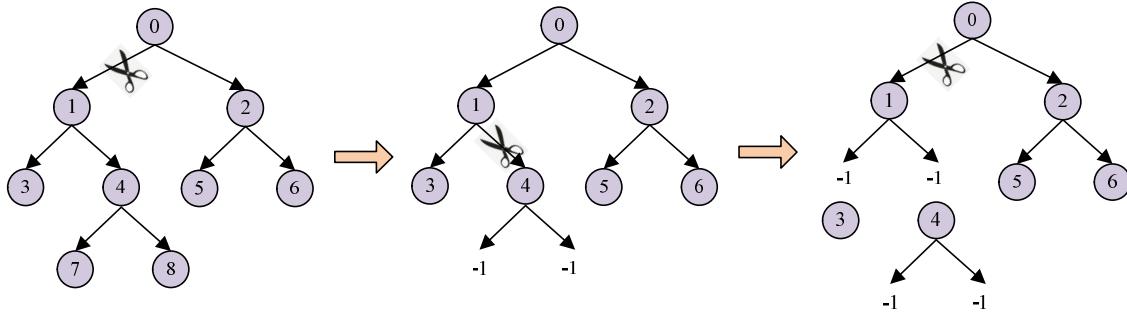


图 14-10 从下往上剪枝

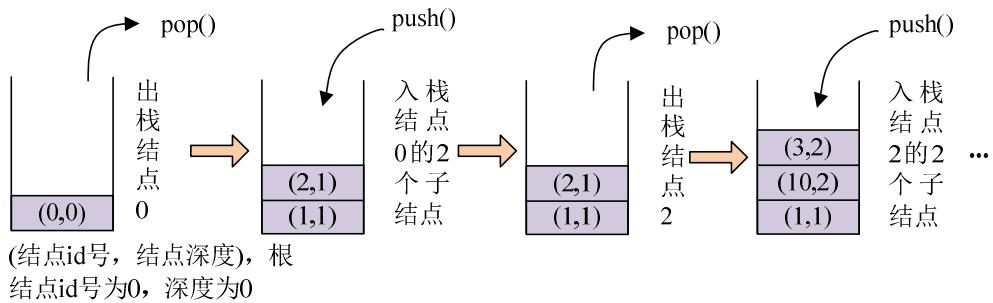


图 14-11 用堆栈遍历决策树

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

图 14-12 CSR 矩阵的表示

```

Console 1/A ×
Reloaded modules: treeOperation
剪枝前对训练数据集的精度: 0.9833333333333333
剪枝前对验证数据集的精度: 0.9666666666666667
第 3 层结点 20 要剪枝
剪去第23个结点
剪去第21个结点
剪去第20个结点
第 2 层结点 3 要剪枝
剪去第6个结点
剪去第5个结点
剪去第3个结点
剪枝后对训练数据集的精度: 0.975
剪枝后对验证数据集的精度: 0.9666666666666667
In [5]:

```

图 14-13 剪枝的过程及剪枝前后精度的对比

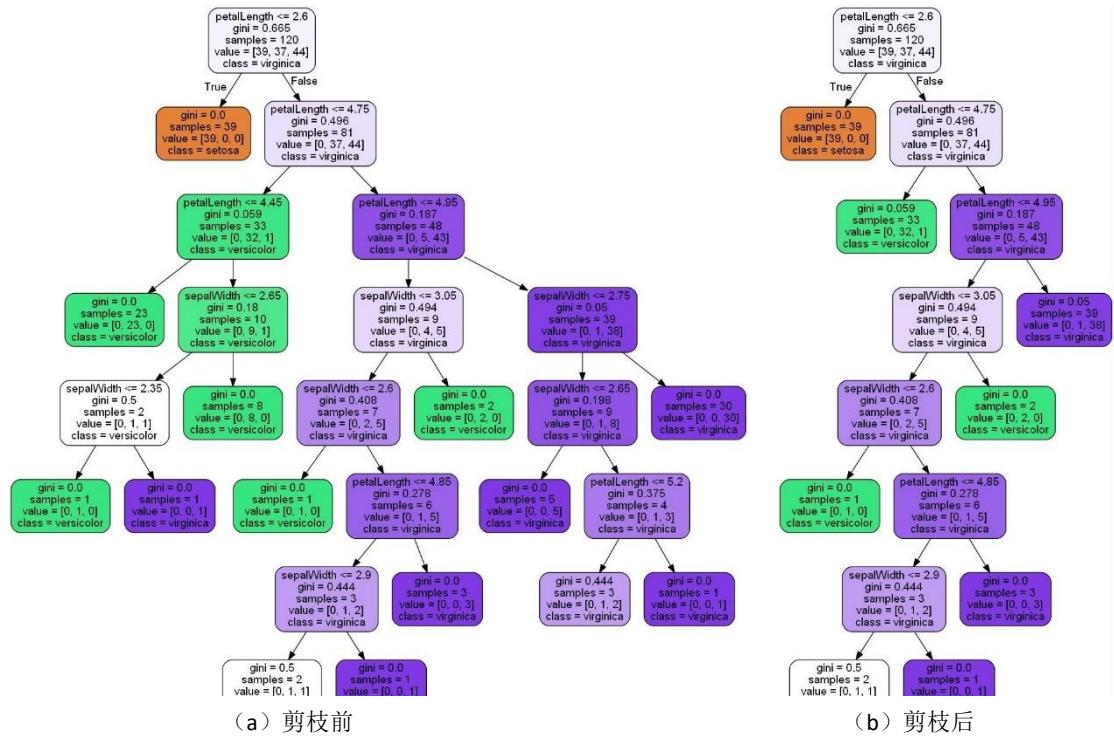


图 14-14 决策树剪枝的前后对比

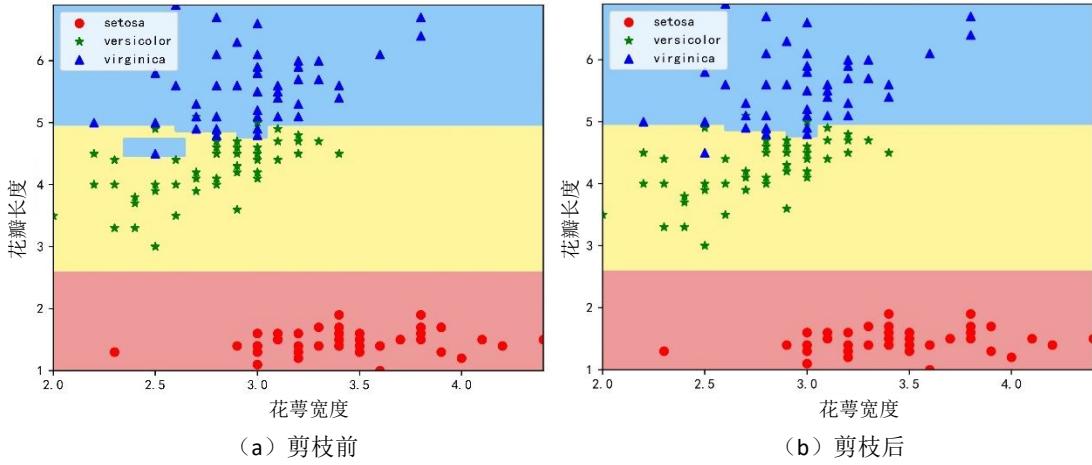


图 14-15 剪枝前后的分界线

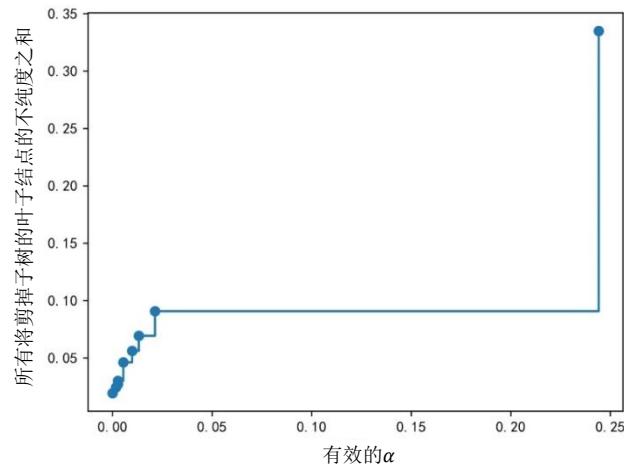


图 14-16  $\alpha$ 与 impurities 之间的关系

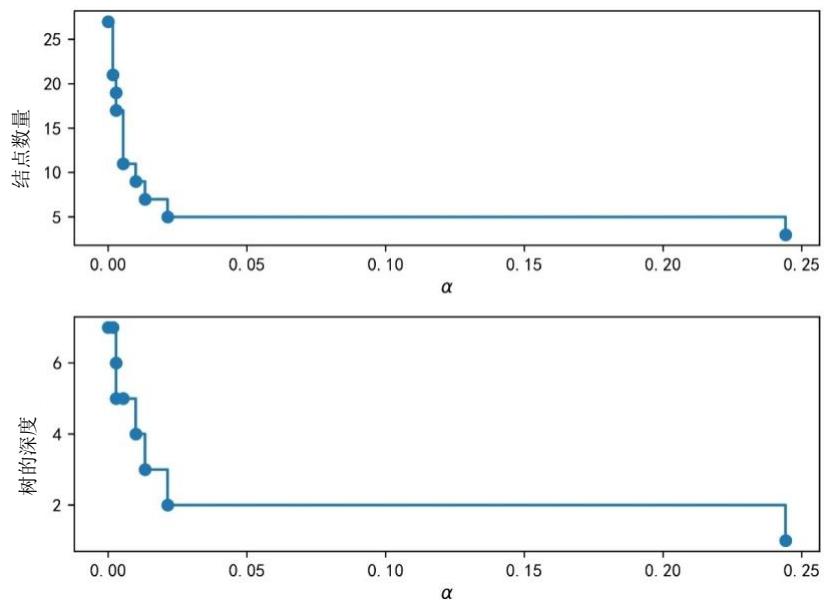


图 14-17  $\alpha$ 值与决策树模型结点数量、深度的关系

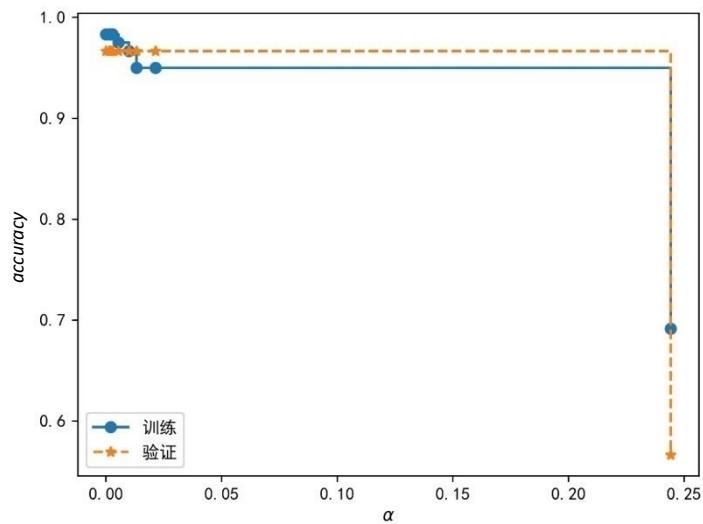


图 14-18  $\alpha$ 与 accuracy 的关系

# 第 15 章

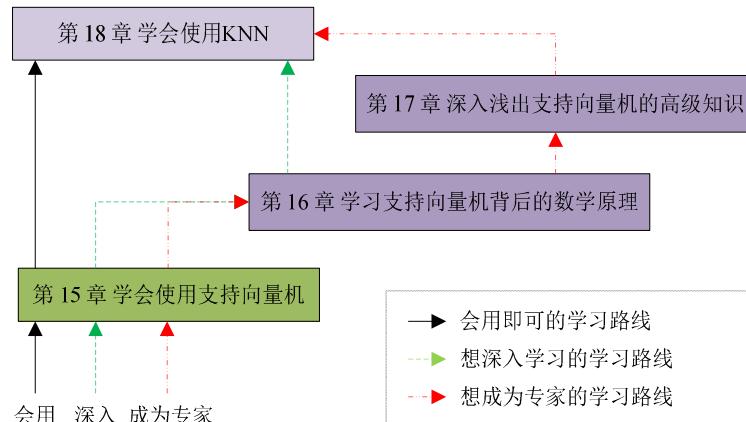


图 15-1 学习路线图

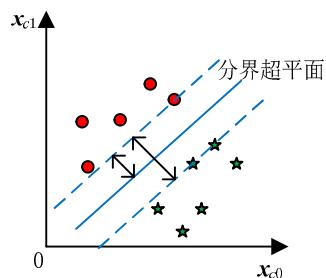


图 15-2 线性支持向量机

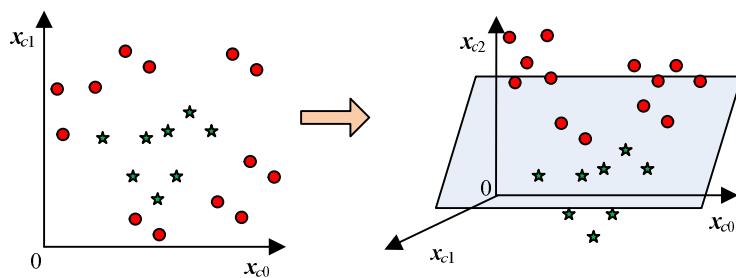


图 15-3 非线性支持向量机

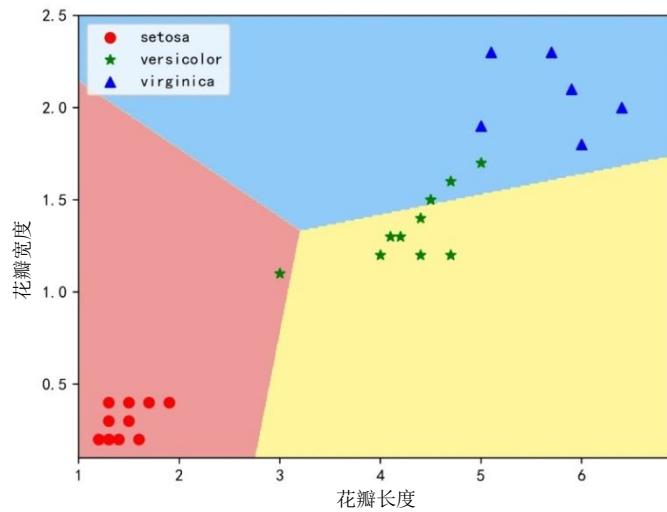


图 15-4 用线性支持向量机做鸢尾花的三分类

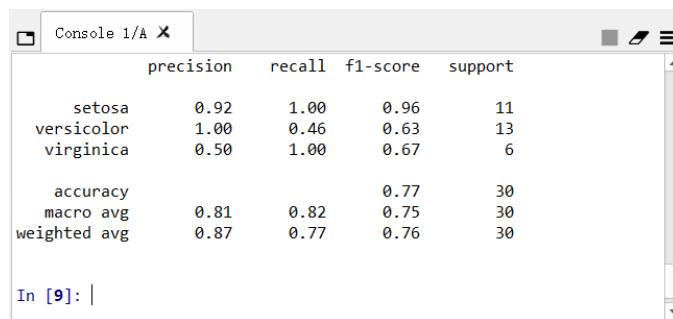


图 15-5 评价线性支持向量机做鸢尾花三分类的效果

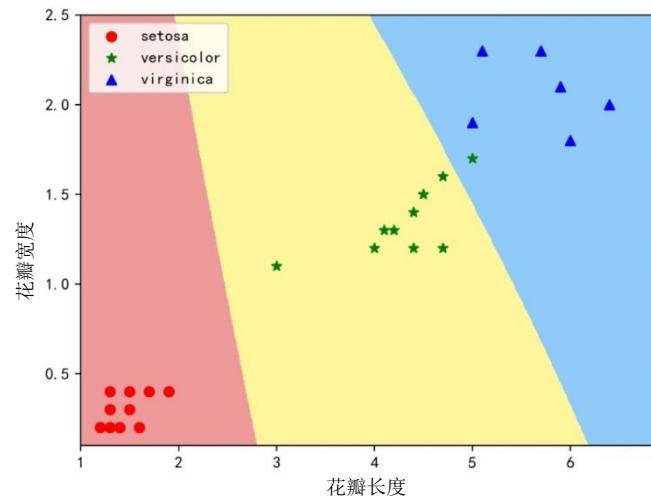


图 15-6 用非线性支持向量机做鸢尾花的三分类

Console 1/A X

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	11
versicolor	1.00	0.92	0.96	13
virginica	0.86	1.00	0.92	6
accuracy			0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

In [10]:

图 15-7 评价非线性支持向量机做鸢尾花三分类的效果

# 第 16 章

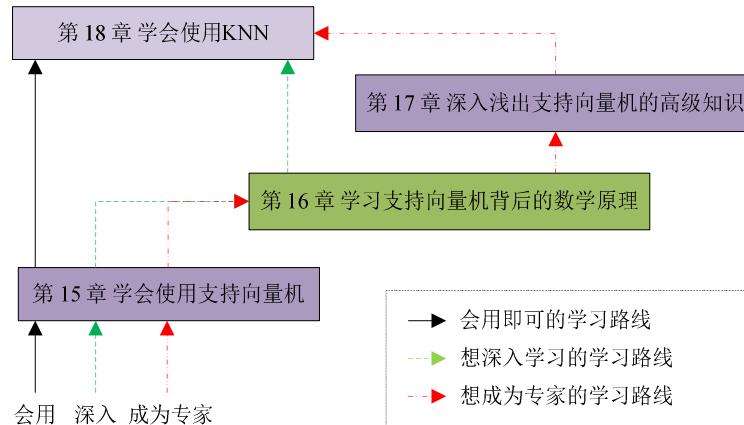


图 16-1 学习路线图

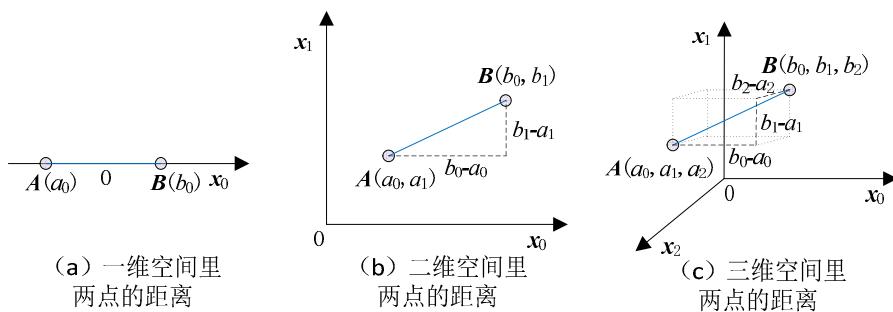


图 16-2 空间中两点之间的距离

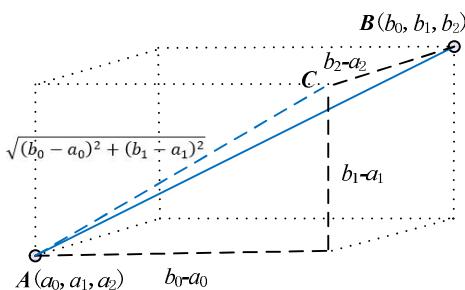


图 16-3 三维空间里两点的距离

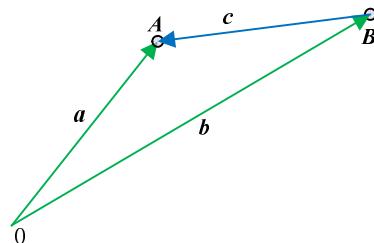


图 16-4 用向量表示两点之间的距离

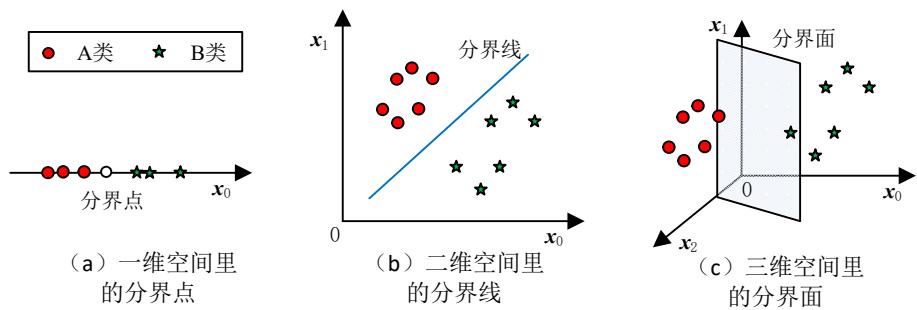


图 16-5 分界的超平面

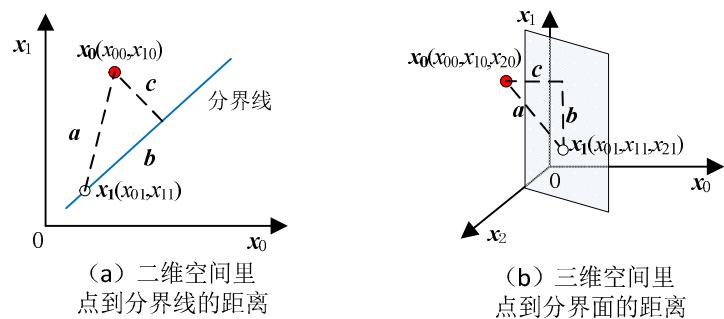


图 16-6 点到超平面的距离

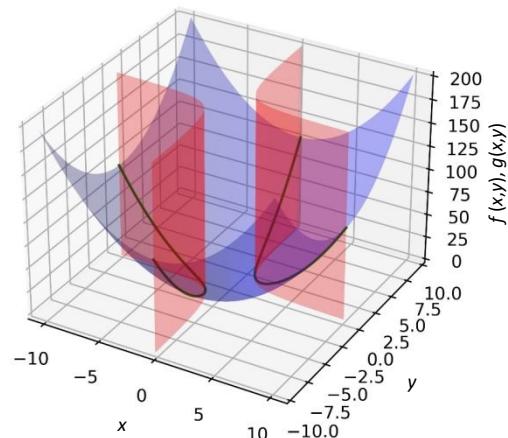


图 16-7  $f(x, y) = x^2 + y^2$  和  $xy = 2$  的图形、相交线

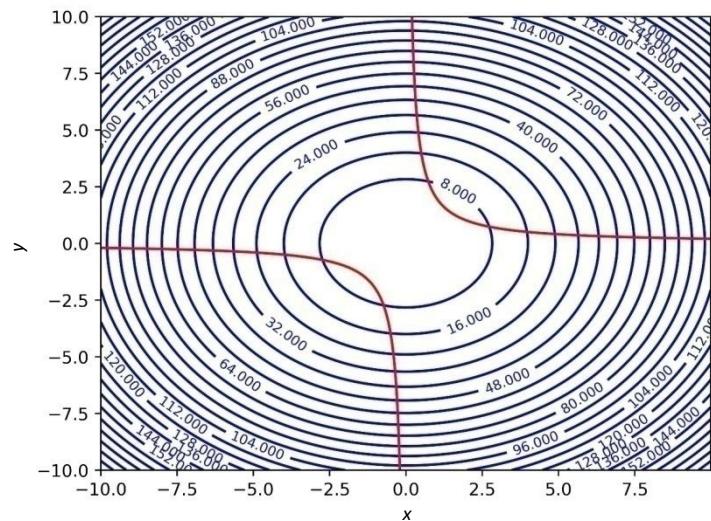


图 16-8 从二维空间看 $f(x, y) = x^2 + y^2$ 和 $xy = 2$ 的图形

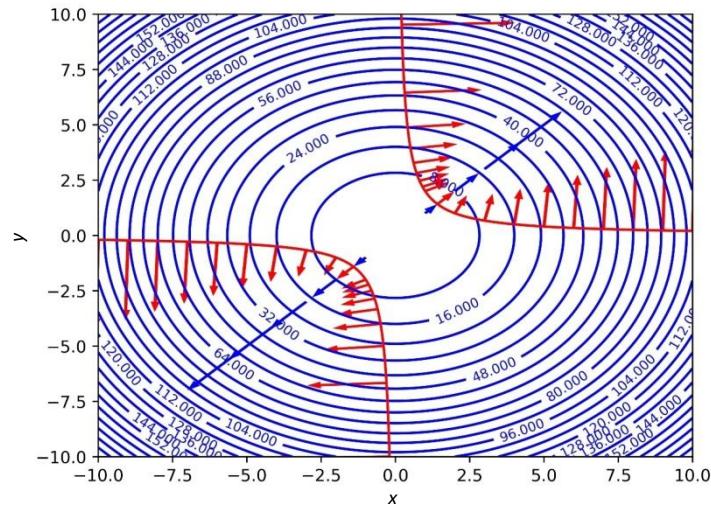


图 16-9 梯度方向的形象展示

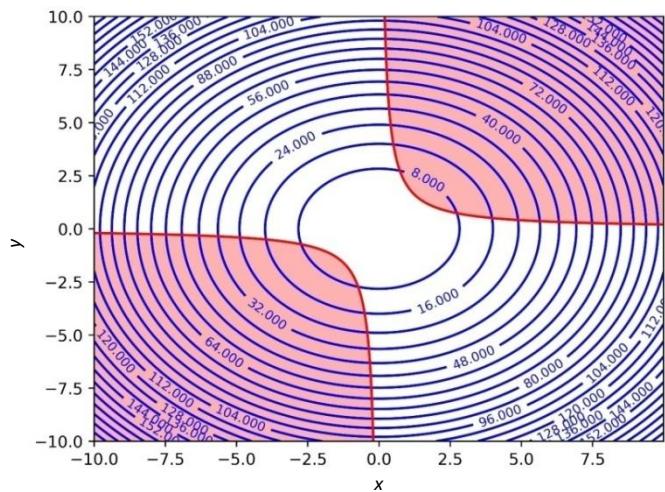


图 16-10  $f(x, y) = x^2 + y^2$  和  $xy \leq 2$  的图形

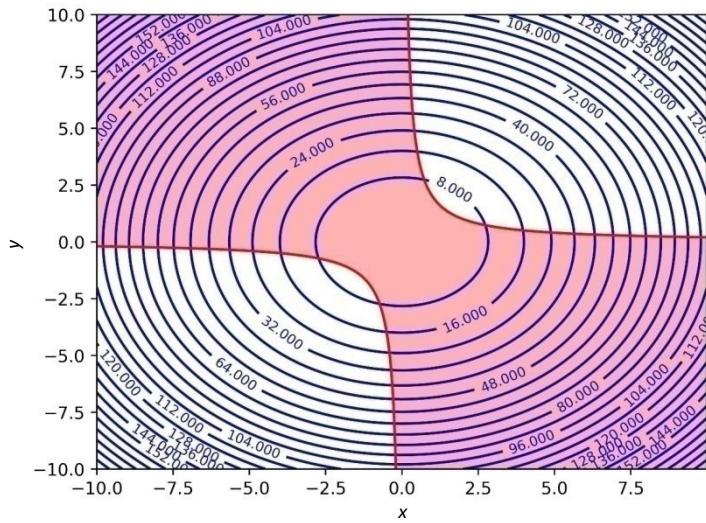


图 16-11  $f(x, y) = x^2 + y^2$  和  $xy \geq 2$  的图形

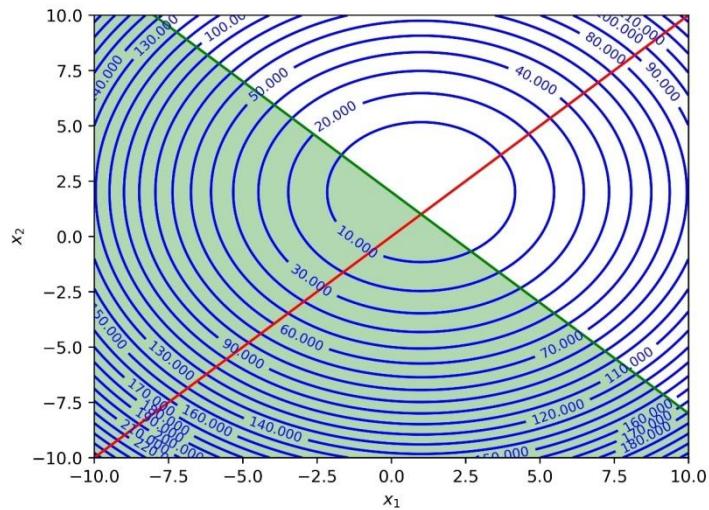


图 16-12 用拉格朗日乘数法和 KKT 应对更复杂的情况

训练数据集 $X$	特征数据项 $x_{c0}$	...	特征数据项 $x_{c(n-1)}$	目标数据项 $Y$
样本 $x_{r0}$	$x_{r0c0}$	...	$x_{r0c(n-1)}$	$y_0$
...	...	...	...	...
样本 $x_{r(m-1)}$	$x_{r(m-1)c0}$	...	$x_{r(m-1)c(n-1)}$	$y_{m-1}$

