Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №4

По дисциплине «Машинное обучение»

Выполнил студент гр 33534/5 Донцов А. Д.

Руководитель И. А. Селин

Санкт-Петербург
2019 г.

Постановка задачи

Данные для обучения и тестирования SVM-моделей, которые необходимо построить в приведенных ниже заданиях, хранятся в файлах с именами svmdataI.txt и svmdataItest.txt, где I номер задания.

1. Постройте алгоритм метода опорных векторов типа "C-classification" с параметром C = 1, используя ядро "linear" (LinearSVC или SVC с ядром “linear”). Визуализируйте разбиение пространства признаков на области с помощью полученной модели. Выведите количество полученных опорных векторов, а также ошибки классификации на обучающей и тестовой выборках.

2. Используя алгоритм метода опорных векторов типа "C-classification" с линейным ядром (LinearSVC или SVC с ядром “linear”), добейтесь нулевой ошибки сначала на обучающей выборке, а затем на тестовой, путем изменения параметра C. Выберите оптимальное значение данного параметра и объясните свой выбор. Всегда ли нужно добиваться минимизации ошибки на обучающей выборке?

3. Среди ядер "poly", "rbf" и "sigmoid" выберите оптимальное в плане количества ошибок на тестовой выборке. Попробуйте различные значения параметра degree для полиномиального ядра.

4. Среди ядер "poly", "rbf" и "sigmoid" выберите оптимальное в плане количества ошибок на тестовой выборке.

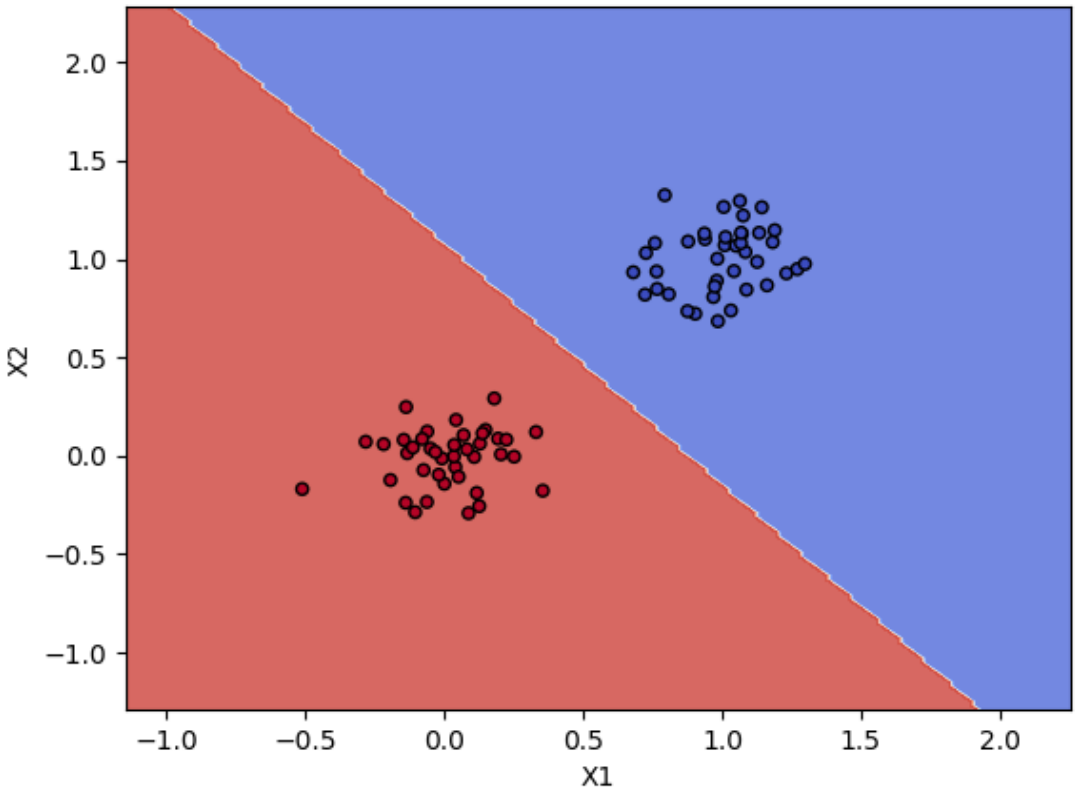
5. Среди ядер "poly ", "rbf " и "sigmoid" выберите оптимальное в плане количества ошибок на тестовой выборке. Изменяя значение параметра gamma, продемонстрируйте эффект переобучения, выполните при этом визуализацию разбиения пространства признаков на области.

