Physical Design Report in Final Project

Fast Timing Model Estimation for new PVT
[ICCAD Contest Problem-D]

學號: **R08921053**

組別: 電機丙研一

姓名: 梁峻瑋

信箱: r08921053@ntu.edu.tw

老師: 張耀文

助教: 呂祐昇, 徐 暘

Contents

1.實作角度切入問題

	Case 1 SS/0.09x/-40	Case 2 SS/0.09x/125	Case 3 TT/0.10x/25	Case 4 FF/0.11x/-40	Case 5 FF/0.11x/125
降壓(8v)	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f
普壓(9v)	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f
升壓(10v)	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f	a, b, c d, e, f

*表格 1 為切割問題後的分類小問題

a := cell_rise	b := rise_transition	c := cell_fall
d := fall_transition	e := rise_power	f := fall_power

*備註 2: a,b,c,d,e,f 的代表意義

#問題描述:

現在，我們給定了 15 種 PVT 條件的資料. 一方面有降壓(8v)，普壓(9v)，升壓(10v)三種類型，一方面有 Process/ Voltage/ Temperature 的五種狀態. 在這兩方面搭配下，有 15 種不同 PVT 條件組合的資料.

#提供數據:

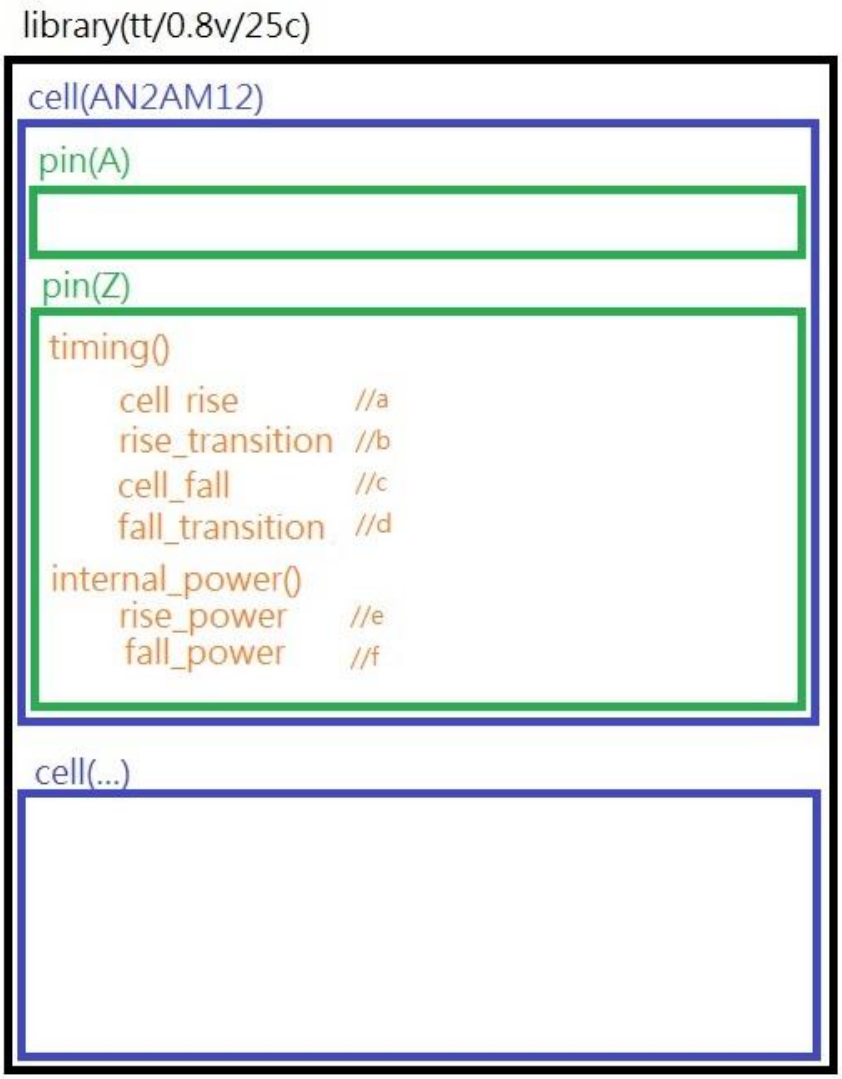
在這 15 種 PVT 條件的資料中，每一種 PVT 條件都給了 400 個 cell，是我們可以使用的 data set. 此外，還給了 100 個挖空數據的 cell，讓模型預測出答案，並且用來評量分數.

在現階段的 Alpha Stage，官方還特別提供了普通電壓(9v)的 100 個 cell 答案，希望我們利用普壓與升壓/降壓之間的關聯性，來提升解的品質.

