ANN_BASIC_3

2018 年 7 月 18 日

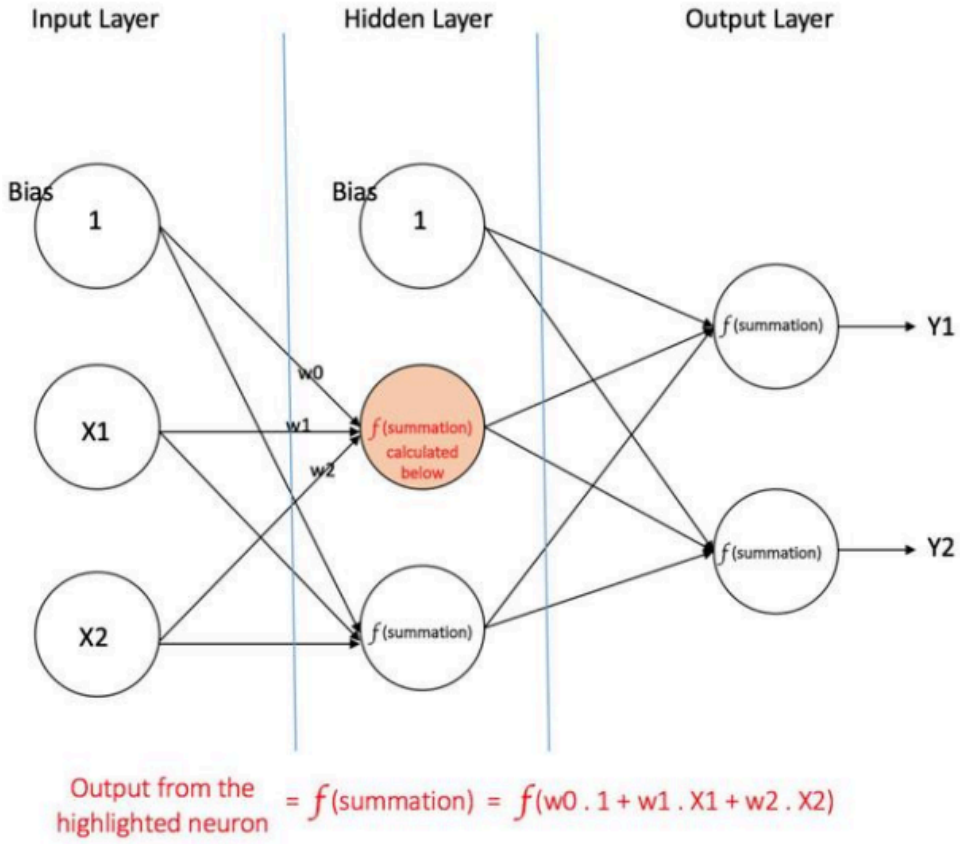
1 再探神經網路 - SVM

1.1 介紹

上一個章節我們介紹了感知器，我們也知道感知器有其限制，只能分類線性可劃分的類別。所以在歷史的發展中，感知器一度被捨棄，最明顯的辯論就是感知器對於 XOR 問題極度的差，接下來神經網路大神們就著重在這問題提出兩種解決方式

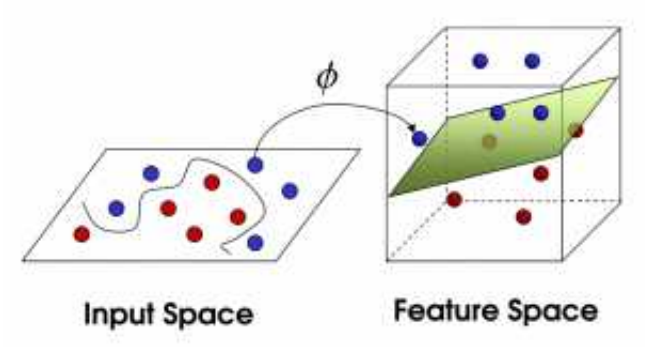
1.1.1 解法 1: 多層感知器 (Multi-layer Perceptron)

模擬神經的方式，利用多個神經元組合下去，也就是所謂的 MLP(Multi-layer Perceptron)，我們留待下一章節再介紹



1.1.2 解法 2:

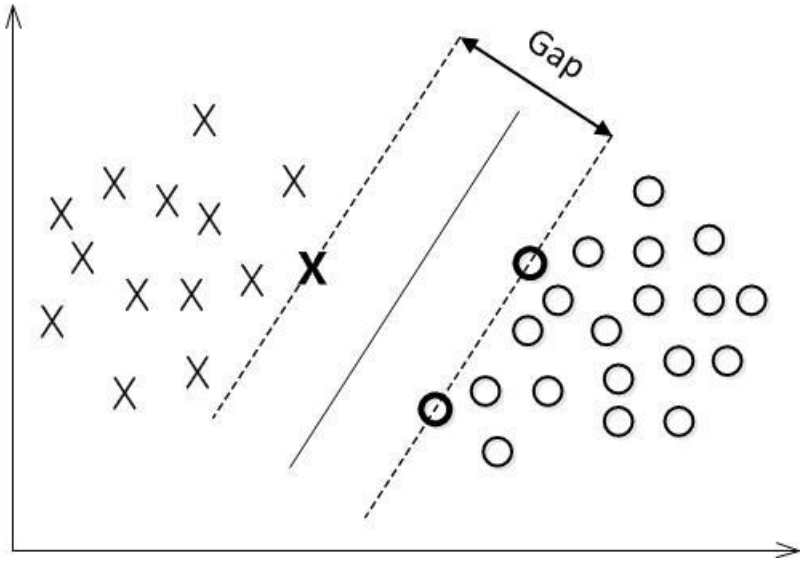
先把你的數據點升高維度，也許就可以變成線性可分，例子如下圖



但是，怎麼把你的網路變成非線性的分類器呢?

我們先從一個線性的分類器說起，但這次我們不使用之前的感知器或者邏輯斯回歸，我們使用一種方法叫做 SVM(支援向量機)

1.2 支援向量機 (Support Vector Machine)



之前我們的線性分類器都只著重在畫出一條可以分割的線就好了，但有時候你畫出的那條線只是把兩邊分開，卻不是最中間，把大家分最開的一條線，所以下一個要判別的資料進來的時候就有可能分錯，於是我們有了支援向量機

支援向量機最重要的一件事，就是畫出一條把正類和負類分最開的線

而我們的問題就變成找到負類距離最近點的平行線 (上方經過 X 的虛線) 和正類距離最近點的平行線 (上方經過 O 的虛線)，當他們距離最大的時候，中間的實線就是我們要找的線了!

1.3 線性 SVM

由於步驟和之前差不多，這裡就直接從頭寫到結束，首先先準備資料集

```
In [3]: from sklearn.datasets import load_iris
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import numpy as np
        from mlxtend.plotting import plot_decision_regions
        %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')
```

```
In [4]: # 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 = df.drop(["petal width (cm)", "sepal width (cm)"], axis = 1)
        df
```

Out[4]:

	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

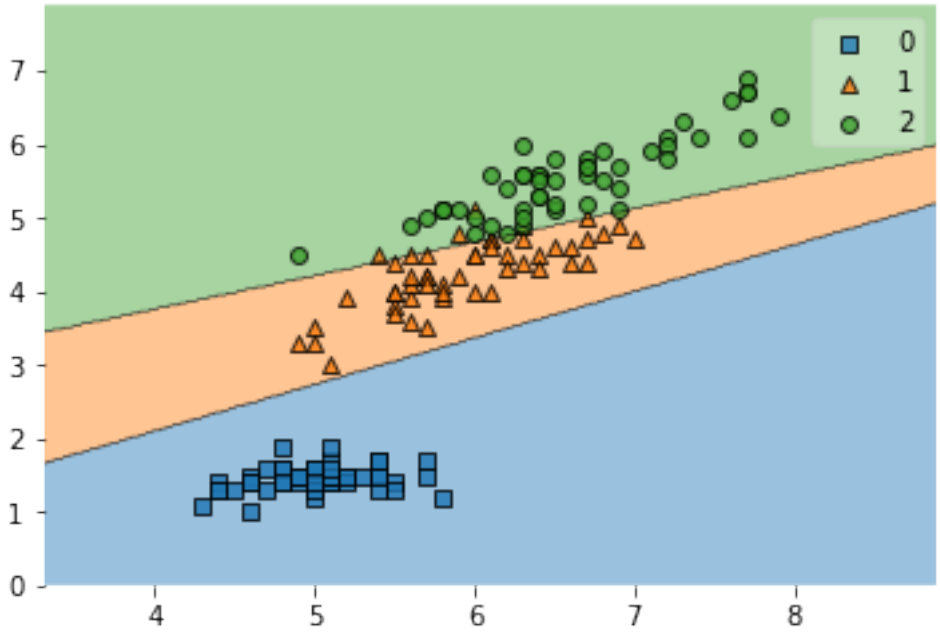
[150 rows x 3 columns]

