Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

Отчет по лабораторной работе:

Лабораторная работа №5 “Метод опорных векторов”

Выполнил: Карп Александр Игоревич

магистрант кафедры информатики

группа №858641

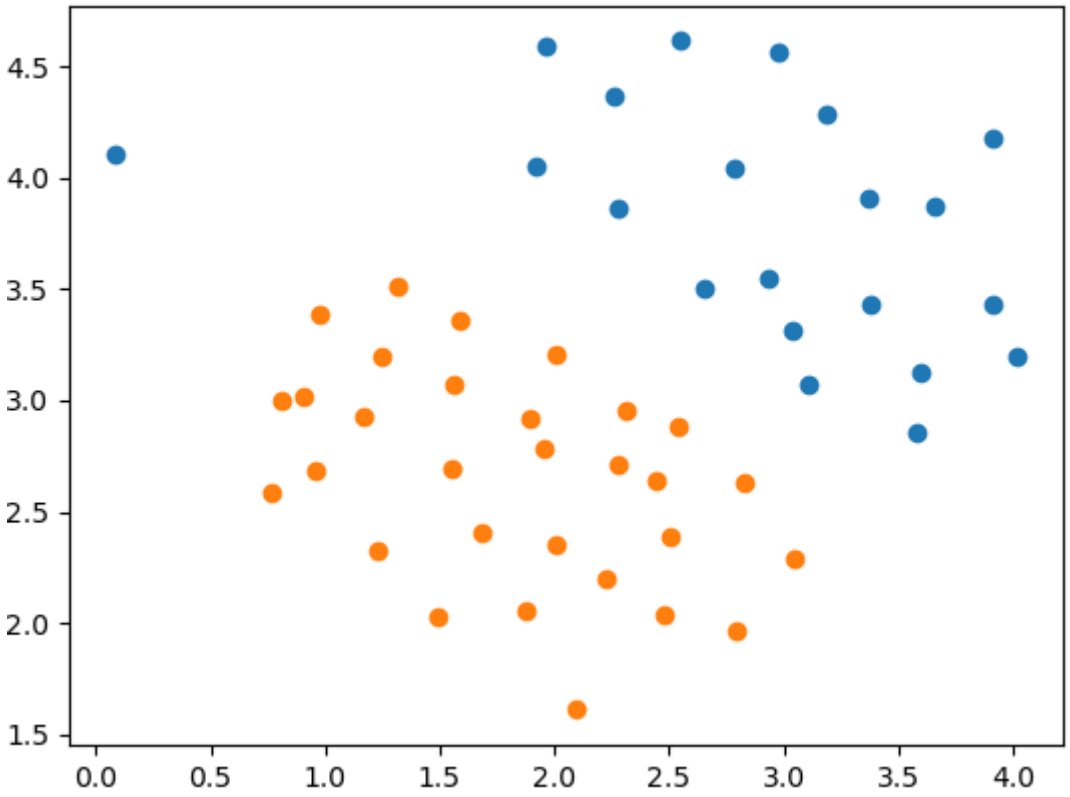
Минск 2019

1. Загрузите данные ex5data1.mat из файла.

```
#1
data = sio.loadmat('ex5data1.mat')
X = data.get('X')
y = data.get('y')
```

2. Постройте график для загруженного набора данных: по осям - переменные X1, X2, а точки, принадлежащие различным классам должны быть обозначены различными маркерами.

```
m, n = X.shape[0], X.shape[1]
pos, neg = (y == 1).reshape(m, 1).flatten(), (y == 0).reshape(m, 1).flatten()
plt.scatter(X[pos, 0], X[pos, 1])
plt.scatter(X[neg, 0], X[neg, 1])
plt.show()
```

