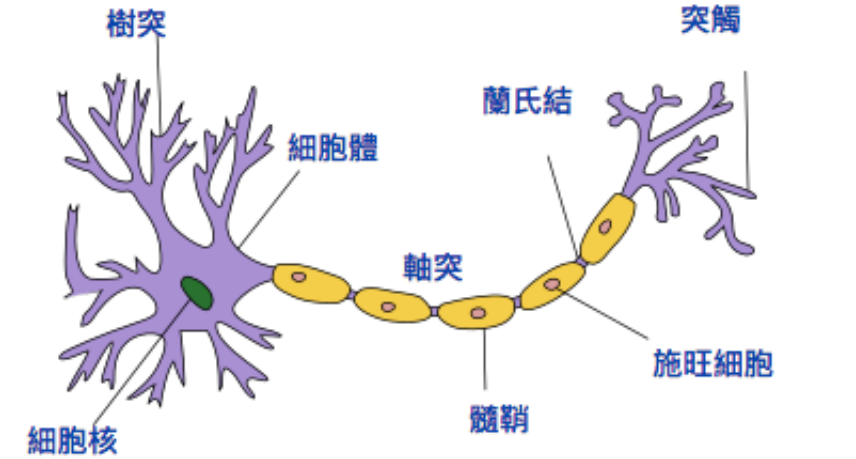
ANN_Basic

2018 年 7 月 18 日

1 初探神經網路 (感知器)

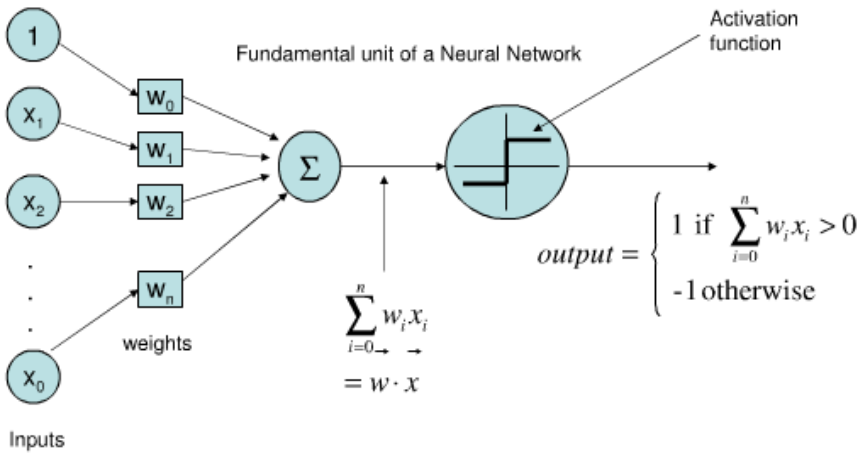
1.1 介紹

在前面的樹類和機率演算法如火如荼的發展和使用的時候，有一脈的演算法也正快速的發展中
這種類的演算法我們稱之為神經網路
為何叫神經網路呢？因為這類的演算法的基礎是一個很類似我們人的神經的一個判斷方式



典型神經元的結構

樹突接受外界刺激，達到一定程度就會激發突觸，繼續往下一層傳遞
最初的神經網路就是我們今天講的感知器 (Perceptron)，模仿單一的神經元



神經元的數學模型

簡單來說，就是把每個特徵 (刺激) 乘上一定的係數再加起來
超過一定的限度就歸類為正類，否則為負類
所以整個步驟是這樣的:

1. 我們有一個特徵向量 [a, b, c]
2. 乘上係數 score = w1 * a + w2 * b + w3 * c
3. 讓我們的 score 經過一個啟動 (激勵) 函數 (Activation Function)，這啟動函數我們先設定成最基本的超過一定的量輸出 1，否則輸出 0

