レポート提出票

科目名: 情報工学実験3

実験課題名: 課題2 パターン認識

実施日: 2021年 6月 10日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者:

1　はじめに

パターン認識とは、入力データをあらかじめ定めていた複数のクラスの 1 つに対応させる処理のことをいう。分類を行うクラスの種類に応じて、パターン認識は大きく、「分類問題」と「予測問題」に分けることができる。また、分類問題、予測問題の両方に関して、「教師あり学習」と「教師なし学習」の 2 つが存在しており、本実験では、入力データと出力の組みが与えられているデータを利用したパターン認識である「教師あり学習」について理解する。

2　授業内課題

2.1　課題内容

(1) 配布した「pima_tr.csv」と「pima_te.csv」を読み込み、2 つのデータを縦方向に結合し、その後 1 列目を削除する。また、結合したデータの type 以外の列を標準化する。

(2) k 近傍法において、$k = 3$ に固定し、このときの訓練データに対する True Positive Rate を求める。また、テストデータに対する True Positive Rate も求める。

2.2　手順

1. Jupiter Notebook 上で Python のプログラムを作成する。
2. 作成したソースコードを実行する。

2.3　結果

作成したソースコードは付録のソースコード 1 と 2 に載せた。ソースコード 2 を実行した結果、k 近傍法において、$k = 3$ に固定したときの訓練データに対する True Positive Rate は、

$$\text{True Positive Rate} \quad = \quad 0.75$$

また、テストデータに対する True Positive Rate は、

$$\text{True Positive Rate} \quad = \quad 0.484848$$

となった。

2.4 考察

訓練データとテストデータの True Positive Rate を比較すると、訓練データの方が高かった。これは、偶然なったわけではなく必然である。理由は、訓練データを学習させたあとに、訓練データとテストデータの True Positive Rate を求めたからである。

訓練データを学習させた後にテストデータの True Positive Rate がどうなっているかを見ると、0.484848 であった。つまり、糖尿病を判断できるのはほぼ半分であると分かる。逆に言えば糖尿病なのに糖尿病と判定されない人が半分もいることになる。したがって、この学習モデルを用いるのは適切でないと考える。

3 レポート課題

3.1 課題内容

(1) 2 クラス分類問題の場合、評価指標として混同行列が使われるが、多クラス分類の際の評価指標について調べ、まとめる。

(2) True Positive Rate を高めることを目標にして、訓練データとテストデータに Stratified になるように 8:2 の割合で分割する。さらに、k 近傍法を用いて、分割した訓練データに対して交差検証を行い最適な k の値を求め、テストデータに対する正解率を予測し、その時の True Positive Rate を求める。

3.2 手順

(1) 検索をして、文献を探す。

(2) 1. Jupiter Notebook 上で Python のプログラムを作成する。
2. 作成したソースコードを実行する。

3.3 結果

(1)

多クラス分類の際も、最初は同じように混同行列作成する。計算するときは、クラス分の True Positive や False Positive などを作成する。2 つのクラスと同様に正しく判定されたものは True Positive として、それ以外はすべて False Positive とする。

多クラス分類の特別な評価指標として、マクロ平均とマイクロ平均がある。マクロ平均は各クラスの評価指標の平均を計算しているため、各クラスの平均の評価指標を表している。各クラスにサンプル数に偏りがあっても、各クラスが公平に扱われる。たとえば、マクロ平均適合率は、クラス L_i の適合率を P_i とすると、

$$\text{マクロ平均適合率} = \frac{\Sigma_{i=1}^{N} P_i}{N}$$

と表される。

マイクロ平均はデータ全体を考慮して計算される指標であり、データセット全体に対する平等的な評価指標といえる。しかし、各クラスのサンプルの数に偏りが大きい場合、サンプル数の大きいクラスの性能が全体の性能を代表してしまう可能性がある。たとえば、マイクロ平均適合率は、クラス L_i の True Positive を TP_i として、False Positive を FP_i とすると、

$$\text{マイクロ平均適合率} = \frac{\Sigma_{i=1}^{N} TP_i}{\Sigma_{i=1}^{N} (TP_i + FP_i)}$$

と表される。

(2)

作成したソースコードは付録のソースコード 3 に載せた。実行した結果は、交差検証したときのグラフとその中で最適な k の値を表示し、その k の値の時のテストデータに対する Accuracy Score と True Positive Rate を表示した。

