SVM 实验报告

江辉--171250506

2020-5-2

南京大学

软件学院

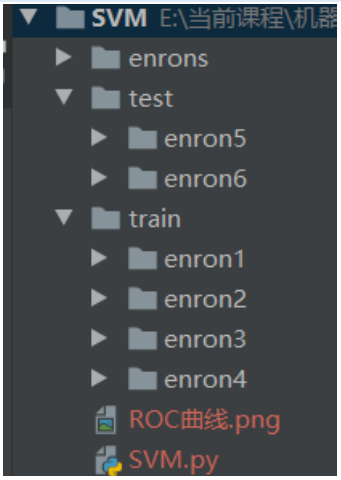
目录

一、实验内容描述

本程序用 python 编写，使用 scikit-learn 构建 SVM 模型。评价指标采用混淆矩阵、精准率、召回率、RUC 曲线、AUC 值

1、项目目录描述

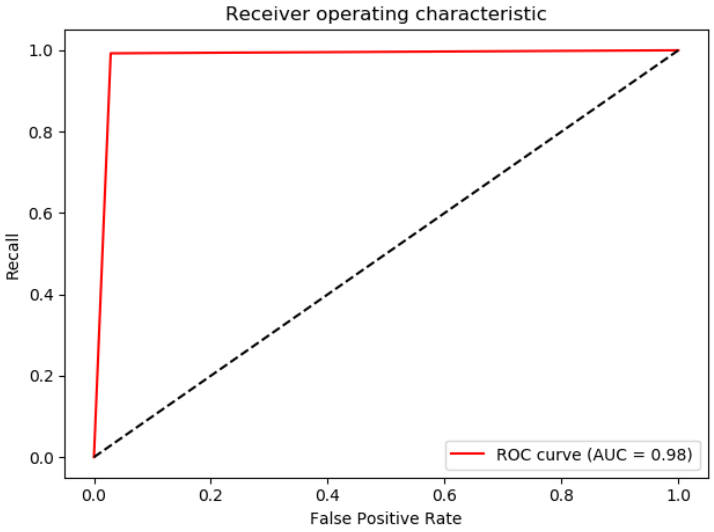
本项目推荐使用 pycharm 打开
SVM.py 里写是代码
train 和 **test** 分别是训练集和测试集，我将 enron1-4 作为训练集，enron5-6 作为测试集



二、实验结果图

下面是实验的结果图，左边打印了采用的训练集和测试集的大小、混淆矩阵、精准率、召回率和 AUC 值。右图是打印的 ROC 曲线。

```
train set size:  22541
    spam:  8996
    non-spam:  13545
test set size:  11175
    spam:  8175
    non-spam:  3000
predict result:
          non-spam  spam
non-spam      2914    86
spam            61  8114
precision socre:  0.9895121951219512
recall score:  0.9925382262996942
auc score:  0.9819357798165138
```



三、 算法原理阐述

SVM 是一种二分类模型，通过计算特征空间上的最大间隔来进行线性分类。SVM 的学习策略是通过求解凸二次规划的最优化问题来求解一个能够正确划分训练数据集并且几何间隔最大的分离超平面。

最大化间隔的原理:

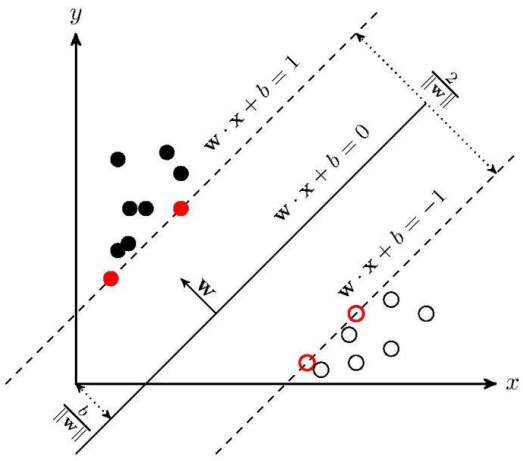
定义每个样本 x 的 n 个特征(x1，x2……xn)和一个类型标记 y∈{-1,+1}