1.2 需要函式庫

1. mlxtend: 可以幫我們快速畫出分類線的函式庫

1.3 資料集

我們利用之前已經用過的鳶尾花數據集，但我們這次做一點比較特別的事
我們只用花瓣和花萼的長度來分類就好
因為我們希望讓你看到決策的邊界

1.4 開始撰寫程式

1.4.1 Step 0. 讀入我們的鳶尾花數據集作為練習

```
In [1]: from sklearn.datasets import load_iris
        import pandas as pd
        import matplotlib.pyplot as plt
```

```
import seaborn as sns
%matplotlib inline

# 為了顯示的漂亮, 我刻意的把印出來的 row 只顯示 15 個和 column 只顯示 10 個
# 大家練習的時候可以去掉下面兩行
pd.set_option('display.max_rows', 15)
pd.set_option('display.max_columns', 10)

# 為了顯示的漂亮, 有時候 sklearn 會提示一下 Future Warning
# 我也把關掉了
import warnings
warnings.filterwarnings('ignore')

# 使用 scikit-learn 提供的鳶尾花資料庫
iris = load_iris()
df = pd.DataFrame(iris['data'], columns = iris['feature_names'])
df["target"] = iris["target"]
df
```

Out[1]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	\
0	5.1	3.5	1.4	0.2	
1	4.9	3.0	1.4	0.2	
2	4.7	3.2	1.3	0.2	
3	4.6	3.1	1.5	0.2	
4	5.0	3.6	1.4	0.2	
5	5.4	3.9	1.7	0.4	
6	4.6	3.4	1.4	0.3	
..	...	...	...	...	
143	6.8	3.2	5.9	2.3	
144	6.7	3.3	5.7	2.5	
145	6.7	3.0	5.2	2.3	
146	6.3	2.5	5.0	1.9	
147	6.5	3.0	5.2	2.0	
148	6.2	3.4	5.4	2.3	
149	5.9	3.0	5.1	1.8	

target

```
0         0
1         0
2         0
3         0
4         0
5         0
6         0
..      ...
143       2
144       2
145       2
146       2
147       2
148       2
149       2

[150 rows x 5 columns]
```

```
In [2]: df = df.drop(["petal width (cm)", "sepal width (cm)"], axis = 1)
        df
```

Out[2]:

	sepal length (cm)	petal length (cm)	target
0	5.1	1.4	0
1	4.9	1.4	0
2	4.7	1.3	0
3	4.6	1.5	0
4	5.0	1.4	0
5	5.4	1.7	0
6	4.6	1.4	0
..	...	...	...
143	6.8	5.9	2
144	6.7	5.7	2
145	6.7	5.2	2
146	6.3	5.0	2
147	6.5	5.2	2
148	6.2	5.4	2
149	5.9	5.1	2

```
        [150 rows x 3 columns]
```

```
In [3]: # 我們把我們擁有的資料集分成兩份，一份測試，一份訓練
        from sklearn.model_selection import train_test_split
        # 把資料分成兩部分 (1. 訓練資料 2. 測試資料)
        data = df.drop(["target"], axis = 1)
        data_train, data_test, target_train, target_test = train_test_split(data,
                                                                            df['target'],
                                                                            test_size=0.1)
```

1.4.2 Step 1. 建立模型

我們使用 sklearn 的 Perceptron 來建造我們的感知器
這裡我並沒有調整過多的參數，目的只是為了讓你看看感知器是如何做出分類的

```
In [4]: from sklearn.linear_model import Perceptron
        clf = Perceptron()
        clf = clf.fit(data_train, target_train)
```

```
In [5]: from sklearn.metrics import accuracy_score

        predict = clf.predict(data_test)
        print("預測:", list(predict))
        print("正確標籤:", list(target_test))
        print("正確率: ", accuracy_score(target_test, predict) * 100, "%")
```

