Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

ЛабоРАТОРНАЯ РАБОТА №5

«Метод опорных векторов»

Выполнила: Шпаковская Валерия

магистрант кафедры информатики

группа №858641

Проверил: доцент, кандидат технических наук Стержанов Максим Валерьевич

Минск 2019

ХОД РАБОТЫ

**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.

**Выполнение:**

1. Загрузите данные ex5data1.mat из файла:

file\_path = 'ex5data1.mat'

dataset = sio.loadmat(file\_path)

X = dataset["X"]

y = dataset["y"]

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

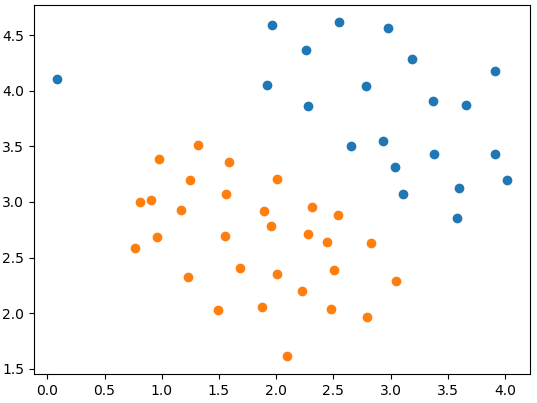
****

Рисунок 1 – исходные данные

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

classifier = SVC(kernel="linear")

classifier.fit(X, y.flatten()) # default C=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?

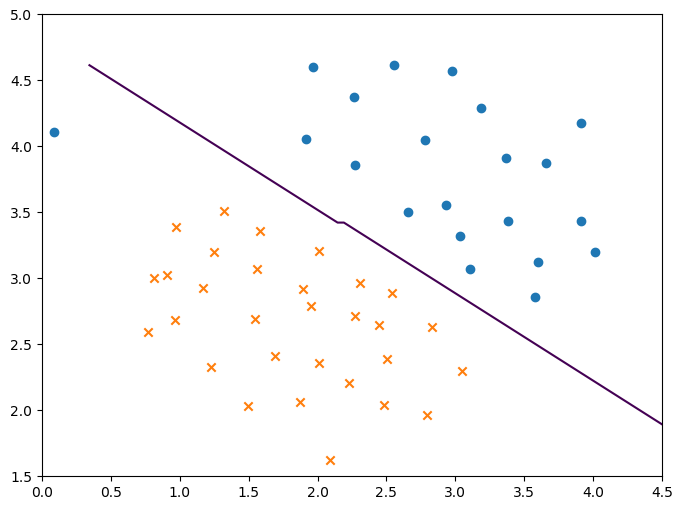


Рисунок 2 – исходные данные и график разделяющей прямой при С=1

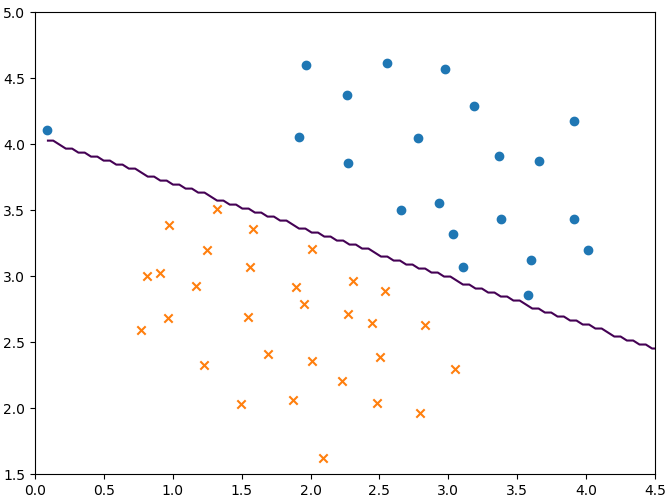


Рисунок 3 – исходные данные и график разделяющей прямой при С=100

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:

file\_path = 'ex5data2.mat'

dataset = sio.loadmat(file\_path)