目标数据项  $Y$  的取值为 {1, -1}

图 16-13 训练数据集

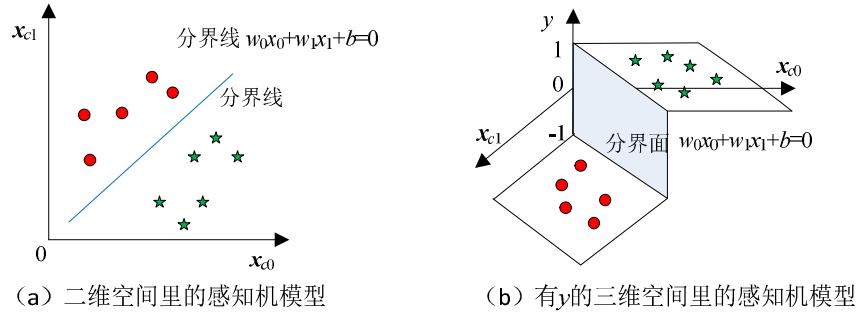


图 16-14 感知机模型的图示

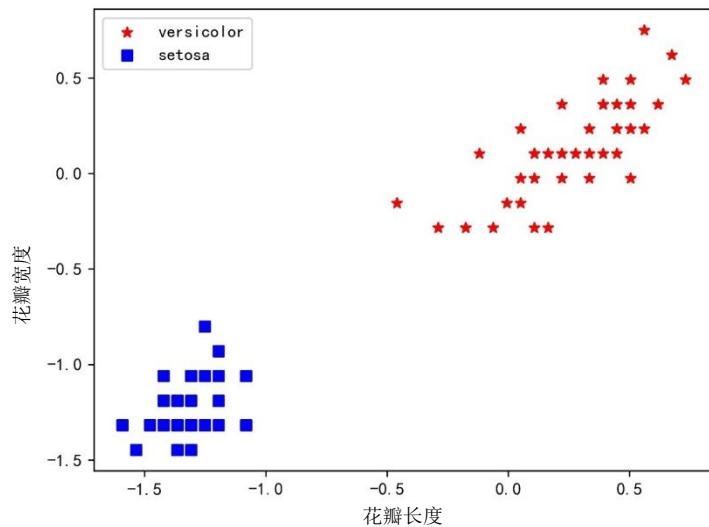


图 16-15 鸢尾花数据集中的 setosa 类型和 versicolor 类型

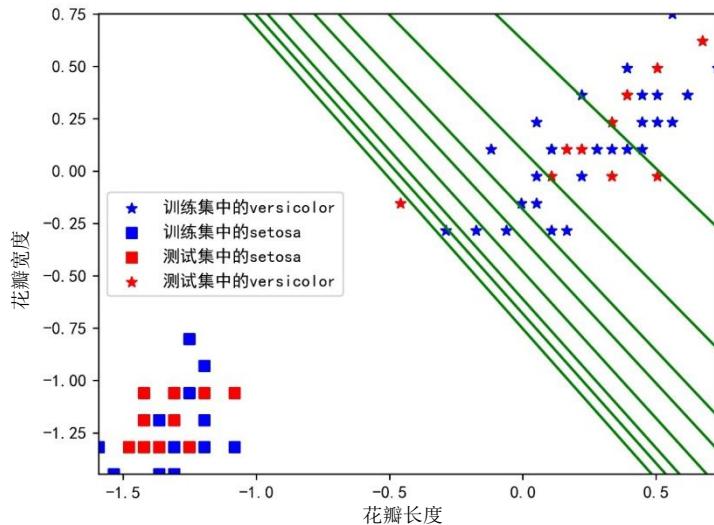


图 16-16 找到分类超平面的过程

```

Console 1/A ×
=====第 6 次迭代=====
误差函数值: 5.527529095017535
76 个训练样本, 4 个错误
=====第 7 次迭代=====
误差函数值: 1.7006642012811337
76 个训练样本, 2 个错误
=====第 8 次迭代=====
误差函数值: 0.30408057895723317
76 个训练样本, 2 个错误
=====第 9 次迭代=====
误差函数值: -0.0
76 个训练样本, 0 个错误
退出迭代。w= [14.29258283 9.96282861] , b= 7.5
In [5]:

```

图 16-17 用感知机模型做鸢尾花分类的控制台输出

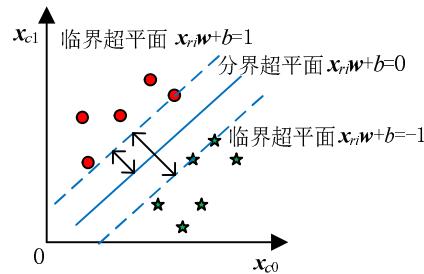


图 16-18 线性可分情况下的理想分界超平面

# 第 17 章

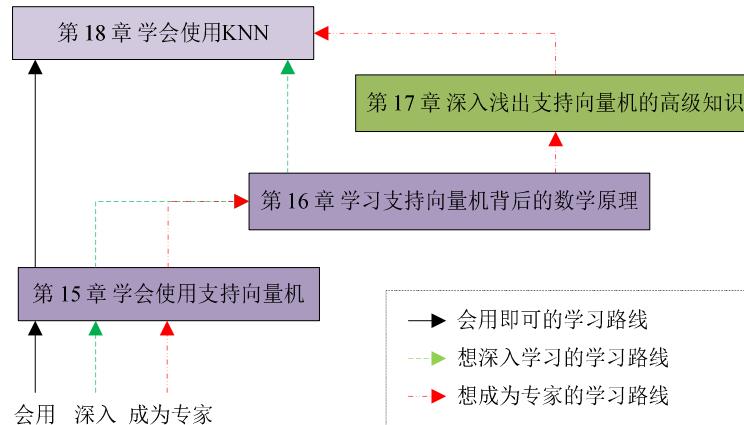


图 17-1 学习路线图

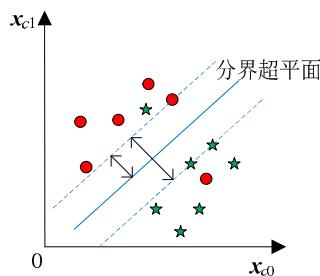


图 17-2 软间隔支持向量机

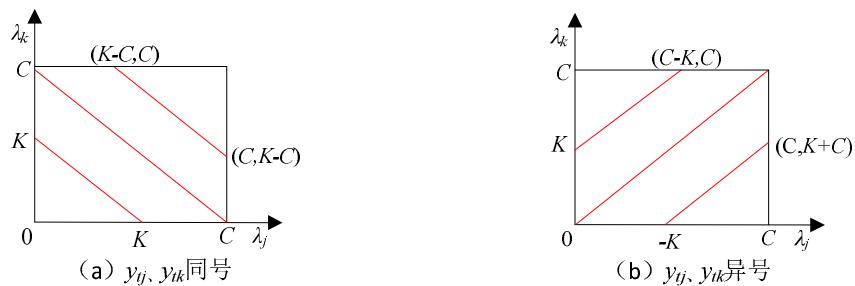


图 17-3  $y_{tj}, y_{tk}$  同号和异号时的情况

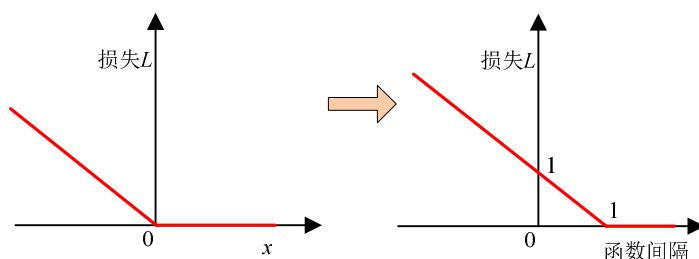
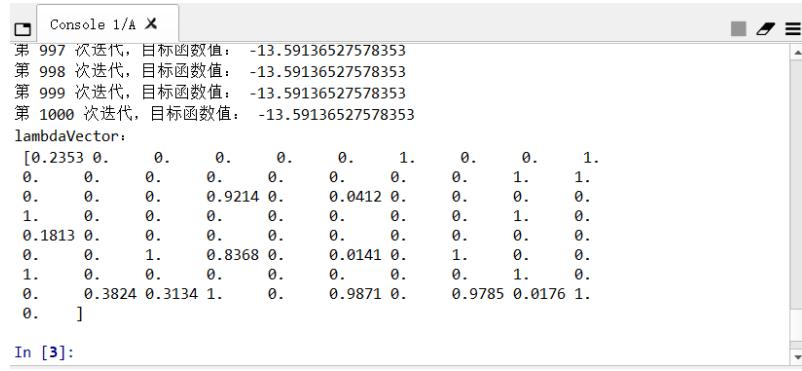


图 17-4 合页损失函数的图形



```

Console 1/A ×
第 997 次迭代, 目标函数值: -13.59136527578353
第 998 次迭代, 目标函数值: -13.59136527578353
第 999 次迭代, 目标函数值: -13.59136527578353
第 1000 次迭代, 目标函数值: -13.59136527578353
lambdaVector:
[0.2353 0. 0. 0. 0. 1. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 1. 1.
 0. 0. 0. 0.9214 0. 0.0412 0. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 1. 0.
 0.1813 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0.8368 0. 0.0141 0. 1. 0.
 1. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0.3824 0.3134 1. 0. 0.9871 0. 0.9785 0.0176 1.
 0. ]