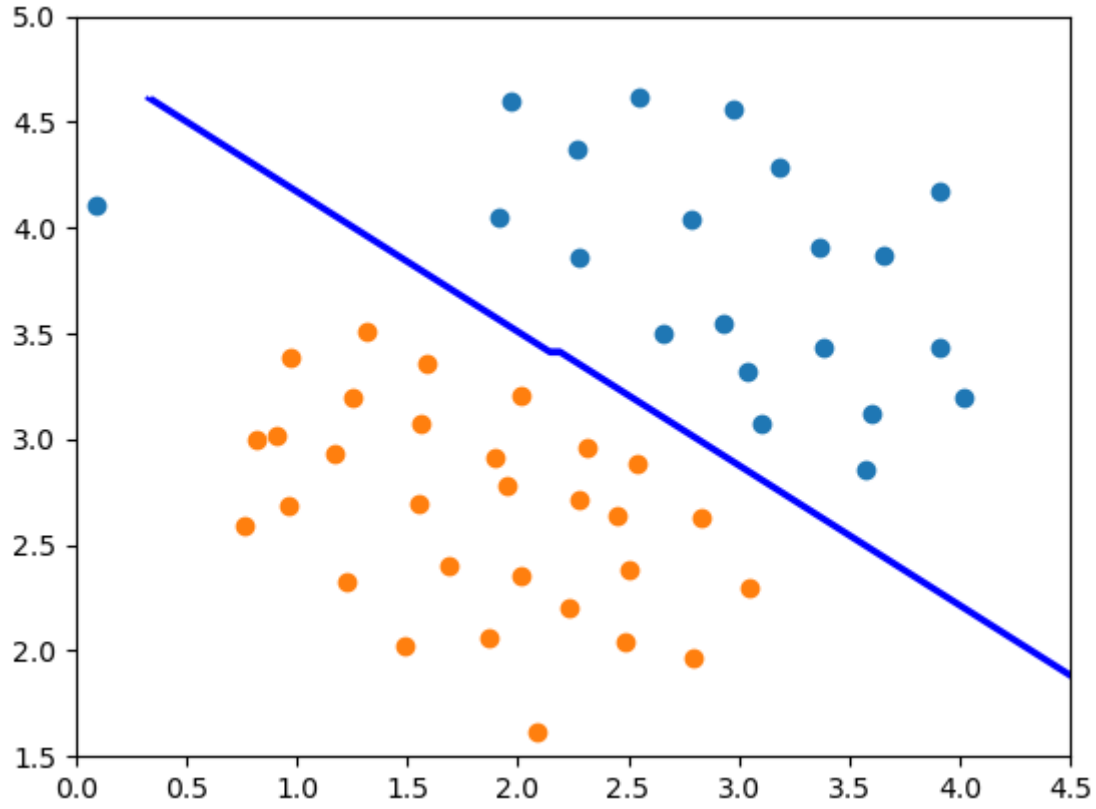


3. Обучите классификатор с помощью библиотечной реализации SVM с линейным ядром на данном наборе.

