支持向量机

刘德华

2023 年 6 月 20 日

例子 1

使用具有线性核的支持向量机分类器绘制两类可分离数据集内的最大间隔分离超平面[a]。

[a]实现的程序见例1

例子 2

使用具有 RBF 核的非线性 SVC 执行二进制分类[a]。

[a]实现的程序见例1

例子 3

iris 数据集的 2D 投影上的不同线性 SVM 分类器的比较。我们只考虑这个数据集的前两个特征[a]。

[a]实现的程序见例1

例子 4

使用 SVC 为不平衡的类找到最佳分离超平面[a]。

[a]实现的程序见例1

例子 5

使用线性、多项式和 RBF 核的一维回归的示例[a]。

[a]实现的程序见例1

1 代码

Python 代码 1

```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap3\sourcecode")
# 导入绘图库
```

```
import matplotlib.pyplot as plt
# 导入支持向量机模型
from sklearn import svm
# 导入数据生成工具
from sklearn.datasets import make_blobs
# 导入决策边界显示工具
from sklearn.inspection import DecisionBoundaryDisplay
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 生成样本
X, y = make_blobs(
    n_samples=40, # 样本量
    centers=2, # 两个类
    random_state=6
)
# 建立没有正则化的 SVM
clf = svm.SVC(kernel="linear", C=1000)
# 模型拟合
clf.fit(X, y)
# 绘图
fig, ax = plt.subplots(figsize=(6,6), tight_layout=True)
ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Paired)
# 绘制决策边界
DecisionBoundaryDisplay.from_estimator(
    clf,
    X,
    plot_method="contour",
    colors="k",
    levels=[-1, 0, 1],
    alpha=0.5,
    linestyles=["--", "-", "--"],
    ax=ax
)
# 绘制支持向量
ax.scatter(
    clf.support_vectors_[:, 0], # 支持向量
    clf.support_vectors_[:, 1], # 支持向量
    s=100,
    linewidth=1,
```

```
    facecolors="none",
    edgecolors="k",
)
plt.show()
fig.savefig("../codeimage/code1.pdf")
```

