

# 32

## Classification Methods in Scikit-Learn

# Scikit-Learn 分类

$k$  最近邻、朴素贝叶斯、支持向量机、核技巧



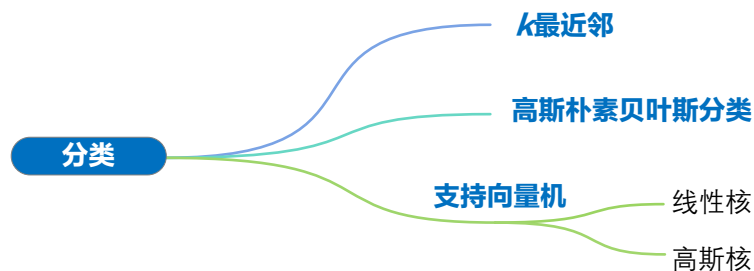
错误，是进步的代价。

*Error is the price we pay for progress.*

—— 阿尔弗雷德·怀特海 (Alfred Whitehead) | 英国数学家、哲学家 | 1861 ~ 1947



- ▶ `matplotlib.colors.ListedColormap()` 创建离散颜色映射的函数。函数接受一个颜色列表作为输入，并生成一个离散的颜色映射对象，用于在可视化中区分不同的类别或数据值
- ▶ `sklearn.datasets.load_iris()` 加载鸢尾花数据
- ▶ `sklearn.naive_bayes.GaussianNB()` 实现高斯朴素贝叶斯分类器算法
- ▶ `sklearn.neighbors.KNeighborsClassifier()` 实现  $k$  最近邻分类器算法
- ▶ `sklearn.svm.SVC()` 实现支持向量机分类器算法



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

## 32.1 什么是分类？

本书前文介绍过，**分类** (classification) 是**有监督学习** (supervised learning) 中的一类问题。分类是指根据给定的数据集，通过对样本数据的学习，建立分类模型来对新的数据进行分类的过程。

如图 1 所示，大家已经清楚鸢尾花数据集分三类 (setosa ●、versicolor ●、virginica ●)。

以**花萼长度** (sepal length)、**花萼宽度** (sepal width) 作为特征，大家如果采到一朵鸢尾花，测量后发现这朵花的花萼长度为 6.5 厘米，花瓣长度为 4.0 厘米，即图 1 中 ×，又叫**查询点** (query point)。

根据已有数据，猜测这朵鸢尾花属于 setosa ●、versicolor ●、virginica ● 三类的哪一类可能性更大，这就是分类问题。

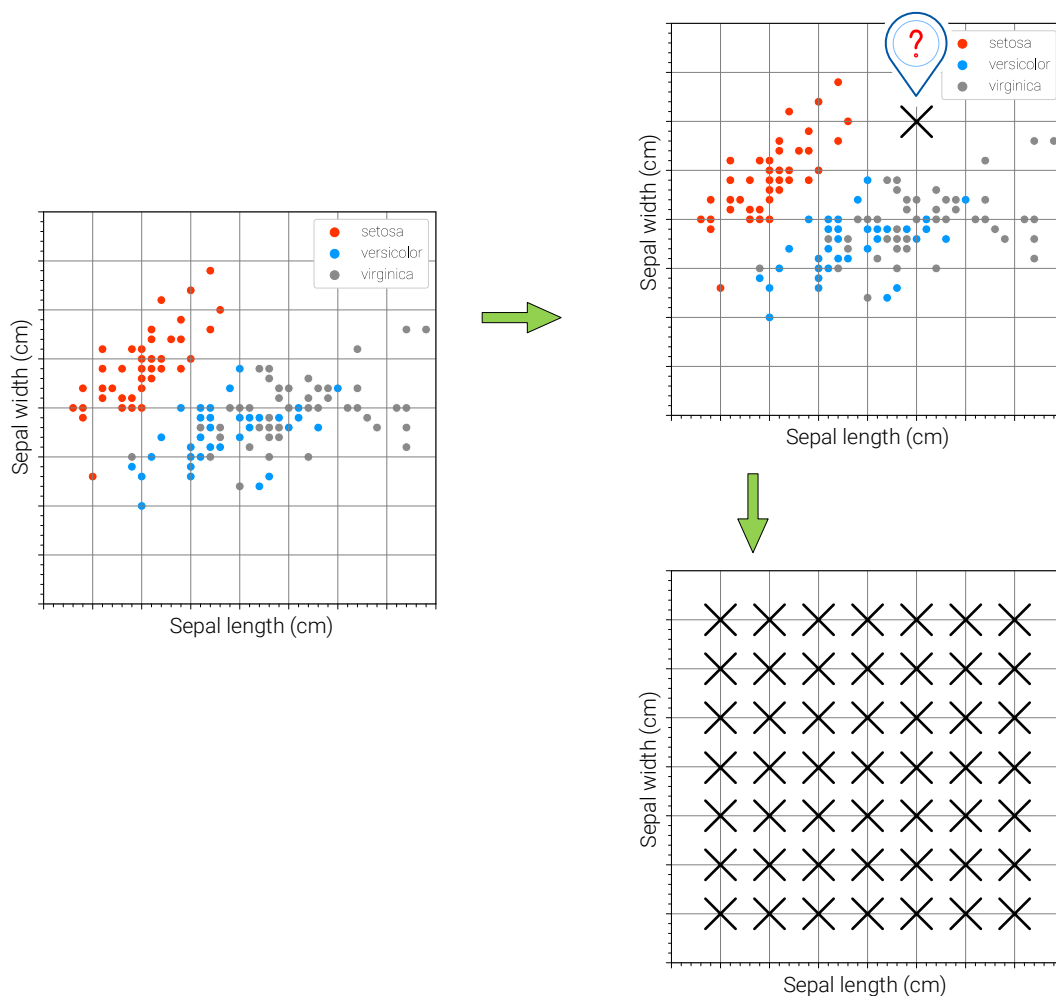


图 1. 用鸢尾花数据介绍分类算法

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

**决策边界** (decision boundary) 是分类模型在特征空间中划分不同类别的分界线或边界。通俗地说，决策边界就像是一道看不见的墙，把不同类别的数据点分隔开。

对于鸢尾花数据集，决策边界就是将 **setosa** ●、**versicolor** ●、**virginica** ● 这三类点“尽可能准确地”区分开的线或曲线。

大家会在本章中看到，为了获得不同算法的决策边界，我们一般会用 `numpy.meshgrid()` 生成一系列均匀网格数据，然后再分别预测每个网格点的分类，以此划定决策边界。

