SVM_spam_classifier

November 19, 2019

0.0.1 Project and data are based on a free, online course of machine learning https://www.coursera.org/learn/machine-learning. I wholeheartedly recommend this!

0.1 I will show how do it in Python:

```
+ apply SVM (support vector machine) to spam classification,
+ use regular expressions.
```

[1]:
```
import re
import string
import scipy.io
import numpy as np
import scipy.io
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import warnings
import sys


# ignore warnings
warnings.filterwarnings('ignore')

# write packages and python version to file
'''
! python -m pip list > packages_versions.txt
# a append to file
with open('packages_versions.txt', 'a') as f:
    f.write('Python version ' + str(sys.version))
''';
```

[2]:
```
def remove_html_tags(text):
    '''
    Removes html tags.
    '''
    return re.sub('<[^<>]+>', ' ', text)
assert remove_html_tags('<a href="onet.com">deal with it</a>') == ' deal with␣
 ↪it '
```

```
def normalizing_URLs(text):
    '''
    Replaces URL to " httpaddr ".
    '''
    return re.sub('(http|https|ftp)://[^\s]*', ' httpaddr ', text)

assert normalizing_URLs('herehttps://en.wikipedia.org/wiki/URL') == 'here␣
 ↪httpaddr '

def normalizing_emails(text):
    '''
    Replaces email adress to " emailaddr ".
    '''
    return re.sub('[^\s]+@[^\s]+', ' emailaddr ', text)

assert normalizing_emails('contact awokado@gmail.com') == 'contact  emailaddr '

def normalize_dolar(text):
    '''
    Replaces $ sign to " dolar ".
    '''
    return re.sub('[$]+', ' dollar ', text)

assert normalize_dolar('less$$$bargain$') == 'less dollar bargain dollar '

def normalize_numbers(text):
    '''
    Replaces $ sign to " number ".
    '''
    return re.sub('[\d]+', ' number ', text)

assert normalize_numbers('for 3 day') == 'for  number  day'

def remove_punct(text):
    '''
    Removes all punctuation marks.
    '''
    translator = str.maketrans(string.punctuation, ' '*len(string.punctuation))
    return text.translate(translator)

assert remove_punct('only#today!!!') == 'only today    '

def remove_multi_spaces(text):
    '''
    Replaces multi spaces to one.
    '''
    return re.sub(' +', ' ', text)
```

```python
def proccess_email(text):
    '''
    Normalizes email.
    '''
    text = text.lower()
    text = remove_html_tags(text)
    text = normalizing_URLs(text)
    text = normalizing_emails(text)
    text = normalize_dolar(text)
    text = normalize_numbers(text)
    text = remove_punct(text)
    text = text.replace('\n', ' ')
    text = remove_multi_spaces(text)
    return text

def email_to_features(email_name, ps, vocab):
    '''
    Sets 1 at right position, when word from vocab occurs in email.
    '''
    with open(email_name, 'r') as file:
        email = file.read()
        email = proccess_email(email).split()
        email = [ps.stem(w) for w in email]
        email = sorted(set(email), key=email.index)
    ind_list = []
    for word in email:
        try:
            ind_list.append(vocab_dict[word])
        except KeyError:
            pass
    init = np.zeros((1,1899))
    for ind in ind_list:
        init[0,ind] = 1
    return init
```

[3]:
```python
# use porterStemmer to treat program and programmer as 'program'
ps = PorterStemmer()

with open('vocab.txt', 'r') as file:
    vocab = file.read().replace('\n', ' ')
    vocab = vocab.replace('\t', ' ')
    vocab = remove_multi_spaces(vocab)
    vocab = vocab.split()
    vocab_dict = {}
    for i, ind in enumerate(range(1, len(vocab), 2)):
        vocab_dict[vocab[ind]] = i
```

```python
rev_vocab_dict = dict(zip(vocab_dict.values(), vocab_dict.keys()))
```

[4]:
```python
# Train model
data = scipy.io.loadmat('spamTrain.mat')
# X indicates which words from spam vocabulary list are used in email.
X = data['X']
Y = data['y'].flatten()
from sklearn.svm import SVC
model = SVC(kernel='linear', C=0.1)
model.fit(X, Y);
```

[5]:
```python
# Lets find words, which are most common in spam.
coef = model.coef_.flatten()
sorted_ind = np.argsort(coef)
top_15 = sorted_ind[:-16:-1]
print('word        weight')
print('*' * 18)
for top in top_15:
    print('{:10}  {:.3f}'.format(rev_vocab_dict[top], coef[top]))
```

```
word        weight
******************
our         0.501
click       0.466
remov       0.423
guarante    0.384
visit       0.368
basenumb    0.345
dollar      0.324
will        0.270
price       0.267
pleas       0.261
most        0.257
nbsp        0.254
lo          0.253
ga          0.248
hour        0.246
```

[6]:
```python
for i, name in enumerate(['emailSample1.txt', 'emailSample2.txt', 'spamSample1.
 ↪txt', 'spamSample2.txt']):
    features_arr = email_to_features(name, ps, vocab)
    print('Predicted value for {} {}, it should be {}'.format(name[:-4], model.
 ↪predict(features_arr)[0], i//2))
```

```
Predicted value for emailSample1 0, it should be 0
Predicted value for emailSample2 0, it should be 0
Predicted value for spamSample1 1, it should be 1
Predicted value for spamSample2 1, it should be 1
```

[7]:
```
data_test = scipy.io.loadmat('spamTest.mat')
X_test = data_test['Xtest']
Y_test = data_test['ytest'].flatten()
print('Prediction on test set = {}%'.format(100 * np.mean(model.predict(X_test)␣
 ↪== Y_test)))
```

```
Prediction on test set = 98.9%
```