IIE Project 3

利用SVM分類脊椎異常患者

108034552 葉紹康

主題簡介

- 利用SVM模型針對脊椎異常的患者做分類，並計算其準確率。
- 資料集共有三種類別
  - 正常(100)
  - 椎間盤突出(60)
  - 腰椎滑脫(150)

	A	B	C	D	E	F	G
1	63.03	22.55	39.61	40.48	98.67	-0.25	DH
2	39.06	10.06	25.02	29	114.41	4.56	DH
3	68.83	22.22	50.09	46.61	105.99	-3.53	DH
4	69.3	24.65	44.31	44.64	101.87	11.21	DH
5	49.71	9.65	28.32	40.06	108.17	7.92	DH
6	40.25	13.92	25.12	26.33	130.33	2.23	DH
7	53.43	15.86	37.17	37.57	120.57	5.99	DH
8	45.37	10.76	29.04	34.61	117.27	-10.68	DH
9	43.79	13.53	42.69	30.26	125	13.29	DH
10	36.69	5.01	41.95	31.68	84.24	0.66	DH
11	49.71	13.04	31.33	36.67	108.65	-7.83	DH
12	31.23	17.72	15.5	13.52	120.06	0.5	DH
13	48.92	19.96	40.26	28.95	119.32	8.03	DH
14	53.57	20.46	33.1	33.11	110.97	7.04	DH

Fearture1=骨盆入射角

Fearture2=骨盆傾斜度

Fearture3=腰椎前凸角

Fearture4=骶骨傾斜坡

Fearture5=骨盆前凸弧度

Fearture6=脊椎滑脫等級

實驗流程

Data Collection → Data Pre-processing → Classifier Training → Accuracy Testing

01

Data Collection

由UCI公開數據集取得資料

02

Data Pre-processing

資料前處理

03

Classifier Training

建立模型與進行訓練

04

Accuracy Testing

投入測試集並得出最終準確率

資料前處理

- 將資料匯入，共有310筆資料、七列不同的資訊

```
import pandas as pd
data = pd.read_csv('C:/Users/Lab905/Desktop/data.csv',header=None)
print(data.head)
print("data shape :", data.shape)
```

```
<bound method NDFrame.head of          0       1       2       3       4       5   6
0     63.03   22.55   39.61   40.48    98.67  -0.25  DH
1     39.06   10.06   25.02   29.00   114.41   4.56  DH
2     68.83   22.22   50.09   46.61   105.99  -3.53  DH
3     69.30   24.65   44.31   44.64   101.87  11.21  DH
4     49.71    9.65   28.32   40.06   108.17   7.92  DH
5     40.25   13.92   25.12   26.33   130.33   2.23  DH
6     53.43   15.86   37.17   37.57   120.57   5.99  DH
7     45.37   10.76   29.04   34.61   117.27 -10.68  DH
```

資料前處理

- 將資料加入標籤，並將三種類別分別以數字表示

```
data.columns = [
    'pelvic_incidence', 'pelvic_tilt',
    'lumbar_lordosis_angle',
    'sacral_slope', 'pelvic_radius', 'grade_of_spondylolisthesis', 'labels'
]
for i in range(0,len(data)):
    if(data['labels'][i] == 'DH'):
        data['labels'][i] = 1
    elif(data['labels'][i] == 'SL'):
        data['labels'][i] = 2
    elif(data['labels'][i] == 'NO'):
        data['labels'][i] = 3
print(data.head)
```

資料前處理

- 資料集前六列為特徵、第七列為標籤，並將資料集做分割。

```
all_attribute = data.iloc[:, 0:6]
all_label = data.iloc[:, 6]
all_label = all_label.values.reshape((310))
print('all_attribute : ', all_attribute.shape)
print('all_label : ', all_label.shape)
```

```
all_attribute :  (310, 6)
all_label :  (310,)
```

```
DH = all_attribute.iloc[0:60, :]
DH_label = all_label[0:60]

SL = all_attribute.iloc[60:210, :]
SL_label = all_label[60:210]

normal = all_attribute.iloc[210:310, :]
normal_label = all_label[210:310]

print('DH = ', DH.shape)
print('SL = ', SL.shape)
print('normal = ', normal.shape)
```

```
DH =  (60, 6)
SL =  (150, 6)
normal =  (100, 6)
```