X = dataset["X"]

y = dataset["y"]

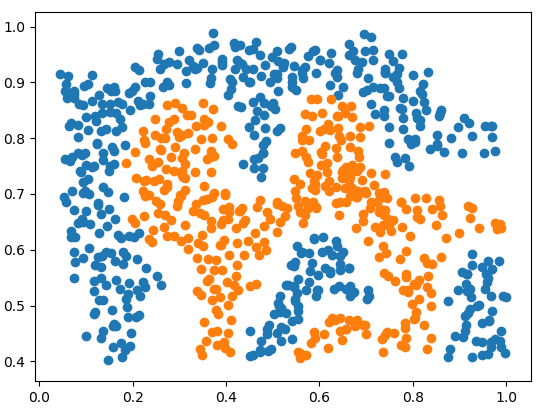


Рисунок 4 – визуализация исходных данных

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):

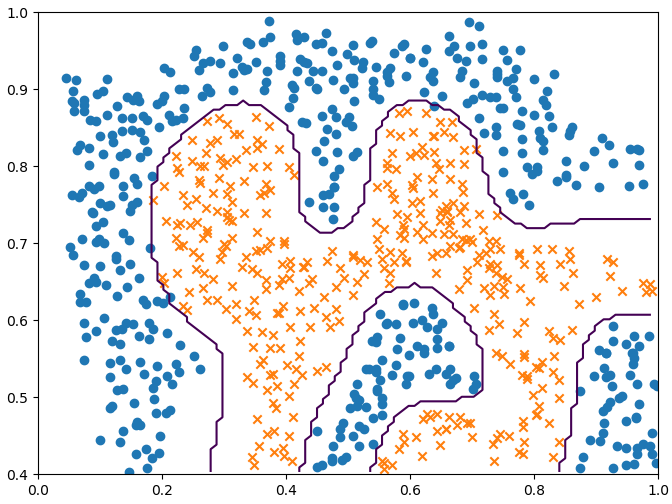


Рисунок 5 – визуализация разделяющей кривой

10. Загрузите данные ex5data3.mat из файла:

file\_path = 'ex5data3.mat'

dataset = sio.loadmat(file\_path)

X = dataset["X"]

y = dataset["y"]

Xval = dataset["Xval"]

yval = dataset["yval"]

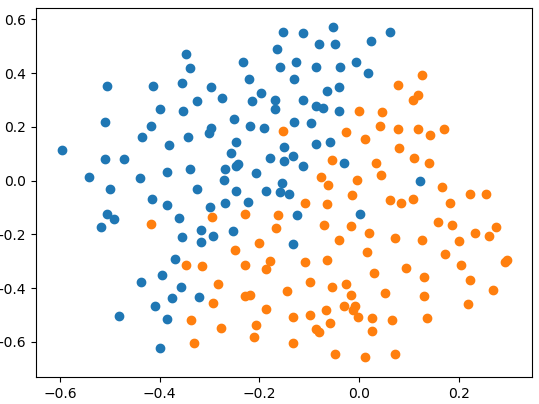


Рисунок 6 – визауализация исходных данных

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):

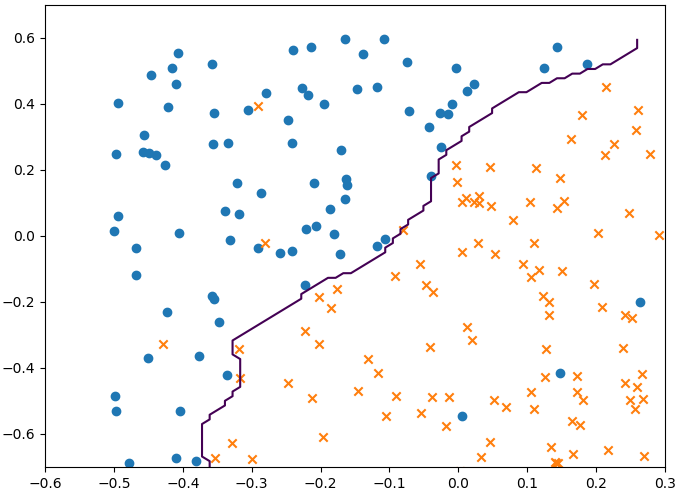


Рисунок 7 – график исходных данных вместе с разделяющей кривой

13. Загрузите данные spamTrain.mat из файла:

file\_path = 'spamTrain.mat'

dataset = sio.loadmat(file\_path)