假设所求的决策边界是 w·x+b=0，对于线性可分的数据集，w·x+b 的值就是预测值

定义样本点到决策边界的距离为γ = y·(w/||w|| · x + b/||w||)

因为希望求得一个能将所有样本点的间隔最明显(最大)的边界，所以以间隔最小的点作为支持向量，最大化决策边界到支持向量的最小间隔γ即可，如下图中红色点为支持向量，希望找到最大化间隔γ的边界。即求下面的最优化函数

$$\arg\max_{w,b}\{\min(\ y(w^T x+b))\cdot\frac{1}{\|w\|}\}$$



上述定义中γ = min γi = min yi·(w/||w|| · xi + b/||w||)

同除γ得 yi·(w/||w||γ · xi + b/||w||γ)>=1

此时||w||和γ都是标量，可以简化为 yi·(w · xi + b)>=1

因此上面的问题可以简化为

$$\arg\max(\frac{1}{\|w\|})$$

$$s.t.\ y(w^T x+b)-1\geq 0$$

在 xi 给定的情况下，最大化 γ 等价于最大化 1/||w||，也就等价于 1/2||w||^2 ，即

$$\min\ \frac{1}{2}\|w\|^2$$

约束条件是 yi·(w · xi + b) >= 1

对于含不等式约束的凸二次规划问题，使用拉格朗日乘子法来处理

定义目标函数

$$L(\boldsymbol{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_{i=1}^{N}\alpha_i\left(y_i\left(\boldsymbol{w}\cdot\boldsymbol{x_i}+b\right)-1\right)$$

求 L 关于求偏导数得

$$\begin{cases}\dfrac{\partial L(w,b,\alpha)}{\partial w}=0\Rightarrow w=\sum_{i=1}^{n}\alpha_i y_i x_i\\ \dfrac{\partial L(w,b,\alpha)}{\partial b}=0\Rightarrow\sum_{i=1}^{n}\alpha_i y_i=0\end{cases}$$

带入前面的 L(w,b,α)化简后得到右边最终的优化目标函数

$$\max\ L(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

$$s.t.$$

$$\begin{cases}\sum_{i=1}^{n}\alpha_i y_i=0\\ \alpha_i>0, i=1,2...n\end{cases}$$

该问题对应的 KKT 条件为

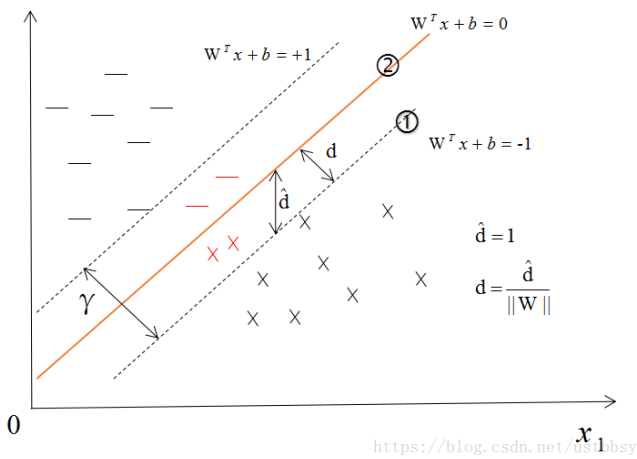
$$\begin{cases}\alpha_i\geq 0\\ y_i f(x_i)-1\geq 0\\ \alpha_i(y_i f(x_i)-1)=0\end{cases}$$

