



湖南大學

HUNAN UNIVERSITY

人工智能实验五报告
支持向量机

目录

一、 实验目的	3
二、 实验描述	3
三、 实验及结果分析	4
(1) 开发语言及运行环境;	4
(2) 实验的具体步骤;	4
① 方法 1, 自定义:	4
② 方法 2, sklearn:	6
(3) 根据实验数据集, 按实验要求给出相应的结果(截图)并对实验结果进行简要分析。	7
① 方法 1, 自定义:	7
② 方法 2, sklearn:	7
四、 心得	8
五、 程序文件名清单	9
六、 附录	9

一、实验目的

建议使用python编程实现，并在Mindspore框架下实现。

二、实验描述

支持向量机（SVM）是一种分类算法，用于预测一个或多个类别的分类问题。它通过在特征空间中找到一个最大化类别间隔的超平面来进行分类。

基本思想：

1. 最大化间隔：支持向量机的目标是在特征空间中找到一个超平面，这个超平面最大化了不同类别之间的间隔（即间隔）。这个超平面被称为最优分割超平面。

2. 支持向量：支持向量是那些最接近分割超平面的数据点。这些点对于定义最优分割超平面至关重要。

3. 软间隔：在实际应用中，并非所有数据都能被一个硬间隔超平面完美分割。因此，SVM引入了软间隔的概念，允许一些数据点违反间隔规则，以换取更好的泛化能力。

4. 核技巧：SVM通过核技巧可以处理非线性可分的数据。核函数允许在高维空间中隐式地执行点积，从而在原始特征空间中创建一个超平面，这个超平面在高维空间中是线性的。

5. 正则化：为了防止过拟合，SVM在优化问题中引入了正则化项。正则化参数控制了模型的复杂度和拟合程度之间的权衡。

6. 模型评估：使用诸如准确率、精确率、召回率、F1分数或ROC曲线等指标来评估SVM模型的性能和预测能力。

7. 预测：训练完成后，SVM模型可以用于预测新数据的类别。SVM通过计算新数据点与分割超平面的距离来确定其类别。

8. 概率输出：一些SVM实现允许输出预测概率。这通常通过校准方法实现，如Platt Scaling，它将SVM的输出转换为概率估计。

9. 特征重要性：虽然SVM不像逻辑回归那样直接提供系数来衡量特征的重要性，但可以通过检查支持向量或使用基于核的方法来间接评估特征的重要性。

数据集处理：

`nload_iris_data(filename)`中，使用`open` 打开指定的文件逐行读取数据，

对数据内容进行提取和处理，转换为NumPy数组，并进行标准化。最后输出数据集的一些特征。

三、实验及结果分析

(1) 开发语言及运行环境；

与实验一相同，不再赘述。

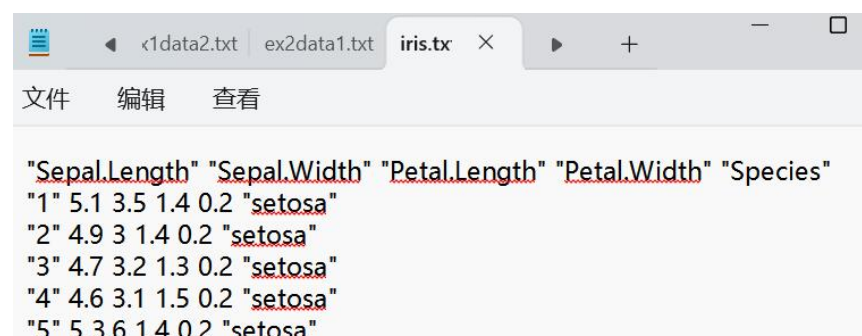
(2) 实验的具体步骤；

题目：

鸢尾花数据集（Iris data set）是模式识别中著名的数据集。本实验通过花萼（sepal）和花瓣（petal）的长和宽，建立SVM分类器来判断样本属于山鸢尾（Iris Setosa）、变色鸢尾（Iris Versicolor）还是维吉尼亚鸢尾（Iris Virginica）。请按要求完成实验。