X = dataset["X"]

y = dataset["y"]

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:

file\_path = 'spamTest.mat'

dataset = sio.loadmat(file\_path)

Xtest = dataset["Xtest"]

ytest = dataset["ytest"]

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:

vocabList = open('vocab2.txt', "r").read()

vocabList = vocabList.split("\n")

vocabList\_d = {}

for ea in vocabList:

[value, key] = ea.split("\t")

vocabList\_d[key] = value

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:

spam\_sample = "Hi, John Smith, Items on your wishlist are now discounted! Some items you've added to your personal wishlist are currently discounted on http://www.gog.com/ - you can find all the details below or directly on your https://www.gog.com/account/wishlist."

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

print('\n Gaussian Kernel: ')

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

email\_sample = "Hi John Smith, Your job search status is open, but not actively looking, but we need more information before accelerating your matches. Once you tell us just a little bit more about yourself, we'll get you in front of companies and send you any new jobs that match your interests."

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

print('Email (should be false - at least gmail doesn\'t classify this message as a spam)', classifier6.predict(email\_sample.T))

spam\_sample3 = "Hi, Hope you are doing great. This is Josh from Avco Consulting Inc. Inc is a global IT company based in Worcester, MA. Our leadership in the industry has been established by our excellence in helping clients use Information Technology to achieve their business objectives. Our core competencies are Information Technology (IT)services and Project Management. I have available consultants with Excellent communication, analytical, and team work skills. I would appreciate if you - or someone you can recommend share the suitable requirements accordingly. Below is the list of the available consultant with different skills set and their preferred locations."

spam\_sample3 = transform\_email\_to\_features(spam\_sample3, vocabList\_d)

print('Spam', classifier6.predict(spam\_sample3.T))

Результат выполнения:

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

("Email (should be false - at least gmail doesn't classify this message as a spam)", array([0], dtype=uint8))

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

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

Были выбраны 2 архива данных: 20030228\_easy\_ham.tar.bz2 и 20030228\_spam.tar.bz2. Файл со спамом содержит ~500 сообщений, не спам - ~2500 сообщений. Данные были отфильтрованы, а именно, из исходных данных были вырезаны заголовки, информация о отправителе и пр. Были проведены этапы удаление html-тегов, стэмминг, перевод в нижний регистр, удаление лишних символов и замена специальных. После этого в спам сообщениях были найдено 1899 наиболее встречающихся слов. В дальнейшем эти слова были отсортированы в алфавитном порядке и записаны в файл.

24. Постройте собственный словарь:

Семпл словарей в сравнение с оригинальным словарём от Andrew Ng:

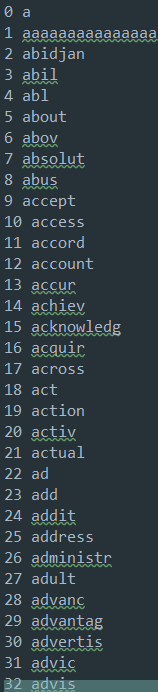
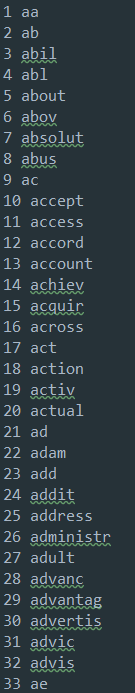


Рисунок 8 – Наглядное сравнение двух словарей (слева – Andrew Ng, справа – мой)

25. Как изменилось качество классификации? Почему?:

('Training Accuracy: ', 99.54166666666666)

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

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

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