在简单的情况下，决策边界可能是一条直线；但在复杂的问题中，决策边界可能是一条弯曲的曲线，甚至是多维空间中的超平面。

模型训练过程就是调整模型的参数，使得决策边界能够最好地拟合训练数据，并且在未见过的数据上也能表现良好。

要注意的是，决策边界的好坏直接影响分类模型的性能。一个好的决策边界能够很好地将数据分类，而一个不合适的决策边界可能导致模型预测错误。因此，选择合适的分类算法和调整模型参数是非常重要的，以获得有效的决策边界和准确的分类结果。

下面我们就用最通俗的语言，以几乎没有数学公式的方式，介绍几种常用分类算法。

## 32.2 $k$ 最近邻分类：近朱者赤，近墨者黑

**$k$  最近邻分类** ( $k$ -nearest neighbors), 简称  $k$ NN。

本书前文提过， $k$ NN 思路很简单——“近朱者赤，近墨者黑”。更准确地说，小范围投票，**少数服从多数** (majority rule)，如图 2 所示。 $k$  是参与投票的最近邻的数量， $k$  为用户输入值。

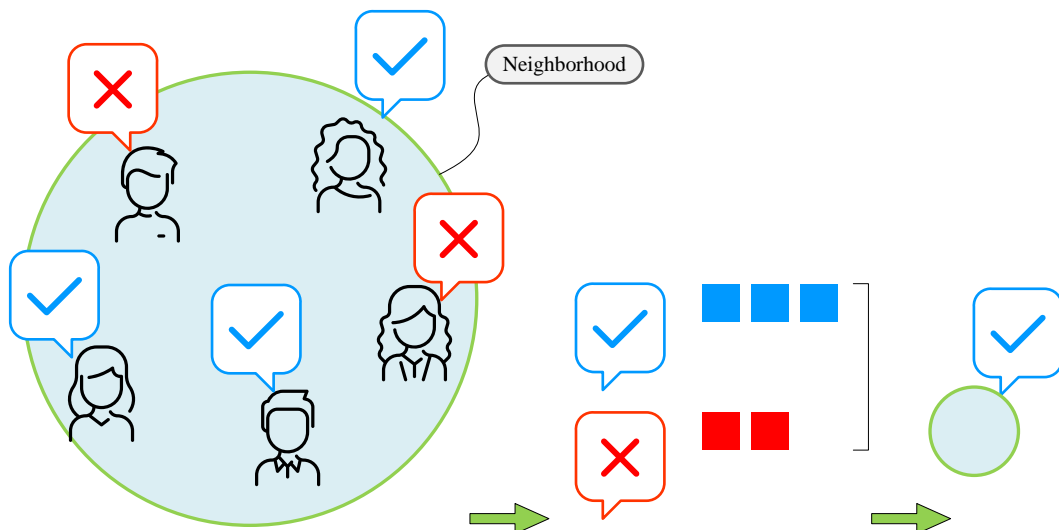


图 2.  $k$  近邻分类核心思想——小范围投票，少数服从多数

最近邻数量  $k$  直接影响查询点分类结果；因此，选取合适  $k$  值格外重要。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

图 3 所示为  $k$  取四个不同值时，查询点  $\times$  预测分类结果变化情况。如图 3 (a) 所示，当  $k = 4$  时，查询点  $\times$  近邻中，3 个近邻为  $\bullet$  ( $C_1$ )，1 个近邻为  $\bullet$  ( $C_2$ )；采用等权重投票，查询点  $\times$  预测分类为  $\bullet$  ( $C_1$ )。

当近邻数量  $k$  提高到 8 时，近邻社区中，4 个近邻为  $\bullet$  ( $C_1$ )，4 个近邻为  $\bullet$  ( $C_2$ )，如图 3 (b) 所示；等权重投票的话，两个标签各占 50%。因此  $k = 8$  时，查询点  $\times$  恰好在决策边界上。

如图 3 (c) 所示，当  $k = 12$  时，查询点  $\times$  近邻中 5 个为  $\bullet$  ( $C_1$ )，7 个为  $\bullet$  ( $C_2$ )；等权重投票条件下，查询点  $\times$  预测标签为  $\bullet$  ( $C_2$ )。当  $k = 16$  时，如图 3 (d) 所示，查询点  $\times$  预测标签同样为  $\bullet$  ( $C_2$ )。