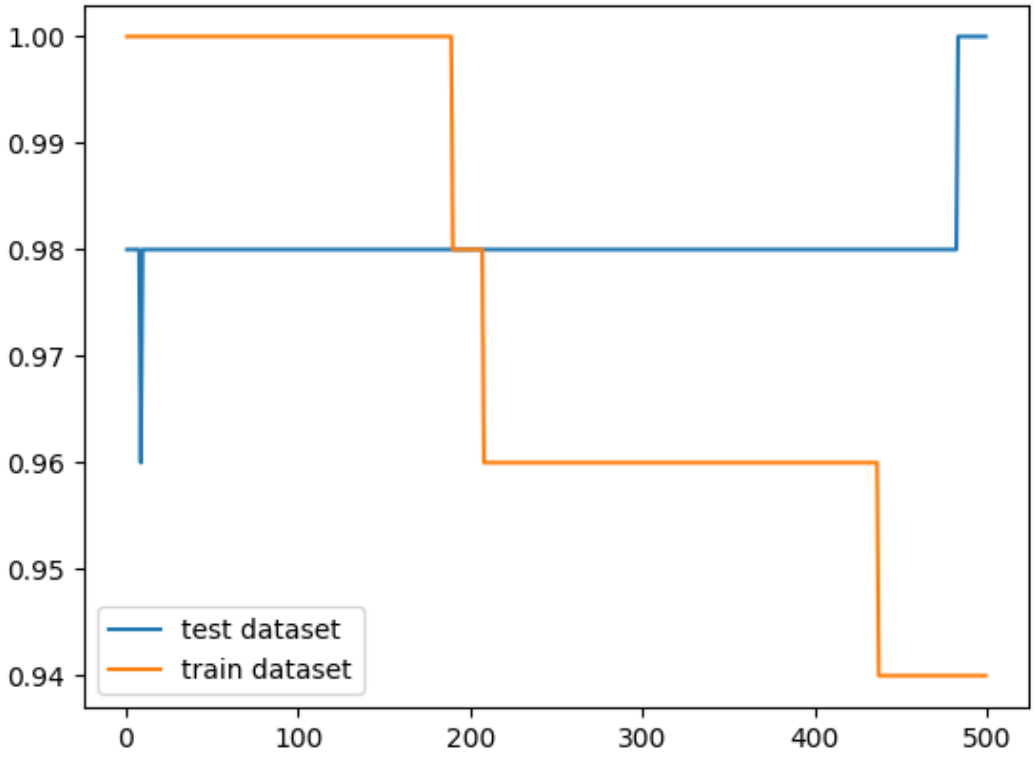
Ход работы

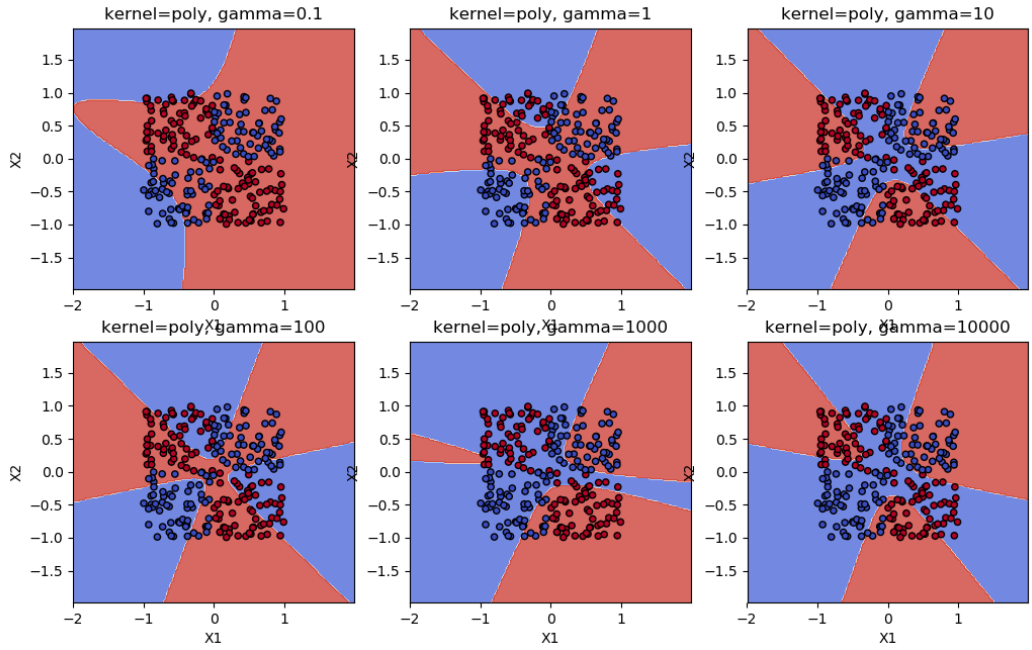
1. Был построен классификтор с параметрами:
   clf = SVC(kernel='linear', C=1.0)
   Для визуализации результатов работы был построен график.
2. Для классификатора с параметрами SVC(kernel='linear', C=i) параметр i изменялся в диапазоне от 1 до 499. Таким образом, нулевая ошибка классификации была получена при 189-и значениях обучающей выборки и 17-и значениях тестовой выборки.
3. Были протестированы различные параметры kernel на обучающей выборке, оптимальным оказался параметр rbf.
4. На тестовой выборке были протестированы различные варианты параметров kernel и degree. Оптимальным снова оказалось значение rbf
5. Для ядер были проведены тесты параметров gamma на значениях gamma = [0.1, 1, 10, 100, 1000, 10000]. Для результатов были построены графики. Также была оценена точность предсказания, были получены следующие результаты:

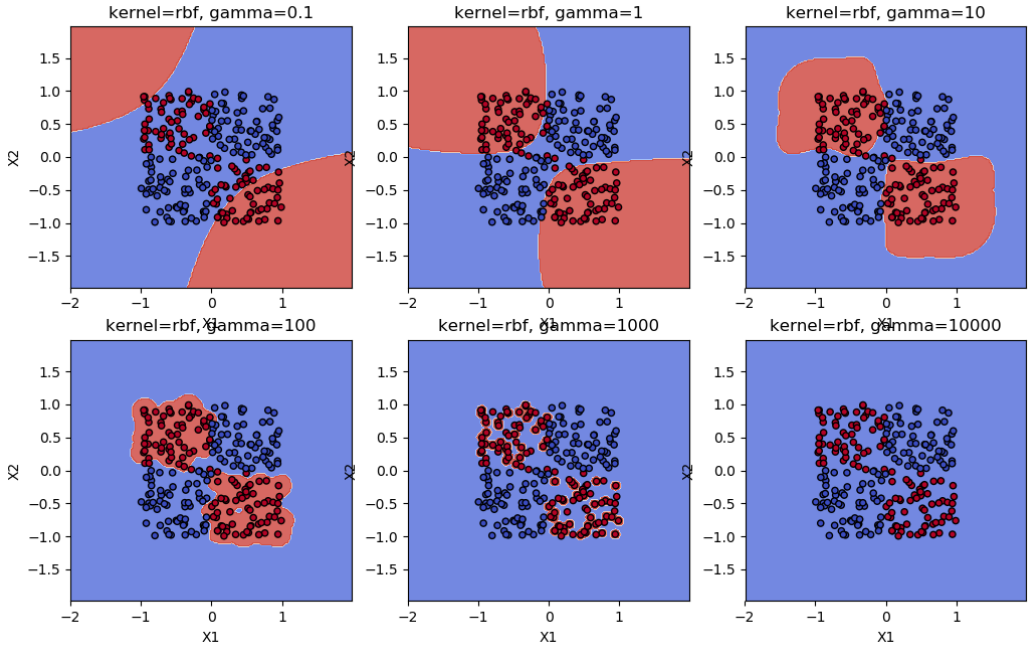
kernel	gamma	accuracy
Poly	0.1	0.53
	1	0.51
	10	0.525
	100	0.516
	1000	0.4916
	10000	0.475
Rbf	0.1	0.625
	1	0.916
	10	0.925
	100	0.9083
	1000	0.675
	10000	0.575
Sigmoid	0.1	0.641
	1	0.583
	10	0.5
	100	0.45
	1000	0.4583
	10000	0.4583

