Министерство образования Республики Беларусь

Учреждение образования

“Белорусский государственный университет информатики и радиоэлектроники”

Факультет компьютерных систем и сетей

Специальность “Обработка больших объемов информации”

Лабораторная работа №5

**Метод опорных векторов**

Выполнил:

магистрант гр. 858641 Кальман В.А.

Проверил:

Стержанов М. В.

Минск 2019

**Данные.**

Набор данных ex5data1.mat представляет собой файл формата \*.mat (т.е. сохраненного из Matlab). Набор содержит три переменные X1 и X2 (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 X1 и X2 (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 X1 и X2 (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.

1. **Загрузите данные ex5data1.mat из файла.**

**import** **re**

**import** **email**

**import** **os**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**from** **scipy** **import** io

**from** **sklearn.svm** **import** SVC

**from** **nltk.stem** **import** PorterStemmer

**import** **matplotlib.pyplot** **as** **plt**

%matplotlib inline

ex5data1 = io.loadmat('Data/Lab 5/ex5data1.mat')

ex5data1\_df = pd.DataFrame(ex5data1['X'], columns=['x1', 'x2'])

ex5data1\_df['y'] = ex5data1['y']

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

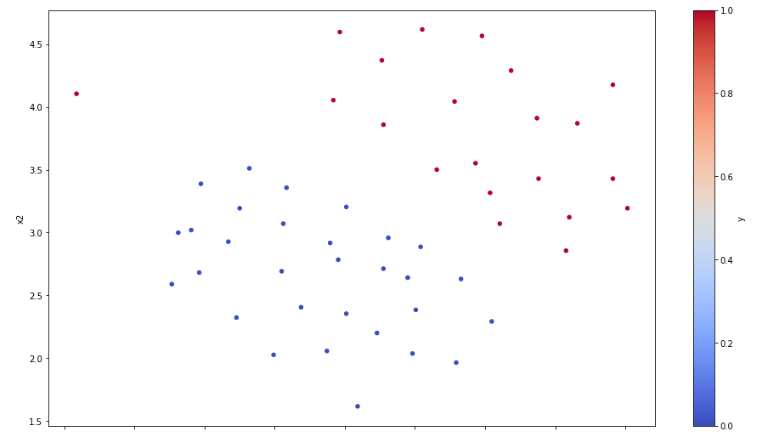
ex5data1\_df.plot(kind='scatter', x='x1', y='x2', c='y', cmap='coolwarm', figsize=(16,9))

Рисунок 1 – исходные данные

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

svc\_c1 = SVC(kernel='linear', C=1)

svc\_c1.fit(ex5data1\_df[['x1', 'x2']], ex5data1\_df['y'])

svc\_c100 = SVC(kernel='linear', C=100)

svc\_c100.fit(ex5data1\_df[['x1', 'x2']], ex5data1\_df['y'])

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

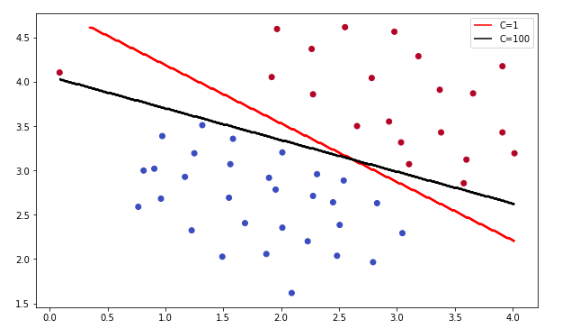


Рисунок 2 – исходные данные и график разделяющей прямой при С=1 и С=100

SVM cost function преследует две optimization objectives: увеличение длинны margin и уменьшение theta. Параметр C контролирует приоритет этих целей.

В первом случае была допущена misclasification в пользу простоты theta (вторая optimization objective).

Во втором случае с параметром C = 100 цена misclasification возрасла в 100 раз. Соответственно была выбранна такая decision boundary, которая классифицировала все элементы верно, при этом margin уменьшилась.

