Университет ИТМО
Кафедра ИПМ

Машинное обучение
Лабораторная работа 4
«Метод опорных векторов»

Выполнил:
Шаймарданов Руслан
группа P4117
Преподаватель:
Жукова Н. А.

Санкт-Петербург
2017

Выбранный датасет: «Statlog (Shuttle) Data Set»

http://archive.ics.uci.edu/ml/datasets/Statlog+%28Shuttle%29

Количество записей: 14500

Описание:

The shuttle dataset contains 9 attributes all of which are numerical. The first one being time. The last column is the class which has been coded as follows :

1	Rad Flow
2	Fpv Close
3	Fpv Open
4	High
5	Bypass
6	Bpv Close
7	Bpv Open

Алгоритм lab4.py

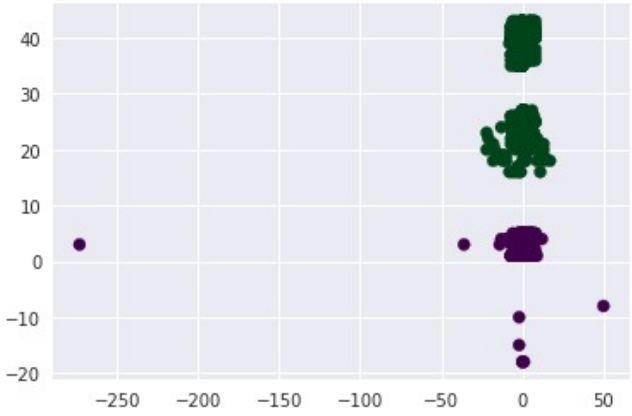
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

dataset = pd.read_csv("shuttle.csv", header=None).values.astype(np.int32, copy=False)
data_train = dataset[0:int(len(dataset) * 0.6)]
data_test = dataset[int(len(dataset) * 0.6) + 1:]
X, y = np.array([]), np.array([])
for row in dataset:
    if (row[-1] == 4 or row[-1] == 5):
        X = np.vstack((X, [row[3], row[6]])) if len(X) != 0 else [row[3], row[6]]
        y=np.append(y, row[-1]-4)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='PRGn_r');

from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X, y)

def plot_svc_decision_function(clf, ax=None):
    if ax is None:
        ax = plt.gca()
    x = np.linspace(plt.xlim()[0], plt.xlim()[1], 30)
    y = np.linspace(plt.ylim()[0], plt.ylim()[1], 30)
    Y, X = np.meshgrid(y, x)
    P = np.zeros_like(X)
    for i, xi in enumerate(x):
        for j, yj in enumerate(y):
            P[i, j] = clf.decision_function(np.array([xi, yj]).reshape(1, -1))
    # plot the margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='PRGn_r')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
            s=200, facecolors='red');

r = np.exp(-(X[:, 0] ** 2 + X[:, 1] ** 2))

from IPython.html.widgets import interact

def plot_3D(elev=30, azim=30):
```

```
    ax = plt.subplot(projection='3d')
    ax.scatter3D(X[:, 0], X[:, 1], r, c=y,cmap='PRGn_r')
    ax.view_init(elev=elev, azim=azim)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('r')

interact(plot_3D, elev=[-90, 90], azip=(-180, 180));

clf = SVC(kernel='rbf')
clf.fit(X, y)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='PRGn_r')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
            facecolors='red');

for kern in ['linear', 'rbf']:
    svc = SVC(kernel=kern)
    svc = svc.fit(data_train[:, :-1], data_train[:, -1])
    svc = svc.score(data_test[:, :-1], data_test[:, -1])
    print(kern, ': ', svc)
```

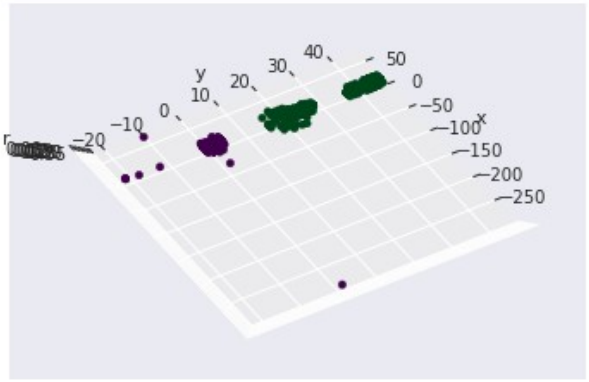
Вывод программы

Представление классов

2d

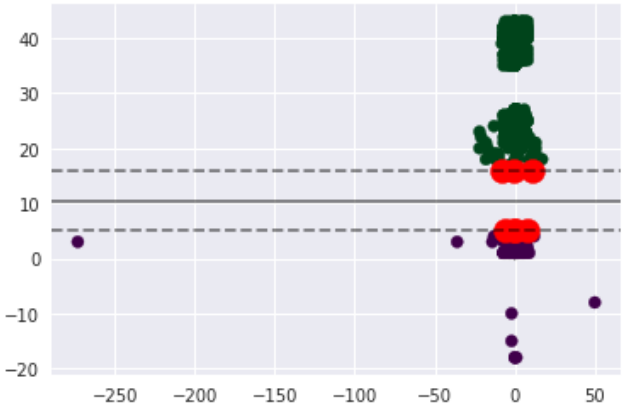


3d

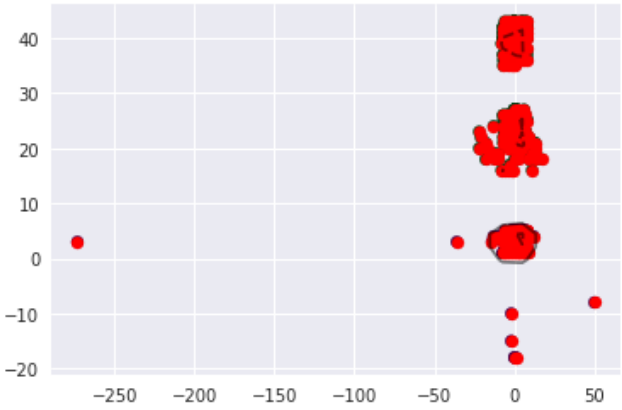


Разбиение классов (с опорными точками)

2d



3d



Результат работы классификатора:

```
linear :  0.982755647525
rbf :  0.931022590102
```

Вывод

В ходе лабораторной изучен метод опорных векторов и получены навыки работы с Jupyter Notebook. В результате работы стало понятно, что Jupyter Notebook хороший инструмент для визуализации, продемонстрировавший, работу метода опорных векторов: SVC рассматривает область вокруг разбивающей классы линии, максимизируя расстояние между классами, а на посторение векторов и, как следствие, результаты классификации, влияют только опорные точки.

Для визуализации были выбраны те же данные, что и для лабораторной работы 3.

По изображению классов становится очевидно, при выборе ядра стоит выбрать линейный вариант. Это же подтверждается результатами сравнения классификации: при разбиении классов с помощью гиперплоскости, результат получился хуже на 0,5.