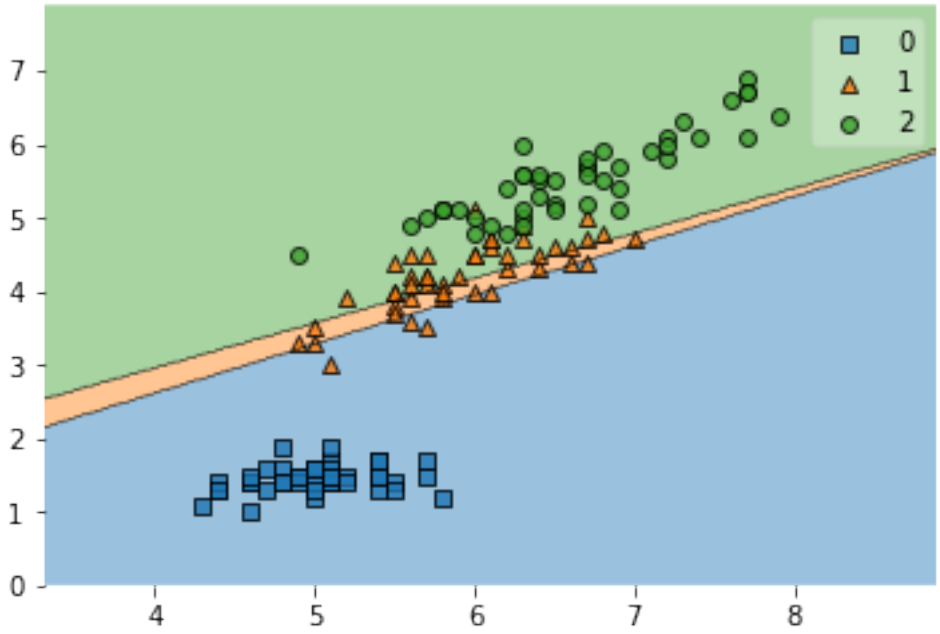
```
預測: [2, 2, 2, 0, 2, 2, 2, 0, 0, 0, 2, 0, 2, 0, 0]
正確標籤: [2, 2, 1, 0, 1, 1, 2, 0, 0, 0, 2, 0, 1, 0, 0]
正確率:  73.3333333333 %
```

1.4.3 Step 2. 畫出決策邊界

接著我們用剛剛的分類器畫出他的決策邊界

```
In [6]: from mlxtend.plotting import plot_decision_regions
        import numpy as np
        plot_decision_regions(X=np.array(data_train),
                              y=np.array(target_train),
                              clf=clf)
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x10a63cb70>



你應該發現了，感知器畫出一條直線來分出兩個區間，所以那種無法用直線分開的混濁類別我們就無法分類了

現在我們將分類或回歸分成兩種

1. 線性分類器: 分類畫出的標準是直來直去的，二維畫出的分類是一條線，三維畫出的分類是一個平面，簡單來說就是每個特徵的衡量都是一次方！用數學的方式表示就是 f(x) = w1 * x1 + w2 * x2 + w3 * x3 . . .

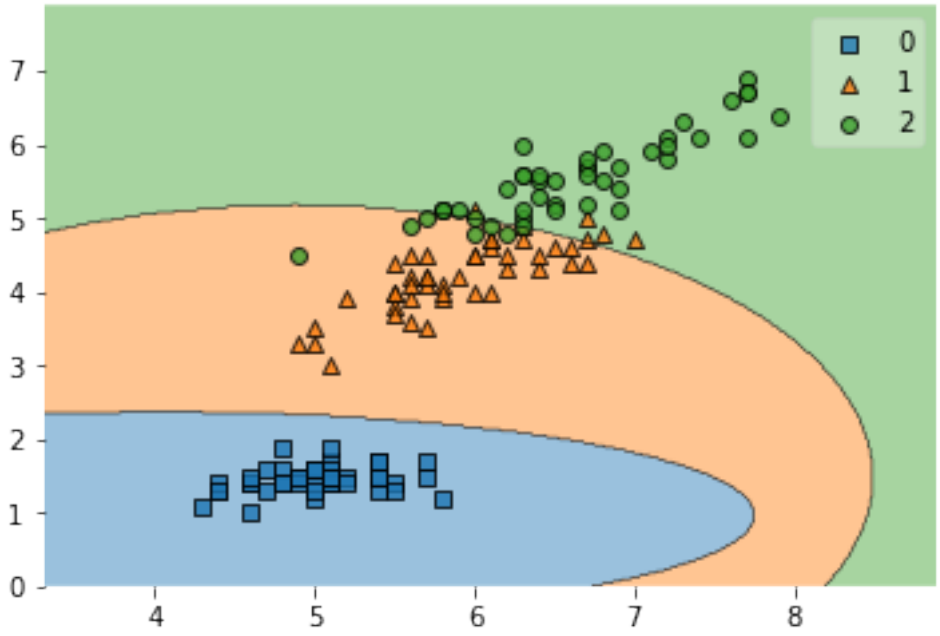
2. 非線性分類器: 去模擬非線性分布的方法很多，最直覺的方式就是使用一個多次方的函式去擬合你的資料，ex: f(x) = w1 * x1^2 + w2 * x2^2. . .，來模擬橢圓形，這種方式類似我們下面的單純貝氏畫出來的方式 (但用單純貝氏舉例比較不直覺，你可以想成貝氏做了很多特徵機率的相乘，所以並不是個一次方的擬合)。另外一種方式是多條直線切割模擬非線性分類，最容易理解的例子就是下面的決策樹，決策樹分類非線性的方法是不斷二切，就可以切出一個用很多直線構成的非線性擬合

單純貝氏擬合非線性的決策邊界如下