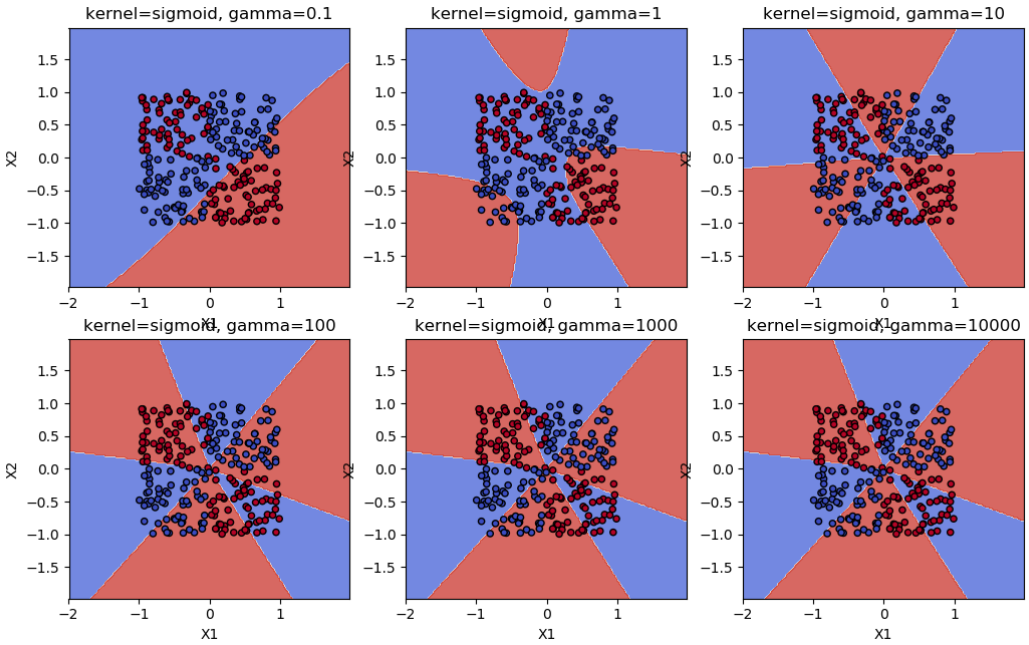
Результаты работы






kernel=poly, gamma=0.1
kernel=poly, gamma=1
kernel=poly, gamma=10
kernel=poly, gamma=100
kernel=poly, gamma=1000
kernel=poly, gamma=10000
X1
X2


kernel=rbf, gamma=0.1
kernel=rbf, gamma=1
kernel=rbf, gamma=10
kernel=rbf, gamma=100
kernel=rbf, gamma=1000
kernel=rbf, gamma=10000
X1
X2


kernel=sigmoid, gamma=0.1
kernel=sigmoid, gamma=1
kernel=sigmoid, gamma=10
kernel=sigmoid, gamma=100
kernel=sigmoid, gamma=1000
kernel=sigmoid, gamma=10000
X1
X2

Вывод

В ходе работы был изучен классификатор SVM.

1. Был построен классификатор, для результатов работы был построен график.
2. Для обучающей выборки нулевая точность достигнута при 189 различных значений параметра C, для 17 различных значений на тестовой выборке. При этом, общая точность предсказания не оказывалась ниже 0.94
3. Наилучшим значением kernel оказалось rbf с точностью предсказания 0.96, точность предсказания не изменилась в заданном диапазоне параметра degree
4. Наилучшим значением kernel оказалось rbf, с точностью предсказания 0.935, точность предсказания не изменилась в заданном диапазоне параметра degree
5. Были протестированы различные значения gamma для различных значений параметра kernel. На выбранных значениях эффект переобучения наиболее заметен при kernel = rbf, gamma >= 100.

Текст программы

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
import pandas as pd
from sklearn import preprocessing, metrics


def make_meshgrid(x, y, h=.02):
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    return xx, yy


def plot_contours(ax, clf, xx, yy, **params):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    out = ax.contourf(xx, yy, Z, **params)
    return out


def make_subplot(ax, clf, x, y, title=None):
    xx, yy = make_meshgrid(x[:, 0], x[:, 1])
    plot_contours(ax, clf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
    ax.scatter(x[:, 0], x[:, 1], c=y, cmap=plt.cm.coolwarm, s=20,
edgecolors='k')
    ax.set_xlabel('X1')
    ax.set_ylabel('X2')
    ax.set_title(title)