```

图 17-5 用 SMO 算法求解 $\lambda$

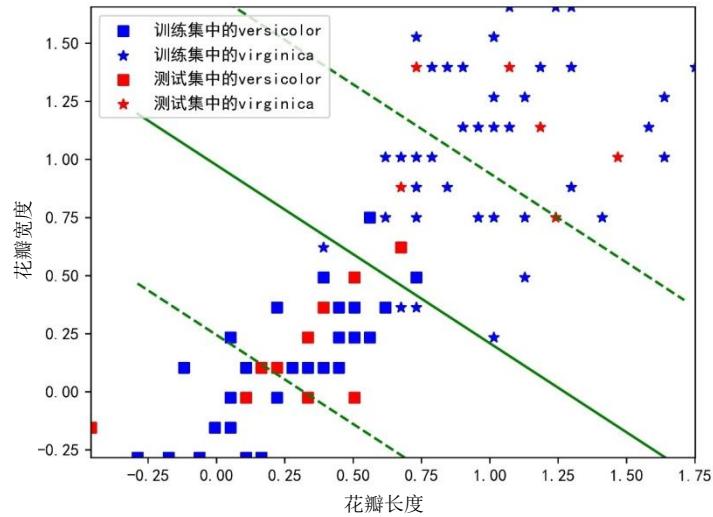


图 17-6 用软间隔支持向量机做二分类

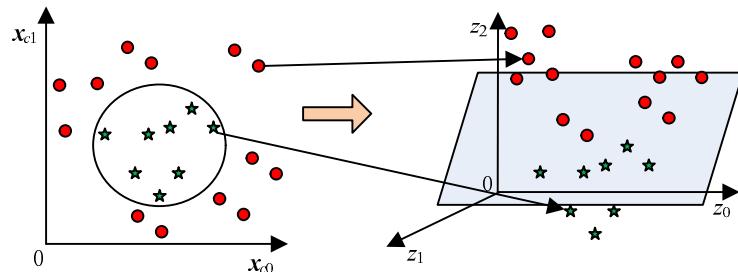
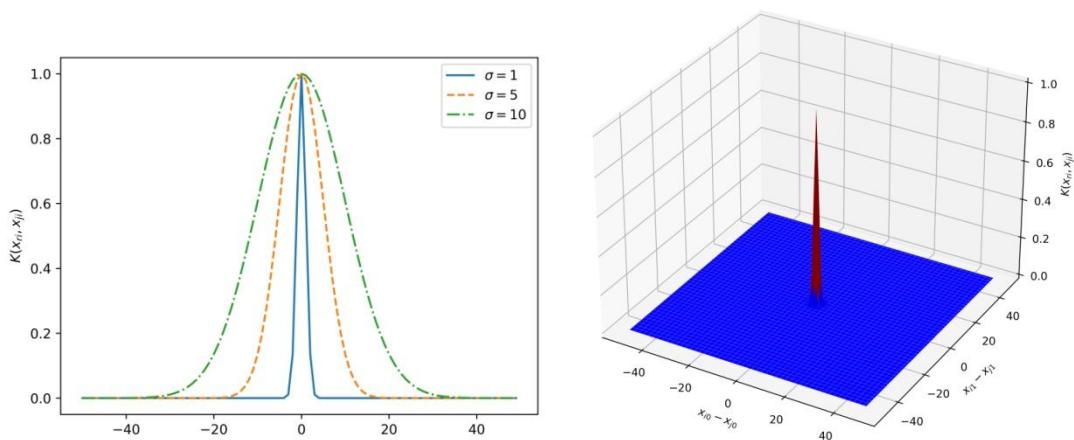
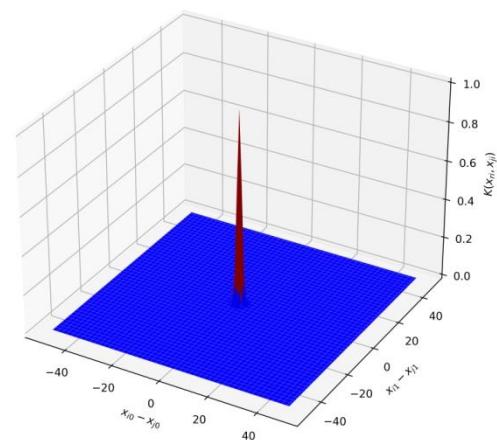


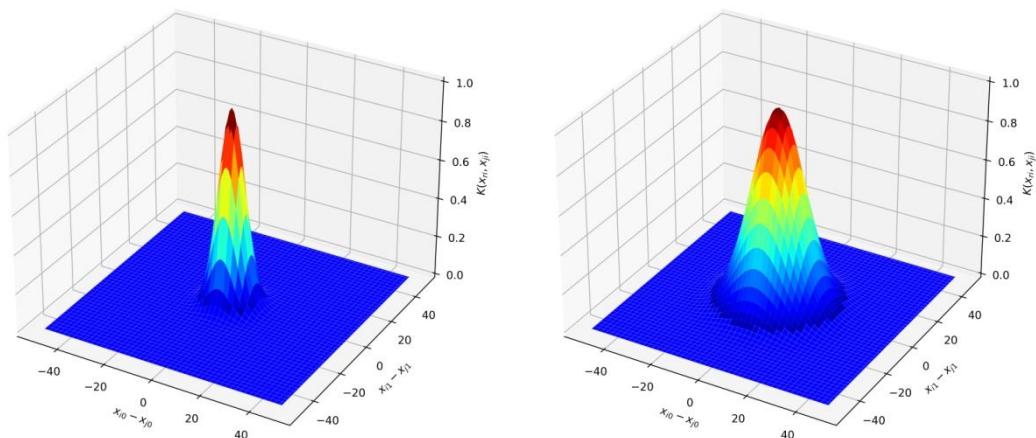
图 17-7 空间的映射



(a) 向量只有一维



(b) 向量有二维且 $\sigma = 1$



(c) 向量有二维且 $\sigma = 5$

(d) 向量有二维且 $\sigma = 10$

图 17-8 高斯核函数的图形

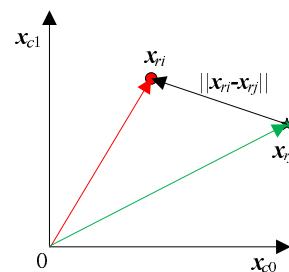


图 17-9 两个向量之间的距离

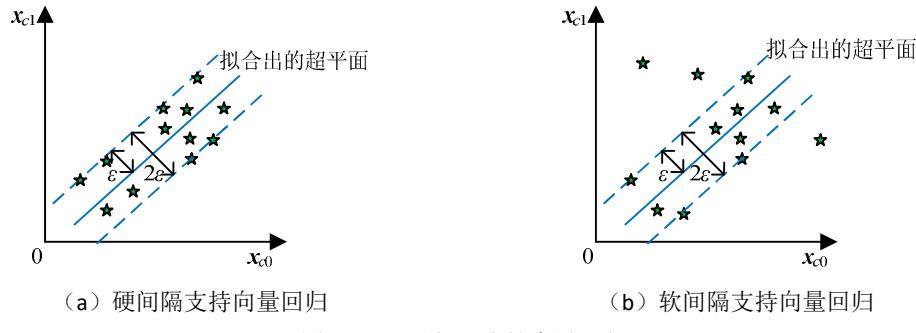


图 17-10 硬间隔支持向量回归

```

Console 1/A x
最优的模型是: SVC(C=17, gamma=10)
最好的成绩(准确度)是: 0.9666666666666666
最优的参数是: {'C': 17, 'gamma': 10}
      precision    recall   f1-score   support
setosa      1.00     1.00     1.00      11
versicolor  1.00     1.00     1.00      13
virginica   1.00     1.00     1.00       6
accuracy          1.00     1.00     1.00      30
macro avg     1.00     1.00     1.00      30
weighted avg  1.00     1.00     1.00      30

```

图 17-11 找到的最理想的非线性支持向量机模型

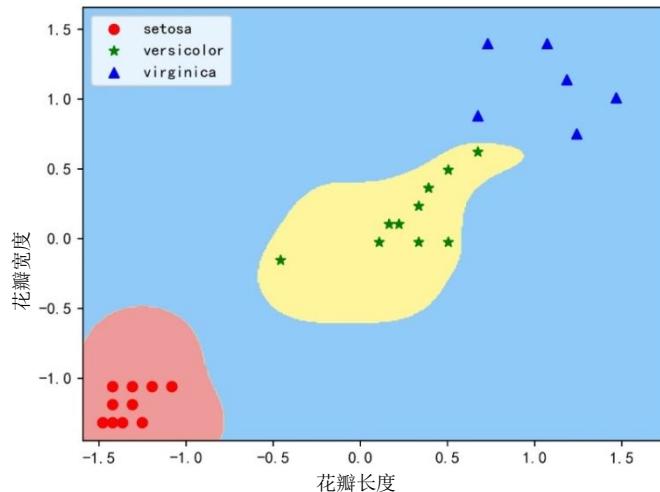


图 17-12 最理想的非线性支持向量机模型的分类效果

```

Console 1/A x
最优的模型是: SVR(epsilon=0.08, gamma=0.4)
最好的成绩(R2)是: 0.49934662516264616
最优的参数是: {'epsilon': 0.08, 'gamma': 0.4}
找到的最理想的非线性支持向量回归模型对训练数据的R^2: 0.5304044254865699
找到的最理想的非线性支持向量回归模型对测试数据的R^2: 0.642795320639945
找到的最理想的非线性支持向量回归模型对训练数据的MSE: 39.823970611756145
找到的最理想的非线性支持向量回归模型对测试数据的MSE: 29.560402856974147
In [38]: |

```

图 17-13 找到的最理想的非线性支持向量回归模型

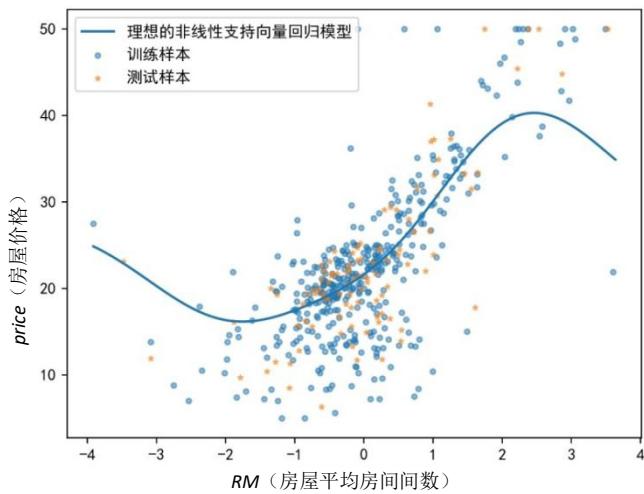


图 17-14 最理想的非线性支持向量回归模型的拟合效果

# 第 18 章

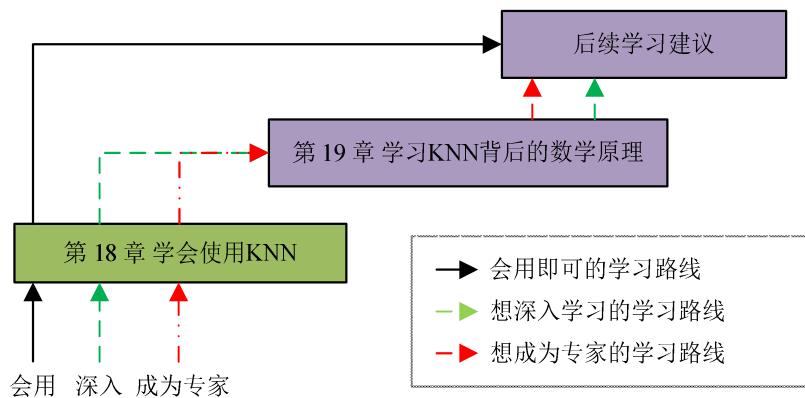


图 18-1 学习路线图

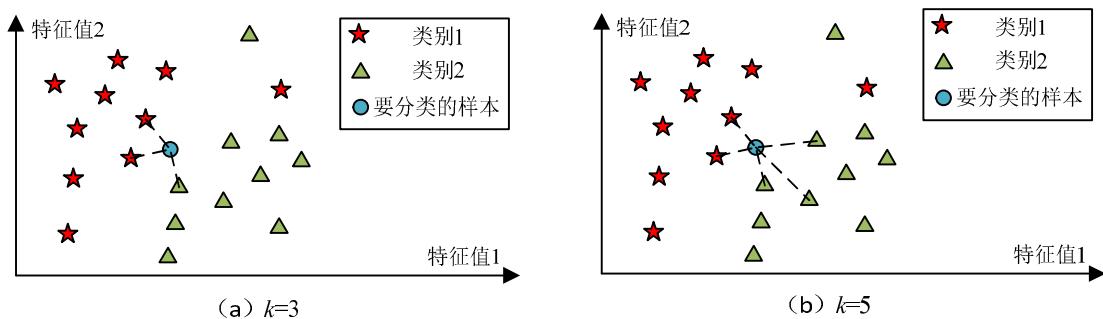


图 18-2 KNN 的基本原理

A screenshot of a Jupyter Notebook cell titled 'Console 1/A'. The cell displays a table of evaluation metrics for a KNN model on the Iris dataset.