数据集：

文件iris.txt为该实验的数据集，包含150个样本，对应数据集的每行数据。每行数据包含每个样本的四个特征（按顺序分别为花萼长度、花萼宽度、花瓣长度、花瓣宽度）和样本的类别信息（Iris Setosa、Iris Versicolor、Iris Virginica中的一种）。



```
"Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" "Species"
"1" 5.1 3.5 1.4 0.2 "setosa"
"2" 4.9 3 1.4 0.2 "setosa"
"3" 4.7 3.2 1.3 0.2 "setosa"
"4" 4.6 3.1 1.5 0.2 "setosa"
"5" 5 3.6 1.4 0.2 "setosa"
```

步骤与要求：

① 方法 1，自定义：

(1) 建立SVM分类器并用交叉验证法进行分析。

```

17 class CustomSVM(nn.Cell):
18     def __init__(self, input_dim, num_classes):
19         super(CustomSVM, self).__init__()
20         # 增加隐藏层, 使模型能够学习更复杂的特征
21         self.fc1 = nn.Dense(input_dim, 64)
22         self.relu = nn.ReLU()
23         self.dropout = nn.Dropout(0.3)
24         self.fc2 = nn.Dense(64, num_classes)
25
26     def construct(self, x):
27         x = self.fc1(x)
28         x = self.relu(x)
29         x = self.dropout(x)
30         x = self.fc2(x)
31         return x

```

CustomSVM 类是使用 MindSpore 框架实现的自定义支持向量机（SVM）模型：

`self.fc1`: 第一个全连接层（Dense Layer），它将输入特征从 `input_dim` 维映射到 64 维。
`self.relu`: 激活函数层，使用 ReLU（Rectified Linear Unit）激活函数，增加模型的非线性能力。
`self.dropout`: Dropout 层，用于防止过拟合。这里设置丢弃率为 0.3，意味着在训练过程中有 30% 的神经元输出会被暂时丢弃。
`self.fc2`: 第二个全连接层，将 64 维的特征映射到 `num_classes` 类，用于多类分类。

`construct` 方法中，定义了数据通过网络的前向传播过程：

输入数据 `x` 首先通过第一个全连接层 `fc1`。然后应用 ReLU 激活函数。接着通过 Dropout 层。最后通过第二个全连接层 `fc2`，输出最终的分类结果。

```

119 def cross_validation(X, y, k_folds=10):
120     indices = np.arange(len(X))
121     np.random.shuffle(indices)
122     fold_size = len(X) // k_folds
123     accuracies = []
124     for i in range(k_folds):
125         print(f"Processing fold {i + 1}/{k_folds}")
126         val_indices = indices[i * fold_size:(i + 1) * fold_size]
127         train_indices = np.concatenate([indices[:i * fold_size],
128                                         indices[(i + 1) * fold_size:]])
129
130         X_train, X_val = X[train_indices], X[val_indices]
131         y_train, y_val = y[train_indices], y[val_indices]
132
133         train_dataset = create_dataset(X_train, y_train)
134         model = train_model(train_dataset, X.shape[1])
135
136         # 评估
137         val_logits = model(ms.Tensor(X_val))
138         val_preds = ops.argmax(val_logits, 1)
139         accuracy = (val_preds.asnumpy() == y_val).mean()
140         accuracies.append(accuracy)

```

`cross_validation` 是实现 10 折交叉验证的过程，每次都调用 `train_model` 来进

行训练，其中定义定义损失函数为 Softmax 交叉熵，使用 Adam 优化器，添加 L2 正则化。

`create_dataset` 函数用于创建一个用于训练的数据集对象。

(2) 利用PCA降维，将数据转化为二维，然后绘制出分类决策边界。
与方法2相同。

② 方法 2，sklearn:

(1) 建立SVM分类器并用交叉验证法进行分析。

```
85 # 直接使用sklearn的交叉验证
86 print("\nPerforming cross-validation...")
87 svm = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
88 scores = cross_val_score(svm, X, y, cv=10)
89 print("Cross-validation scores:", scores)
90 print("Average accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

8788这两行是最核心的两行代码。

87: 创建SVM分类器实例，使用径向基函数（RBF）作为核，设置正则化参数 $C=1.0$ ，核函数参数 $\gamma='scale'$ ，随机种子42。

88: 使用 `cross_val_score` 函数进行交叉验证，设置 $cv=10$ 表示使用10折交叉验证。

(2) 利用PCA降维，将数据转化为二维，然后绘制出分类决策边界。