```
In [14]: # 由於我們的特徵不是不連續的整數，我們必須使用 GaussianNB 來
         from sklearn.naive_bayes import GaussianNB
```

```
clf = GaussianNB()
clf = clf.fit(data_train, target_train)
plot_decision_regions(X=np.array(data_train),
                      y=np.array(target_train),
                      clf=clf)
```

```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x11227f2b0>
```

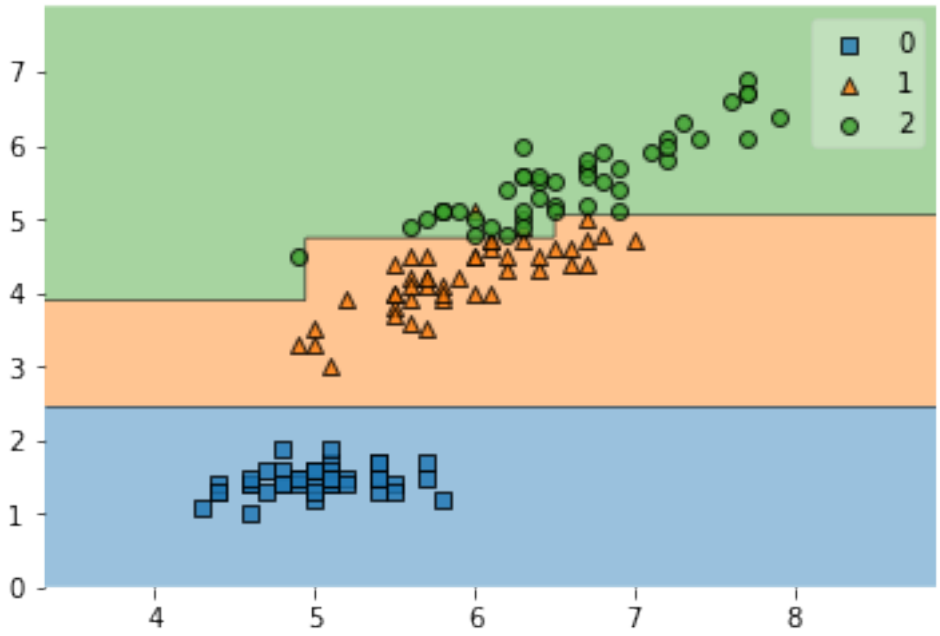


決策樹擬合非線性的決策邊界如下