	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	11
versicolor	1.00	0.92	0.96	13
virginica	0.86	1.00	0.92	6
accuracy			0.97	30
macro avg	0.95	0.97	0.96	30
weighted avg	0.97	0.97	0.97	30

图 18-3 评价 KNN 模型做鸢尾花分类的效果

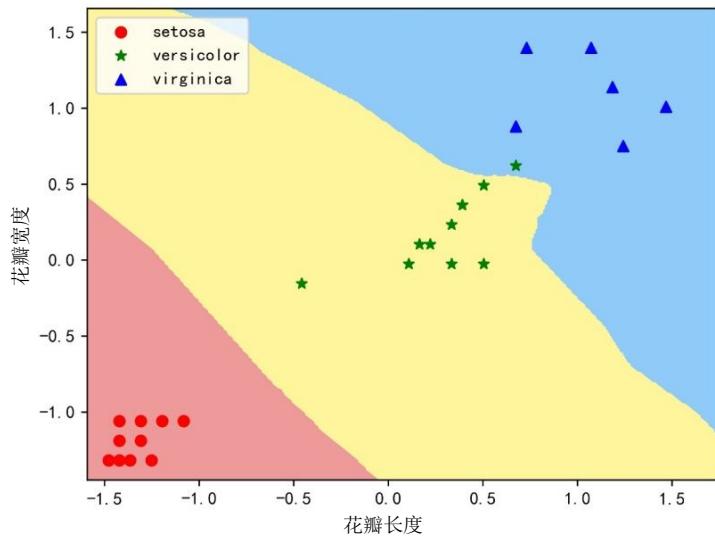


图 18-4 用 KNN 模型做鸢尾花分类的效果图示

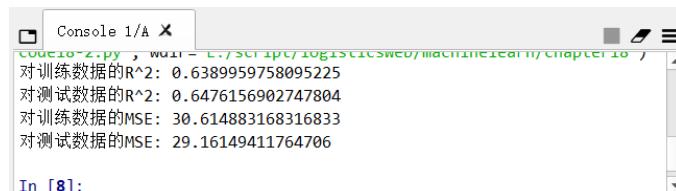


图 18-5 评价 KNN 模型做房屋价格回归的效果

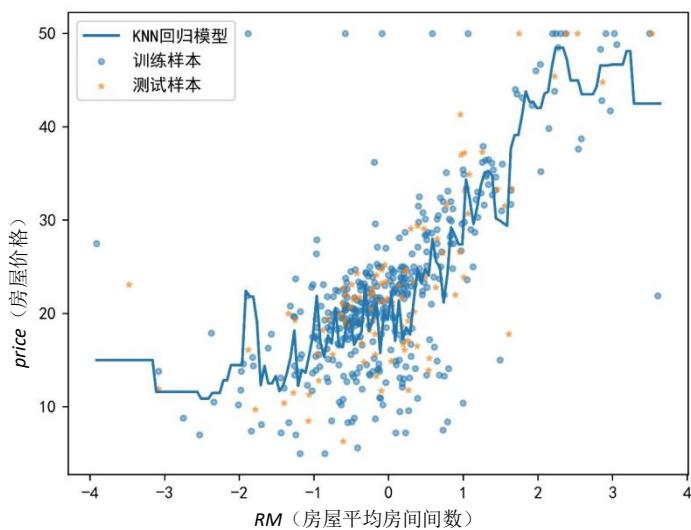


图 18-6 用 KNN 模型做房屋价格回归的效果图示

# 第 19 章

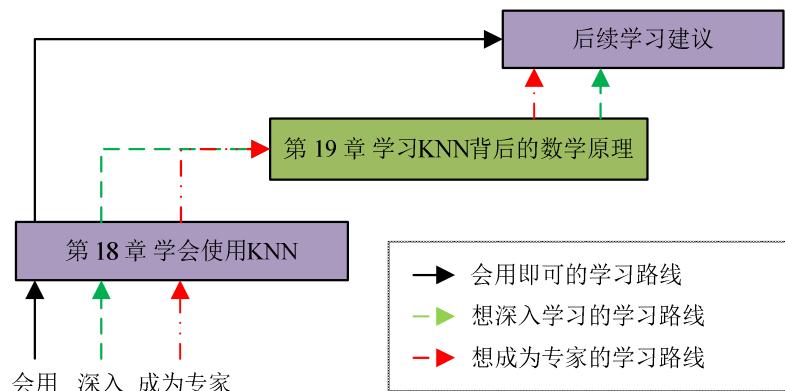


图 19-1 学习路线图

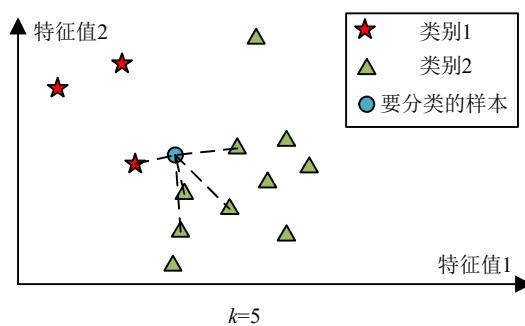


图 19-2 样本类型中的样本数量不均衡

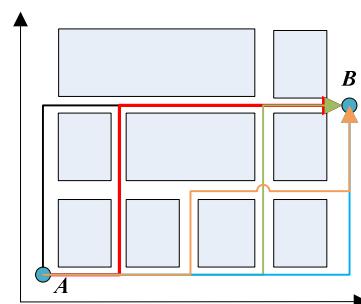


图 19-3 两点之间的曼哈顿距离

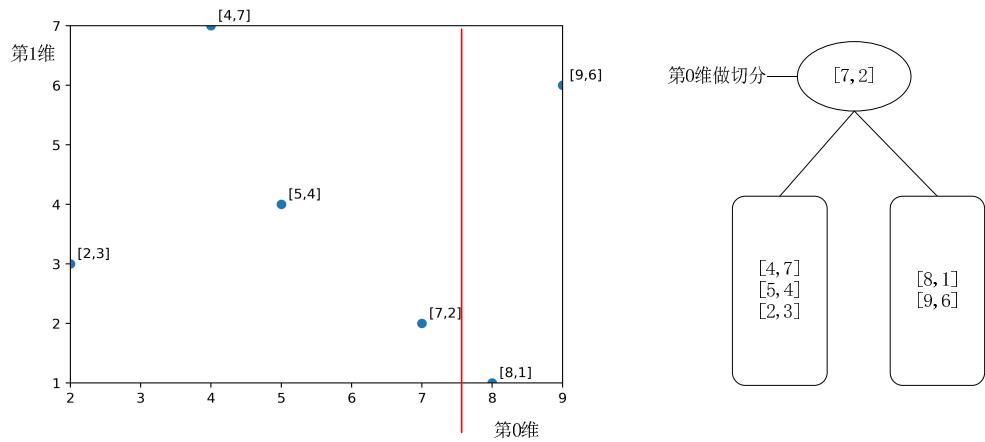


图 19-4 构建 KD 树的根

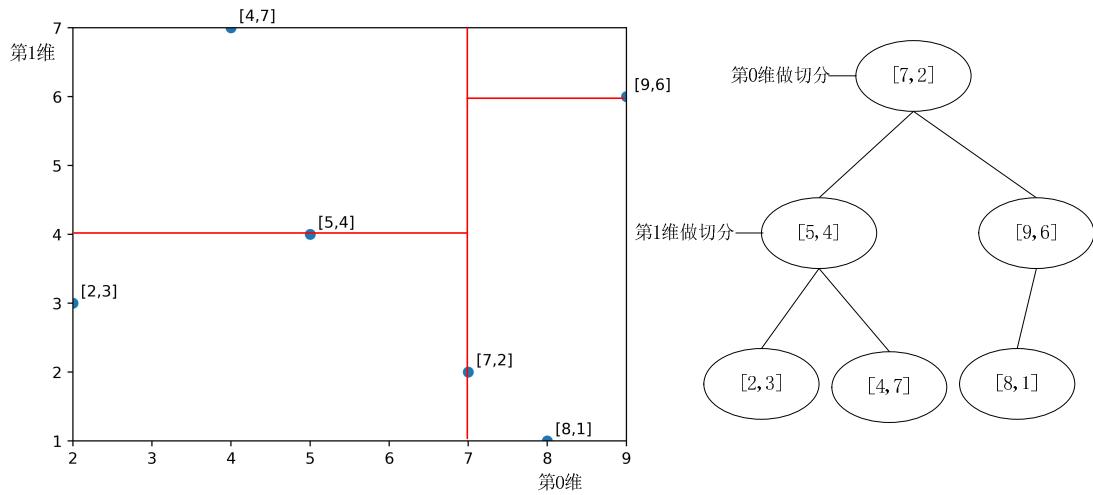


图 19-5 继续以第 1 维做切分

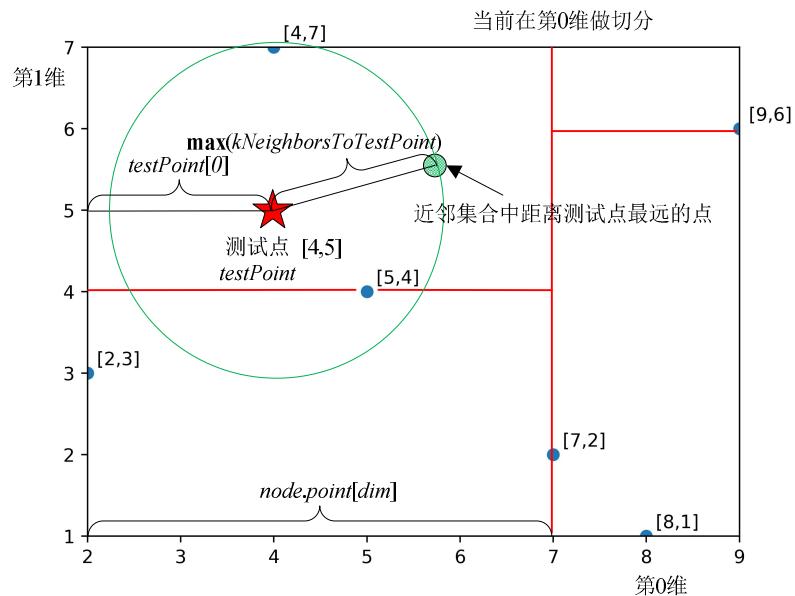


图 19-6 近邻点可能出现在比当前结点值更小的区间中

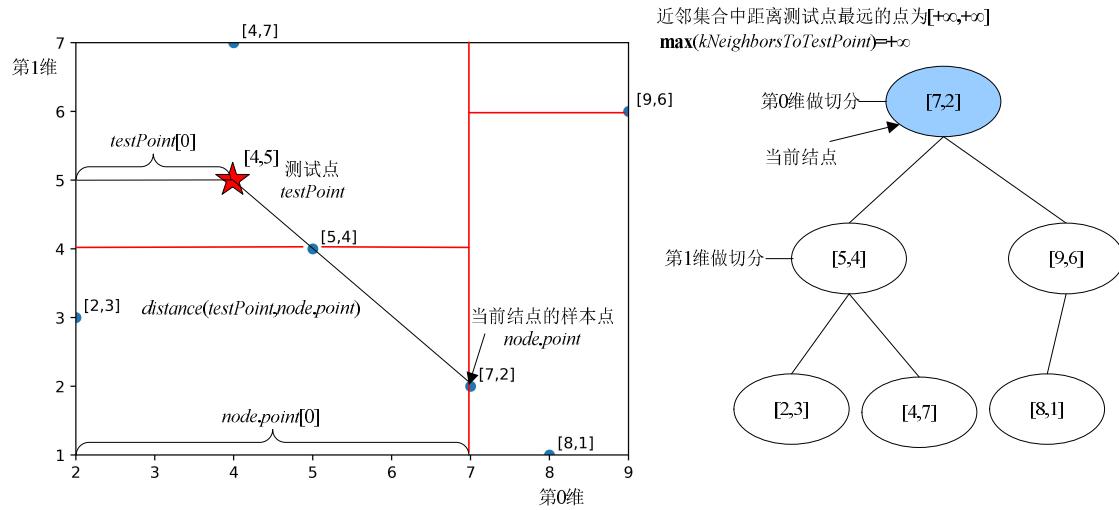


图 19-7 第 1 次调用算法  $findKNeighbors(3, [4,5], node([7,2]), 0)$