**5. Реализуйте функцию вычисления Гауссового ядра для алгоритма SVM.**

def gauss\_kernel\_carried(sigma):

def gauss\_kernel(x1, x2):

sigma\_squared = np.power(sigma, 2)

matrix = np.power(x1-x2, 2)

return np.exp(-np.sum(matrix)/(2\*sigma\_squared))

return gauss\_kernel

x1 = np.array([1, 2, 1])

x2 = np.array([0, 4, -1])

sigma = 2

sim = gauss\_kernel\_carried(sigma)(x1, x2)

print('Gaussian Kernel between x1 = [1, 2, 1], x2 = [0, 4, -1], sigma = %0.2f:'

'\n\t%f\n(for sigma = 2, this value should be about 0.324652)\n' % (sigma, sim))

**6. Загрузите данные ex5data2.mat из файла.**

ex5data2 = io.loadmat('Data/Lab 5/ex5data2.mat')

ex5data2\_df = pd.DataFrame(ex5data2['X'], columns=['x1', 'x2'])

ex5data2\_df['y'] = ex5data2['y']

**7. Обработайте данные с помощью функции Гауссового ядра.**

sigma = 0.1

kernel = gauss\_kernel\_carried(sigma)

**8. Обучите классификатор SVM.**

gamma = np.power(sigma, -2.)

classifier3 = SVC(C=1, kernel='rbf', gamma=gamma)

classifier3.fit(X, y.flatten())

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

xx1, xx2 = get\_meshgrid(ex5data2\_df['x1'], ex5data2\_df['x2'], 100)

preds = np.zeros(xx1.shape)

**for** i **in** range(xx1.shape[1]):

new\_X = np.column\_stack((xx1[:, i], xx2[:, i]))

preds[:, i] = svc2.predict(gauss\_kernel\_preprocess(new\_X, ex5data2['X']))

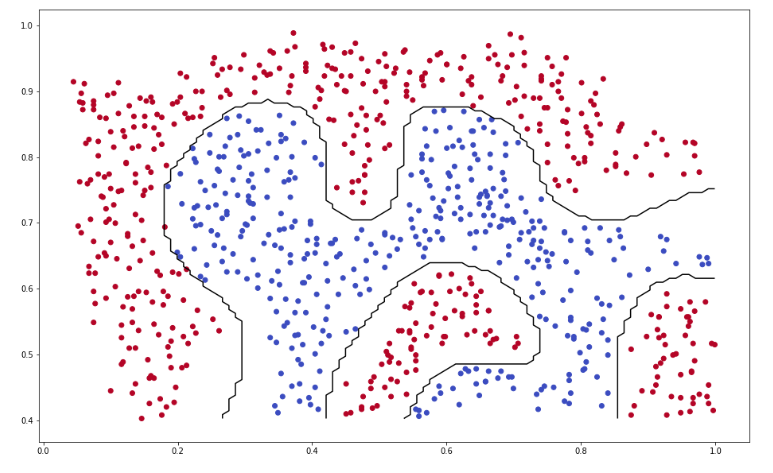
plt.figure(figsize=(16, 10))

plt.scatter(x=ex5data2\_df['x1'], y=ex5data2\_df['x2'],

c=ex5data2\_df['y'],

cmap='coolwarm')

plt.contour(xx1, xx2, preds, colors='black', levels=[0,0])

 Рисунок 5 – визуализация разделяющей кривой

**10. Загрузите данные ex5data3.mat из файла.**

ex5data3 = io.loadmat('Data/Lab 5/ex5data3.mat')

X, Xval, y, yval = (ex5data3['X'], ex5data3['Xval'],

ex5data3['y'].flatten(),

ex5data3['yval'].flatten())

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

def dataset3Params(X, y, Xval, yval, values):

C = values[0]

sigma = values[0]

result\_score = 0

for i in values:

for j in values:

gamma = 1 / j

classifier = SVC(C=i, gamma=gamma, kernel='rbf')

classifier.fit(X, y)

prediction = classifier.predict(Xval)

score = classifier.score(Xval, yval)

print("i: ", i, "j: ", j, "score: ", score)

if score > result\_score:

result\_score = score

C = i

sigma = gamma

return C, sigma

vals = [0.01, 0.03, 0.1, 0.3, 0.5, 1, 3, 10, 30, 50, 100]