$kNN$  算法选取较小的  $k$  值虽然能准确捕捉训练数据的分类模式；但是，缺点也很明显，容易受到噪声影响。



鸢尾花书《机器学习》会专门介绍  $kNN$  算法。

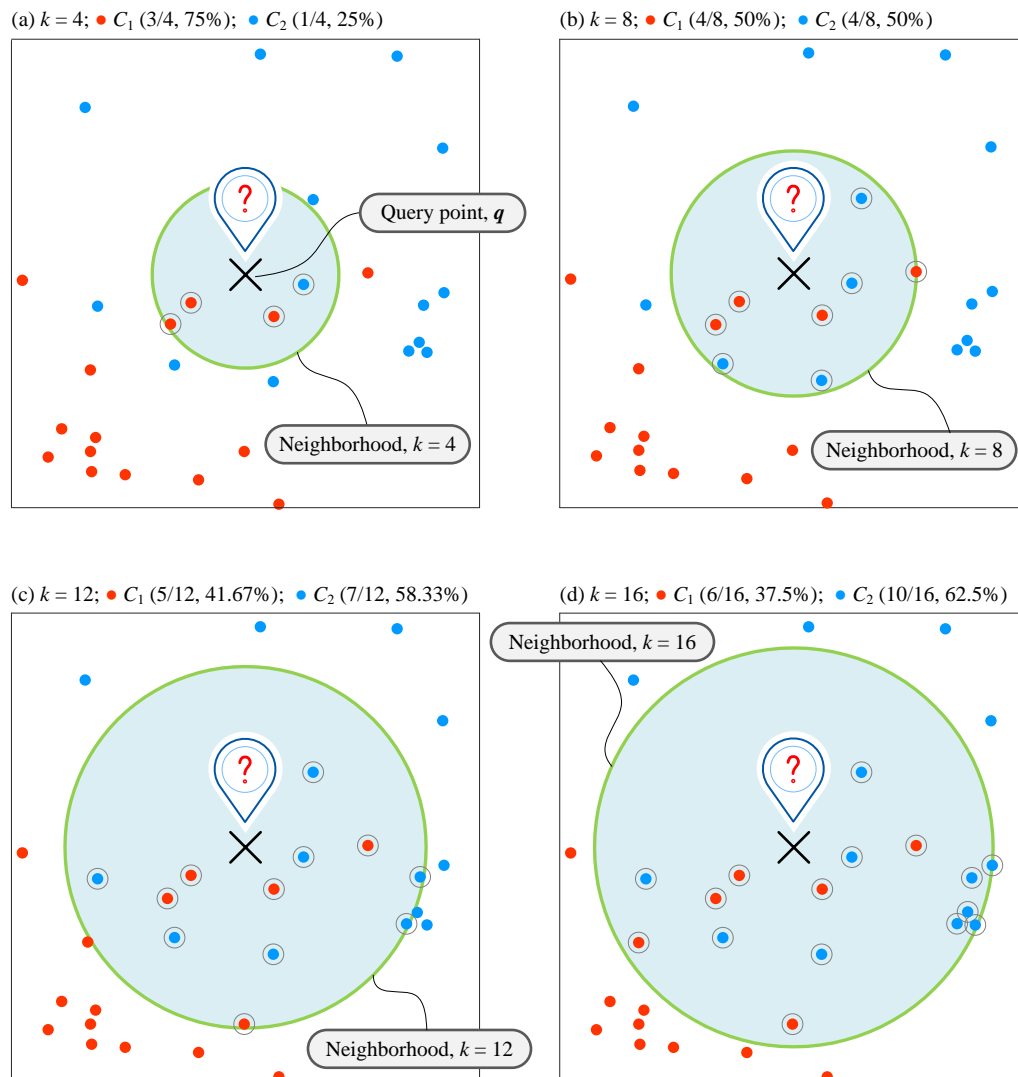


图 3. 近邻数量  $k$  值影响查询点的分类结果

图 4 所示为利用  $k$ NN 算法确定的鸢尾花数据决策区域和决策边界。

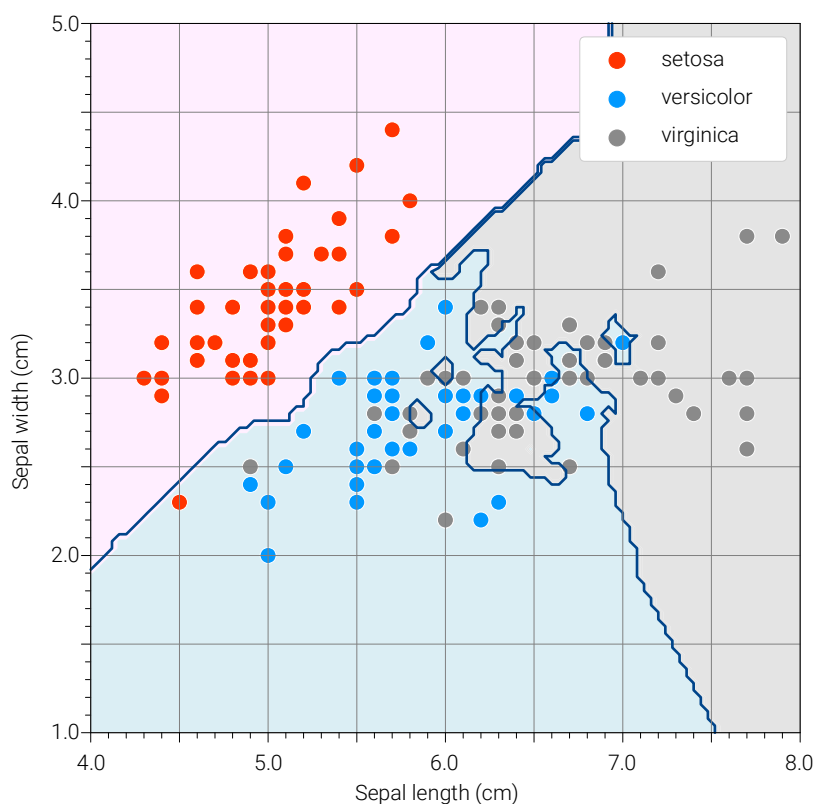


图 4. 根据花萼长度、花萼宽度，用  $k$ NN 算法确定决策边界

代码 1 用  $k$ NN 算法确定决策边界。下面介绍其中重要的语句。

**a** 利用 `sklearn.datasets.load_iris()` 加载了鸢尾花数据集。本书前文介绍过，在 `scikit-learn` 中，`datasets` 模块提供了一些经典的示例数据集。

**b** 提取了鸢尾花数据集的前两列——花萼长度、花萼宽度——作为分类特征。

**c** 提取鸢尾花分类标签。

**d** 用 `numpy.meshgrid()` 生成网格化数据，这些就是用来预测分类的查询点。

**e** 用 `matplotlib.colors.ListedColormap()` 创建离散色谱，即颜色映射，展示鸢尾花预测分类的区域。

**f** 用 `sklearn.neighbors.KNeighborsClassifier(k_neighbors)` 创建了一个  $k$  最近邻分类器对象  $k$ NN，并将 `k_neighbors` 作为参数传递给这个分类器。这里的 `k_neighbors` 指定了算法中要使用的最近邻居数量。

**g** 这行代码用训练数据  $X$  和相应的标签  $y$  来训练  $k$  最近邻分类器  $k$ NN。在训练过程中，分类器会学习如何根据特征向量  $X$  将其分配到相应的标签  $y$  上。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

h 利用 `numpy.c_()` 将两个一维数组按列合并，形成一个新的二维数组，即查询点。  
`numpy.ravel()` 函数将二维数组展平成一维数组。

i 这行代码用之前训练好的  $k$  最近邻分类器  $kNN$  对查询点进行预测，得到预测的标签 `y_predict`。

j 利用 `numpy.reshape()` 将预测的标签 `y_predict` 调整为与 `xx1` 相同形状，以便后续可视化。

k 利用 `matplotlib.pyplot.contourf()` 绘制分类区域。

l 利用 `matplotlib.pyplot.contour()` 绘制分类决策边界。

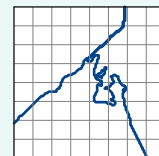
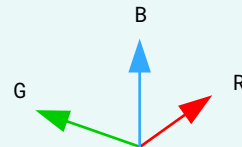
m 利用 `seaborn.scatterplot()` 绘制散点图展示鸢尾花数据集。

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap
from sklearn import neighbors, datasets

