Psychoinformatics - Week 9 (Exercises)

by Yung-Yi Hsu (r12227104@ntu.edu.tw)

In [ ]:
```
import numpy as np
from sklearn import *
from sklearn import model_selection
from matplotlib.pyplot import *
%matplotlib inline
```

1 檢查 machine learning pipeline (8 points)

1.1 請打亂原本的Y觀察正確率是否和chance level (0.33)有差異? 若有, why? (4 points)

In [ ]:
```
# 本題在研究打亂X和打亂Y有差別嗎?
iris = datasets.load_iris()
X=iris.data
Y=iris.target
Y2=np.random.permutation(Y)
print(Y)
clf=neighbors.KNeighborsClassifier(1)
clf.fit(X,Y2)
accuracy=np.mean(clf.predict(X)==Y2)
print(accuracy)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
1.0
```

討論一

結果發現正確率為100%，因為模型訓練的時候是拿X與Y2使用k-NN(1-NN)分類演算法去訓練，理所當然地在拿X預測Y2時會得到100%正確。 細節說明如下： k-NN演算法是以最接近的k個資料點去分類資料，當k=1表示只用1個點做預測，因此當用模型去預測X時，模型會選擇X本身（最接近的點就是自己）對應的Y2，故理論上和實際結果皆得到100%正確率。

In [ ]:
```
#用不同的k值來看看用X預測Y的準確度會不會有差別
for i in range(2,6,1):
    clf2=neighbors.KNeighborsClassifier(i)
    clf2.fit(X,Y)
    acc2=np.mean(clf2.predict(X)==Y)
    print(f"K={i}, accuracy={round(acc2,2)}")
```

```
K=2, accuracy=0.98
K=3, accuracy=0.96
K=4, accuracy=0.96
K=5, accuracy=0.97
```

In [ ]:
```
#用不同的k值來看看用X預測打亂的Y的準確度會不會有差別
for i in range(2,6,1):
    clf3=neighbors.KNeighborsClassifier(i)
    clf3.fit(X,Y2)
    acc3=np.mean(clf3.predict(X)==Y2)
    print(f"K={i}, accuracy={round(acc3,2)}")
```

```
K=2, accuracy=0.69
K=3, accuracy=0.69
K=4, accuracy=0.61
K=5, accuracy=0.57
```

討論二

結果發現當k值增加時，以X預測Y的正確率仍非常高（皆 > 0.95）。 但若將Y打散（即Y2），以X預測Y的正確率則下降到0.6-0.7左右，且有k越大正確率越低的趨勢 細節說明如下： 因為已知X與Y確實存在關係（鳶尾花的品種與花萼花瓣長寬間有相關），因此即使k值增加，都能大致正確。 而Y2（打散的Y）是X的隨機標籤，兩者理論上並無相關，因此k越多時，採用的無效資料點越多（除了自己都是無效點），因此正確率下降，且有k越大正確率越低的趨勢。

In [ ]:
```
# 用打亂的Y訓練能預測正確的Y嗎?
import scipy.stats as stats
# shuffle Y and calculate predict accuracy
clf4=neighbors.KNeighborsClassifier(1)
acclist=[]
for i in range(30):
    Yshuffle=np.random.permutation(Y)
    clf4.fit(X,Yshuffle)
    acc = np.mean(clf4.predict(X)==Y)
    acclist.append(acc)

meanacc = np.mean(acclist)
print(round(meanacc,4))
#t-test
t = stats.ttest_1samp(a=acclist, popmean=1/3)
print(f"t({t.df}) = {round(t.statistic,4)}, p-value = {round(t.pvalue,4)}")
```

```
0.3282
t(29) = -0.6641, p-value = 0.5119
```

討論三

將X及打散後的Y作為k-NN(1-NN)的訓練資料產生模型，並拿此模型預測X對應的Y。 結果發現30次訓練中，平均正確率為0.3282，與1/3(chance level) 相當接近（差異值僅約0.0051）。 將30筆正確率與1/3(chance level) 做one-sample t-test，統計上沒有顯著差異（p-value = 0.5119）。 細節說明如下： 因為打散後的Y是X的隨機標籤，因此在預測X對應的Y上，理論上和實際上都是無效的，故正確率大致為chance level(0.33)。