C, gamma = dataset3Params(X, y.flatten(), Xval, yval.flatten(), vals)

print("C: ", C, ", gamma: ", gamma)

classifier4 = SVC(C=C, gamma=gamma, kernel='rbf')

classifier4.fit(X, y.flatten())

Результат выполнения:

('C: ', 0.3, ', gamma: ', 100.0)

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

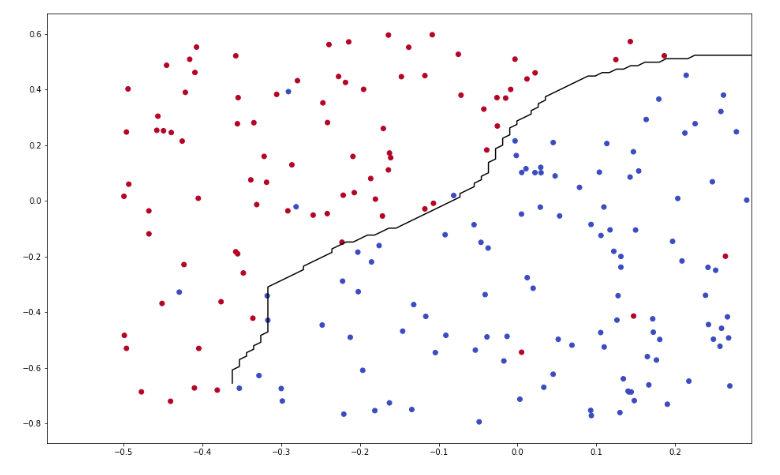


Рисунок 7 – график исходных данных вместе с разделяющей кривой

**13. Загрузите данные spamTrain.mat из файла.**

spam\_train = io.loadmat('Data/Lab 5/spamTrain.mat')

**14. Обучите классификатор SVM.**

C = 0.1

classifier5 = SVC(C=C, kernel='linear')

classifier5.fit(X, y.flatten())

print('Training Accuracy: ', (classifier5.score(X, y.flatten())) \* 100)

Результат выполнения:

'Training Accuracy: ', 99.825

**15. Загрузите данные spamTest.mat из файла.**

spam\_test = io.loadmat('Data/Lab 5/spamTest.mat')

**16. Подберите параметры C и σ2.**

C = 30, gamma=0.001 ~ 99.1

**17. Реализуйте функцию предобработки текста письма, включающую в себя:**

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

def process\_email(email\_contents):

"""

Preprocesses the body of an email and returns a list of indices of the words contained in the email.

"""

# a - Lower case

email\_contents = email\_contents.lower()

# b - remove html/xml tags

email\_contents = re.sub("<[^>]\*>", " ", email\_contents).split(" ")

email\_contents = filter(len, email\_contents)

email\_contents = ' '.join(email\_contents)

# c - Handle URLS

email\_contents = re.sub("[http|https]://[^\s]\*", "httpaddr", email\_contents)

# d - Handle Email Addresses

email\_contents = re.sub("[^\s]+@[^\s]+", "emailaddr", email\_contents)

# e - Handle numbers

email\_contents = re.sub("[0-9]+", "number", email\_contents)

# f - Handle $ sign

email\_contents = re.sub("[$]+", "dollar", email\_contents)

# Strip all special characters

special\_chars = [

"<", "[", "^", ">", "+", "?", "!", "'", ".", ",", ":",

"\*", "%", "#", "\_", "="

]

for char in special\_chars:

email\_contents = email\_contents.replace(str(char), "")

email\_contents = email\_contents.replace("\n", " ")

# Stem the word

ps = PorterStemmer()

email\_contents = [ps.stem(token) for token in email\_contents.split(" ")]

email\_contents = " ".join(email\_contents)

return email\_contents

**18. Загрузите коды слов из словаря vocab.txt.**

**with** open('Data/Lab 5/vocab.txt', 'r') **as** f: vocab = {} **for** line **in** f.readlines(): i, word = line.split() vocab[word] = int(i)

**19. Реализуйте функцию замены слов в тексте письма после предобработки на их соответствующие коды.**

def find\_word\_indices(processed\_email, vocabList\_d):

# Process the email and return word\_indices

word\_indices = []

for char in processed\_email.split():

if len(char) > 1 and char in vocabList\_d:

word\_indices.append(int(vocabList\_d[char]))

return word\_indices

**20. Реализуйте функцию преобразования текста письма в вектор признаков (в таком же формате как в файлах spamTrain.mat и spamTest.mat).**

def email\_features(word\_indices, vocabList\_d):

"""

Takes in a word\_indices vector and produces a feature vector from the word indices.