SVM 引入了松弛变量解决无法完全线性分割的噪点问题
如右图中分割边界中间有一些散落的点，这些点不满足 γ>=1 的限制条件。因此需要加入一个松弛变量 ξ
这时约束条件不再是 1，而是 1 - ξ

$$y_i(w^T x_i + b) \geq 1 - \xi$$

数学模型中加入一个惩罚因子 C，C 值越大，对分类的惩罚越大

$$\begin{cases} \min\limits_{w,b} \frac{1}{2}\|W\|^2 + C\sum\limits_{i=1}^{m} \xi_i \\ y_i(w^T x_i + b) \geq 1 - \xi \\ \xi_i \geq 0 \\ i = 1,2,\ldots m \end{cases}$$

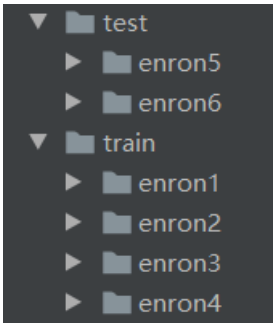


目标函数推导过程同前面没有加入松弛变量的一样，结果也一样，不过修改惩罚参数 C 更能解决噪点数据的问题

四、代码简述

先切分了 train 和 test 的数据集并根据 ham 和 spam 文件夹来打上对应标签(0 和 1)

```
train_dir = "./train"
train_matrix, train_labels = load_data(train_dir)
test_dir = "./test"
test_matrix, test_labels = load_data(test_dir)
```




```
def load_data(dir):
    labels = []
    ham_files = []
    spam_files = []
    for enron in os.listdir(dir):
        sub_dir = os.path.join(dir, enron)
        if os.path.isdir(sub_dir):
            for d in os.listdir(sub_dir):
                if d == "ham":
                    for fi in os.listdir(os.path.join(sub_dir, "ham")):
                        ham_files.append(os.path.join(sub_dir, "ham", fi))
                        labels.append(0)
                elif d == "spam":
                    for fi in os.listdir(os.path.join(sub_dir, "spam")):
                        ham_files.append(os.path.join(sub_dir, "spam", fi))
                        labels.append(1)
    text_matrix = np.ndarray([len(ham_files) + len(spam_files)], dtype=object)
    index = 0
    for file in ham_files:
        with open(file, 'r', errors="ignore") as fi:
            next(fi)
            data = fi.read().replace('\n', ' ')
            text_matrix[index] = data
            index += 1
    for file in spam_files:
        with open(file, 'r', errors="ignore") as fi:
            next(fi)
            data = fi.read().replace('\n', ' ')
            text_matrix[index] = data
            index += 1
    return text_matrix, labels
```

整个代码实现处理过程参考了作业 pdf 里给出的样例代码，使用 TF-IDF 特征，用 sklearn 的默认的 SVM 模型。

```
train_dir = "./train"
train_matrix, train_labels = load_data(train_dir)
print("train set size: ", train_matrix.shape[0])
print("\tspam: ", train_labels.count(True))
print("\tnon-spam: ", train_labels.count(False))

count_v1 = CountVectorizer(stop_words="english", max_df=0.5, decode_erro
counts_train = count_v1.fit_transform(train_matrix)
tfidftransformer = TfidfTransformer()
tfidf_train = tfidftransformer.fit(counts_train).transform(counts_train)

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

```
test_dir = "./test"
test_matrix, test_labels = load_data(test_dir)
print("test set size: ", test_matrix.shape[0])
print("\tspam: ", test_labels.count(True))
print("\tnon-spam: ", test_labels.count(False))

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

result = model.predict(tfidf_test)
```

在测试评估时采用了精确率、召回率、ROC 曲线、AUC 值作为评估指标

```
print("precision socre: ", precision_score(test_labels, result))
print("recall score: ", recall_score(test_labels, result))
auc = roc_auc_score(test_labels, result)
print("auc score: ", auc)

FPR, recall, thresholds = roc_curve(test_labels, result, pos_label=1)

# 画出ROC曲线
plt.figure()
plt.plot(FPR, recall, color='red'
         , label='ROC curve (AUC = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], color='black', linestyle='--')
# 为了让曲线不黏在图的边缘
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('Recall')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```