資料前處理

- 訓練集(80%)
- 測試集(20%)

```
DH_train = DH.iloc[0:48, :]
DH_train_label = DH_label[0:48]
DH_test = DH.iloc[48:60, :]
DH_test_label = DH_label[48:60]

SL_train = SL.iloc[0:120, :]
SL_train_label = SL_label[0:120]
SL_test = SL.iloc[120:150, :]
SL_test_label = SL_label[120:150]

normal_train = normal.iloc[0:80, :]
normal_train_label = normal_label[0:80]
normal_test = normal.iloc[80:100, :]
normal_test_label = normal_label[80:100]

print('DH_train = ', DH_train.shape)
print('DH_test = ', DH_test.shape)
print('SL_train = ', SL_train.shape)
print('SL_test = ', SL_test.shape)
print('normal_train = ', normal_train.shape)
print('normal_test = ', normal_test.shape)
```

```
DH_train =  (48, 6)
DH_test =  (12, 6)
SL_train =  (120, 6)
SL_test =  (30, 6)
normal_train =  (80, 6)
normal_test =  (20, 6)
```

資料前處理

- 將訓練集與測試集的資料做整合。

```
train = np.vstack((normal_train, DH_train, SL_train))
label = np.hstack((normal_train_label, DH_train_label, SL_train_label))
test = np.vstack((normal_test, DH_test, SL_test))
test_label = np.hstack((normal_test_label, DH_test_label, SL_test_label))
```

- 將資料型態 float 轉為 int。

```
train = train.astype(int)
label = label.astype(int)
test = test.astype(int)
test_label = test_label.astype(int)
```

模型參數設定

- kernel：核函式型別 (預設為rbf)
- C：懲罰係數，即對誤差的寬容度。
- decision_function_shape：決策函式型別

```
clf = SVC(kernel='rbf',C=100,decision_function_shape=None)
clf.fit(train,label)
```

```
C:\Users\Lab905\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: Fu
'auto' to 'scale' in version 0.22 to account better for unscaled feature
his warning.
  "avoid this warning.", FutureWarning)
```

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto_deprecated',
    kernel='rbf', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

測試結果

- 最初結果

```
prediction = clf.predict(test)
```

```
from sklearn import metrics
print('Accuracy: ', metrics.accuracy_score(test_label, prediction))
```

```
Accuracy:  0.4838709677419355
```

➢準確度為 48.38%

參數調整

- kernel：rbf → poly

Accuracy：48.38% → 74.19%

```
clf = SVC(kernel='poly',C=100,decision_function_shape=None)
clf.fit(train,label)
```

```
C:\Users\Lab905\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: Futu
'auto' to 'scale' in version 0.22 to account better for unscaled feature
his warning.
  "avoid this warning.", FutureWarning)
```

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto_deprecated',
    kernel='poly', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
prediction = clf.predict(test)
```

```
from sklearn import metrics
print('Accuracy: ', metrics.accuracy_score(test_label, prediction))
```

```
Accuracy:  0.7419354838709677
```

參數調整

- kernel：poly → linear　　Accuracy：74.19% → 80.64%

```
clf = SVC(kernel='linear',C=100,decision_function_shape=None)
clf.fit(train,label)
```

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape=None, degree=3, gamma='auto_deprecated',
    kernel='linear', max_iter=-1, probability=False, random_state=None,
    shrinking=True, tol=0.001, verbose=False)
```

```
prediction = clf.predict(test)
```

```
from sklearn import metrics
print('Accuracy: ', metrics.accuracy_score(test_label, prediction))
```

```
Accuracy:  0.8064516129032258
```