結果簡述

1. 即使打散Y和X沒有相關，1-NN模型仍能正確的用X預測打散Y（自己預測自己）。 2. 當k變大時，k-NN用X預測打散Y（兩者無相關）的正確率變低，而X預測Y（兩者高相關）則保持非常高的正確率。 3. 若用打散Y訓練1-NN模型，然後輸入X預測Y，得到的正確率為chance level。

1.2 請用母數或無母數統計檢定以下accuracies中的結果是否和chance level (0.5)有差異? 若有, why? (4 points)

In [ ]:
```
Y3=np.remainder(range(200),2)
print(Y3) #Y3的0和1個數一樣多
```

```
[0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1]
```

In [ ]:
```
# 跑一百次測試:
clf=svm.SVC()
accuracies=[]
for i in range(100):
 X=np.random.rand(200,2) # X取亂數
 kf=model_selection.KFold(len(Y3),shuffle=True) # Leave-one-out cross-validation
 sc=model_selection.cross_val_score(clf,X,Y3,cv=kf)
 accuracies.append(sc.mean())
```

In [ ]:
```
# According to central limit theorem, the distribution of the mean of a large nu
# So, we can use t-test to test the mean of accuracies.
# t-test
from scipy.stats import ttest_1samp
t2 = ttest_1samp(accuracies,0.5)
accmean2 = np.mean(accuracies)
print(round(accmean2,4))
print(f"t({t2.df}) = {round(t2.statistic,4)}, p-value = {round(t2.pvalue,4)}")
```

```
0.48
t(99) = -2.8157, p-value = 0.0059
```

討論一

結果發現正確率的平均值為0.48，略低於平均值 t-test結果發現有顯著差異，p-value約為0.006 因為使用Leave-one-out cross-validation，在0和1一樣多的情況下，testing data的正確答案在training data的比例較低 (99:100)，因此在X和Y3無關的情況下，正確率會略低於chance level。

In [ ]:
```
#測試不同比例的Y

Y4 = np.array([0] * 120 + [1] * 80)
Y4 = np.random.permutation(Y4)
print(Y4)
# 跑一百次測試:
clf=svm.SVC()
accuracies2=[]
for i in range(100):
 X=np.random.rand(200,2) # X取亂數
```

```
    kf=model_selection.KFold(len(Y4),shuffle=True) # Leave-one-out cross-validation
    sc=model_selection.cross_val_score(clf,X,Y4,cv=kf)
    accuracies2.append(sc.mean())
t3 = ttest_1samp(accuracies2,0.5)
accmean3 = np.mean(accuracies2)
print(round(accmean3,4))
print(f"t({t3.df}) = {round(t3.statistic,4)}, p-value = {round(t3.pvalue,4)}")
```

```
[0 1 1 0 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1
 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0
 0 1 1 0 0 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1
 0 0 1 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0
 0 1 0 0 1 0 1 1 1 0 0 1 1 0 0 1 0 0 1 1 0 1 1 1 0 1 0 1 1 1 1 0 1 1 0 1 1
 1 0 0 0 0 0 1 0 1 0 1 0 0 1 0]
0.5846
t(99) = 28.4879, p-value = 0.0
```

In [ ]:

```
Y5 = np.array([0] * 150 + [1] * 50)
Y5 = np.random.permutation(Y5)
print(Y5)
# 跑一百次測試:
clf=svm.SVC()
accuracies3=[]
for i in range(100):
    X=np.random.rand(200,2) # X取亂數
    kf=model_selection.KFold(len(Y5),shuffle=True) # Leave-one-out cross-validation
    sc=model_selection.cross_val_score(clf,X,Y5,cv=kf)
    accuracies3.append(sc.mean())
t4 = ttest_1samp(accuracies3,0.5)
accmean4 = np.mean(accuracies3)
print(round(accmean4,4))
print(f"t({t4.df}) = {round(t4.statistic,4)}, p-value = {round(t4.pvalue,4)}")
```

```
[0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 1
 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0
 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0
 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0
 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0]
0.7496
t(99) = 935.3144, p-value = 0.0
```