```
45 def pca_visualization(X, y):
46     # PCA降维
47     pca = PCA(n_components=2)
48     X_pca = pca.fit_transform(X)
49
50     # 创建和训练SVM分类器
51     svm = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
52     svm.fit(X_pca, y)
53
54     # 创建网格
55     x_min, x_max = X_pca[:, 0].min() - 0.5, X_pca[:, 0].max() + 0.5
56     y_min, y_max = X_pca[:, 1].min() - 0.5, X_pca[:, 1].max() + 0.5
57     xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
58                           np.arange(y_min, y_max, 0.02))
```

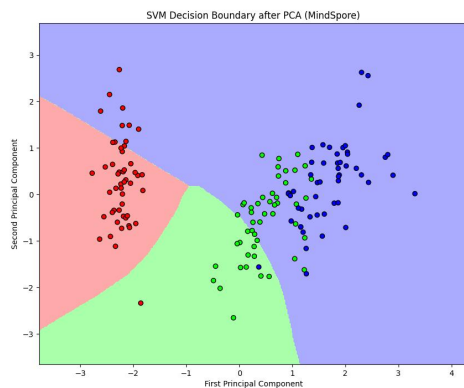
使用PCA降维至二维（之前算是四维），然后重新训练，与上面的步骤相同，然后绘图可视化结果。

这种方法下仅使用函数调用就可以完成训练过程，非常方便简洁，效果也很好。

(3) 根据实验数据集，按实验要求给出相应的结果（截图）并对实验结果进行简要分析。

① 方法 1，自定义：

```
Fold 10 accuracy: 0.8000
Cross-validation scores: [0.73333333 0.66666667 0.8        0.86666667 0.26666667 0.66666667
0.73333333 0.8        0.6        0.8        ]
Average accuracy: 0.69 (+/- 0.32)
```



平均精度只有 0.69 (+/- 0.32)，PCA 降维的效果相同（本应该相同）。

画图出来的效果一般，而且很不稳定。

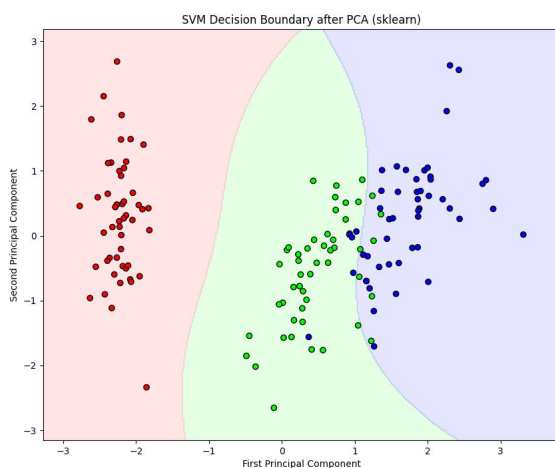
② 方法 2，sklearn:

```
Loading data...
Data shape: (150, 4)
Number of classes: 3
Sample of features: [[5.1 3.5 1.4 0.2]
 [4.9 3. 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5. 3.6 1.4 0.2]]
Sample of labels: [0 0 0 0 0]

Performing cross-validation...
Cross-validation scores: [1.          0.93333333 1.          0.93333333 1.          0.93333333
0.86666667 1.          1.          1.          ]
Average accuracy: 0.97 (+/- 0.09)

Performing PCA and visualization...
PCA explained variance ratio: [0.7296245 0.22850753]