```
In [12]: from sklearn.tree import DecisionTreeClassifier

         # 讀者可以試試看從 max_depth = 1 慢慢往上調整，觀看決策邊界的變化
         clf = DecisionTreeClassifier()
         clf = clf.fit(data_train, target_train)
         plot_decision_regions(X=np.array(data_train),
                               y=np.array(target_train),
                               clf=clf)
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x112301dd8>
```



1.5 問題 1

我們發現感知器有一點點小小的問題

因為只有正負的分類，並沒有把相對於分類線的距離做成人類比較容易理解的量度，所以會比較難跟人解釋 (你無法說出：降雨的機率是多少，你只能說出：會不會降雨)

為了解決這個問題，我們發展出邏輯斯回歸 (Logistic Regression)，我們在下一章節介紹這個問題

1.6 問題 2

感知器身為神經網路的鼻祖，現在多是一個精神象徵的地位，而不是直接拿來實際使用了，因為我們在現實的問題大部分都是非線性問題居多，所以我們繼續往下擴展，來繼續看神經網路的發展。順便讓大家看看那代人對感知器最後下的結論，『對於 XOR(Exclusive OR) 問題完全沒有辦法分類』

XOR 問題：類似 OR 問題，不過在左右都是 True 的時候會是 False。

舉例：一個女生對於帥或者有才華只要有一個就可以了，兩者都有更好，這叫 OR 問題。但是如果他對於又帥又有才華的男生有恐懼感，這就叫 XOR 問題。

OR	True	False
True	True	True
False	True	False

XOR	True	False
True	False	True
False	True	False

In [27]:
```
from numpy import random
# 可以用 numpy 快速產生隨機，第一個參數是你產生有多少種類
# 第二個參數是你要幾個
# x1 是我們的第一特徵，你可以想像成帥
# x2 是我們的第二特徵，你可以想像成有才華
x1 = random.choice([True, False], 100)
x2 = random.choice([True, False], 100)
# y 是我們的 target，你可以想像成會不會喜歡
y = np.logical_xor(x1, x2)
df = pd.DataFrame(columns = ["x1", "x2", "y"])
df["x1"] = x1
df["x2"] = x2
df["y"] = y
df = df.astype(int)
df
```

Out[27]:

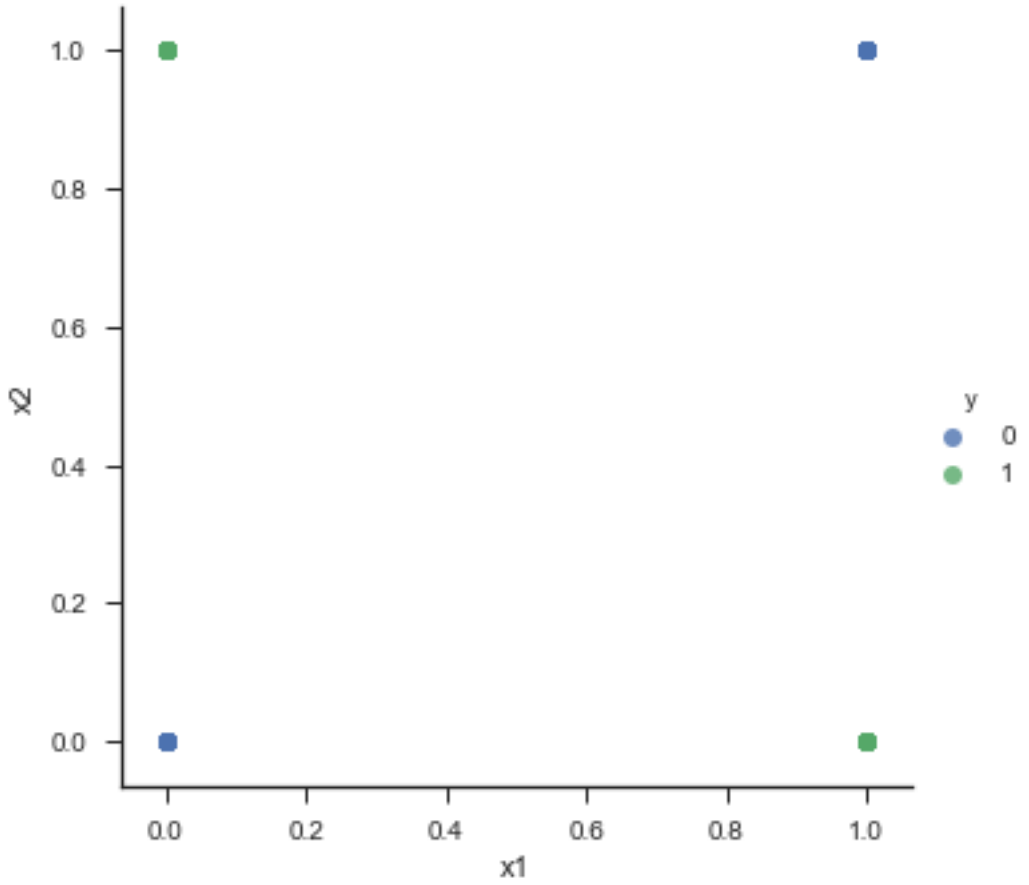
	x1	x2	y
0	0	0	0
1	0	0	0
2	1	1	0
3	1	1	0
4	0	0	0
5	0	1	1
6	1	0	1
..	..	..	..
93	0	1	1
94	1	0	1
95	0	0	0