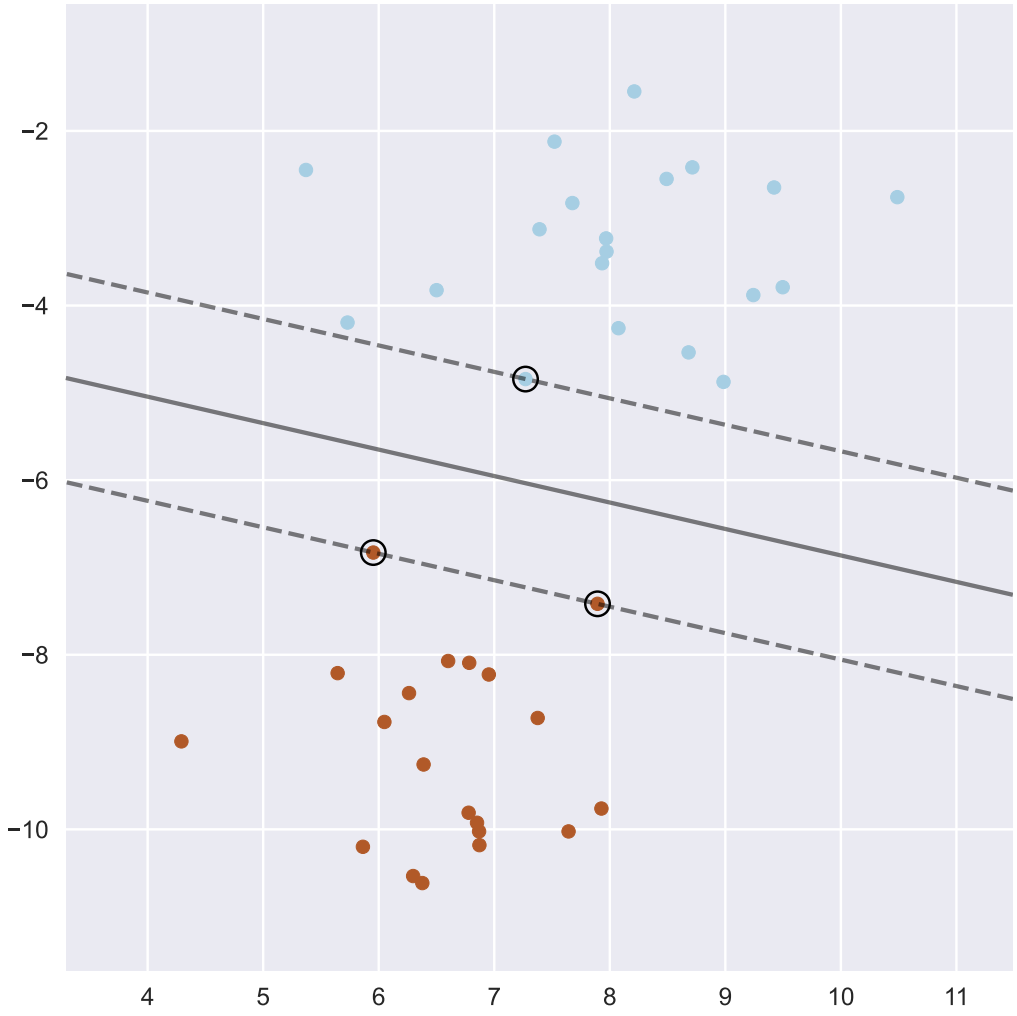

图 1: code1

Python 代码 2

```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap3\sourcecode")
# 导入基础计算库
import numpy as np
# 导入绘图库
```

```
import matplotlib.pyplot as plt
# 导入支持向量机模型
from sklearn import svm
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 生成样本
xx, yy = np.meshgrid(
    np.linspace(-3, 3, 500),
    np.linspace(-3, 3, 500)
)
np.random.seed(0)
X = np.random.randn(300, 2)
Y = np.logical_xor(X[:, 0] > 0, X[:, 1] > 0)
# 拟合 NuSVM 模型
clf = svm.NuSVC(gamma="auto")
# 模型拟合
clf.fit(X, Y)
# 决策函数
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# 开始绘图
fig, ax = plt.subplots(figsize=(6,6), tight_layout=True)
# 图形
ax.imshow(
    Z,
    interpolation="nearest",
    extent=(xx.min(), xx.max(), yy.min(), yy.max()),
    aspect="auto",
    origin="lower",
    cmap=plt.cm.PuOr_r,
)
# 等高线
contours = ax.contour(xx, yy, Z, levels=[0], linewidths=2, linestyles="dashed")
ax.scatter(X[:, 0], X[:, 1], s=30, c=Y, cmap=plt.cm.Paired, edgecolors="k")
ax.set_xticks(())
ax.set_yticks(())
ax.set_xlim([-3, 3])
ax.set_ylim([-3, 3])
plt.show()
```

```
fig.savefig("../codeimage/code2.pdf")
```