Process finished with exit code 0
```



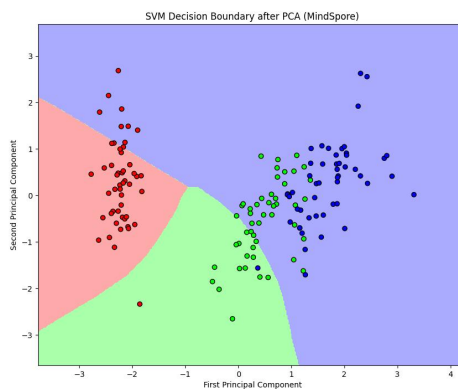
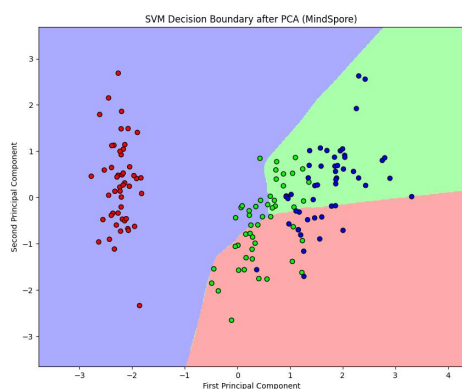
首先输出一些数据集特征，然后对十折交叉验证的结果精度进行输出，给出平均精度 0.97 (+0.09)，最后输出 PCA 降维的两个主成分：第一个主成分解释了大约 72.96% 的方差，第二个解释了大约 22.85%，前两个主成分共同捕捉了数据中 95.81% 的变异性，这两个主成分在描述数据集的变异性方面非常有效。

非常有效且好看的效果，画图出来也可以看出来效果很好。

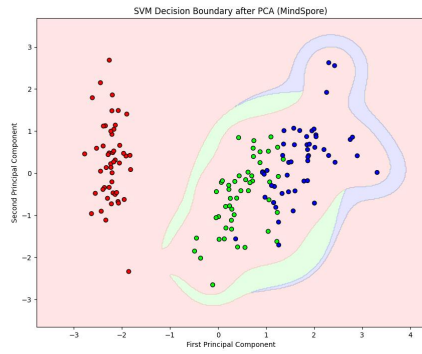
四、心得

最开始因为 mindspore 没有提供 SVM 的实现，只能自定义实现 SVM 分类器，因为是神经网络模拟的，分类的效果一般。后来使用 MindSpore 的 LSSVM，出来的效果反而更差了。最后换了 sklearn 的 SVM 实现，代码更精简而且出来的效果很好，所以这里提供了第一种自定义实现和 sklearn 实现的代码。

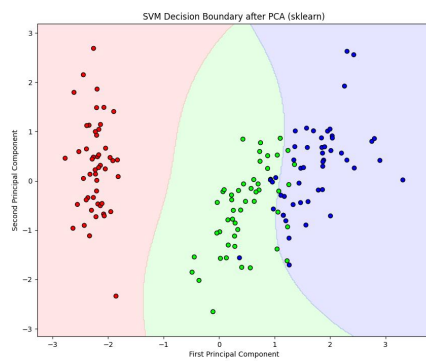
自定义：



LSSVM:



sklearn:



五、程序文件名清单

main.py: 自定义源代码

main.py.pdf: 自定义源代码 pdf 版

main2.py: sklearn 源代码

main2.py.pdf: sklearn 源代码 pdf 版

六、附录

代码如下。

main.py

```
1  import numpy as np
2  import pandas as pd
3  import mindspore as ms
4  import mindspore.dataset as ds
5  from mindspore import nn, ops
6  from mindspore.dataset.transforms import TypeCast
7  from sklearn.preprocessing import StandardScaler
8  from sklearn.decomposition import PCA
9  import matplotlib.pyplot as plt
10 from matplotlib.colors import ListedColormap
11
12 # 设置MindSpore上下文
13 ms.set_context(mode=ms.PYNATIVE_MODE, device_target="CPU")
14
15
16 # 自定义SVM分类器
17 class CustomSVM(nn.Cell):
18     def __init__(self, input_dim, num_classes):
19         super(CustomSVM, self).__init__()
20         # 增加隐藏层, 使模型能够学习更复杂的特征
21         self.fc1 = nn.Dense(input_dim, 64)
22         self.relu = nn.ReLU()
23         self.dropout = nn.Dropout(0.3)
24         self.fc2 = nn.Dense(64, num_classes)
25
26     def construct(self, x):
27         x = self.fc1(x)
28         x = self.relu(x)
29         x = self.dropout(x)
30         x = self.fc2(x)
31         return x
32
33
34 def load_iris_data(filename):
35     # 使用空格分隔符读取数据, 跳过第一行
36     data = []
37     with open(filename, 'r') as file:
38         # 跳过标题行
39         next(file)
40         for line in file:
41             # 移除开头的引号和结尾的换行符
42             if line.startswith('"'):
43                 line = line[1:]
44             # 分割行并处理每个字段
45             row = line.strip().split(' ')
46             # 提取数值和类别
47             values = [float(row[1]), float(row[2]), float(row[3]), float(row[4])]
48             label = row[5].strip('"')
49             data.append(values + [label])
50
51     # 转换为numpy数组
52     data_array = np.array(data)
53     X = data_array[:, :4].astype(np.float32)
54     y = pd.Categorical(data_array[:, 4]).codes.astype(np.int32)
55
56     # 数据标准化
57     scaler = StandardScaler()
58     X_scaled = scaler.fit_transform(X)
59
60     print("Data shape:", X.shape)
61     print("Number of classes:", len(np.unique(y)))
62     print("Sample of features:", X[:5])
63     print("Sample of labels:", y[:5])
64
65     return X_scaled, y
66
67
68 def create_dataset(features, labels, batch_size=32, shuffle=True):
69     data = {'features': features, 'labels': labels}
70     dataset = ds.NumpySlicesDataset(data, shuffle=shuffle)
71
```