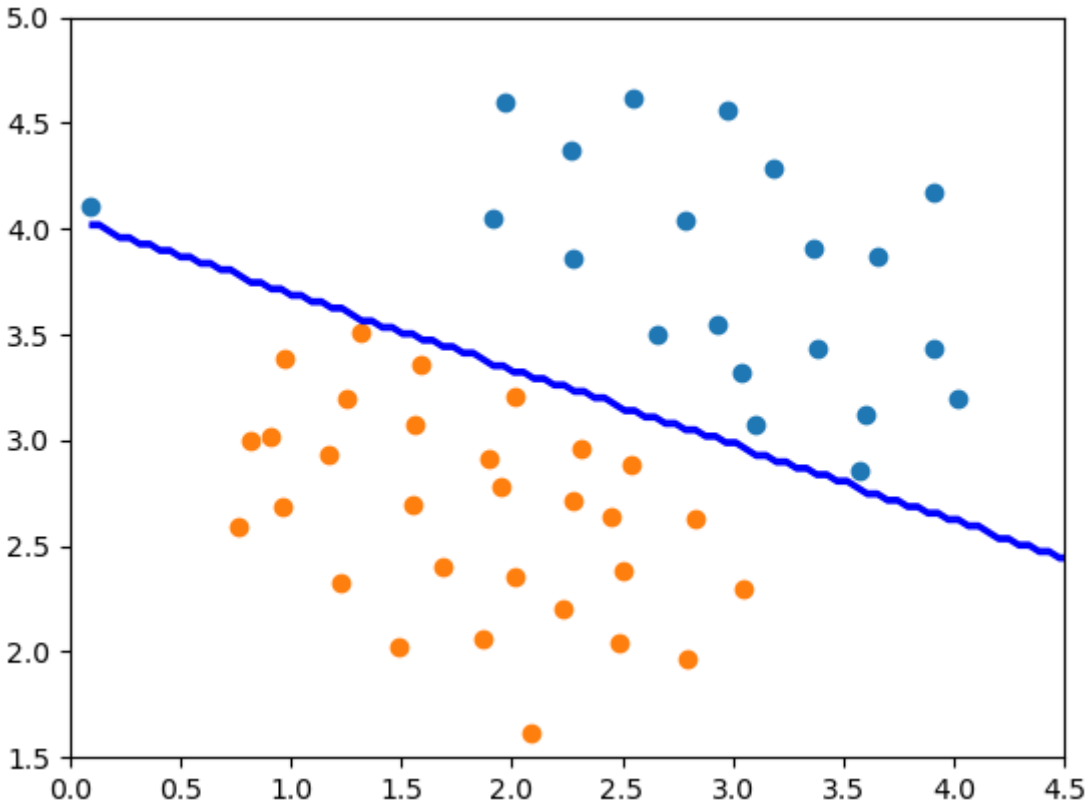
```
classifier = SVC(kernel="linear")
classifier.fit(X, y[:, 0])
```

4. Постройте разделяющую прямую для классификаторов с различными параметрами C = 1, C = 100 (совместно с графиком из пункта 2). Объясните различия в полученных прямых?

C=1


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

C=100


5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

```
#4
# plotting the decision boundary
X_1,X_2 =
np.meshgrid(np.linspace(X[:,0].min(),X[:,1].max(),num=100),np.linspace(X[:,1]
.min(),X[:,1].max(),num=100))
plt.contour(X_1,X_2,classifier.predict(np.array([X_1.ravel(),X_2.ravel()]).T)
.reshape(X_1.shape),1,colors="b")
pos, neg = (y == 1).reshape(m, 1).flatten(), (y == 0).reshape(m, 1).flatten()
plt.scatter(X[pos, 0], X[pos, 1])
plt.scatter(X[neg, 0], X[neg, 1])
plt.xlim(0,4.5)
plt.ylim(1.5,5)
plt.show()


classifier = SVC(kernel="linear", C=100)
classifier.fit(X, y[:, 0])
X_1,X_2 =
np.meshgrid(np.linspace(X[:,0].min(),X[:,1].max(),num=100),np.linspace(X[:,1]
.min(),X[:,1].max(),num=100))
plt.contour(X_1,X_2,classifier.predict(np.array([X_1.ravel(),X_2.ravel()]).T)
.reshape(X_1.shape),1,colors="b")
pos, neg = (y == 1).reshape(m, 1).flatten(), (y == 0).reshape(m, 1).flatten()
plt.scatter(X[pos, 0], X[pos, 1])
plt.scatter(X[neg, 0], X[neg, 1])
plt.xlim(0,4.5)
plt.ylim(1.5,5)
plt.show()
```

Параметр C отвечает за то, как сильно мы хотим избежать неправильной классификации. Как видно из графика, при C=1 нас есть одна ошибка классификации, но при этом расстояние между разделяющей прямой и классами максимально. При C=100 приоритет стоит на правильно классификации, следовательно, выброс из второго класса классифицирован правильно, но расстояние между разделяющей прямой и классами уменьшилось.

5. Реализуйте функцию вычисления Гауссового ядра для алгоритма SVM.
6. Загрузите данные **ex5data2.mat** из файла.

