SVM 支持向量机

目标

选取不同的卷积核并利用SVM方法来对对第二次作业数据进行分类，并和作业二进行比较。

实现

首先我们还是通过 load_words_data 读取excel 表，进行数据的读取和组合，和作业二不同的是，我们直接把结果二元化，用 0，1表示而不是字符串，如下述代码：

```
    fix_word_data = []
    if search_index > 4800:
        fix_word_data.append(1)
    else:
        fix_word_data.append(0)

    if search_result < 1000:
        fix_word_data.append(1)
    else:
        fix_word_data.append(0)

    if search_popular > 50:
        fix_word_data.append(1)
    else:
        fix_word_data.append(0)
```

接下来用sklearn 进行 svm 准确率计算：

我们使用 train_test_split 进行训练集和测试集的拆分，定义使用传入的kernel，然后进行数据训练和预测得分。

```
def svm(k):
    print('========' + k + '========')
    labels, data = load_words_data()
    x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.3, random_state=33)
    clf = SVC(kernel=k, C=10)
    clf.fit(x_train, y_train)
    print('The result score is', clf.score(x_test, y_test))
```

传入不同的核函数

```
kernels = ['linear', 'rbf', 'poly', 'sigmoid']
for kernel in kernels:
    svm(kernel)
```

结果

```
/usr/local/Cellar/python3/3.7.1/bin/python3.7 /Users/jackrex/Desktop/AILesson/L5/SVM.py
=======linear=======
The result score is 0.874
Time usage: 0.8820810317993164
=======rbf=======
The result score is 0.8706666666666667
Time usage: 0.925457239151001
=======poly=======
The result score is 0.8706666666666667
Time usage: 0.8478808403015137
=======sigmoid=======
The result score is 0.8706666666666667
Time usage: 0.8874490261077881
```

从结果上来看其实其差别不是太大，其中linear 最好，也有可能我处理的数据问题本身就是一个强线性关系，所以使用其他非线性核并没有太好的提升。

和作业L2 比起来，之前使用DT 和随机森林的结果分别是：
DT：

```
base_entropy is : 0.7106768538561 23
new_entropy is + 0.710676853856123
prob is 0.712
time gap is:0.033406972885131836
{'Search-Result': {'invalid search result': 'failed', 'valid search result': {'App-Name': {'valid words'
base_entropy is : 0.5332398959774798
```

准确率 71.2%

随机森林

```
new_entropy is + 0.7256040085121113
new_entropy is + 0.7128481244254435
base_entropy is : 0.7168319999345276
new_entropy is + 0.7165149469934304
{'Search-Result': {'invalid search result': 'failed', 'valid search result': {'Search-Index': {'valid search index': {'Ap
prob is 0.875751503006012
max is 0.9018036072144289
time t1 gap is:0.578096866607666
```

准确率 90.18036%

使用Gridsearch 选择最优参数

```
def best_svm(k):
    print('=======' + k + '=======')
    labels, data = load_words_data()
    x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.3, random_state=33)
    tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                         'C': [1, 10, 100, 1000]},
                        {'kernel': ['linear'], 'C': [1, 10, 100, 1000]},
                        {'kernel': ['poly'], 'C': [1, 10, 100, 1000]},
                        {'kernel': ['sigmoid'], 'C': [1, 10, 100, 1000]}]
    clf = GridSearchCV(SVC(), tuned_parameters, cv=5)
    clf.fit(x_train, y_train)
    print(clf.best_params_)
    print('The result score is', clf.score(x_test, y_test))
```

我们定义了不同的kernel，不同的gamma 和 C 错误容忍度来去算出最好的参数，结果如下：

```
{'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
The result score is 0.874
```

结论

1、对应纯线性问题不同核函数之间分类结果差异不大
2、和DT 比，SVM 效果明显好于DT，但是不一定优于随机森林
3、SVM 好处和优势既可应用于线性分类，也可应用于非线性分类
4、通过Gridsearch 各种 gamma C kernel 的不同选择，可以得到最优结果。