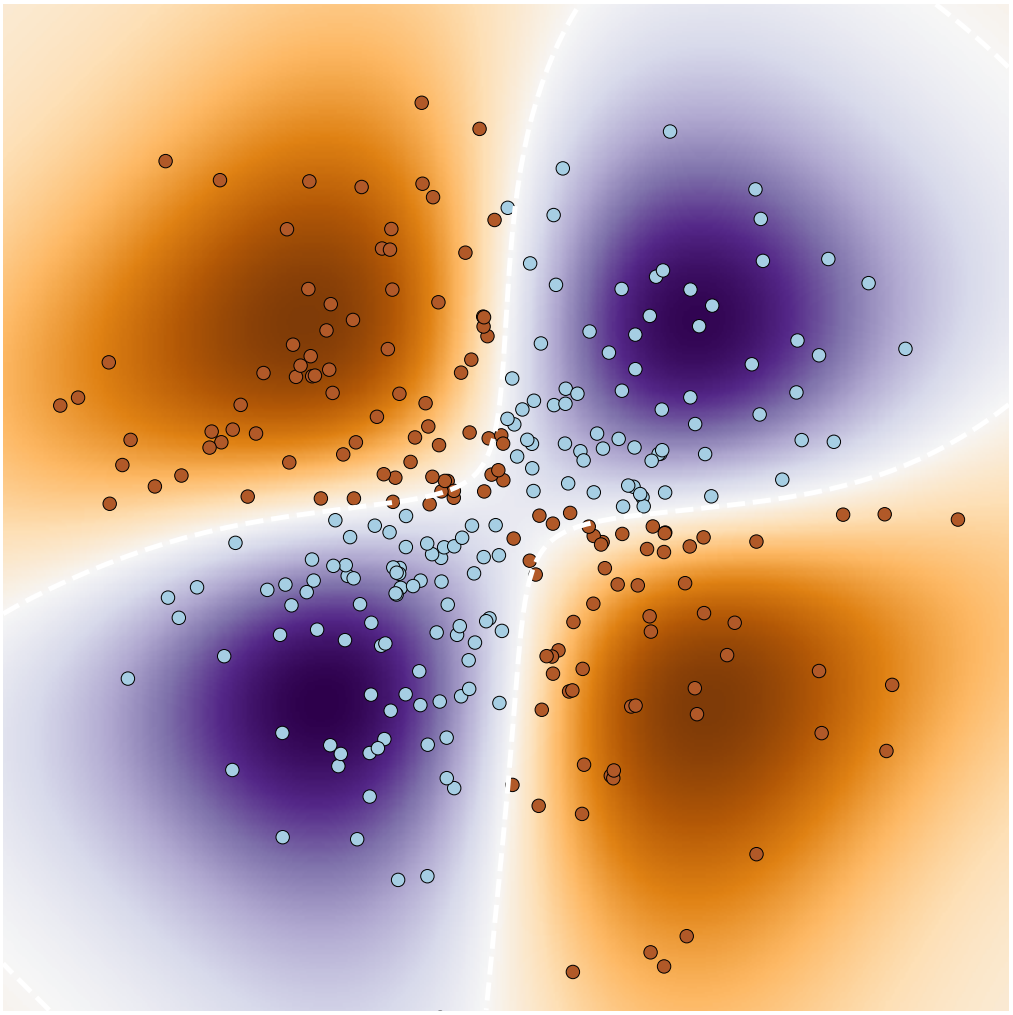

图 2: code2

Python 代码 3

```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap3\sourcecode")
# 导入绘图库
import matplotlib.pyplot as plt
# 导入支持向量机模型
from sklearn import svm
# 导入决策边界可视化工具
from sklearn.inspection import DecisionBoundaryDisplay
# 导入 iris 数据集
```

```
from sklearn.datasets import load_iris
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 生成样本
iris = load_iris()
# 取前两个变量
X = iris.data[:, :2]
y = iris.target
# 惩罚系数
C = 1.0  # SVM regularization parameter
models = (
    svm.SVC(kernel="linear", C=C),
    svm.LinearSVC(C=C, max_iter=10000),
    svm.SVC(kernel="rbf", gamma=0.7, C=C), # 径向核函数
    svm.SVC(kernel="poly", degree=3, gamma="auto", C=C) # 多项式核
)
# 模型拟合
models = (clf.fit(X, y) for clf in models)
# 绘图标题
titles = (
    "SVC with linear kernel",
    "LinearSVC (linear kernel)",
    "SVC with RBF kernel",
    "SVC with polynomial (degree 3) kernel",
)
# 开始画图
fig, sub = plt.subplots(2, 2, figsize=(14,14), tight_layout=True)
plt.subplots_adjust(wspace=0.4, hspace=0.4)
# 第一、二个维度的 X
X0, X1 = X[:, 0], X[:, 1]
for clf, title, ax in zip(models, titles, sub.flatten()):
    # 绘制决策边界
    disp = DecisionBoundaryDisplay.from_estimator(
        clf,
        X,
        response_method="predict",
        cmap=plt.cm.coolwarm,
        alpha=0.8,
        ax=ax,
```

```
        xlabel=iris.feature_names[0],
        ylabel=iris.feature_names[1],
    )
    # 绘制散点图
    ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors="k")
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)
plt.show()
fig.savefig("../codeimage/code3.pdf")
```