討論二

試著用0和1的比例不同的Y和亂數X來進行Leave-one-out cross-validation。 在0和1的比例為3:2時(Y4, 120:80)，正確率的平均值為0.5846； 在0和1的比例為3:1時(Y5, 150:50)，正確率的平均值為0.7496。 發現比例越不平衡時，正確率越高。 細節說明： 當Y的比例越不平衡時，在X和Y無關的情況下，模型應該越有高的傾向將資料歸類為數量多的類別 (模型中為0)，同時數量多的類別(0)是testing data的正確答案的比例也越高，因此正確率越高。

In [ ]:

```
Y6 = np.array([0] * 100 + [1] * 100 + [2] *100)
Y6 = np.random.permutation(Y6)
print(Y6)
# 跑一百次測試:
clf=svm.SVC()
accuracies4=[]
for i in range(100):
```

```
 X=np.random.rand(300,2) # X取亂數
 kf=model_selection.KFold(len(Y6),shuffle=True) # Leave-one-out cross-validation
 sc=model_selection.cross_val_score(clf,X,Y6,cv=kf)
 accuracies4.append(sc.mean())
t5 = ttest_1samp(accuracies4,1/3)
accmean5 = np.mean(accuracies4)
print(round(accmean5,4))
print(f"t({t5.df}) = {round(t5.statistic,4)}, p-value = {round(t5.pvalue,4)}")
```

```
[1 2 2 1 2 1 0 1 2 2 0 0 0 2 2 1 0 0 0 0 2 2 1 1 1 0 2 1 2 0 0 0 1 0 0 1 2
 2 1 2 2 2 0 1 2 2 2 1 1 2 1 2 0 2 0 0 2 0 0 2 1 1 0 1 2 1 1 0 1 0 1 1 1 0
 1 1 0 1 0 2 1 2 2 1 2 0 2 1 1 0 2 1 1 2 1 1 0 0 2 1 1 2 0 0 0 0 1 0 1 1 1
 0 2 0 2 1 2 1 2 1 2 2 2 0 1 0 1 2 1 0 1 2 0 2 0 2 0 0 1 1 0 0 1 1 2 0 2 0
 1 0 1 2 2 2 1 0 0 0 2 0 2 1 2 2 0 0 0 0 0 2 2 0 2 0 1 1 0 2 0 0 1 0 2 2 1
 0 1 0 1 2 0 0 2 1 2 0 0 0 1 2 2 2 1 2 1 1 1 0 0 1 2 1 2 2 1 2 2 2 1 2 1 2
 0 1 1 2 1 1 0 1 2 1 0 2 1 0 0 2 1 1 0 2 2 2 1 2 2 0 1 2 0 2 0 1 2 1 0 2 1
 0 2 0 0 1 2 0 2 0 1 2 2 1 2 0 1 2 0 0 0 1 0 0 0 2 1 2 1 0 0 0 1 2 2 0 1 2
 1 0 1 1]
0.312
t(99) = -4.5837, p-value = 0.0
```

In [ ]:
```
print(t5)
```

```
TtestResult(statistic=-4.583748905283841, pvalue=1.3354060839215416e-05, df=99)
```

討論三

測試當Y有等量三類別時，和亂數X來進行Leave-one-out cross-validation的正確率 結果發現平均正確率為0.312，統計上達顯著(t(99) = -4.5831, p-value < 0.0001) 與我在討論一中說明的預期結果相符（Y比例一致時，正確率會顯著略低於平均）

總結論

當X和Y無關時，用Leave-one-out cross-validation建立X預測Y模型的正確率受Y中類別的比例影響。 如果Y是等比例的，正確率會略低於chance level；而當Y的比例越懸殊時，正確率越高。

Please submit your notebook in PDF to NTU Cool by next Friday (11/10).