"""

n = len(vocabList\_d)

features = np.zeros((n, 1))

for i in word\_indices:

features[i] = 1

return features

**21. Проверьте работу классификатора на письмах из файлов emailSample1.txt, emailSample2.txt, spamSample1.txt и spamSample2.txt.**

email\_sample1 = open('emailSample1.txt', "r").read()

email\_sample2 = open('emailSample2.txt', "r").read()

spam\_sample1 = open('spamSample1.txt', "r").read()

spam\_sample2 = open('spamSample2.txt', "r").read()

email\_sample1 = transform\_email\_to\_features(email\_sample1, vocabList\_d)

email\_sample2 = transform\_email\_to\_features(email\_sample2, vocabList\_d)

spam\_sample1 = transform\_email\_to\_features(spam\_sample1, vocabList\_d)

spam\_sample2 = transform\_email\_to\_features(spam\_sample2, vocabList\_d)

print('Spam -> 1\nEmail -> 0')

print('\n Gaussian Kernel: ')

print('False', classifier6.predict(email\_sample1.T))

print('False', classifier6.predict(email\_sample2.T))

print('Spam', classifier6.predict(spam\_sample1.T))

print('Spam', classifier6.predict(spam\_sample2.T))

print('\n')

Результат выполнения:

Spam -> 1

Email -> 0

Gaussian Kernel:

('False', array([0], dtype=uint8))

('False', array([0], dtype=uint8))

('Spam', array([0], dtype=uint8))

('Spam', array([1], dtype=uint8))

**22. Также можете проверить его работу на собственных примерах.**

'my email prediction: **{}**'.format(

svc\_spam.predict([get\_email\_features('Data/Lab 5/myEmailSample.txt')])[0]

),

'**\n**my spam prediction: **{}**'.format(

svc\_spam.predict([get\_email\_features('Data/Lab 5/mySpamSampe.txt')])[0]

)

)

Результат выполнения:

my email prediction: 0

my spam prediction: 1

### 23. Создайте свой набор данных из оригинального корпуса текстов - <http://spamassassin.apache.org/old/publiccorpus/>.

**def** make\_emails\_list(path):

emails = []

**for** i **in** os.listdir(path):

**with** open(os.path.join(path, i)) **as** f:

**try**:

email\_data = email.message\_from\_string(''.join(f.readlines()))

**if** isinstance(email\_data.get\_payload(), str):

email\_text = email\_data.get\_payload()

emails.append(email\_text)

**except**:

print('error while parse email')

**return** emails

spams = make\_emails\_list('Data/Lab 5/spam/')

emails = make\_emails\_list('Data/Lab 5/easy\_ham/')

**24. Постройте собственный словарь.**

new\_vocab = {}

**25. Как изменилось качество классификации?**

('Training Accuracy: ', 99.54166666666666)

('Test Accuracy (linear):', 98.82747068676717, '%')

('Training Accuracy (gaussian):', 99.33333333333333, '%')

('Test Accuracy (gaussian):', 98.99497487437185, '%')

**Вывод**

Метод опорных векторов — набор схожих алгоритмов обучения с учителем, использующихся для задач классификации и регрессионного анализа. Принадлежит семейству линейных классификаторов. Особым свойством метода опорных векторов является непрерывное уменьшение эмпирической ошибки классификации и увеличение зазора, поэтому метод также известен как метод классификатора с максимальным зазором. Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве. Две параллельных гиперплоскости строятся по обеим сторонам гиперплоскости, разделяющей классы. Разделяющей гиперплоскостью будет гиперплоскость, максимизирующая расстояние до двух параллельных гиперплоскостей. Алгоритм работает в предположении, что чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка классификатора.