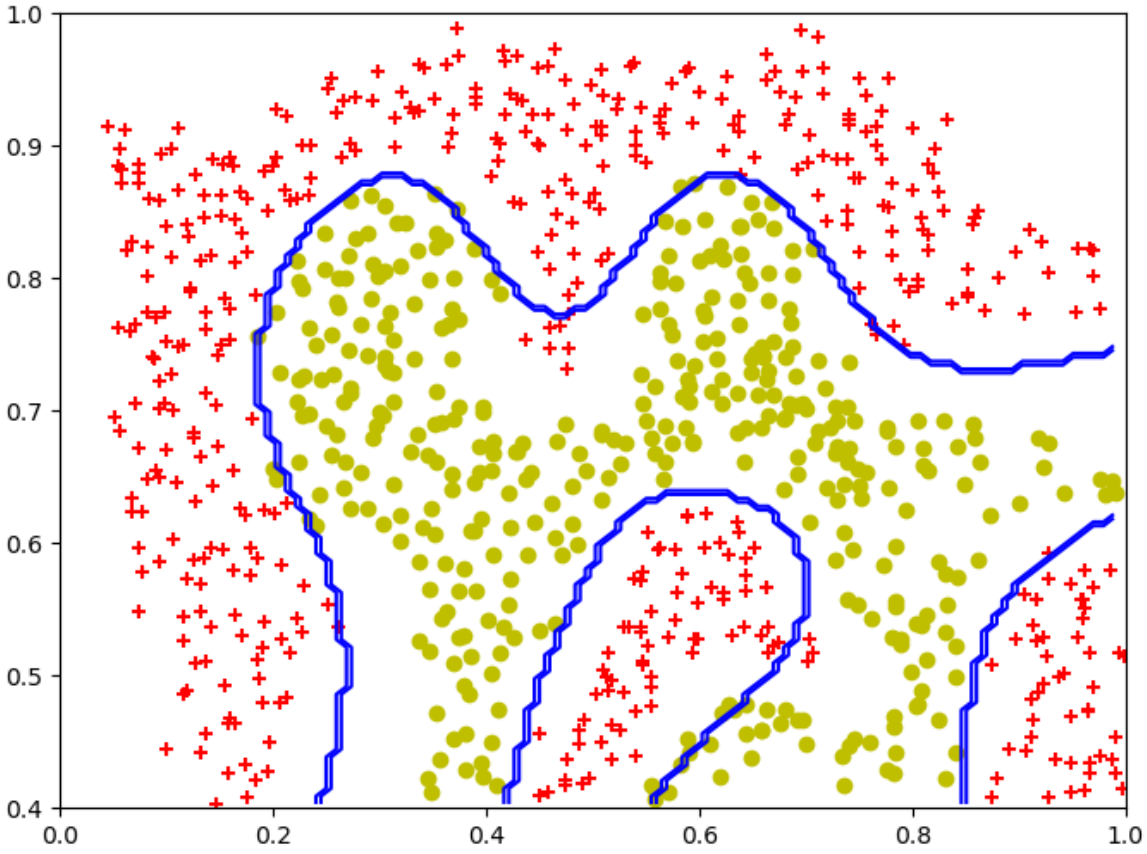
```
data = sio.loadmat('ex5data2.mat')
X = data.get('X')
y = data.get('y')
```

7. Обработайте данные с помощью функции Гауссового ядра.
8. Обучите классификатор SVM.

```
m2,n2 = X.shape[0],X.shape[1]
pos2,neg2= (y==1).reshape(m2,1), (y==0).reshape(m2,1)
classifier = SVC(kernel="rbf",gamma=30)
classifier.fit(X,y.ravel())
```

9. Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4).

```
plt.figure(figsize=(8,6))
plt.scatter(X[pos2[:,0],0],X[pos2[:,0],1],c="r",marker="+")
plt.scatter(X[neg2[:,0],0],X[neg2[:,0],1],c="y",marker="o")
# plotting the decision boundary
X_1,X_2 =
np.meshgrid(np.linspace(X[:,0].min(),X[:,1].max(),num=100),np.linspace(X[:,1]
.min(),X[:,1].max(),num=100))
plt.contour(X_1,X_2,classifier.predict(np.array([X_1.ravel(),X_2.ravel()]).T)
.reshape(X_1.shape),1,colors="b")
plt.xlim(0,1)
plt.ylim(0.4,1)
plt.show()
```