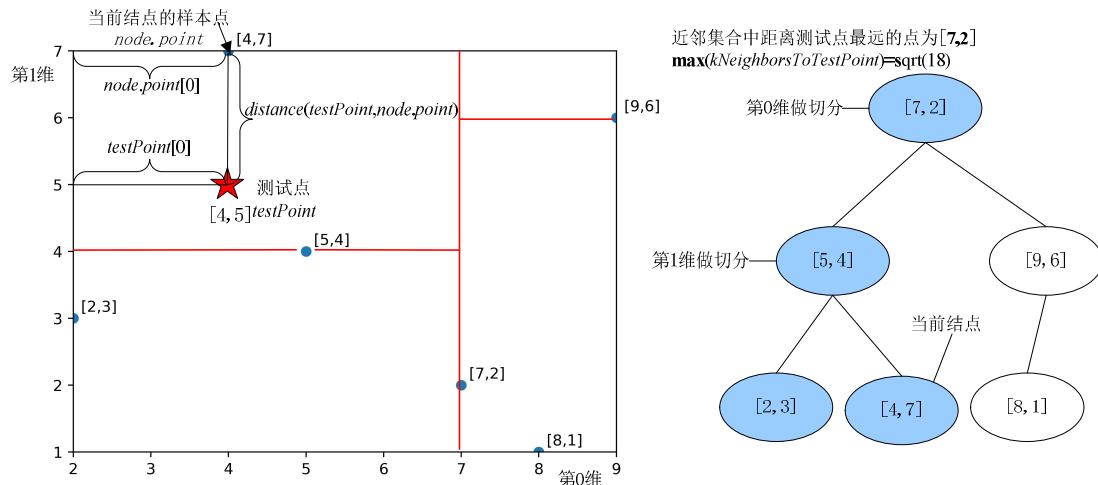


图 19-8 第 4 次调用算法  $findKNeighbors(3, [4,5], node([4,7]), 0)$

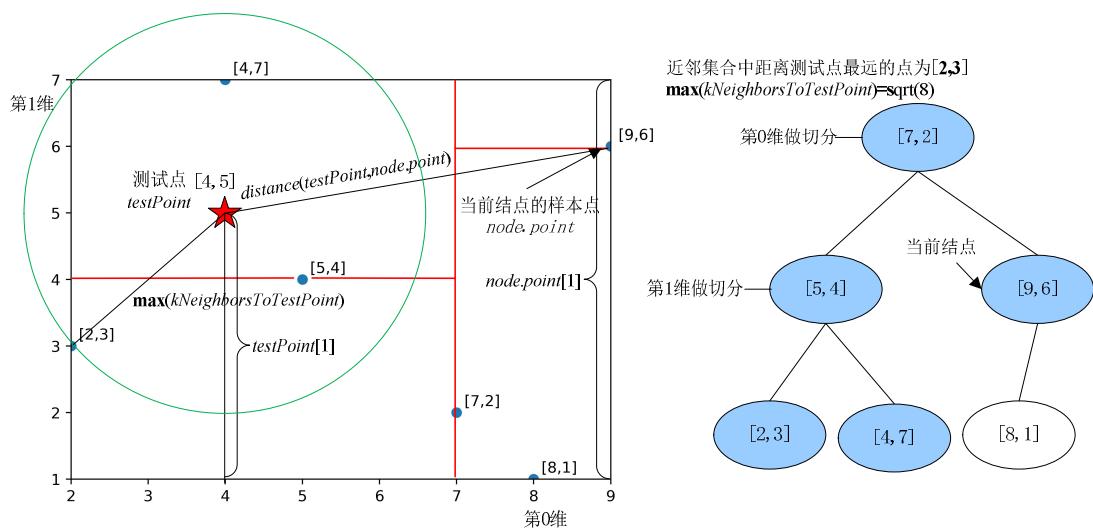


图 19-9 第 5 次调用算法  $findKNeighbors(3, [4,5], node([9,6]), 1)$

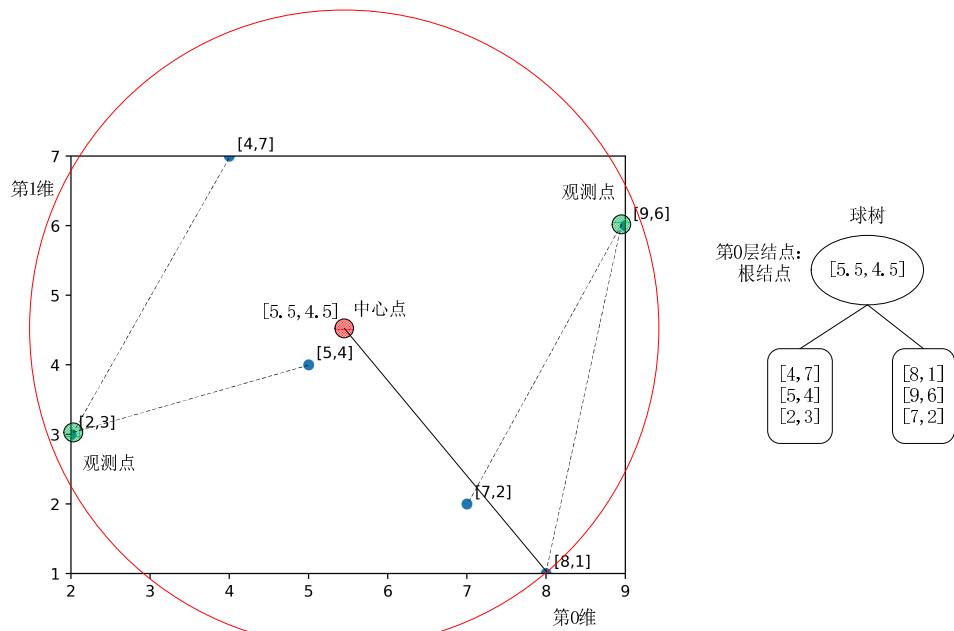


图 19-10 Ball 树的根结点及其超球体

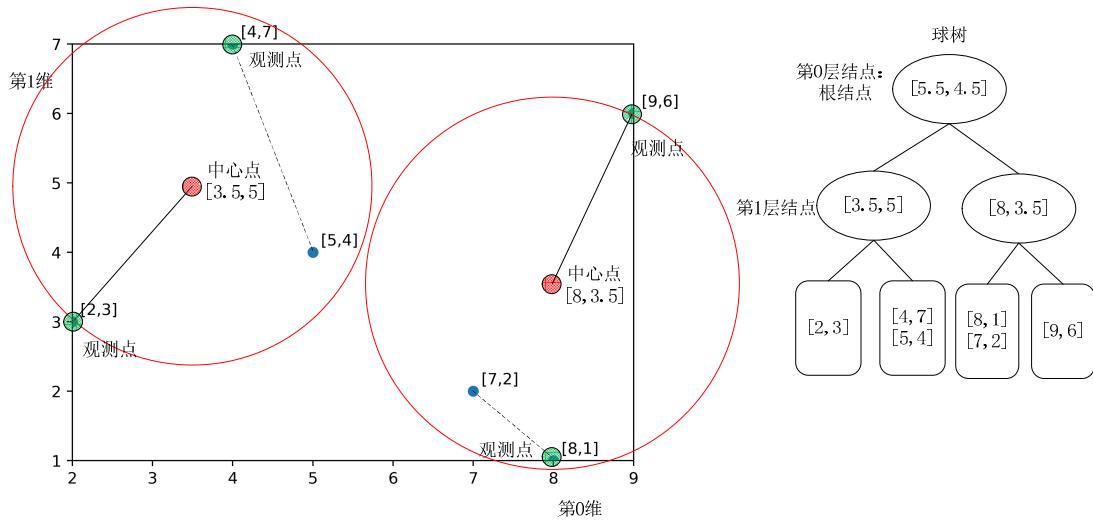


图 19-11 Ball 树的第 1 层结点及其超球体

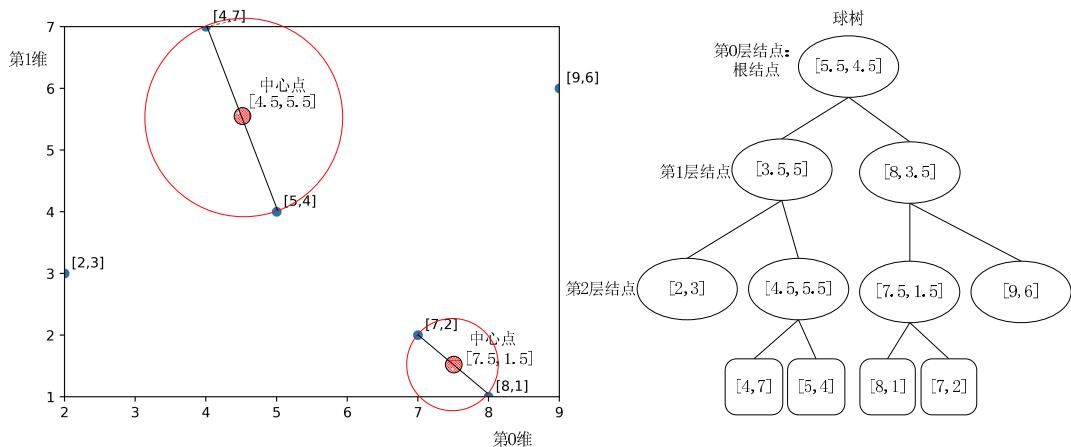


图 19-12 Ball 树的第 2 层结点及其超球体

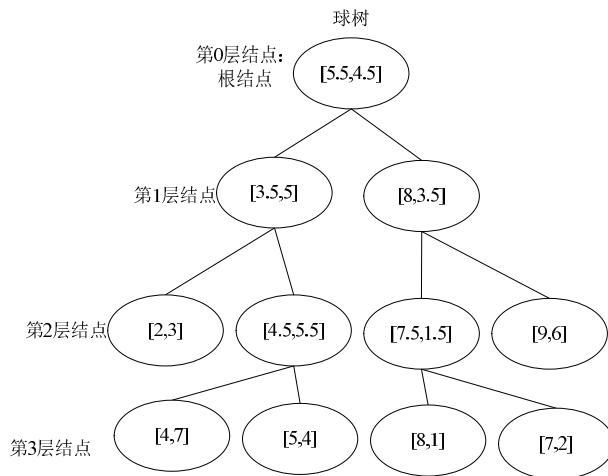
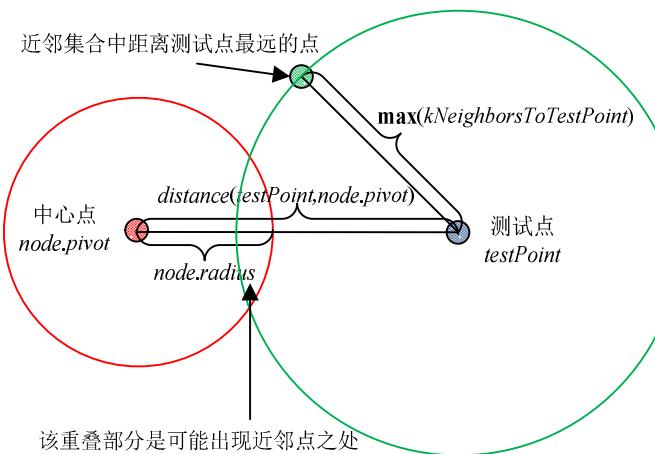
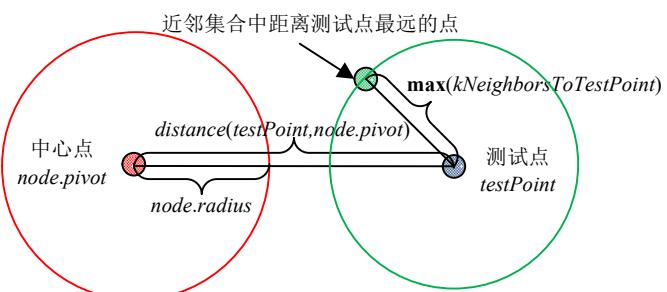


图 19-13 构造出的 Ball 树



(a)  $distance(testPoint, node.pivot) - node.radius < \max(kNeighborsToTestPoint)$



(b)  $distance(testPoint, node.pivot) - node.radius \geq \max(kNeighborsToTestPoint)$

图 19-14 判断超球体中是否有近邻点的原理

```
Console 1/A X
最优的模型是: KNeighborsClassifier(n_neighbors=1)
最好的成绩(准确度)是: 0.9666666666666666