```

72     type_cast_op = TypeCast(ms.float32)
73     dataset = dataset.map(operations=type_cast_op, input_columns=['features'])
74     dataset = dataset.map(operations=TypeCast(ms.int32), input_columns=['labels'])
75     dataset = dataset.batch(batch_size)
76
77     return dataset
78
79
80 def train_model(dataset, input_dim, num_classes=3, epochs=200): # 增加训练轮数
81     # 创建模型和损失函数
82     model = CustomSVM(input_dim, num_classes)
83     loss_fn = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction='mean')
84
85     # 使用更小的学习率
86     optimizer = nn.Adam(model.trainable_params(), learning_rate=0.01, weight_decay=0.01) # 添加L2正则化
87
88     def forward_fn(data, label):
89         logits = model(data)
90         loss = loss_fn(logits, label)
91         return loss, logits
92
93     grad_fn = ops.value_and_grad(forward_fn, None, optimizer.parameters)
94
95     def train_step(data, label):
96         (loss, _), grads = grad_fn(data, label)
97         optimizer(grads)
98         return loss
99
100    # 训练循环
101    print("Starting training...")
102    for epoch in range(epochs):
103        total_loss = 0
104        steps = 0
105        for data in dataset.create_dict_iterator():
106            features = data['features']
107            labels = data['labels']
108            loss = train_step(features, labels)
109            total_loss += float(loss)
110            steps += 1
111
112        if (epoch + 1) % 20 == 0: # 每20轮打印一次
113            print(f'Epoch {epoch + 1}, Average Loss: {total_loss / steps}')
114
115    return model
116
117
118 def cross_validation(X, y, k_folds=10):
119     indices = np.arange(len(X))
120     np.random.shuffle(indices)
121     fold_size = len(X) // k_folds
122
123     accuracies = []
124     for i in range(k_folds):
125         print(f"Processing fold {i + 1}/{k_folds}")
126         val_indices = indices[i * fold_size:(i + 1) * fold_size]
127         train_indices = np.concatenate([indices[:i * fold_size],
128                                         indices[(i + 1) * fold_size:]])
129
130         X_train, X_val = X[train_indices], X[val_indices]
131         y_train, y_val = y[train_indices], y[val_indices]
132
133         train_dataset = create_dataset(X_train, y_train)
134         model = train_model(train_dataset, X.shape[1])
135
136         # 评估
137         val_logits = model(ms.Tensor(X_val))
138         val_preds = ops.argmax(val_logits, 1)
139         accuracy = (val_preds.asnumpy() == y_val).mean()
140         accuracies.append(accuracy)
141         print(f"Fold {i + 1} accuracy: {accuracy:.4f}")
142
143     return np.array(accuracies)
144
145

```