使用 sklearn 的 LinearSVC 做出判斷，並且畫出邊界

SVM 也可以用來做迴歸: 擬合出方程式來預測下一個連續的數值
> 迴歸: Support Vector Regression(SVR)
> 分類: Support Vector Classification(SVC)

有興趣的讀者可以往前翻一下，你可以看出 SVC 畫出的邊界和 Logistic Regression 畫出的邊界截然不同

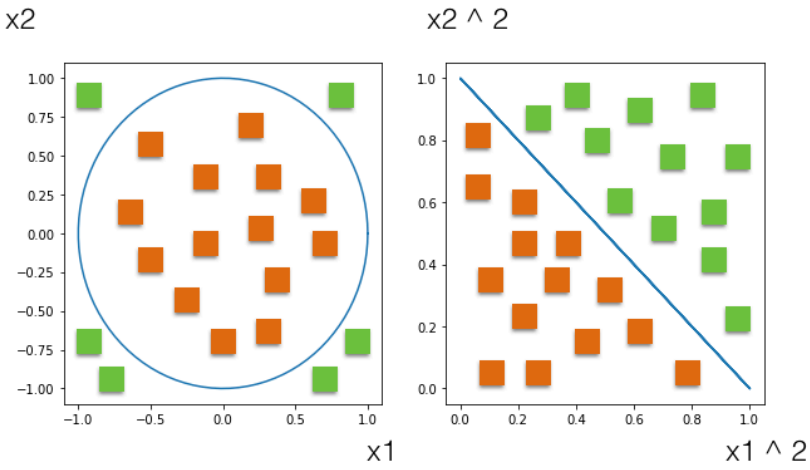
```
In [5]: from sklearn.svm import LinearSVC
        from sklearn.metrics import accuracy_score

        clf = LinearSVC()
        clf = clf.fit(df.drop(["target"], axis = 1), df["target"])

In [6]: plot_decision_regions(X=np.array(df.drop(["target"], axis = 1)),
                              y=np.array(df["target"]),
                              clf=clf)

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x10ae8b978>
```



1.4 非線性 SVM

會了 S V M我們就可以開始解決剛剛非線性的問題了
剛剛說到，我們在 S V M解決非線性的問題使用的是“提高維度”的方法
我們先用一個最簡單的例子來簡單描述一下什麼是提高維度

1.4.1 例子 1



我們從左邊的圖看到我們的資料可以被一個二次曲線 (圓) 劃分
圓的曲線表示是 ***x1^2 + x2^2 = r^2*** (r 為半徑)
用我們之前的線性分類器，當然行不太通
但是如果我們製造出兩個新的特徵 x1^2 和 x2^2
然後使用這兩個製造出來的特徵分類，你會發現！這兩個新的特徵可以簡單地用一個直線擬合

1.4.2 例子 2

我們用比較生活的方式再來跟你說一次提高維度
假設有一個情況是今天我們有兩個特徵來決定一個人會不會愛這個食物

1. x1: 食物的軟硬度 (越軟越不愛)
2. x2: 食物的臭味 (越臭越不愛)

但你驚訝的發現當兩個很不愛的東西加起來的時候，反而很多人很愛，這是什麼食物呢？臭豆腐！

那我們如果只用 x1 和 x2 這兩個特徵來看的話，只能用一個二次以上的曲線來擬合，但是如果我們製造出一個新的特徵叫做 x1 * x2

這個特徵也許就可以幫我們來辨別上面的 case(乘起來極度小的時候代表愛)

1.4.3 總結

你可以發現，上面我們只做了一件事情，利用已有的特徵製造新的特徵，但如果你把它當成資料的維度的時候，你可以把它想像成資料的維度增加了！(從兩個特徵變成三或四個特徵)

這就是所謂的增加維度!!!

那增加維度有什麼意義呢?

經由較為嚴謹的數學證明: 當維度增加的時候，能夠線性劃分的機率會增加
用比較想當然爾的想法想: 當維度增加也就代表你在空間的分佈變廣了，那就越有機會做出線性的劃分

注意，這裡只是能夠劃分的機率增加，並不是 100% 保證能線性劃分，但就算不能完全劃分，因為分佈變廣，你的線性分割還是可以取得比較好的效果

1.5 核函數

剛剛我們已經大概敘述了我們解決非線性的核心思維了

但還有個問題沒解決，我們轉換過的維度有時候實在太大了，如果對每個轉換過的向量做 w * x 來算分數的話計算量會非常非常的大

那怎麼辦呢?