# 导入并整理数据
a iris = datasets.load_iris()
b X = iris.data[:, :2]
c y = iris.target
# 生成网格化数据
x1_array = np.linspace(4, 8, 101)
x2_array = np.linspace(1, 5, 101)
d xx1, xx2 = np.meshgrid(x1_array, x2_array)
# 创建色谱
rgb = [[255, 238, 255],
        [219, 238, 244],
        [228, 228, 228]]
rgb = np.array(rgb)/255.
e cmap_light = ListedColormap(rgb)
cmap_bold = [[255, 51, 0],
              [0, 153, 255],
              [138, 138, 138]]
cmap_bold = np.array(cmap_bold)/255.
k_neighbors = 4 # 定义kNN近邻数量k
# 创建kNN分类器对象
f kNN = neighbors.KNeighborsClassifier(k_neighbors)
g kNN.fit(X, y) # 用训练数据训练kNN
h q = np.c_[xx1.ravel(), xx2.ravel()]
# 用kNN对一系列查询点进行预测
i y_predict = kNN.predict(q)
j y_predict = y_predict.reshape(xx1.shape)
# 可视化
fig, ax = plt.subplots()
k plt.contourf(xx1, xx2, y_predict, cmap=cmap_light)
l plt.contour(xx1, xx2, y_predict, levels=[0, 1, 2],
             colors=np.array([0, 68, 138])/255.)
m sns.scatterplot(x=X[:, 0], y=X[:, 1],
                  hue=iris.target_names[y],
                  ax=ax,
                  palette=dict(setosa=cmap_bold[0, :],
                              versicolor=cmap_bold[1, :],
                              virginica=cmap_bold[2, :]),
                  alpha=1.0,
                  linewidth=1, edgecolor=[1, 1, 1])
plt.xlim(4, 8); plt.ylim(1, 5)
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
ax.grid(linestyle='--', linewidth=0.25,
        color=[0.5, 0.5, 0.5])
ax.set_aspect('equal', adjustable='box')

```



代码 1. 根据花萼长度、花萼宽度，用 k-NN 算法确定决策边界； Bk1\_Ch32\_01.ipynb

