SVM 实验报告

191250111 裴为东

1.算法原理:

普遍意义上的 SVM 是一个二分类线性分类器，通过在分类超平面的正负两边各找到一个离分类超平面最近的点，使得这两个点距离分类器超平面的距离和最大

2.实验过程:

1）数据选用了已经过预处理的数据，故不需要再进行预处理

2）读取数据的内容，将数据集中 enron5、enron6 作为测试集，其余作为训练集

```
def load_data(mail_dir):
    files = []
    labels = []
    for dir in os.listdir(mail_dir):
        subdir = os.path.join(mail_dir, dir)
        if os.path.isdir(subdir):
            for sub_sub_dir in os.listdir(subdir):
                if sub_sub_dir == "ham":
                    for fi in os.listdir(os.path.join(subdir, "ham")):
                        files += [os.path.join(subdir, "ham", fi)]
                        labels.append(1)
                elif sub_sub_dir == "spam":
                    for fi in os.listdir(os.path.join(subdir, "spam")):
                        files += [os.path.join(subdir, "spam", fi)]
                        labels.append(0)
    test_matrix = np.ndarray((len(files)), dtype=object)
    id = 0
    for fil in files:
        with open(fil, 'r', errors="ignore") as fi:
            next(fi)
            data = fi.read().replace('\n', ' ')
            test_matrix[id] = data
            id += 1
    return test_matrix, labels
```

3）提取特征并训练模型，CountVectorize 函数会将文本中的词语转换为词频矩阵，通过 fit_transform 函数计算各个词语出现的次数，再经过 TfidfTransformer 函数输出词频逆反文档频率，最后通过 sklearn 库中的线性核 SVM 模型进行训练

```
train_dir = "./train"
train_matrix, train_labels = load_data(train_dir)
print("train size: ", train_matrix.shape[0])

count_v1 = CountVectorizer(stop_words="english", max_df=0.5, decode_error="ignore", binar
counts_train = count_v1.fit_transform(train_matrix)
tf_idf_transformer = TfidfTransformer()
tfidf_train = tf_idf_transformer.fit(counts_train).transform(counts_train)

model = LinearSVC()
model.fit(tfidf_train, train_labels)
```

4）评估模型

```
test_dir = "./test"
test_matrix, test_labels = load_data(test_dir)
print("test size: ", test_matrix.shape[0])

count_v2 = CountVectorizer(vocabulary=count_v1.vocabulary_, stop_words="english", max_df=
                           binary=True)
counts_test = count_v2.fit_transform(test_matrix)
tfidf_test = tf_idf_transformer.fit(counts_test).transform(counts_test)

result = model.predict(tfidf_test)
cm = pd.DataFrame(
    confusion_matrix(test_labels, result), index=["non-spam", "spam"], columns=["non-spam
)

print("result:")
print(cm)
print("precision score: ", precision_score(test_labels, result))
print("recall score: ", recall_score(test_labels, result))
```

3.实验结果

```
train size:  22541
test size:  6000
result:
          non-spam  spam
non-spam      4470    30
spam            61  1439
precision score:  0.9795779441797141
recall score:  0.9593333333333334
```