

通識計算機程式設計期中考

臺灣大學 鄭士康 11/11/2016

試題共 7 題，兩面印製 11 頁，滿分 100



本講義除另有註明外，採創用CC姓名標示-非商業性-相同方式分享3.0臺灣版授權釋出

1. 撰寫一或數個C#敘述達成下列要求: (假設`using System;`敘述已經包含於程式中)
 - (a) 宣告 `bool` 變數 `valid`，`int` 變數 `xi`，`byte` 變數 `xb` (3%)
 - (b) 在螢幕顯示一行字，要求使用者輸入一個 0 到 255 之間的整數值 (3%)
 - (c) 自鍵盤讀入一個整數，並將其值存入已宣告之 `int` 變數 `xi` (3%)
 - (d) 將邏輯關係式 `xi >= 0 && xi <= 255` 設定給已宣告之 `bool` 變數 `valid` (3%)
 - (e) 寫一個 `if-else` 敘述，當已宣告之 `bool` 變數 `valid` 為 `true` 時，將已宣告設值之 `int` 變數 `xi` 強制轉型為 `byte` 數值，存入已宣告之 `byte` 變數 `xb`，否則令 `xb` 為 0 (3%)
2. 撰寫一或數個C#敘述達成下列要求: (假設`using System;`敘述已經包含於程式中)
 - (a) 將已宣告設值之 `int` 變數 `m`，用遞減算子 `--` 減 1 (3%)
 - (b) 令他處已宣告設值之 `double` 變數 `x1` 及 `x2`，由 `x1` 減去 `x2` 後，設定給他處已宣告之 `double` 變數 `dx` (3%)
 - (c) 宣告 `double` 變數 `distance`，並設定其值為單一算式 (expression): $\sqrt{dx^2 + dy^2}$ ，假設 `dx` 及 `dy` 已於他處宣告設值 (3%)
 - (d) 利用三元運算子，使他處已宣告設值之 `double` 變數 `x`，數值大於等於 0 時，設定變數 `y` 的數值為 `x`，反之則令 `y` 值為 0 (3%)
 - (e) 宣告變數 `c` 為 `char` 型別，並令其值為單引號「'」 (apostrophe) (3%)
3. 撰寫一或數個C#敘述達成下列要求: (假設`using System;`敘述已經包含於程式中)
 - (a) 先以一個敘述宣告一個亂數產生器物件`rand`，讓系統以時間及網路卡ID決定其種子數 (3%)
 - (b) 宣告一個 `int` 常數 `N`，設其值為10。同時宣告一個長度為 `N` 的一維

bool 陣列 **labels** (3%)

(c) 寫一個 **for** 迴圈，整數迴圈控制變數 **i** 由 0 開始，每次加 1，遞增至 **N-1**。迴圈中每個 iteration 以 **rand.Next()** 產生一個隨機整數。如果隨機整數是奇數，則設 **bool** 陣列 **labels** 的第 **i** 個元素為 **true**，否則設為 **false**。假定迴圈前已宣告設值 **int** 常數 **N**，以及長度 **N** 的 **bool** 陣列 **labels** (3%)

(d) 宣告一個長度為 **N** 的 **double** 陣列 **distances**。利用 **for** 迴圈，在迴圈內的第 **i** 個 iteration，使用 **rand.NextDouble()** 產生一個 0 與 1 之間均勻分布的隨機小數，設值給 **distances[i]** (3%)

(e) 寫一個 **static bool** 函式 **LabelOfNearestElement**，設其輸入參數為一維 **double** 陣列 **distances** 及長度相同的一維 **bool** 陣列 **labels**。以 **distances** 為 key，**labels** 為 term，利用 **Array.Sort()** 將 **distances** 由小而大排列，同時對應的 **labels** 也同步隨之移動。找出最小的 **distances** 元素，傳回 (return) 對應的 **labels** 元素 (3%)

4. 找出以下程式片段之錯誤，並在盡量保持原先程式碼之前提下，予以更正。

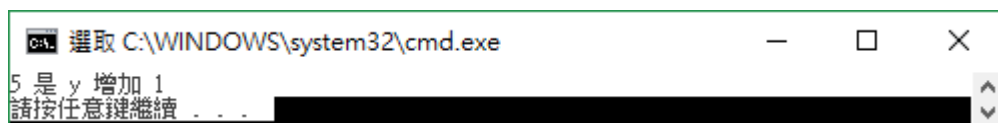
假設 **using System;** 敘述已經包含於程式中。

(a) (3%) (一個語義錯誤) 執行時螢幕應顯示



```
Console.WriteLine("請輸入學號: ");  
string registerNumber = Console.ReadLine();
```