## 32.3 高斯朴素贝叶斯分类：贝叶斯定理的应用

**高斯朴素贝叶斯分类** (Gaussian Naive Bayes, GNB) 是一种基于**贝叶斯定理** (Bayes' theorem) 的分类算法。



### 什么是贝叶斯定理？

贝叶斯定理是一种概率论中用于计算条件概率的重要公式。它描述了在已知某个条件下，另一事件发生的概率。根据贝叶斯定理，我们可以通过已知的先验概率和条件概率，来计算更新后的后验概率。这个定理在统计学、机器学习和人工智能等领域广泛应用，尤其在贝叶斯推断和贝叶斯分类中起着重要作用。

贝叶斯定理、贝叶斯分类、贝叶斯推断中有两个重要概念——**先验概率** (prior probability, prior)、**后验概率** (posterior probability, posterior)。

先验概率是指在考虑任何新证据之前，我们对一个事件或假设的概率的初始估计。它基于以前的经验、先前的观察或领域知识。这种概率是“先验”的，因为它不考虑新数据或新证据，只是基于我们事先已经了解的信息。

假设我们要研究某地区的流感发病率。在流感季节之前，我们可能会查阅历史数据、了解流感传播的模式以及人口的健康状况，从而得出在流感季节中某人患上流感的初始估计概率，这就是先验概率。

后验概率是指在考虑了新证据或数据后，我们对一个事件或假设的概率进行更新后的估计。在得到新的信息后，我们根据贝叶斯定理来更新先验概率，以得到后验概率。贝叶斯定理将先验概率和新的证据结合起来，提供了一个更准确的概率估计。

在流感季节中，我们开始收集实际发病数据，比如每天有多少人确诊患上流感。根据这些新数据，我们可以使用贝叶斯定理来更新先前的先验概率，得到一个更准确的后验概率，以更好地预测未来发病率或做出相关决策。

图 5 所示为高斯朴素贝叶斯分类的流程图。

高斯朴素贝叶斯分类假设每个特征在给定类别下是条件独立的，即给定类别的情况下，每个特征与其他特征之间条件独立。这便是高斯朴素贝叶斯分类中“朴素”两个字的来由。然后，将每个类别的特征分布建模为高斯分布，这则是高斯朴素贝叶斯分类中“高斯”两个字的来由。

以图 5 为例，给定标签为  $C_1$  (红色点)，分别独立获得  $f_{X1|Y}(x_1 | C_1)$  和  $f_{X2|Y}(x_2 | C_1)$ 。假设条件独立， $f_{Y,X1,X2}(C_1, x_1, x_2) = p_Y(C_1) \cdot f_{X1|Y}(x_1 | C_1) \cdot f_{X2|Y}(x_2 | C_1)$ 。



大家如果对上述内容有疑惑的话，请参考鸢尾花书《统计至简》第 18、19 章。

在训练时，算法从训练数据中学习每个类别的各个特征的（条件）均值和方差，用于计算每个特征在该类别下的概率密度函数，即**似然概率** (likelihood)。

当有新的未标记样本输入时，算法将计算该样本在每个类别下的条件概率（后验概率），并选择具有最高概率的类别作为预测结果。



高斯朴素贝叶斯分类算法的优点是简单快速、易于实现和适用于高维数据。它能够处理连续型数据，因为它假设数据分布是高斯分布。

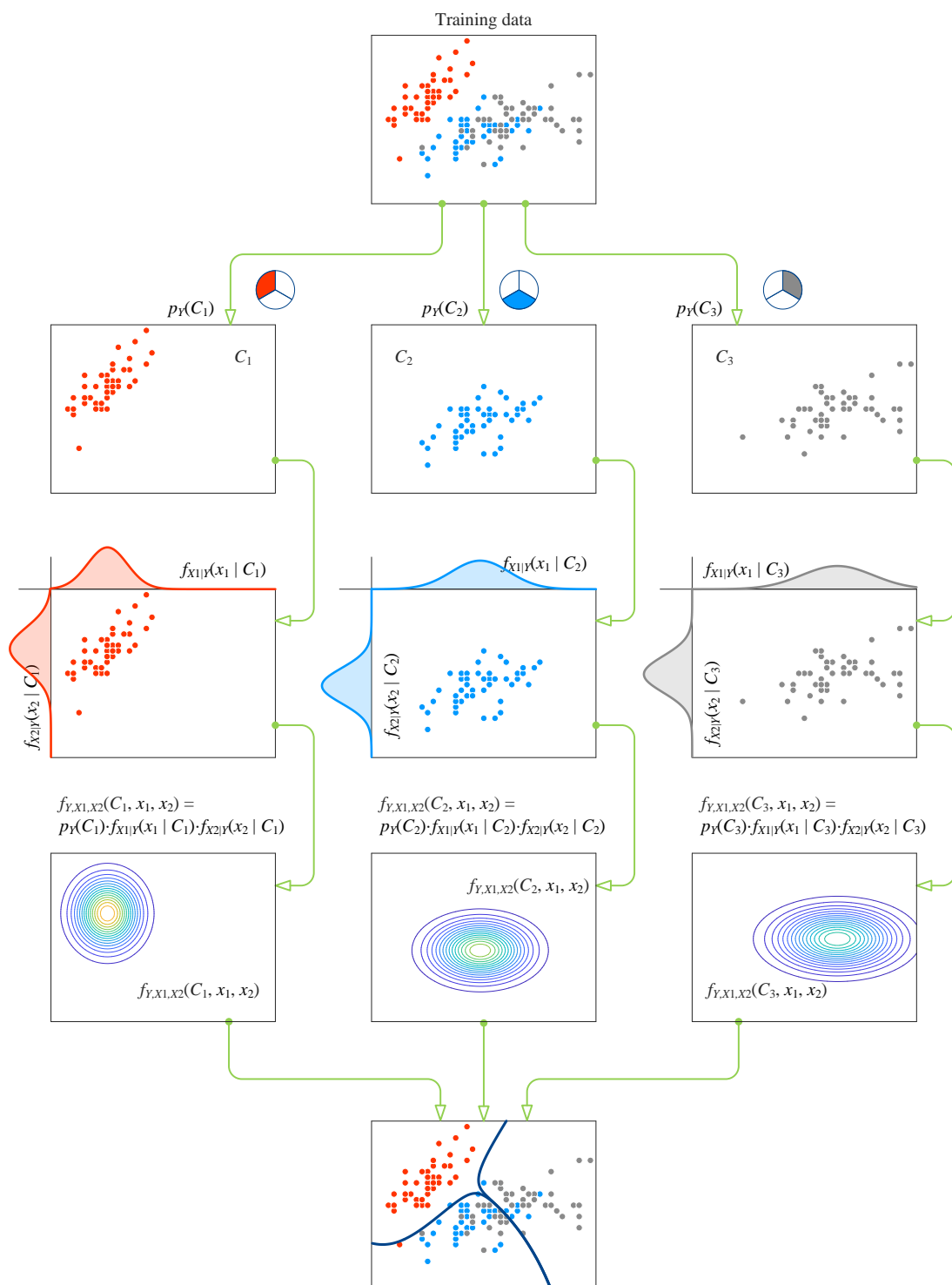


图 5. 高斯朴素贝叶斯分类过程

图 6 所示为利用高斯朴素贝叶斯分类算法获得的决策边界。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

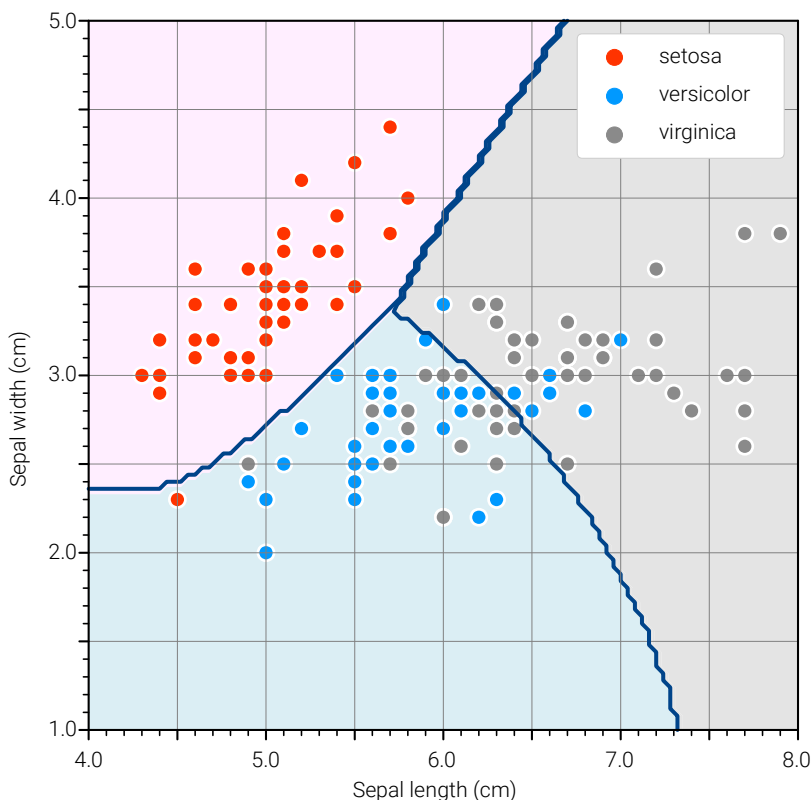



图 6. 根据花萼长度、花萼宽度，用高斯朴素贝叶斯算法确定决策边界

代码 2 所示为高斯朴素贝叶斯分类算法部分代码，请大家用代码 2 替换代码 1 对应语句。  
Bk1\_Ch32\_02.ipynb 有完整代码，请大家自行分析并逐行注释。