我們先把之前的 wx + b 的優化問題拿出來討論

我們在解這個問題的時候，主要是在解 w 向量，但是在一個限制條件下，就是不能有任何落在支援向量以內的空間

嚴格的證明你要將這優化問題轉化成『對偶問題』解決

但我們今天用比較直覺的想法來想這個問題

w 應該是根據你餵進去的訓練資料所決定的!!! 那就應該是跟你訓練資料的特徵和分類有關

所以我們大概得到下面的簡化

$$\begin{aligned} w^T x + b &= \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b \\ &= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b. \end{aligned}$$

簡單來說 w = 常數 (a) x 每一筆資料的特徵 (xi) x 資料的分類 (正類或負類 yi)

但 y 不是向量，只是個常數，所以我們可以把他拉出來，最後就只剩 xi 乘上 x (向量的乘法我們叫做內積)

所以對於你要判斷的新資料，只要傳進來跟所有舊的資料做內積就結束了!

你會說! 疑，我們的原始問題還沒解決啊! 維度的數目並沒有被縮小啊!

是的，但如果你在這世界上能夠找到一個函式

$$k(x, y) = \Phi(x)\, \Phi(y)$$

指的是一個某種低維向高維的轉換

如果你能找到兩個轉換過的向量的內積 = 沒轉換過的向量的某種計算結果
上面的問題得到解決了!!! 因為所有的轉換過的內積計算都可以換成沒轉換過的向量計算
最後我們的式子變成

$$g(x) = \sum_{i=1}^{n} a_i y_i K(x_i, x) + b$$

那最後最後的疑問只剩，真的有這個函式嗎？有的話長怎樣呢？

- linear: $K(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{x}_i^T \mathrm{x}_j$.
- polynomial: $K(\mathrm{x}_i, \mathrm{x}_j) = (\gamma \mathrm{x}_i{}^T \mathrm{x}_j + r)^d, \gamma > 0$.
- radial basis function (RBF): $K(\mathrm{x}_i, \mathrm{x}_j) = \exp(-\gamma \|\mathrm{x}_i - \mathrm{x}_j\|^2), \gamma > 0$.
- sigmoid: $K(\mathrm{x}_i, \mathrm{x}_j) = \tanh(\gamma \mathrm{x}_i{}^T \mathrm{x}_j + r)$.

這些都是我們可以使用的核函數，不過我們最常使用的一個核函數是 RBF 函數，原因是 RBF 是一個往無窮維度的轉換，所以能找出線性切割的機率較大

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}$$

e 的展開是一個無窮的維度，所以 rbf 展開當然也是一個無窮的維度

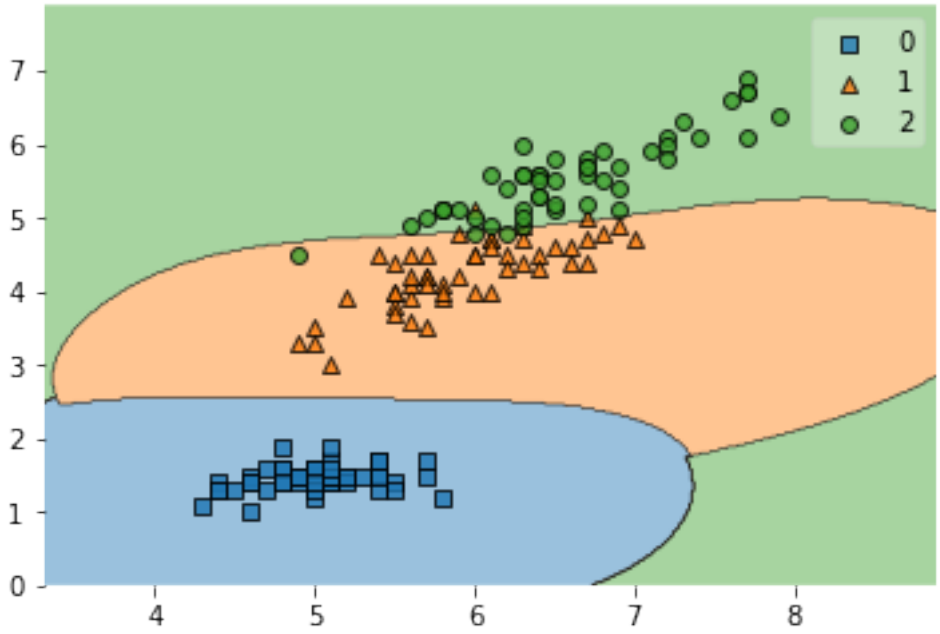
最後根據上面的說明，你發現在使用 RBF 核函數的時候，兩個高維度向量的內積變成原維度向量的距離度量!
多麼美好的一件事啊!

