Assignment 1 - Speech and not speech detection Report
夏伊倩（21631152）

[Assignment Requirements]

Write a program to detect whether a person in a video speaks or not.

Features:

S: Depicts the motion of mouth;

V: Depicts the degree of mouth opening.

Feature vector:

X = [Si-1 Si Si+1 Vi-1 Vi Vi+1]

So that, the input feature vector X is an N*6 matrix, where N is the number of feature vectors.

Label:

+1: Speaking

-1: Not-speaking

So that, the output label vector is an N*1 vector predY, predY(i)=-1 or 1.

[Solutions]

1. SVM

使用 Scikit-learn，即 sklearn，一个 Python 的机器学习包。

对于 SVM 问题，我们可以调用 sklearn 包中基于 libsvm 的 svm 模块。对于本次任务中的分类问题，我们调用其中的 SVC 函数（针对回归问题，可以调用 SVR 函数）。

任务实现的主要步骤为：

数据预处理-加载数据集-训练模型（网格搜索参数）-测试模型

1.1 数据预处理

原数据集 training.data 格式为：

```
0.65307 0.9135 0.99926 0.18748 0.14696 0.16105 -1
```

我们通过 FormatDatalibsvm.xls，按照

http://blog.csdn.net/pangpang1239/article/details/7435842 中所说的方式

将数据处理为 libsvm 要求的格式，得到 training.txt：

```
-1 1:1.57790 2:1.84050 3:1.48320 4:0.28122 5:0.23753 6:0.22297
```

1.2 加载数据集

获得特征向量 xData 和标签向量 yData：

```
xData, yData = datasets.load_svmlight_file("training.txt")
```

我们再在数据中留出一部分验证集：

```
training_data_x, test_data_x, training_data_y, test_data_y =
train_test_split(xData, yData)
```

train_test_split()函数是交叉验证中常用的函数，

功能是：从样本中随机按比例选取 trainData 和 testData；

参数有：train_data, train_target, test_size, random_state，

train_data：所要划分的样本特征集

train_target：所要划分的样本结果

test_size：样本占比，默认为 0.25，即验证集占 25%

random_state：是随机数的种子，为 0 或者不填，则每次产生的随机数不同

1.3 训练模型

通过网格搜索的方式对参数空间进行测试，寻求最佳的参数。

```
C = np.logspace(-1,1,5,base=2)
gamma = np.logspace(-2,5,5,base=2)
param_grid = dict(C=C, gamma=gamma)
grid = GridSearchCV(estimator=clf,param_grid=param_grid,n_jobs=-1)
```

得到 grid.best_estimator_.C 和 grid.best_estimator_.gamma

即，得到了最后的 SVC 模型：

```
clf = SVC()
clf.fit(training_data_x, training_data_y)
SVC(C=bestC, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma=bestGamma,
kernel='rbf',max_iter=-1, probability=False, random_state=None,
shrinking=True,tol=0.001, verbose=True)
```

保存模型：

```
joblib.dump(clf, 'svcmodel.pkl')
```

1.4 测试模型

定义接口函数 speakingDetection.py :

```
def speakingDetection(X):
    clf = SVC()
    clf = joblib.load('svcmodel.pkl')
    predY = clf.predict(X)
    return predY
```

输入 test_data_x 得到 prediction

```
prediction = speakingDetection(test_data_x)
```

得到测试报告

```
report = classification_report(test_data_y, prediction)
```

1.5 函数调用说明

文件夹 SVM 下的 speakingDetection.py 是用 python 编写的接口函数，调用形式可以参照 SK_TestSVM.py，主要就是 prediction = speakingDetection(test_data_x)这一句，就能得到 preY。训练得到的 SVC 模型为 svcmodel.pkl

2. NN

使用 keras，一个用 python 编写的基于 Theano/tensorflow 的深度学习框架，是一个高度模块化的神经网络库。

（非常）易于上手，把很多内部运算都隐藏了，类似一个黑箱，调用 API 就行了。

有扩展性，可以用 theano 或 TensorFlow 的语句来写扩展功能。

任务实现的主要步骤为：

数据预处理-加载数据集-训练模型-测试模型

值得一提的是，神经网络做二分类问题时，label 值为 0 或 1，故需要将原 label 进行处理。

除此之外，数据预处理&加载数据集和 SVM 部分中一致，以下略去不表。

2.1 训练模型

通过 Sequential()初始化一个神经网络

通过 add 方法添加一层神经网，需要添加输入层、隐层、输出层。通过 input_dim 定义输出维度，units 定义输出维度，activation 定义激励函数。

本次任务中，定义 optimizer 为 SGD，loss 为 sparse_categorical_crossentropy，即稀疏的多类的对数损失，epochs 为迭代次数，batch_size=128.

```
# Create a model
model = Sequential()
# Input
model.add(Dense(64, activation='relu', input_dim=6))
model.add(Dropout(0.5))
# Hidden
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
# Output
model.add(Dense(2, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='sparse_categorical_crossentropy', optimizer=sgd,
metrics=['accuracy'])
model.fit(training_data_x, training_data_y, epochs=3000,
batch_size=128)
```

保存模型

```
model.save('nn_model.h5')
```

2.2 测试模型

定义接口函数 speakingDetection.py :

```
def speakingDetection(X):
    model = Sequential()
    model = load_model('nn_model.h5')
    predY = model.predict_classes(X, batch_size=128)
    return predY
```

输入 test_data_x 得到 prediction

```
prediction = speakingDetection(test_data_x)
```

得到测试报告

```
report = classification_report(test_data_y, prediction)
```

2.3 函数调用说明

文件夹 NN 下的 speakingDetection.py 是用 python 编写的接口函数，调用形式可以参照 NN_Test.py，主要就是 prediction = speakingDetection(test_data_x)这一句，就能得到 preY。训练得到的 NN 模型为 nn_model.h5

[Results]

在随机截取的 25%的验证集上的正确率：

SVM：74.92%

NN： 77.37%

[踩过的坑]

1. 最一开始还是想用 Matlab 做的，但是在调用 libsvm 库的时候总是报错，编译.c 文件也失败，只好转战 python，后来才知道是 Matlab 版本太低，换上 2015 版，什么问题都没了；
2. 转战 python 后，本来想直接使用调用 libsvm 库，但只是跑了个 10*11 的网格进行参数搜索就快要奔溃了，跑一个参数 pair 大概耗时 20mins+，充分理解到了 SVM 的“计算量大”说的是什么，当然电脑配置不高是主要原因，只能果断放弃，选择 sklearn，果然又快又方便（关于其运行速度比 sklearn 慢的问题，推测是 sklearn 做了某些优化）；
3. 本次神经网络的编写完全速成，大概花了半天时间，查了网上的教程就基本完成本次的神经网络任务的初步编写，若是要扎实学习，还是要直接使用 Tensorflow 做实践吧。

附上本次任务的 github 地址：

https://github.com/BigRabbit71/speakingDetection