Лабораторная работа №5: «Метод опорных векторов»

Набор данных **ex5data1.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит три переменные X_1 и X_2 (независимые переменные) и y (метка класса). Данные являются линейно разделимыми.

Набор данных **ex5data2.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит три переменные X_1 и X_2 (независимые переменные) и y (метка класса). Данные являются нелинейно разделимыми.

Набор данных **ex5data3.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит три переменные X_1 и X_2 (независимые переменные) и y (метка класса). Данные разделены на две выборки: обучающая выборка (X, y), по которой определяются параметры модели; валидационная выборка (Xval, yval), на которой настраивается коэффициент регуляризации и параметры Гауссового ядра.

Набор данных **spamTrain.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две переменные X - вектор, кодирующий отсутствие (0) или присутствие (1) слова из словаря vocab.txt в письме, и y - метка класса: 0 - не спам, 1 - спам. Набор используется для обучения классификатора.

Набор данных **spamTest.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две переменные Xtest - вектор, кодирующий отсутствие (0) или присутствие (1) слова из словаря vocab.txt в письме, и ytest - метка класса: 0 - не спам, 1 - спам. Набор используется для проверки качества классификатора.

Задание.

1. Загрузите данные **ex5data1.mat** из файла.
2. Постройте график для загруженного набора данных: по осям - переменные X_1, X_2, а точки, принадлежащие различным классам должны быть обозначены различными маркерами.
3. Обучите классификатор с помощью библиотечной реализации SVM с линейным ядром на данном наборе.
4. Постройте разделяющую прямую для классификаторов с различными параметрами C = 1, C = 100 (совместно с графиком из пункта 2). Объясните различия в полученных прямых?
5. Реализуйте функцию вычисления Гауссового ядра для алгоритма SVM.
6. Загрузите данные **ex5data2.mat** из файла.
7. Обработайте данные с помощью функции Гауссового ядра.

8. Обучите классификатор SVM.
9. Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4).
10. Загрузите данные **ex5data3.mat** из файла.
11. Вычислите параметры классификатора SVM на обучающей выборке, а также подберите параметры C и σ^2 на валидационной выборке.
12. Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4).
13. Загрузите данные **spamTrain.mat** из файла.
14. Обучите классификатор SVM.
15. Загрузите данные **spamTest.mat** из файла.
16. Подберите параметры C и σ^2.
17. Реализуйте функцию предобработки текста письма, включающую в себя:

a. перевод в нижний регистр;
b. удаление HTML тэгов;
c. замена URL на одно слово (например, “httpaddr”);
d. замена email-адресов на одно слово (например, “emailaddr”);
e. замена чисел на одно слово (например, “number”);
f. замена знаков доллара ($) на слово “dollar”;
g. замена форм слов на исходное слово (например, слова “discount”, “discounts”, “discounted”, “discounting” должны быть заменены на слово “discount”). Такой подход называется stemming;
h. остальные символы должны быть удалены и заменены на пробелы, т.е. в результате получится текст, состоящий из слов, разделенных пробелами.

18. Загрузите коды слов из словаря **vocab.txt**.
19. Реализуйте функцию замены слов в тексте письма после предобработки на их соответствующие коды.
20. Реализуйте функцию преобразования текста письма в вектор признаков (в таком же формате как в файлах **spamTrain.mat** и **spamTest.mat**).
21. Проверьте работу классификатора на письмах из файлов **emailSample1.txt, emailSample2.txt, spamSample1.txt** и **spamSample2.txt**.
22. Также можете проверить его работу на собственных примерах.
23. Создайте свой набор данных из оригинального корпуса текстов - http://spamassassin.apache.org/old/publiccorpus/.
24. Постройте собственный словарь.
25. Как изменилось качество классификации? Почему?
26. Ответы на вопросы представьте в виде отчета.

Реализация:

In[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pnd
```

1.1 Загрузите набор данных ex5data1.mat из файла.

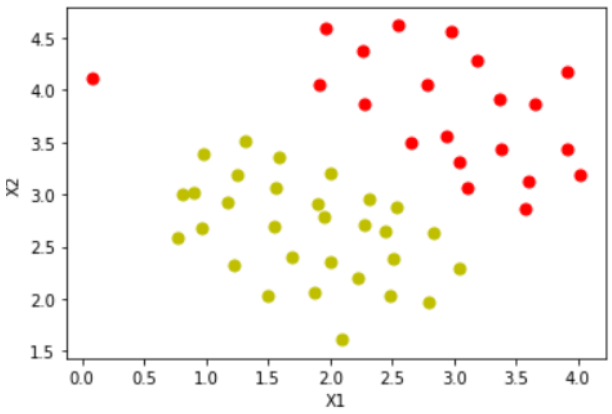
In[2]:

```
mat = loadmat("data/ex5data1.mat")
X = mat["X"]
y = mat["y"]
y = y.reshape(y.shape[0])
```

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

In[3]:

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

fig, ax = plt.subplots()
ax.scatter(X[pos[:,0],0],X[pos[:,0],1],c="r", s=50)
ax.scatter(X[neg[:,0],0],X[neg[:,0],1],c="y", s=50)
ax.set_xlabel('X1')
ax.set_ylabel('X2')
plt.show()
```



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

In[4]:

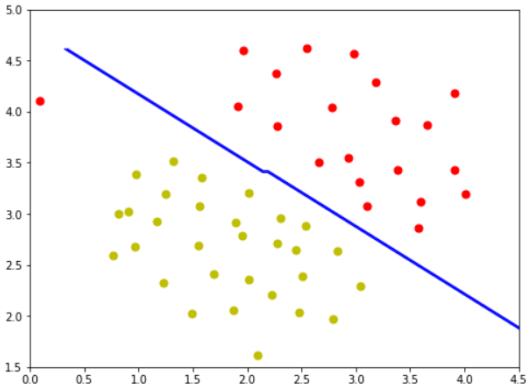
```
from sklearn.svm import SVC
classifier = SVC(kernel="linear", C=1)
classifier.fit(X,np.ravel(y))
```

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

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

In[5]:

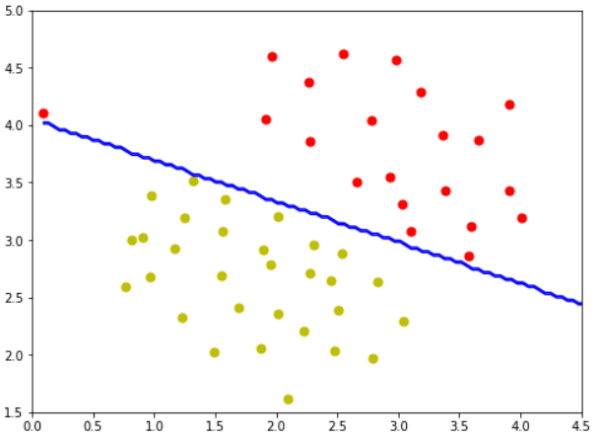
```
def plot_decision_line(classifier):
    plt.figure(figsize=(8,6))
    plt.scatter(X[pos[:,0],0], X[pos[:,0],1], c="r", s=50)
    plt.scatter(X[neg[:,0],0], X[neg[:,0],1], c="y", s=50)

    # plotting the decision boundary
    X_1,X_2 = np.meshgrid(np.linspace(X[:,0].min(),X[:,1].max(),num=100),np.lin-
space(X[:,1].min(),X[:,1].max(),num=100))
    plt.contour(X_1,X_2, classifier.predict(np.ar-
ray([X_1.ravel(),X_2.ravel()]).T).reshape(X_1.shape),1,colors="b")
    plt.xlim(0,4.5)
    plt.ylim(1.5,5)

plot_decision_line(classifier)
```



In[6]:

```
classifier = SVC(kernel="linear", C=100)
classifier.fit(X,np.ravel(y))
```

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```



При C = 100 мы видим, что модель реагирует на один пример, который отклонился от всех остальных. Параметр C это штраф за ошибку и в sklearn значение по умолчанию для C равно 1. Таким образом, при C = 100 наша модель скорее всего переобучается.

1.5 Реализуйте функцию вычисления Гауссового ядра для алгоритма SVM

In[7]:

```
def gaussian(x, l, sigma):
    degree = ((x - l)**2).sum(axis=1)
    return np.e ** (-degree) / (2 * sigma**2)
```

1.6. Загрузите данные ex5data2.mat из файла.

In[8]:

```
mat2 = loadmat("data/ex5data2.mat")
X = mat2["X"]
y = mat2["y"]
y = y.reshape(y.shape[0])
```

1.7 Обработайте данные с помощью функции Гауссового ядра

In[9]:

```
X_gaussian = np.array([gaussian(X, l, 1) for l in X])
```

1.8 Обучите классификатор SVM

In[10]:

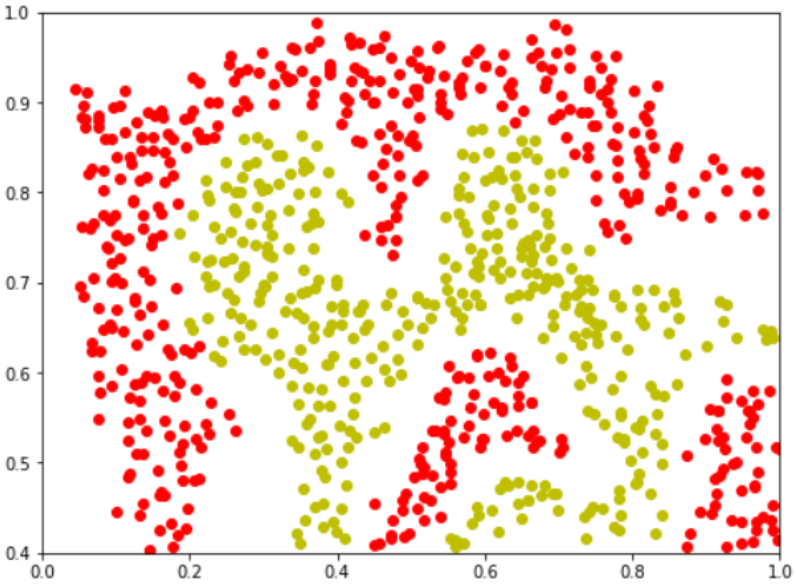
```
clf_gaussian = SVC(kernel='rbf', C=1, gamma=30)
clf_gaussian.fit(X, y)
```

```
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=30, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

1.9 Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4)

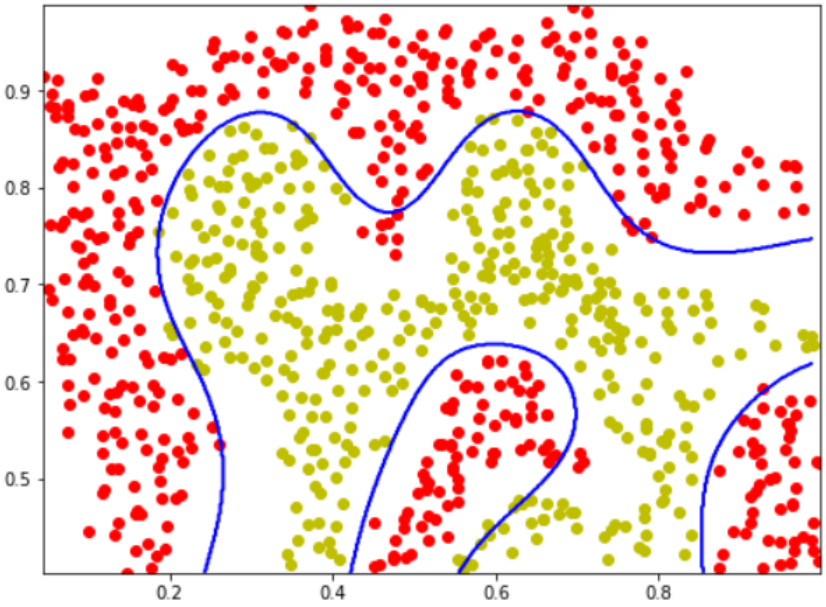
In[11]

```
m, n = X.shape[0], X.shape[1]
pos, neg = (y==1).reshape(m,1), (y==0).reshape(m,1)
plt.figure(figsize=(8,6))
plt.scatter(X[pos[:,0],0], X[pos[:,0],1], c="r")
plt.scatter(X[neg[:,0],0], X[neg[:,0],1], c="y")
plt.xlim(0,1)
plt.ylim(0.4,1)
plt.show()
```



In[12]:

```
def plot_decision_line(classifier, X, y):
    m, n = X.shape[0], X.shape[1]
    pos, neg = (y==1).reshape(m, 1), (y==0).reshape(m, 1)

    plt.figure(figsize=(8,6))
    plt.scatter(X[pos[:,0],0], X[pos[:,0],1], c="r")
    plt.scatter(X[neg[:,0],0], X[neg[:,0],1], c="y")


    # plotting the decision boundary
    X_5, X_6 = np.meshgrid(np.linspace(X[:,0].min(), X[:,1].max(), num=500),
                           np.linspace(X[:,1].min(), X[:,1].max(), num=500))
    plt.contour(X_5, X_6, classifier.predict(np.array([X_5.ravel(), X_6.ravel()])
.T).reshape(X_5.shape), 1, colors="b")
    plt.xlim(X[:, 0].min(), X[:, 0].max())
    plt.ylim(X[:, 1].min(), X[:, 1].max())

plot_decision_line(clf_gaussian, X, y)
```



1.10 Загрузите данные ex5data3.mat из файла

In[13]:

```
mat3 = loadmat("data/ex5data3.mat")
X = mat3["X"]
y = mat3["y"]
y = y.reshape(y.shape[0])

Xval = mat3["Xval"]
yval = mat3["yval"]
yval = yval.reshape(yval.shape[0])
```

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

In[14]:

```
def calculate_best_params(X, y, Xval, yval, C_list, gamma_list):
    best_score = -np.inf
    best_params = None
    for C in C_list:
        for gamma in gamma_list:
            s = SVC(kernel='rbf', C=C, gamma=gamma)
            s.fit(X, y)
            score = s.score(Xval, yval)
            if score > best_score:
                best_score = score
                best_params = (C, gamma)
    return best_params

C, gamma = calculate_best_params(X, y, Xval, yval,
                                 C_list=np.logspace(-
1, 3, 100), gamma_list=np.linspace(0.0001, 10, 100))

classifier = SVC(C = C, gamma = gamma)
classifier.fit(X, y.ravel())
```

1.12 Визуализируйте данные вместе с разделяющей кривой (аналогично пункту 4)

In[15]:

```
plot_decision_line(classifier, X, y)
```