def make_subplot2(ax, clf, x1, y1, x2, y2, title=None):
    make_subplot(ax, clf, x1, y1, title)
    ax.scatter(x2[:, 0], x2[:, 1], c=y2, cmap=plt.cm.coolwarm, s=20,
edgecolors='k')


def make_plot1(clf, x, y, title=None):
    fig, sub = plt.subplots(1, 1)
    make_subplot(sub, clf, x, y, title)
    return sub


def make_plot2(clf, x1, y1, x2, y2, title=None):
    sub = make_plot1(clf, x1, y1, title)
    sub.scatter(x2[:, 0], x2[:, 1], c=y2, cmap=plt.cm.coolwarm, s=20,
edgecolors='k')


def getData(filename, y_encoder):
    df1 = pd.read_csv(filename, delim_whitespace=True)
    x = df1[['X1', 'X2']].values
    y = y_encoder.fit_transform(df1['Color'].values)
    return x, y


def svm_point1():
    le = preprocessing.LabelEncoder()
    data_train, res_train = getData('svmdata1.txt', le)
    data_test, res_test = getData('svmdataltest.txt', le)
    clf = SVC(kernel='linear', C=1.0)
```

```
    clf.fit(data_train, res_train)
    make_plot2(clf, data_train, res_train, data_test, res_test)
    print('number of support vectors: ', clf.n_support_)
    print('train accuracy: ', metrics.accuracy_score(res_train,
clf.predict(data_train)))
    print('test accuracy: ', metrics.accuracy_score(res_test,
clf.predict(data_test)))


def svm_point2():
    le = preprocessing.LabelEncoder()
    data_train, res_train = getData('svmdata2.txt', le)
    data_test, res_test = getData('svmdata2test.txt', le)
    test_count = 0
    train_count = 0
    for i in range(1, 500):
        clf = SVC(kernel='linear', C=i)
        clf.fit(data_train, res_train)
        if metrics.accuracy_score(res_train, clf.predict(data_train)) >= 1.0:
            train_count +=1
            print(i)
        if metrics.accuracy_score(res_test, clf.predict(data_test)) >= 1.0:
            test_count +=1
            print(i)
    print('test dataset {0}'.format(test_count))
    print('train dataset {0}'.format(train_count))


def svm_point3():
    le = preprocessing.LabelEncoder()
    data_train, res_train = getData('svmdata3.txt', le)
    data_test, res_test = getData('svmdata3test.txt', le)
    for i in ['poly', 'rbf', 'sigmoid']:
            for j in range(1, 5):
                clf = SVC(kernel=i, C=1.0, gamma='auto', degree=j)
                clf.fit(data_train, res_train)
                print('current accuracy: {0}; current kernel: {1}; current
degree: {2}'.format(metrics.accuracy_score(res_train,

clf.predict(data_train)), i, j))

def svm_point4():
    le = preprocessing.LabelEncoder()
    data_train, res_train = getData('svmdata4.txt', le)
    data_test, res_test = getData('svmdata4test.txt', le)
    for i in ['poly', 'rbf', 'sigmoid']:
            for j in range(1, 5):
                clf = SVC(kernel=i, C=1.0, gamma='auto', degree=j)
                clf.fit(data_train, res_train)
                print('current accuracy: {0}; current kernel: {1}; current
degree: {2}'.format(metrics.accuracy_score(res_test,

clf.predict(data_test)), i, j))

def svm_point5():
    le = preprocessing.LabelEncoder()
    data_train, res_train = getData('svmdata5.txt', le)
    data_test, res_test = getData('svmdata5test.txt', le)

    gammas = [0.1, 1, 10, 100, 1000, 10000]

    for i in ['poly', 'rbf', 'sigmoid']:
        fig, sub = plt.subplots(2, 3)
        for d, j in enumerate(gammas):
```

```
            clf = SVC(kernel=i, gamma=j)
            clf.fit(data_train, res_train)
            print('current accuracy: {0}; current kernel: {1}; current gamma:
{2}'.format(
                metrics.accuracy_score(res_test, clf.predict(data_test)), i,
j))
            ax = sub.flatten()[d]
            make_subplot2(ax, clf, data_train, res_train, data_test,
res_test, 'kernel={0}, gamma={1}'.format(i, j))

print('SVM 1\n')
#svm_point1()
print('SVM 2\n')
#svm_point2()
print('SVM 3\n')
#svm_point3()
print('SVM 4\n')
svm_point4()
print('SVM 5\n')
#svm_point5()
plt.show()
```