```

a from sklearn.naive_bayes import GaussianNB
  # 创建高斯朴素贝叶斯分类器对象
b gnb = GaussianNB()
  # 用训练数据训练kNN
  gnb.fit(X, y)
  # 用高斯朴素贝叶斯分类器对一系列查询点进行预测
  y_predict = gnb.predict(q)

```

代码 2. 用高斯朴素贝叶斯算法确定决策边界，部分代码;  Bk1\_Ch32\_02.ipynb

## 32.4 支持向量机：间隔最大化

图 7 所示为支持向量机 (Support Vector Machine, SVM) 核心思路。

如图 7 所示，一片湖面左右散布着蓝色 ● 红色 ● 礁石，游戏规则是，皮划艇以直线路径穿越水道，保证船身恰好紧贴礁石。寻找一条路径，让该路径通过的皮划艇宽度最大。很明显，图 7 (b) 中规划的路径好于图 7 (a)。

图 7 (b) 中加黑圈 ○ 的五个点，就是所谓的**支持向量** (support vector)。

图 7 中深蓝色线，便是**决策边界**，也称**分离超平面** (separating hyperplane)。特别提醒大家注意一点，加黑圈 ○ 支持向量确定决策边界位置；其他数据并没有起到任何作用。因此，SVM 对于数据特征数量远高于数据样本量的情况也有效。

图 7 中两条虚线之间宽度叫做**间隔** (margin)。支持向量机的优化目标为——间隔最大化。

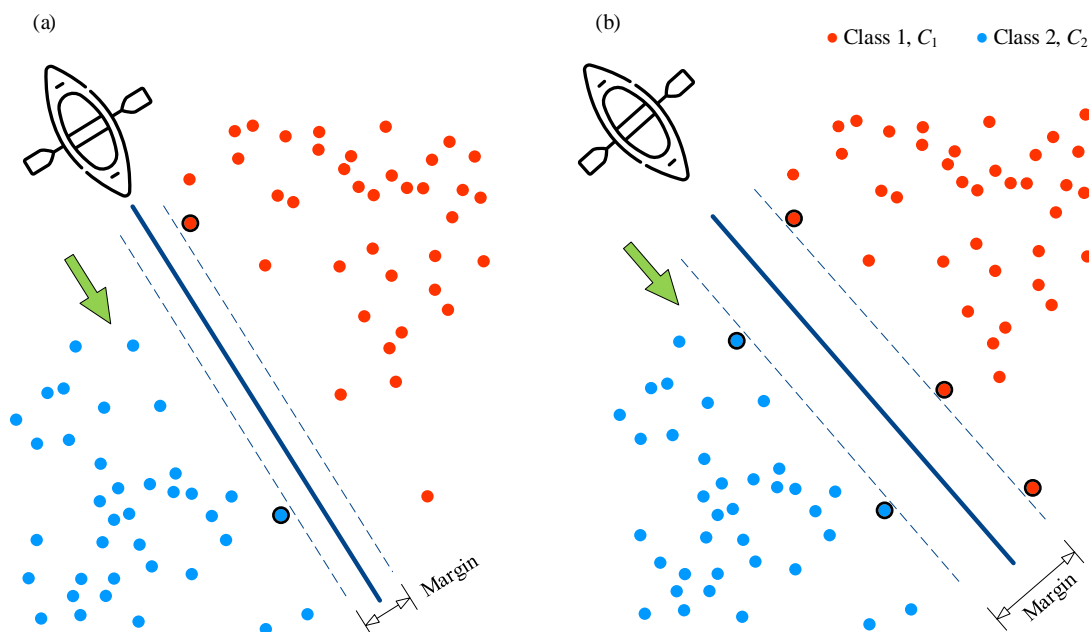


图 7. 支持向量机原理