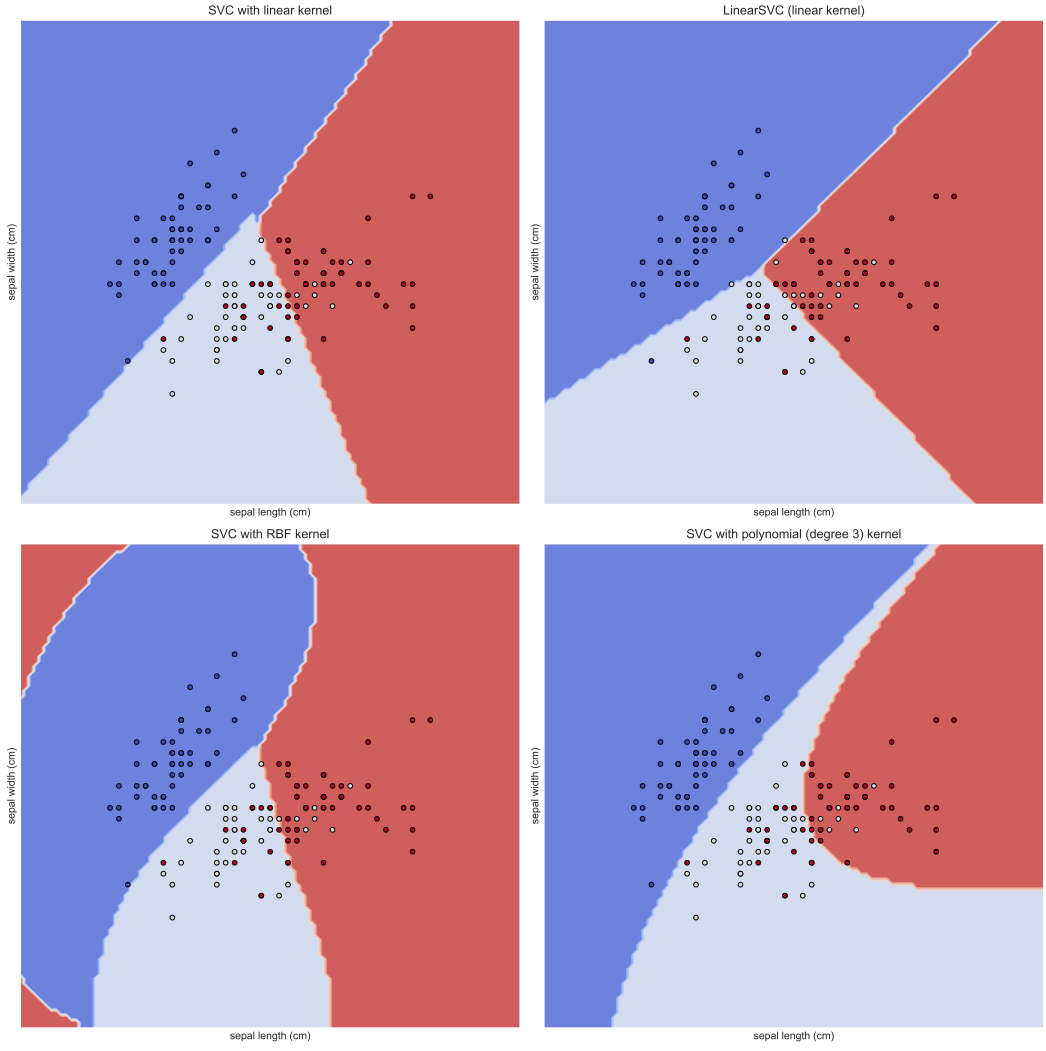

图 3: code3

Python 代码 4

```
# 导入操作系统库
import os
```

```
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap3\sourcecode")
# 导入绘图库
import matplotlib.pyplot as plt
# 导入支持向量机模型
from sklearn import svm
# 导入决策边界可视化工具
from sklearn.inspection import DecisionBoundaryDisplay
# 导入数据集生成工具
from sklearn.datasets import make_blobs
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 生成样本
n_samples_1 = 1000
n_samples_2 = 100
centers = [[0.0, 0.0], [2.0, 2.0]]
clusters_std = [1.5, 0.5]
# 分类数据
X, y = make_blobs(
    n_samples=[n_samples_1, n_samples_2], # 分别的样本量
    centers=centers, # 聚类中心
    cluster_std=clusters_std, # 标准差
    random_state=0,
    shuffle=False,
)
# 线性 SVM 模型
clf = svm.SVC(kernel="linear", C=1.0)
# 模型拟合
clf.fit(X, y)
# 加权的 SVM 模型
wclf = svm.SVC(kernel="linear", class_weight={1: 10})
# 模型拟合
wclf.fit(X, y)
# 开始绘图
fig, ax = plt.subplots(figsize=(6,6), tight_layout=True)
# 绘制散点
ax.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired, edgecolors="k")
# 绘制 SVM 的决策边界
disp = DecisionBoundaryDisplay.from_estimator(
```

```
    clf,
    X,
    plot_method="contour",
    colors="k",
    levels=[0],
    alpha=0.5,
    linestyles=["--"],
    ax=ax
)
# 绘制加权的 SVM 的决策边界
wdisp = DecisionBoundaryDisplay.from_estimator(
    wclf,
    X,
    plot_method="contour",
    colors="r",
    levels=[0],
    alpha=0.5,
    linestyles=["-"],
    ax=ax
)
# 添加图例
ax.legend(
    [disp.surface_.collections[0], wdisp.surface_.collections[0]],
    ["non weighted", "weighted"],
    loc="upper right",
)
plt.show()
fig.savefig("../codeimage/code4.pdf")
```