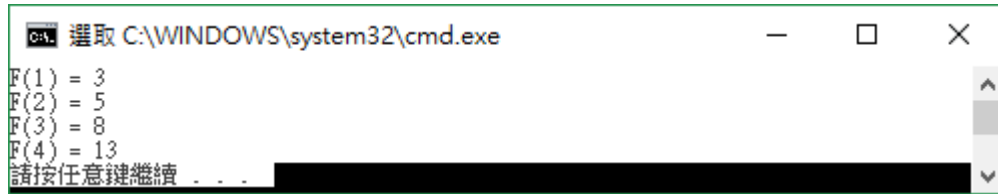
(b) (3%) (一個語義錯誤) 執行時螢幕應顯示



```
int y = 4;  
int z = y++;  
Console.WriteLine("{0} 是 y 增加 1", z);
```

(c) (3%) (一個語義錯誤) 執行時螢幕應顯示前四個 Fibonacci 數列項:

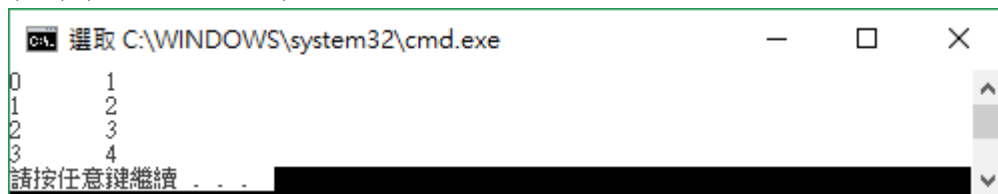
$F(n) = F(n-1) + F(n-2)$



```
選擇 C:\WINDOWS\system32\cmd.exe
F(1) = 3
F(2) = 5
F(3) = 8
F(4) = 13
請按任意鍵繼續...
```

```
int f_n_minus2 = 1;
int f_n_minus1 = 2;
int f;
int n = 1;
do
{
    f = f_n_minus1 + f_n_minus2;
    Console.WriteLine("F({0}) = {1}", n, f);
    f_n_minus2 = f_n_minus1;
    f_n_minus1 = f;
} while(n <= 4);
```

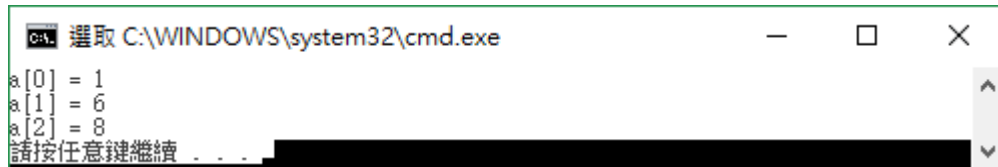
(d) (3%) (一個語義錯誤) 執行時螢幕應顯示如下:



```
選擇 C:\WINDOWS\system32\cmd.exe
0 1
1 2
2 3
3 4
請按任意鍵繼續...
```

```
int[,] table = new int[4, 2];
for (int i = 0; i < table.GetUpperBound(0); ++i)
{
    for(int j = 0; j <= 1; ++j)
    {
        table[i, j] = i + j;
    }
    Console.WriteLine(
        table[i, 0] + "\t" + table[i, 1]);
}
```

(e) (3%) (一個語義錯誤) 執行時螢幕應顯示如下:



```
static void Main(string[] args)
{
    int[] a = {1, 2, 3, 4, 5};
    F00(a);
    for(int i = 0; i < a.Length; ++i)
    {
        Console.WriteLine("a[" + i + "] = " + a[i]);
    }
}

static void F00(int[] a)
{
    for(int i = 0; i < a.Length; ++i)
    {
        a[i] += 2;
    }
    a = new int[3] {1, 6, 8};
}
```

5. 試寫出下列程式的螢幕輸出 (5 %)

```
using System;

namespace Problem5
{
    class Program
    {
        static void Main(string[] args)
        {
            double x = 0.8;
            double r = 0.5;
            const int N = 4;
            for(int n = 0; n < N; ++n)
```

```

    {
        Console.WriteLine("n = {0}, x = {1:F2}", n, x);
        x = r * x * (1.0 - x); // Logistic equation
    }
}
}