參數調整

- C：100 → 0.01

Accuracy：80.64% → 87.09%

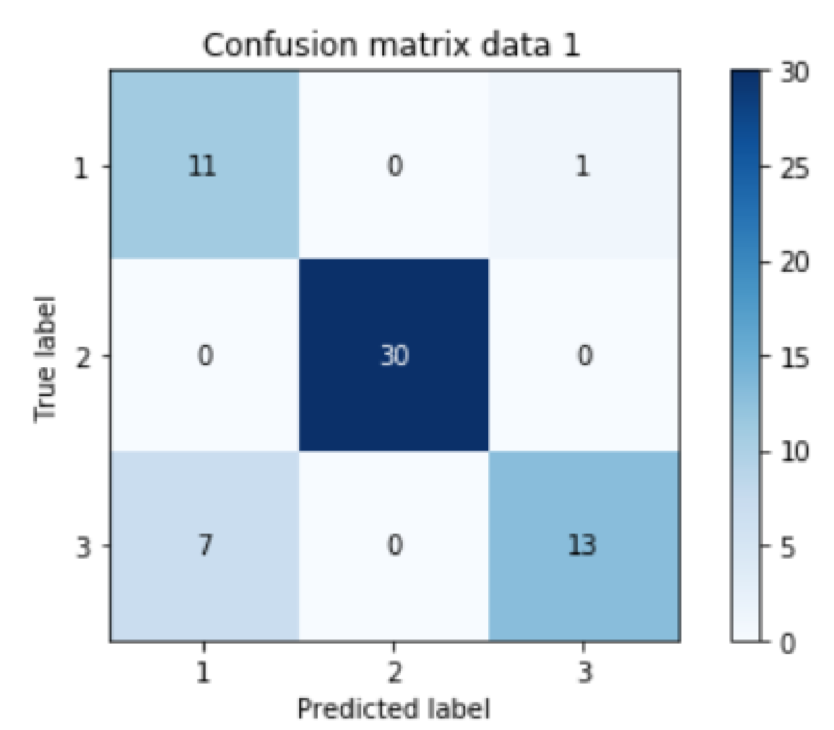
```
clf = SVC(kernel=                                                  pe=None)
clf.fit(train,lab

SVC(C=0.01, cache_                                                  0,
     decision_func                                                  _deprecated',
     kernel='linea                                                  ndom_state=None,
     shrinking=Tru

prediction = clf.

from sklearn impo
print('Accuracy:                                                    prediction))

Accuracy:  0.8709
```



結果統整

改善步驟	準確度
kernel：rbf → poly	48.38% → 74.19%
kernel：poly → linear	74.19% → 80.64%
C：100 → 0.01	80.64% → **87.09%**

討論

- 模型核函式型別的選擇對於準確度的影響。

linear kernel	線性可分時，特徵數量多時，樣本數量多再補充一些特徵時，linear kernel可以是RBF kernel的特殊情況
Polynomial kernel	image processing，引數比RBF多，取值範圍是(0,inf)
Gaussian radial basis function (RBF)	通用，線性不可分時，特徵維數少 樣本數量正常時，在沒有先驗知識時用，取值在[0,1]

NTHU IEEM

討論

- 遇到的困難
  - SVM了解不足
  - 程式debug問題
- 解決方式
  - 透過網路找尋相關資訊

未來可改進方向

- 了解每個特徵的權重，利用更少特徵就能準確判斷是否有問題。
- 增加資料數據

參考資料

- python機器學習庫sklearn——支援向量機svm
  https://www.itread01.com/content/1549429381.html
- SVM的實現多分類的幾種方法以及優缺點詳解
  https://www.itread01.com/content/1546997242.html
- SVM 的核函式選擇和調參
  https://www.itread01.com/content/1550635205.html
- UCI資料集
  http://archive.ics.uci.edu/ml/datasets/vertebral+column

Thank You for Your Listening