Python 代码 5

```
# 导入操作系统库
import os
# 更改工作目录
os.chdir(r"D:\softwares\applied statistics\pythoncodelearning\chap3\sourcecode")
# 导入基础计算库
import numpy as np
# 导入绘图库
import matplotlib.pyplot as plt
# 导入支持向量机模型
from sklearn.svm import SVR
# 导入绘图库中的字体管理包
from matplotlib import font_manager
# 实现中文字符正常显示
```

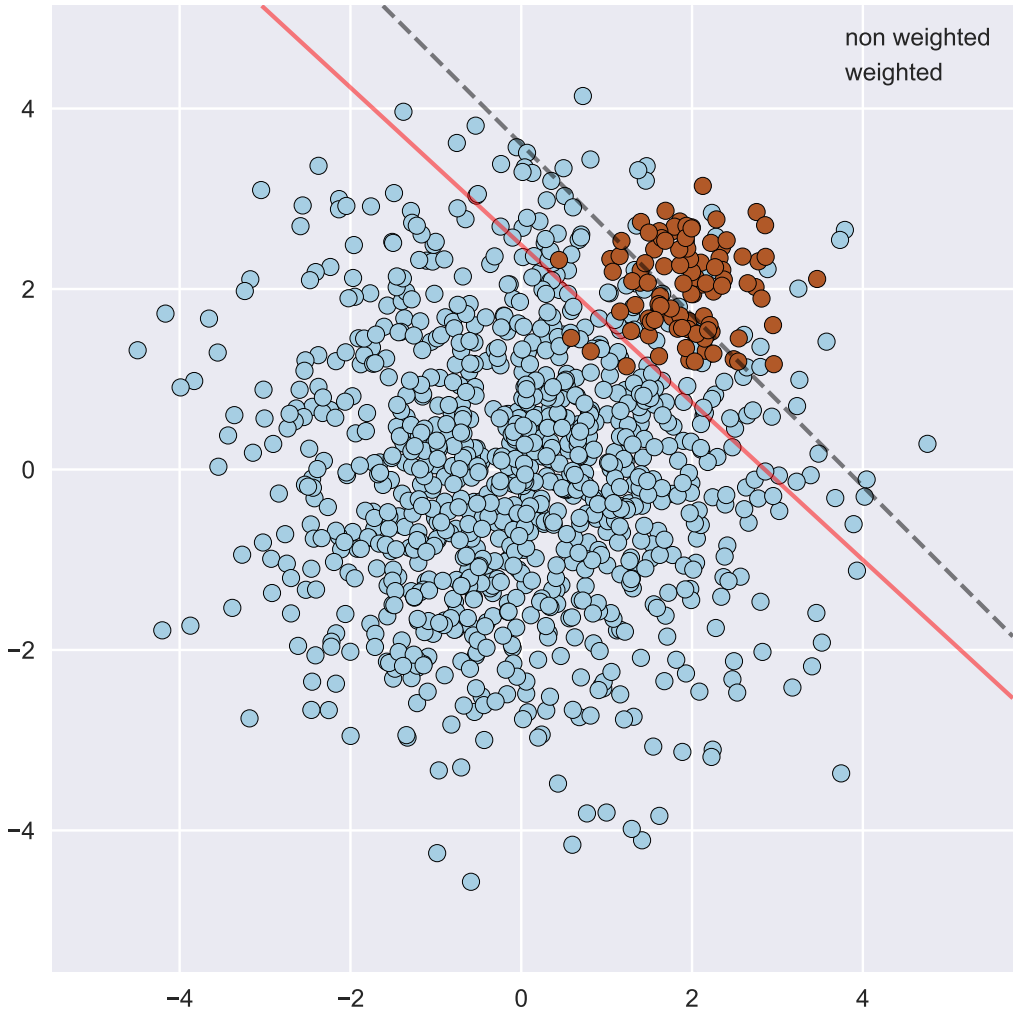

图 4: code4

```
font = font_manager.FontProperties(fname=r"C:\Windows\Fonts\SimKai.ttf")
# 使用 seaborn 风格绘图
plt.style.use("seaborn-v0_8")
# 生成样本
X = np.sort(5 * np.random.rand(40, 1), axis=0)
y = np.sin(X).ravel()
# 添加噪声
y[::5] += 3 * (0.5 - np.random.rand(8))
# rbf 核函数的 SVR
svr_rbf = SVR(kernel="rbf", C=100, gamma=0.1, epsilon=0.1)
# 线性核函数的 SVR
svr_lin = SVR(kernel="linear", C=100, gamma="auto")
# 多项式核函数的 SVR
svr_poly = SVR(kernel="poly", C=100, gamma="auto", degree=3, epsilon=0.1, coef0=1)
lw = 2
# 构造迭代对象列表
svrs = [svr_rbf, svr_lin, svr_poly]
kernel_label = ["RBF", "Linear", "Polynomial"]
model_color = ["m", "c", "g"]
# 开始绘图
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 10), sharey=True)
for ix, svr in enumerate(svrs):