```

6. 試寫出下列程式的螢幕輸出 (10 %)

```

using System;

namespace Problem6
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] lyrics = {
                "Way down upon de Swanee Ribber,",
                "Far, far away,",
                "Dere's wha my heart is turning ebber,",
                "Dere's wha de old folks stay."
            };

            string[] words = new string[30];
            int n = 0;
            string[] wordsInALine;
            for (int i = 0; i < lyrics.Length; ++i)
            {
                wordsInALine = lyrics[i].Split(
                    ' ', ',', '.', '!');
                foreach (string word in wordsInALine)
                {
                    string wordInLowercase = word.ToLower();
                    if (wordInLowercase == "" ||
                        HasAppeared(wordInLowercase, words))
                        continue;
                    words[n] = wordInLowercase;
                }
            }
        }
    }
}

```

```

        ++n;
    }
}
Display(words);
}

static bool HasAppeared(string word,
    string[] words)
{
    int idx = Array.IndexOf(words, word);
    bool result = (idx >= 0);
    return result;
}

static void Display(string[] words)
{
    for(int n = 0; n < words.Length; ++n)
    {
        Console.WriteLine(words[n]);
    }
}
}
}

```

7. 自從「大數據」(big data)及「機器學習」(machine learning)廣泛應用，人類生活近年已經經歷了巨大的改變。這其中，網路商家依照所收集的消費者消費習慣，分析其喜好，推薦適當產品的技術，成為一項顯學。居家生活中，類似的方法也被用來分析書籍或多媒體閱聽者的偏好，主動提供閱聽者可能喜愛的內容。大家熟悉的 Amazon、K K Box 和 YouTube，都有這樣的服務。台大後卓越計畫的多媒體研究團隊，多年前就發展出《互動式數位內容呈現系統》：能依照閱聽者生理訊號，如皮膚導電率、肌電圖、脈搏等，判斷閱聽者喜歡當前播放的音樂與圖片的程度。較喜歡的話，繼續播放內容情緒表現特性相似的內容；反之，則逐漸更改播放內容，逐漸偏向內容情緒表現特性差不多與原先相反的音樂與圖片。詳細一些的介紹，請參看 YouTube 影片 <https://www.youtube.com/watch?v=fINyh5m2UI>。

接著我們用一般的音樂推薦系統，說明如何應用以上所說的機器學習技術。概念上需要先收集足夠多的訓練樣本(樂曲片段，每段大概 30 秒)，每個訓練樣本都附

有喜歡或不喜歡的標籤(label 或 tag)。標籤的來源可能是閱聽者自行加註，也可能是電腦由閱聽者的外在行為(例如網路消費)或內在生理訊號(例如台大的《互動式數位內容呈現系統》)來判斷。從這些附有標籤的訓練資料，系統可以根據內部模型的計算，算出一個標籤。藉由算出來的標籤是否與訓練資料所附的標籤一致，調整內部計算模型的參數(「監督式學習」，supervised learning)。系統「學習」完成的計算模型，可以相當準確的將訓練資料分類(有點像模擬考，考題都是學過的)。然而，機器學習更重要的優點，是以非訓練資料測試時，仍然可以使內部模型跑出來的喜歡或不喜歡的分類結果，達到相當的正確率(像是基測或學測，考題多半沒看過)。從以上說明，可以了解大量加註有標籤的訓練資料，是監督式機器學習的成功要件，因此許多公司收集標籤不遺餘力。現在你就知道為什麼 Win 10 常常改變開機畫面，並要你回答喜歡或不喜歡顯示的圖片吧？

一般用於機器學習的訓練或測試資料，都假設具備一套可以代表資料的「特徵」(features)。例如一段音樂的音色明亮度(brightness)、節奏快慢、旋律和諧程度等等，以訓練資料的特徵和標籤，讓機器學習。應用機器學習於音樂特色辨識系統的先驅 George Tzanetakis 在 2002 年發表的論文 G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Trans. Speech and Audio Processing*, vol. 10, no. 5, 2002, pp. 293 – 302. 中，為建立音樂樂種(genre，例如「古典」、「爵士」、「搖滾」等)自動分類系統，依照每一段音樂(約 30 秒)的訊號變化，舉出十數種特徵，以及各種特徵的計算方法。根據訓練資料及測試資料的所有特徵，運用電腦科學及統計學常見的兩三種機器學習演算法，得到 60%~80%，與一般人類相伴的樂種分類準確率。