10. Загрузите данные ex5data3.mat из файла.

```
data = sio.loadmat('ex5data3.mat')
X = data["X"]
y = data["y"]
```

```
Xval = data["Xval"]
yval = data["yval"]
```

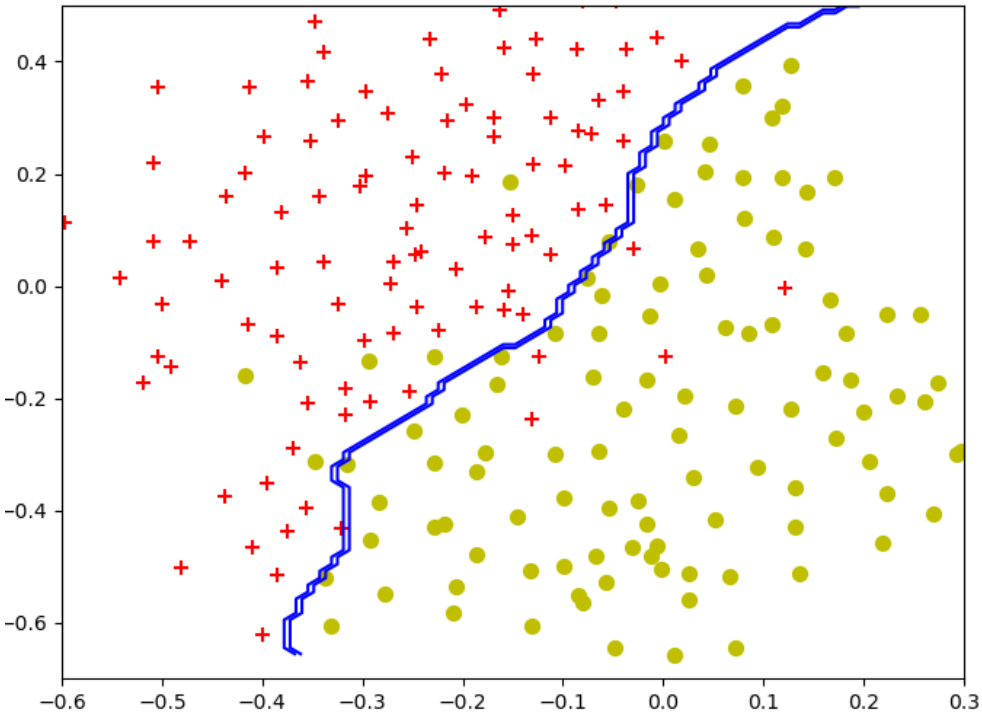
11. Вычислите параметры классификатора SVM на обучающей выборке, а также подберите параметры C и σ2 на валидационной выборке.

```
vals = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]
C, gamma = dataset3Params(X, y.ravel(), Xval, yval.ravel(),vals)
def dataset3Params(X, y, Xval, yval,vals):
    """
    Returns your choice of C and sigma. You should complete this function to
return the optimal C and
    sigma based on a cross-validation set.
    """
    acc = 0
    best_c=0
    best_gamma=0
    for i in vals:
        C= i
        for j in vals:
            gamma = 1/j
            classifier = SVC(C=C,gamma=gamma)
            classifier.fit(X,y)
            prediction = classifier.predict(Xval)
            score = classifier.score(Xval,yval)
            if score>acc:
                acc =score
                best_c =C
                best_gamma=gamma
    return best_c, best_gamma
```

$Best\ C$: 0.3

$Best\ gamma$: 100.0

12. Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4).



Выводы

Метод опорных векторов один из наиболее популярных методов обучения, который применяется для решения задач классификации и регрессии. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки наиболее оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

Преимущества и недостатки SVM:

- это наиболее быстрый метод нахождения решающих функций;
- метод сводится к решению задачи квадратичного программирования в выпуклой области, которая всегда имеет единственное решение;
- метод находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию;
- метод чувствителен к шумам и стандартизации данных;
- не существует общего подхода к автоматическому выбору ядра (и построению спрямляющего подпространства в целом) в случае линейной неразделимости классов.