    axes[ix].plot(
        X,
        svr.fit(X, y).predict(X),
        color=model_color[ix],
        lw=lw,
        label="{} model".format(kernel_label[ix]),
    )
    axes[ix].scatter(
        X[svr.support_],
        y[svr.support_],
        facecolor="none",
        edgecolor=model_color[ix],
        s=50,
        label="{} support vectors".format(kernel_label[ix]),
    )
    axes[ix].scatter(
        X[np.setdiff1d(np.arange(len(X)), svr.support_)],
        y[np.setdiff1d(np.arange(len(X)), svr.support_)],
        facecolor="none",
        edgecolor="k",
        s=50,
```

```
        label="other training data",
    )
    axes[ix].legend(
        loc="upper center",
        bbox_to_anchor=(0.5, 1.1),
        ncol=1,
        fancybox=True,
        shadow=True,
    )

fig.text(0.5, 0.04, "data", ha="center", va="center")
fig.text(0.06, 0.5, "target", ha="center", va="center", rotation="vertical")
fig.suptitle("Support Vector Regression", fontsize=14)
plt.show()
fig.savefig("../codeimage/code5.pdf")
```

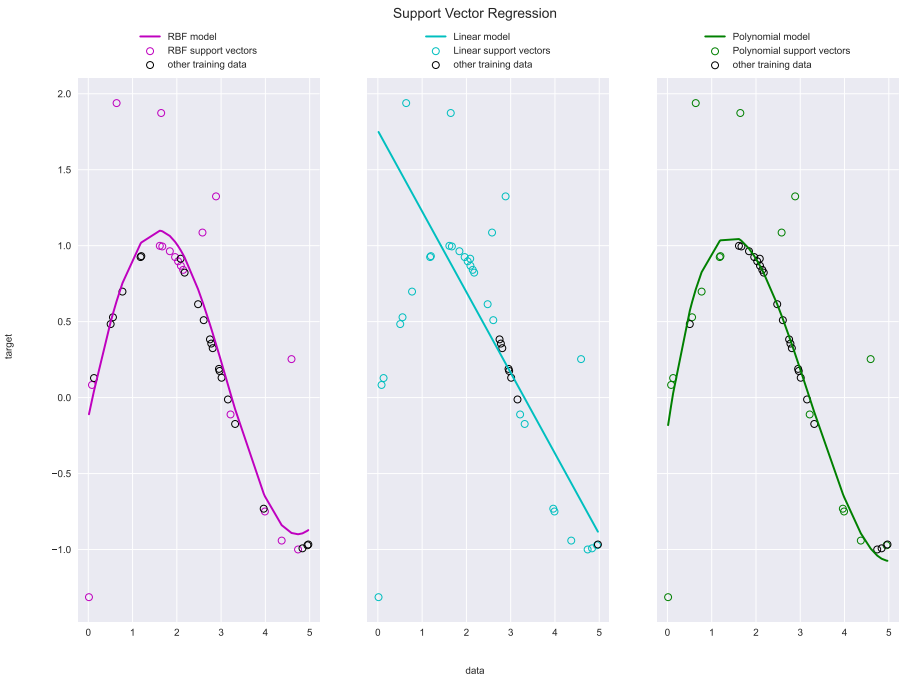

图 5: code5