从数据角度，图 7 两类数据用一条直线便可以分割开来，这种数据叫做**线性可分** (linearly separable)。线性可分问题采用**硬间隔** (hard margin)；白话说，硬间隔指的是，间隔内没有数据点。

实践中，并不是所有数据都是线性可分。多数时候，数据**线性不可分** (non-linearly separable)。如图 8 所示，不能找到一条直线将蓝色 ● 红色 ● 数据分离。

对于线性不可分问题，就要引入两种方法——**软间隔** (soft margin) 和**核技巧** (kernel trick)。

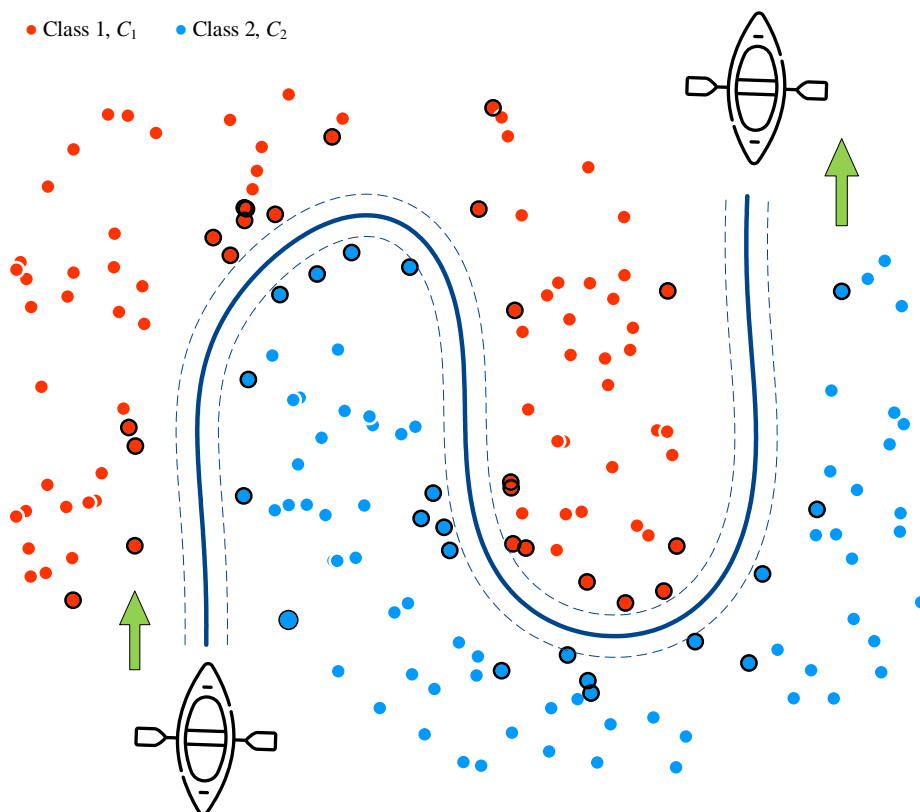


图 8. 线性不可分数据

白话说，如图 9 所示，软间隔相当于一个缓冲区 (buffer zone)。软间隔存在时，用决策边界分离数据时，有数据点侵入间隔，甚至超越间隔带。

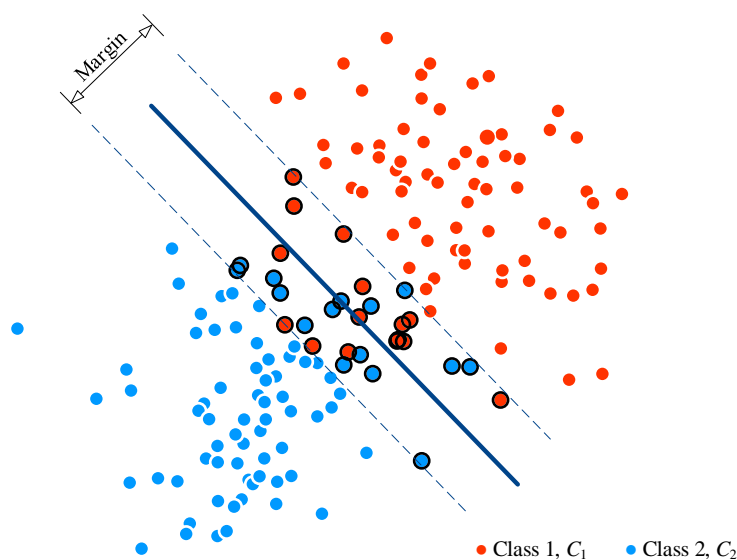


图 9. 软间隔

图 10 所示用支持向量机确定的决策边界。

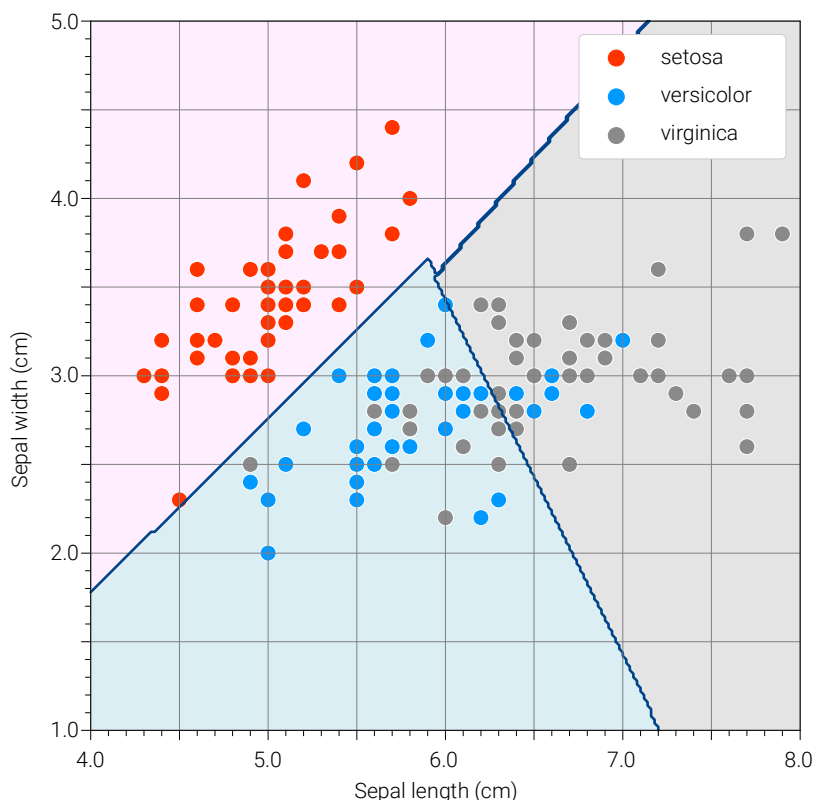


图 10. 根据花萼长度、花萼宽度，用支持向量机（线性核，默认）算法确定决策边界


代码 3 为支持向量机（线性核）算法部分代码，请大家用代码 3 替换代码 1 对应语句。

**线性核** (linear kernel) 是 SVM 中最简单的核函数之一。它适用于处理线性可分的数据集，即可以通过一个直线（在二维空间中）或一个超平面（在高维空间中）将不同类别的样本点分开。