```
96   1   1  0
97   0   0  0
98   0   1  1
99   0   1  1

[100 rows x 3 columns]
```

```
In [28]: import seaborn as sns
         sns.lmplot(x = "x1", y = "x2", hue = "y", data=df, fit_reg = False)
```

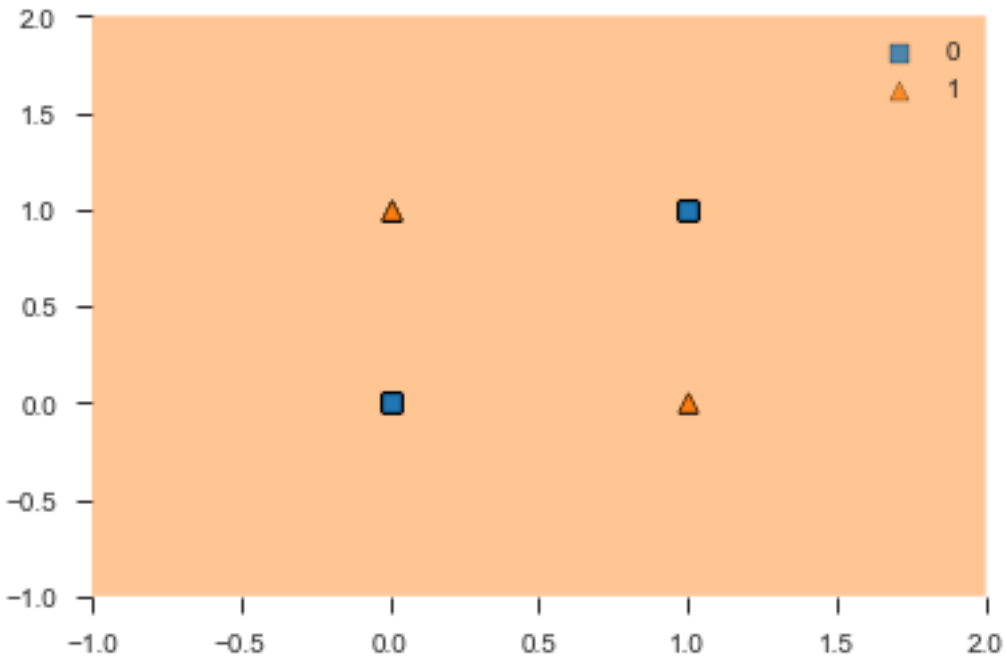
```
Out[28]: <seaborn.axisgrid.FacetGrid at 0x1132fd278>
```




```
In [30]: clf = Perceptron()
         clf = clf.fit(df.drop(["y"], axis = 1), df["y"])
         plot_decision_regions(X=np.array(df.drop(["y"], axis = 1)),
```

```
                                y=np.array(df["y"]),
                                clf=clf)
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x113e92e48>
```



你可以看到完全分不出來，這分類器將所有的類別都當成 True。我們下一章節要就這個問題來一起看看神經網路的發展