最优的参数是: {'n_neighbors': 1, 'weights': 'uniform'}
      precision    recall   f1-score   support
setosa      1.00     1.00     1.00      11
versicolor  1.00     1.00     1.00      13
virginica   1.00     1.00     1.00       6

accuracy          1.00      1.00     1.00      30
macro avg       1.00     1.00     1.00      30
weighted avg    1.00     1.00     1.00      30
```

图 19-15 用 GridSearchCV 调节 KNN 分类模型的参数

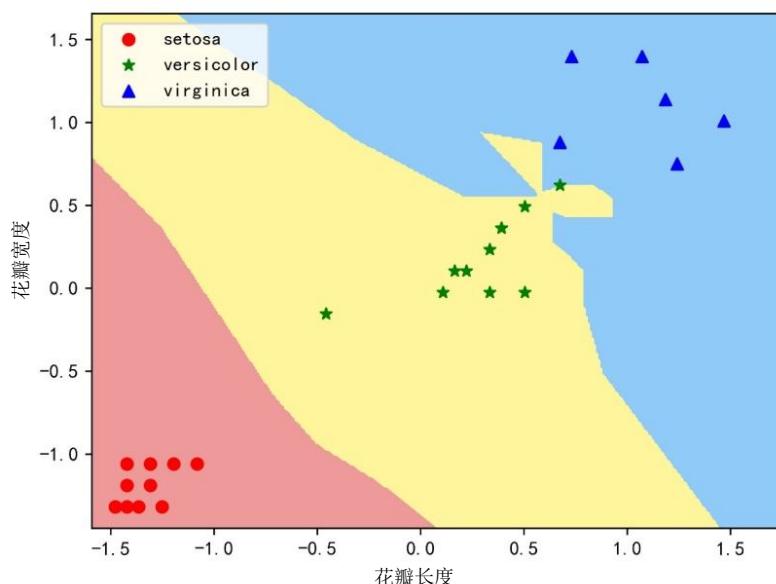


图 19-16 找到的最优 KNN 模型的分类效果

```
Console 1/A X
最优的模型是: KNeighborsRegressor(n_neighbors=34)
最好的成绩(R^2)是: 0.5103225636032028
最优的参数是: {'n_neighbors': 34, 'weights': 'uniform'}
对训练数据的R^2: 0.5525148042728467
对测试数据的R^2: 0.6691521361896275
对训练数据的MSE: 37.948903803658915
对测试数据的MSE: 27.379249779496572
```

图 19-17 用 GridSearchCV 调节 KNN 回归模型的参数

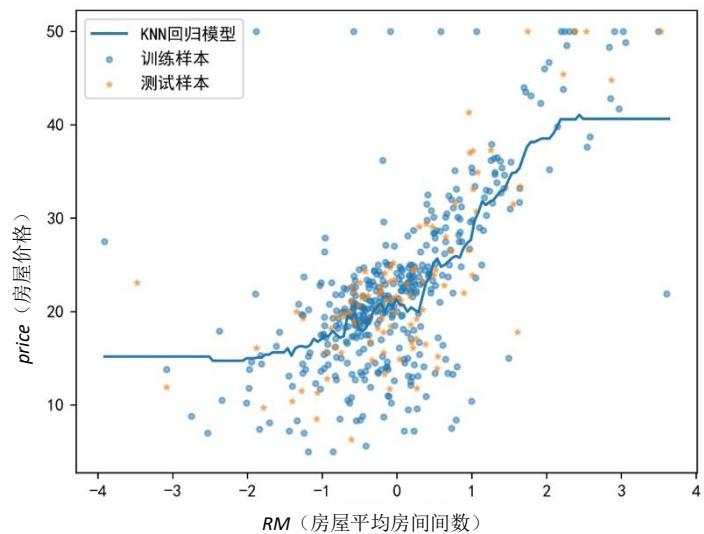
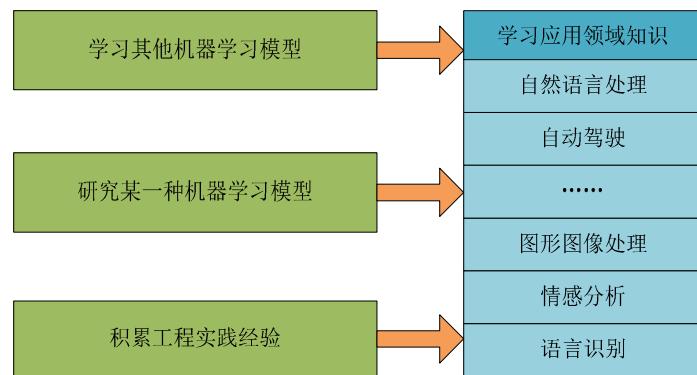


图 19-18 找到的最优 KNN 模型的回归效果



后续学习建议图