目前學界及業界已存在非常多機器學習演算法。最容易了解，也曾應用於 Tzanetakis 與 Cook 論文中的方法，是「 k 近鄰」(k Nearest-Neighbor，簡稱 kNN)法，簡單舉例說明如下：

為簡化問題，假定每個訓練及測試資料(例如，三十秒的音樂)，只需要兩種特徵 x_1 和 x_2 描述(譬如，音樂的音色明亮度和節奏快慢，都是正實數)。應用 Excel 中的 RAND 函數所得亂數，經簡單算術，產生 20 筆訓練資料，顯示為圖 1 中 x_1 - x_2 平面上的紫色(標籤為 True)及淺藍色圓點(標籤為 False)。電腦記住這二十個圓點座標，就完成訓練。接著進行測試：假設 $k=3$ (通常取奇數)，測試用的第一段音樂特徵為(-0.2, -0.2) (圖 1 中左下角的橘色圓點)；顯然最接近的 $k=3$ 個點都是紫色圓點，所以程式會把第一段測試音樂標記為 True。第二個音樂測試片段特徵是(0.05, 0.05)，以圖 1 中央的深青綠色圓點代表。最靠近的三個圓點，一個紫色，兩個淺藍色，所以會由多數決，把第二段音樂歸類為淺藍色圓點標籤 False。第三個音樂測試片段(0.3, 0.4)，就是圖 1 右上方的紅褐色圓點。最接近的三個圓點都是淺藍色，所以標記為 False。