1.6 非線性 SVM 例子 1

我們先拿之前的鳶尾花的例子繼續來看看非線性的分類，你可以看到我的 kernel 參數選擇的是 rbf
最後我們得到一個非線性的邊界!!!

In [7]:
```
from sklearn.svm import SVC

clf = SVC(kernel="rbf")
clf = clf.fit(df.drop(["target"], axis = 1), df["target"])
plot_decision_regions(X=np.array(df.drop(["target"], axis = 1)),
                      y=np.array(df["target"]),
                      clf=clf)
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x118186358>
```



1.7 非線性 SVM 例子 2

把我們上一節懸而未解的問題: XOR 問題拿出來，一樣選擇 rbf 核
看一下分類的效果
你可以看到我們完美的分出了 XOR 問題

```
In [8]: 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
```

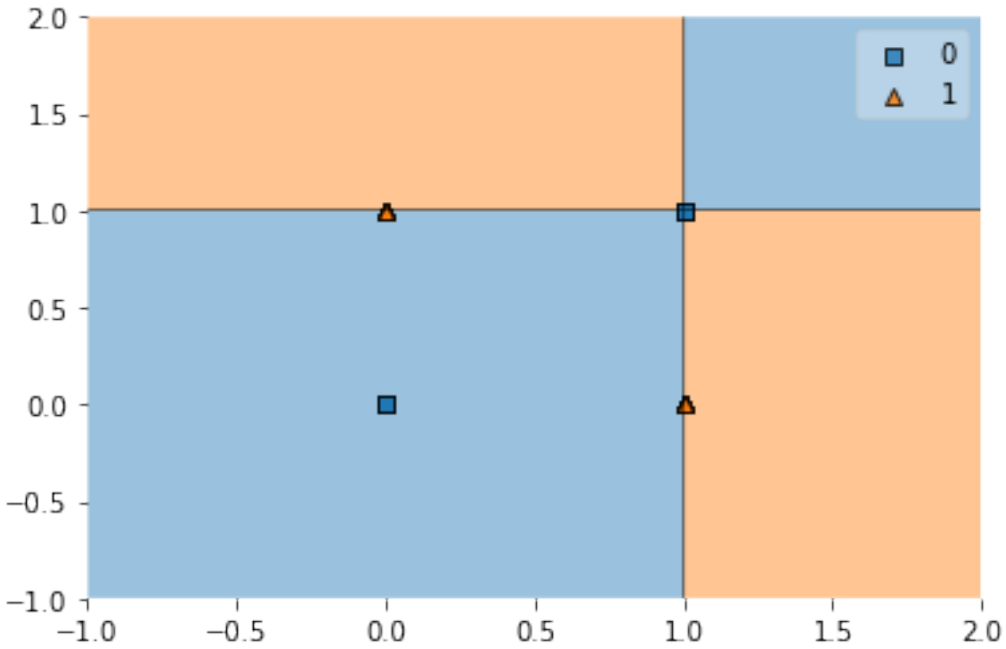
Out[8]:

	x1	x2	y
0	1	0	1
1	0	1	1
2	0	0	0
3	0	1	1
4	1	0	1
5	1	1	0
6	1	0	1
..	..	..	..
93	1	0	1
94	0	0	0
95	1	1	0
96	0	1	1
97	0	1	1
98	0	0	0
99	1	1	0

[100 rows x 3 columns]

In [9]:
```
from mlxtend.plotting import plot_decision_regions
from sklearn.svm import SVC

clf = SVC(kernel="rbf")
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[9]: <matplotlib.axes._subplots.AxesSubplot at 0x10c376358>



1.8 總結

當初非線性 S V M被提出的時候，造成一陣轟動，因為所有的問題都可以試著用非線性 S V M提出一個數學方程式來分類，而且分類的效果非常好，甚至不輸我們之前最常用的隨機森林，而 S V M也成為機器學習界使用好一陣子的主要演算法。但最後我們碰到了一個瓶頸了：抽象問題的分類，介紹到現在的演算法在分類抽象問題都有其極限!

究竟我們要怎麼解決比較抽象的問題呢？我們下一節繼續看下去！一起來進入深度學習的世界吧!!!