SVM机器学习实验——171250574杨逸存

实验简介

- 本实验使用python语言及sklearn、numpy、pandas等相关库实现了SVN分类算法在辨别垃圾邮件上的应用问题。

算法/代码详解

SVM算法原理

SVM算法是建立在感知机分类的基础上的。感知机意图寻找一个超平面Π: $w^Tx+b=0$来区分两个类，但是SVM意图寻找一块最有的平面，即**让决策面离正负样本点的间隔都尽可能大。**对于一个数据集到超平面的距离表示为：

$d(D, Π) = \min d((x, y), Π)$，其中**$d((x, y), Π) = 1/|w| \cdot y \cdot (w^Tx + b)$**

则优化目标为：

$\max d(D, Π)$ ， w.l.o.g. $\min (|w|^2 / 2)$ s.t. $y_i(w^Tx_i + b) \geq 1$

对于非线性可分的数据集，我们需要放松限制条件，同时增加损失函数的惩罚度：

$\min [|w|^2 / 2 + C \cdot \sum_i \xi_i]$ s.t. $y_i(w^Tx_i + b) \geq 1 - \xi_i$

经过数学转化：

$\min L(D) = \min [|w|^2 /2 + C \cdot \sum_i ReLU(1 - y_i(w^Tx_i + b))]$ s.t. $y_i(w^Tx_i + b) \geq 1 - \xi_i$

对改进的损失函数进行梯度下降，先求导：

$L(D) = \min [|w|^2 /2 + C \cdot \sum_i ReLU(1 - y_i(w^Tx_i + b))]$ i.e. $L(x, y) = \min [|w|^2 /2 + C \cdot ReLU(1 - y(w^Tx + b))]$

当$y(w^Tx + b) \geq 1$时，$\partial L(x, y) / \partial w = w$，$\partial L(x, y) / \partial b = 0$

当$y(w^Tx + b) < 1$时，$\partial L(x, y) / \partial w = w - Cyx$，$\partial L(x, y) / \partial b = -Cy$

i.e. 梯度下降表示为：**$w \leftarrow w(1 - \eta)$，η为学习率**

当有**$y(w^Tx + b) < 1$**时，选出某个被错分类的样本实例(x, y)，再进行如下操作：

$w \leftarrow w + \eta Cyx$

$b \leftarrow b + \eta Cy$

最终完成收敛。

核心代码讲解

```
mail_matrix, mail_labels = read_files(mail_dir_path)
# 分割训练测试集，按照8：2的比例分割训练集和测试集
mail_train, mail_test, mail_train_label, mail_test_label \
    = train_test_split(mail_matrix, mail_labels, test_size=0.2,
random_state=1)
# 使用TF-IDF将文本特征提取为向量
# 关键词document frequence最大阈值设为0.6
# 对关键词出现次数进行实际计数，而非0/1
count_vec = CountVectorizer(stop_words='english', max_df=0.6,
decode_error='ignore', binary=False)
count_train = count_vec.fit_transform(mail_train)
tfidfTransformer = TfidfTransformer()
tfidf_train = tfidfTransformer.fit_transform(count_train)
# 训练模型
svn_model = LinearSVC()
svn_model.fit(tfidf_train, mail_train_label)
```

对测试集做同样处理后，进行predict操作，最终混淆矩阵和度量值代码如下：

```
'''
混淆矩阵：    预测正例(1)    预测反例(0)
真实正例(1)     TP            FN
真实反例(0)     FP            TN
设矩阵的labels字段为[1, 0]定义顺序
'''
confusion_m = pd.DataFrame(confusion_matrix(mail_test_label,
predict_label, labels=[1, 0]),
                           index=['actual_ham', 'actual_spam'],
                           columns=['predicted_ham',
'predicted_spam'])
print(confusion_m)
print('精确率Precision=TP/(TP+FP)，预测为正类的实例中真正预测正确的比
例：', precision_score(mail_test_label, predict_label, pos_label=1))
print('准确率Accuracy=(TP+TN)/(TP+FN+FP+TN)，总体预测结果正确率：',
accuracy_score(mail_test_label, predict_label))
print('召回率Recall=TP/(TP+FN)，指对于所有正类实例的预测正确率：',
recall_score(mail_test_label, predict_label, pos_label=1))
```

实验结果分析

混淆矩阵及三个度量值：精确率、准确率、召回率显示如下

```
             predicted_ham  predicted_spam
actual_ham            3299              60
actual_spam             16            3369
精确率Precision=TP/(TP+FP)，预测为正类的实例中真正预测正确的比例：  0.9951734539969834
准确率Accuracy=(TP+TN)/(TP+FN+FP+TN)，总体预测结果正确率：  0.9887307236061684
召回率Recall=TP/(TP+FN)，指对于所有正类实例的预测正确率：  0.982137540934802

Process finished with exit code 0
```

binary设为True后实验结果，各方面度量值都有所下降，因此关键词出现的频数也对模型训练有所影响。

```
             predicted_ham  predicted_spam
actual_ham            3293              66
actual_spam             22            3363
精确率Precision=TP/(TP+FP)，预测为正类的实例中真正预测正确的比例：  0.9933634992458522
准确率Accuracy=(TP+TN)/(TP+FN+FP+TN)，总体预测结果正确率：  0.9869513641755635
召回率Recall=TP/(TP+FN)，指对于所有正类实例的预测正确率：  0.9803512950282822

Process finished with exit code 0
```