```
146 def pca_visualization(X, y):
147     # PCA降维
148     pca = PCA(n_components=2)
149     X_pca = pca.fit_transform(X)
150
151     # 训练模型
152     dataset = create_dataset(X_pca, y)
153     model = train_model(dataset, input_dim=2)
154
155     # 创建网格
156     x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
157     y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
158     xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
159                          np.arange(y_min, y_max, 0.02))
160
161     # 预测
162     grid_points = np.c_[xx.ravel(), yy.ravel()].astype(np.float32)
163     grid_tensor = ms.Tensor(grid_points)
164     logits = model(grid_tensor)
165     Z = ops.argmax(logits, 1).asnumpy()
166     Z = Z.reshape(xx.shape)
167
168     # 可视化
169     plt.figure(figsize=(10, 8))
170     cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
171     cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
172
173     plt.contourf(xx, yy, Z, cmap=cmap_light)
174     plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap=cmap_bold, edgecolors='black')
175
176     plt.xlabel('First Principal Component')
177     plt.ylabel('Second Principal Component')
178     plt.title('SVM Decision Boundary after PCA (MindSpore)')
179
180     plt.show()
181
182     print("PCA explained variance ratio:", pca.explained_variance_ratio_)
183
184
185 def main():
186     print("\nLoading data...")
187     X, y = load_iris_data('iris.txt')
188
189     print("\nPerforming cross-validation...")
190     scores = cross_validation(X, y)
191     print("Cross-validation scores:", scores)
192     print("Average accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
193
194     print("\nPerforming PCA and visualization...")
195     pca_visualization(X, y)
196
197
198 if __name__ == "__main__":
199     main()
```


main2.py

```
1  import numpy as np
2  import pandas as pd
3  from sklearn.preprocessing import StandardScaler
4  from sklearn.decomposition import PCA
5  from sklearn.svm import SVC
6  from sklearn.model_selection import cross_val_score
7  import matplotlib.pyplot as plt
8  from matplotlib.colors import ListedColormap
9
10
11 def load_iris_data(filename):
12     # 使用空格分隔符读取数据，跳过第一行
13     data = []
14     with open(filename, 'r') as file:
15         # 跳过标题行
16         next(file)
17         for line in file:
18             # 移除开头的引号和结尾的换行符
19             if line.startswith('"'):
20                 line = line[1:]
21             # 分割行并处理每个字段
22             row = line.strip().split(' ')
23             # 提取数值和类别
24             values = [float(row[1]), float(row[2]), float(row[3]), float(row[4])]
25             label = row[5].strip('"')
26             data.append(values + [label])
27
28     # 转换为numpy数组
29     data_array = np.array(data)
30     X = data_array[:, :4].astype(np.float32)
31     y = pd.Categorical(data_array[:, 4]).codes.astype(np.int32)
32
33     # 数据标准化
34     scaler = StandardScaler()
35     X_scaled = scaler.fit_transform(X)
36
37     print("Data shape:", X.shape)
38     print("Number of classes:", len(np.unique(y)))
39     print("Sample of features:", X[:5])
40     print("Sample of labels:", y[:5])
41
42     return X_scaled, y
43
44
45 def pca_visualization(X, y):
46     # PCA降维
47     pca = PCA(n_components=2)
48     X_pca = pca.fit_transform(X)
49
50     # 创建和训练SVM分类器
51     svm = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
52     svm.fit(X_pca, y)
53
54     # 创建网格
55     x_min, x_max = X_pca[:, 0].min() - 0.5, X_pca[:, 0].max() + 0.5
56     y_min, y_max = X_pca[:, 1].min() - 0.5, X_pca[:, 1].max() + 0.5
57     xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02),
58                           np.arange(y_min, y_max, 0.02))
59
60     # 预测
61     Z = svm.predict(np.c_[xx.ravel(), yy.ravel()])
62     Z = Z.reshape(xx.shape)
63
64     # 可视化
65     plt.figure(figsize=(10, 8))
66     cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
```

```
67     cmap_bold = ListedColormap(['#FF0000', '#00FF00', '#0000FF'])
68
69     plt.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.3)
70     plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap=cmap_bold, edgecolors='black')
71
72     plt.xlabel('First Principal Component')
73     plt.ylabel('Second Principal Component')
74     plt.title('SVM Decision Boundary after PCA (sklearn)')
75
76     plt.show()
77
78     print("PCA explained variance ratio:", pca.explained_variance_ratio_)
79
80
81 def main():
82     print("\nLoading data...")
83     X, y = load_iris_data('iris.txt')
84
85     # 直接使用sklearn的交叉验证
86     print("\nPerforming cross-validation...")
87     svm = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
88     scores = cross_val_score(svm, X, y, cv=10)
89     print("Cross-validation scores:", scores)
90     print("Average accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
91
92     print("\nPerforming PCA and visualization...")
93     pca_visualization(X, y)
94
95
96 if __name__ == "__main__":
97     main()
```