```

a from sklearn import svm
# 创建支持向量机（线性核）分类器对象
b SVM = svm.SVC(kernel='linear')
# 用训练数据训练kNN
SVM.fit(X, y)
# 用支持向量机（线性核）分类器对一系列查询点进行预测
y_predict = SVM.predict(q)

```

代码 3. 用支持向量机（线性核，默认）算法确定决策边界，部分代码；  Bk1\_Ch32\_03.ipynb

## 32.5 核技巧：数据映射到高维空间

**核技巧** (kernel trick) 将数据映射到高维特征空间，相当于数据升维。如图 11 所示，样本数据有两个特征，用平面可视化数据点位置。很明显图 11 给出的原始数据线性不可分。

采用核技巧，将图 11 二维数据，投射到三维核曲面上；很明显，在这个高维特征空间，容易找到某个水平面，将蓝色 ● 红色 ● 数据分离。利用核技巧，分离线性不可分数据变得更容易。

通常，采用支持向量机解决线性不可分问题，需要并用软间隔和核技巧。如图 12 所示，SVM 分类环形数据中，核技巧配合软间隔。

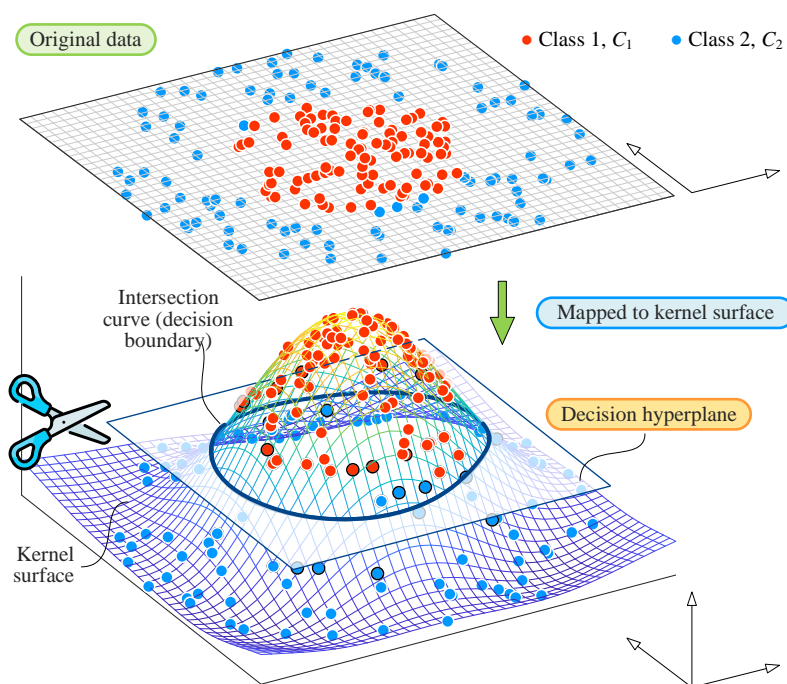


图 11. 核技巧原理

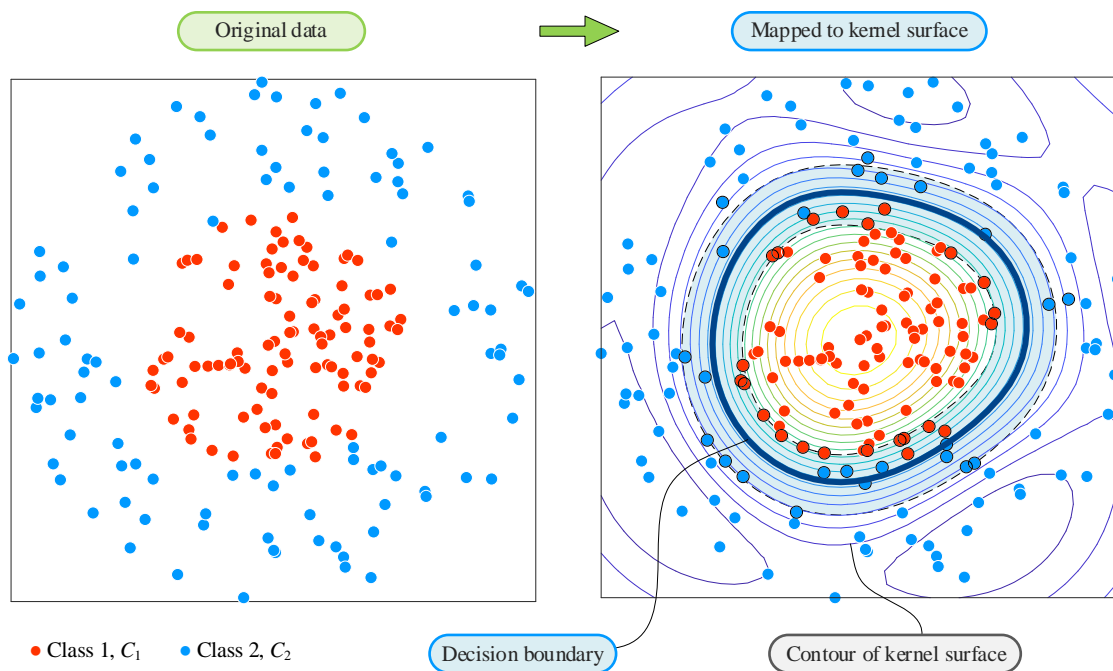


图 12. 核技巧配合软间隔

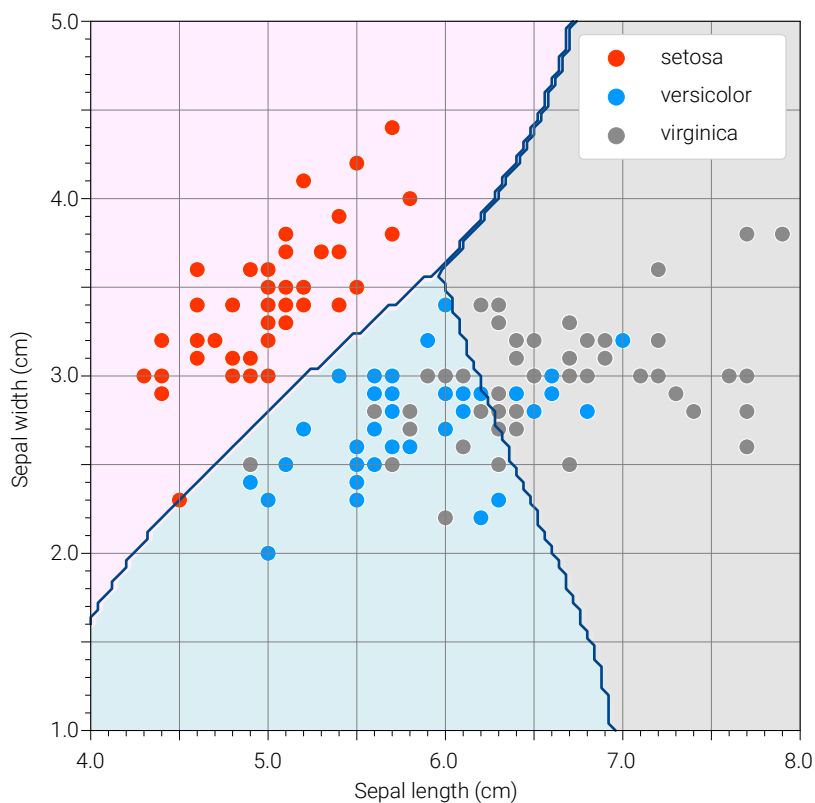


图 13. 根据花萼长度、花萼宽度，用支持向量机（高斯核）算法确定决策边界


代码 4 采用高斯核完成支持向量机算法并确定决策边界，请大家自行分析。

高斯核，也称为**径向基核** (Radial Basis Function Kernel)，是 SVM 中常用的非线性核函数。它能够将数据映射到无穷维的特征空间，从而在低维空间中不可分的数据变得线性可分。

```

a from sklearn import svm
# 创建支持向量机（高斯核）分类器对象
b SVM = svm.SVC(kernel='rbf', gamma='auto')
# 用训练数据训练kNN
SVM.fit(X, y)
# 用支持向量机（线性核）分类器对一系列查询点进行预测
y_predict = SVM.predict(q)

```

代码 4. 用支持向量机（高斯核）算法确定决策边界，部分代码；  Bk1\_Ch32\_04.ipynb



请大家完成如下题目。

Q1. 修改代码 1，调整 kNN 近邻数量，比如说尝试 6、7、8、9 等等，然后说明 kNN 近邻数量对决策边界的影响。

Q2. 使用代码 4 中的高斯核支持向量机时，请调整参数 gamma 的取值，并观察 gamma 大小对决策边界影响。

\* 题目很基础，本书不给答案。



本章几乎在“零公式”的条件下，给大家介绍了机器学习中三个特别重要的分类方法—— $k$  最近邻、高斯朴素贝叶斯分类、支持向量机。

简单来说， $k$  最近邻分类的核心思想就是“近朱者赤，近墨者黑”，小范围投票，少数服从多数。本章在利用  $k$  最近邻完成分类时，用的距离是默认的欧氏距离。鸢尾花书中，会不断地给大家介绍各种各样其他形式的距离，以及它们背后的数学原理。

高斯朴素贝叶斯分类提到了两个重要人名——高斯、贝叶斯。在鸢尾花书《统计至简》中，高斯和贝叶斯是最重要的两个人物。这本书中，我们会用多元高斯分布帮助大家“升维”，用贝叶斯定理完成分类和推断。本章介绍的高斯朴素贝叶斯分类算法则是两者的完美合体。

支持向量机的原理也很简单——间隔最大化。想要理解支持向量机算法背后主要数学工具都在鸢尾花书《矩阵力量》中。在介绍支持向量机时，我们还聊了聊核技巧。核技巧是一种通过将数据映射到高维特征空间来处理非线性问题的方法，也离不开线性代数工具。