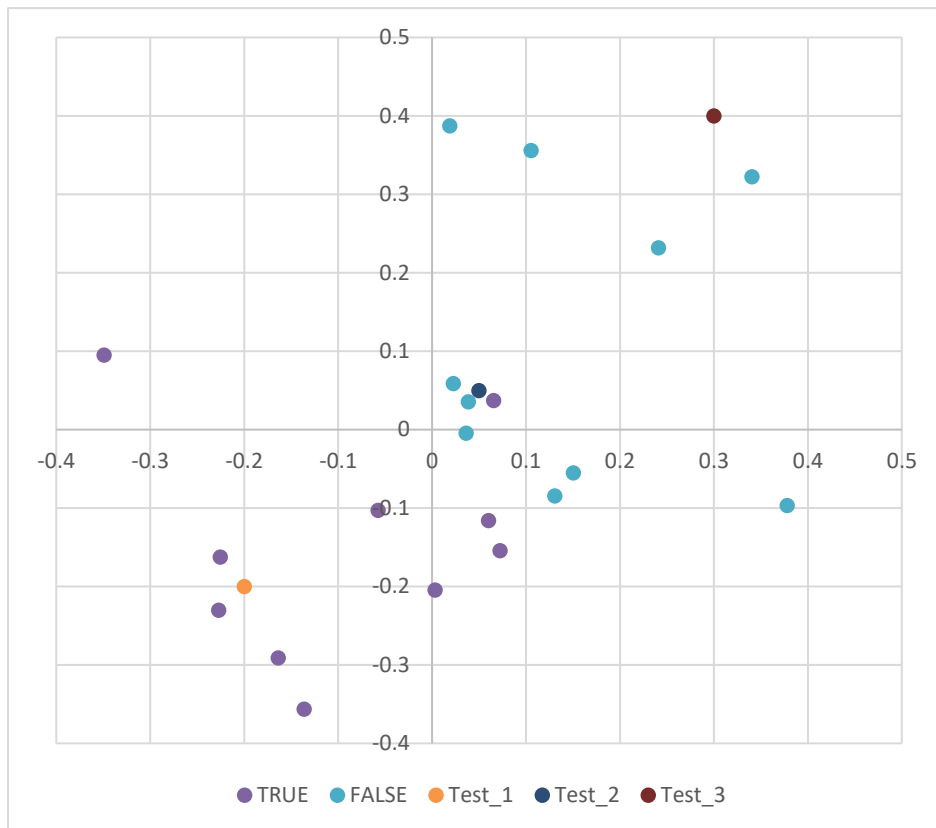


圖 1. k 近鄰法範例說明

本題請你依照如下說明，撰寫一個使用 k 近鄰法的自動標記訓練與測試資料的程式。程式要求先使用同一個亂數產生器 **rand** 產生 N 個訓練資料，其特徵 x_1, x_2 均各自以 **rand.NextDouble()** 產生一個 0 與 1 之間的亂數，存入 $N \times 2$ 的 **double** 二維矩陣，而每一筆訓練資料也用亂數產生器決定其標籤為 True，還是 False，機會各半。隨後要求使用者輸入一筆測試資料的特徵 x_1, x_2 ，程式應用上述 k 近鄰法決定測試資料的標籤並輸出，就完成本題程式。典型的程式執行畫面如圖 2。


```
選取 C:\WINDOWS\system32\cmd.exe
輸入訓練資料個數: 10
產生的訓練資料
編號      x      y      標籤
0      0.488587285153841      0.448777055576806      False
1      0.817182920788034      0.497794808120371      True
2      0.134184992934663      0.185763281390892      True
3      0.970125498236215      0.421138648139843      False
4      0.635121987031364      0.0778962299590447      True
5      0.301378485887022      0.387612989818497      False
6      0.540594956623667      0.753243196640743      False
7      0.437169486394697      0.504069823075118      True
8      0.844820867685983      0.40434863949397      False
9      0.125099904893478      0.135067653905166      True
輸入 k: 3
輸入測試資料的 x, y特徵數值, 以逗點分隔
0.2,0.5
Test data = [ 0.2, 0.5 ]      Label = False
請按任意鍵繼續 . . .
```

圖 2. 程式執行畫面一例

顯然，程式的核心部分在於如何由訓練資料的特徵分布，求出測試資料的標籤。假設這個核心部分可以寫成一個函式

```
public static bool LabelBy_k_NearestNeighbors(int k,
    double[] testData, double[,] trainingData,
    bool[] trainingLabels )
```

則在撰寫所有程式前，先以圖1所示的範例，撰寫單元測試函式如下：

```
[TestMethod]
public void Test_LabelBy_k_NearestNeighbors()
{
    double[,] trainingData = {
        {-0.225542684, -0.162490126},
        {-0.0572955, -0.102926668},
        {-0.227269149, -0.23014595 },
        {-0.136056454, -0.356289445},
        {-0.349125066, 0.095309375},
        { 0.003338106, -0.204311707},
        {-0.163670945, -0.291114938},
        { 0.072415973, -0.154194194},
```

```

        { 0.060265986, -0.115874717},
        { 0.065617297,  0.037235826},

        { 0.036272065, -0.004458326},
        { 0.018839149,  0.387165092},
        { 0.022452643,  0.058591421},
        { 0.038614911,  0.03564152 },
        { 0.378152143, -0.096746229},
        { 0.340427498,  0.322580387},
        { 0.150449783, -0.05505972 },
        { 0.241000371,  0.231978224},
        { 0.130731297, -0.084269977},
        { 0.105557276,  0.355703319}

        };

bool[] trainingLabels = {
    true,      true,      true,      true,      true,
    true,      true,      true,      true,      true,

    false,     false,     false,     false,     false,
    false,     false,     false,     false,     false
        };

double[] testData_1 = { -0.2, -0.2 };
int k = 3;
bool testLabel_1 = Program.LabelBy_k_NearestNeighbors(
    k, testData_1, trainingData, trainingLabels);
Assert.IsTrue(testLabel_1);

double[] testData_2 = { 0.05, 0.05 };
bool testLabel_2 = Program.LabelBy_k_NearestNeighbors(
    k, testData_2, trainingData, trainingLabels);
Assert.IsFalse(testLabel_2);

double[] testData_3 = { 0.3, 0.4 };
bool testLabel_3 = Program.LabelBy_k_NearestNeighbors(
    k, testData_3, trainingData, trainingLabels);
Assert.IsFalse(testLabel_3);
}

```

從以上單元測試函式和圖 1 範例的對照，你可能比較知道寫出來的程式該是甚麼樣子。如果你覺得不太適應這樣的 TDD 過程(Test-Driven Process)，照自己慣用的方式撰寫程式，只要程式邏輯正確，也可以接受。

本題滿分 25 分，全部待寫程式集中寫成一個大 **Main** 函式，不區分函式者，最高得 23 分；善用函式，乃至尚未教到的物件導向程式設計(object-oriented programming)者，最高得 25 分。(25%)