図 1 は、交差検証したときの MSE の大きさを見るグラフである。

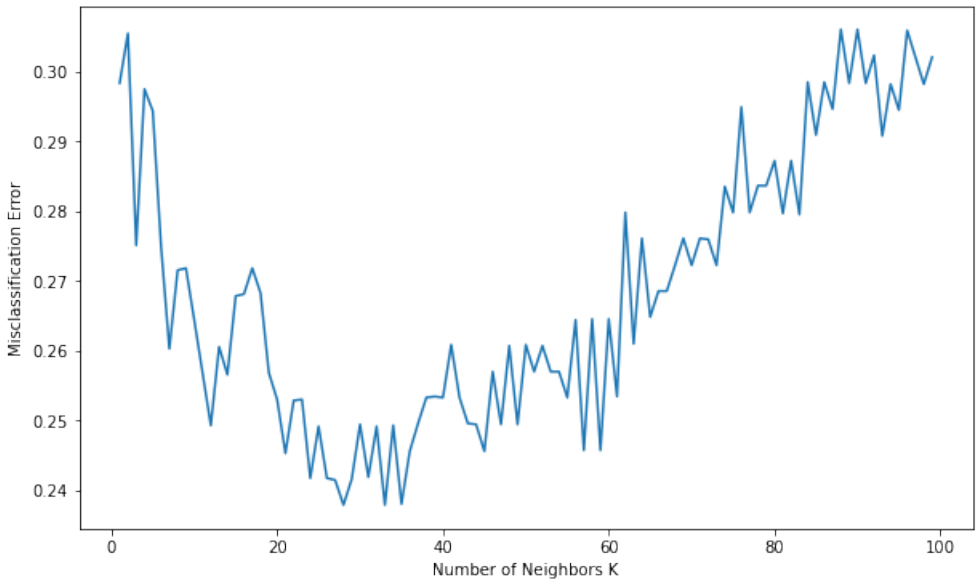

図 1: 交差検証のグラフ

交差検証より、最適な k の値は $k = 33$ と分かった。また、$k = 33$ の時のテストデータに対する Accuracy Score と True Positive Rate は、

$$\begin{aligned} \text{Accuracy Score} &= 0.835821 \\ \text{True Positive Rate} &= 0.545455 \end{aligned}$$

となった。

3.4 考察

授業内課題で作成した学習モデルと比較すると、True Positive Rate の値が高くなっていることが分かる。これは最適な $k = 33$ に変更したことにより良くなっているということである。

4　感想

今回の講義動画に関しては、実際にコードを実行しながら説明をしていたので分かりやすかった。改善はあまりしなくていいと考えるが、あえて言うならば、好みがわかれるが自分的には動画を 1 つにまとめてくれる方が見やすかった。

参考文献

[1] 多クラス分類の性能指標

https://kunsen.net/2021/03/18/post-3653/#i-2

最終閲覧日 2021/06/12

[2] 多クラスの評価指標 — 多クラス予測モデルを評価するときに指標のマクロ平均あるいはマイクロ平均を使う

https://axa.biopapyrus.jp/machine-learning/model-evaluation/multiclass-evaluation.html

最終閲覧日 2021/06/12

A 付録

ソースコード 1: 授業内課題 (1)

```
import pandas as pd
from sklearn import preprocessing
pima_train = pd.read_csv('data/pima_tr.csv',encoding='UTF-8')
pima_test = pd.read_csv('data/pima_te.csv',encoding='UTF-8')

group_data = pd.concat([pima_train, pima_test], ignore_index=True)
del group_data['Unnamed: 0']
X = preprocessing.scale(group_data[['npreg','glu','bp','skin','bmi','ped','age']])
```

ソースコード 2: 授業内課題 (2)

```
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
pima_train = pd.read_csv('data/pima_tr.csv',encoding='UTF-8')
pima_test = pd.read_csv('data/pima_te.csv',encoding='UTF-8')

group_data = pd.concat([pima_train, pima_test], ignore_index=True)
del group_data['Unnamed: 0']

X = preprocessing.scale(group_data[['npreg','glu','bp','skin','bmi','ped','age']])
y = group_data.type

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, train_size=0.7,
    stratify=y)

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_train)
cmat = confusion_matrix(y_train, y_pred, labels=['Yes', 'No'])
TP, FN = cmat[0, 0], cmat[0, 1]
print('訓練データに対する True Positive Rate:',TP/(TP+FN))

y_pred = knn.predict(X_test)
cmat = confusion_matrix(y_test, y_pred, labels=['Yes', 'No'])
TP, FN = cmat[0, 0], cmat[0, 1]
print('テストデータに対する True Positive Rate:',TP/(TP+FN))
```

ソースコード 3: レポート課題 (2)

```
import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
%matplotlib inline
pima_train = pd.read_csv('data/pima_tr.csv',encoding='UTF-8')
pima_test = pd.read_csv('data/pima_te.csv',encoding='UTF-8')

group_data = pd.concat([pima_train, pima_test], ignore_index=True)
del group_data['Unnamed: 0']

X = preprocessing.scale(group_data[['npreg','glu','bp','skin','bmi','ped','age']])
y = group_data.type

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, train_size=0.8,
    stratify=y)

neighbors = list(range(1, 100))
cv_scores = []
for k in neighbors:
    knn = KNeighborsClassifier(n_neighbors = k)
    scores = cross_val_score(
        knn,
        X_train,
        y_train,
        cv = 10,
        scoring = 'accuracy',
        n_jobs = -1
    )
    cv_scores.append(np.mean(scores))

MSE = [1 - x for x in cv_scores]

optimal_k = neighbors[MSE.index(min(MSE))]
print('The optimal number of k is %d' % optimal_k)

fig, ax = plt.subplots(figsize = (10, 6))
ax.plot(neighbors, MSE)
ax.set_xlabel('Number of Neighbors K')
ax.set_ylabel('Misclassification Error')

knn = KNeighborsClassifier(n_neighbors = optimal_k)
knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)
cmat = confusion_matrix(y_test, y_pred, labels=['Yes', 'No'])
TP, FN = cmat[0, 0], cmat[0, 1]
print('テストデータに対する True Positive Rate:',TP/(TP+FN))

print('テストデータに対する Accuracy Score:',accuracy_score(y_test, y_pred))
```