#切割問題: [上述兩個表格所示]

每一種 PVT 條件有 400/100 個 cell. 每 1 個 cell 裡面，會有數個 pin()的資料，數目不固定，但只有 pin(Z)裡面才有可訓練的(x,y)格子，其餘都是輔助資料，如 pin(A), pin(B), pin(C)等等.

在 1 個 pin(Z)內，基本上兩類資訊，一類是 timing，裡面固定有 4 項(x,y)資訊—cell_rise, rise_transition, cell_fall, fall_transition，表示為 a,b,c,d；另外一類是 internal_power，固定含 2 項(X,Y)資訊—rise_power, fall_power，表示為 e,f.

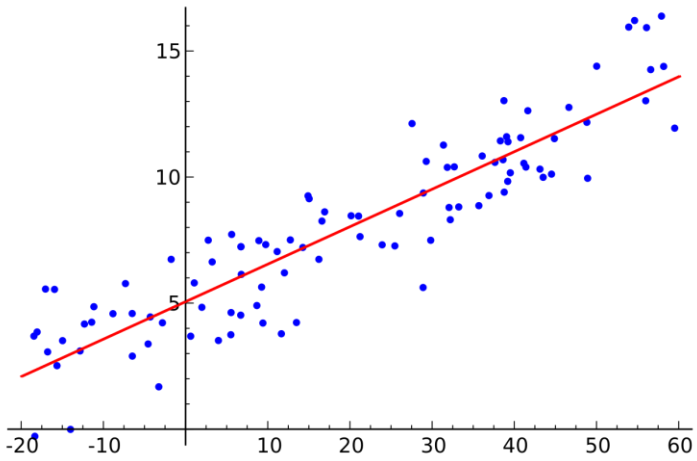


*附圖 3 為某一種 PVT 條件下的 library 內容

```
cell_rise (delay_template_generic_7x7) {
  index_1 ("0.002, 0.0154552, 0.062686, 0.15288, 0.293398, 0.490561, 0.75");
  index_2 ("0.001, 0.00250018, 0.00776614, 0.0178223, 0.0334892, 0.0554717, 0.0843977");
  values ( \
    "0.0680163, 0.0727861, 0.0859738, 0.105594, 0.131942, 0.166545, 0.211125", \
    "0.0725617, 0.0773088, 0.0904731, 0.110124, 0.136468, 0.171067, 0.215665", \
    "0.0874965, 0.0923036, 0.10553, 0.125197, 0.151542, 0.186189, 0.230759", \
    "0.110941, 0.115842, 0.129291, 0.149222, 0.175652, 0.210363, 0.254852", \
    "0.137285, 0.142585, 0.157353, 0.178094, 0.204979, 0.239697, 0.284081", \
    "0.161457, 0.167373, 0.183917, 0.2063, 0.23419, 0.269363, 0.313576", \
    "0.18351, 0.189786, 0.207765, 0.232736, 0.262386, 0.298369, 0.342809" \
  );
}
```

*附圖 4 為實際數據中的一項(x,y)資訊, x 為 index_1/2, y 為 values

2.實作演算法/ ML 模型



*附圖 5 為線性回歸模型，取自於”wiki-線性回歸”頁面

#整體方向:

在監督式學習(supervised learning)的回歸分析中，有一種最簡單的”線性回歸模型”，是我這次所使用的.

我直接用 index_1, index2 的 14 維數據做為 x, values 的 49 維數據做為 y. 也就是只使用 timing()與 internal_power()內的資訊量. 至於 pin 的資訊與 cell 的資訊，即 dummy variable，紀錄(x,y)數據屬於哪一類，則先不使用它.

單從肉眼看就知道，各筆數據之間的 x 都很相近，勢必要先做降維度的動作. 經由觀察變異數，可知保留 dim=3 就足夠，因此我嘗試:
[版本 1] BasicModel: 直接保留 X 最重要的 3 個維度資訊，再加上平移項”1,…,1”.

#演算法:

如果輸入為矩陣 X，輸出為向量 Y，線性模型假設存在矩陣 β，使得 $Y=X\beta+err$. 使用最小平方法估計，則 $\beta=(X^TX)^{-1}X^TY$，其中，在此 X 只取 3 維，有:

$$X=\begin{pmatrix}1 & x_{1,1} & x_{1,2} & x_{1,3}\\ 1 & \vdots & \vdots & \vdots\\ 1 & x_{3186,1} & x_{3186,2} & x_{3186,3}\end{pmatrix}, Y=\begin{pmatrix}y_{1,1} & \cdots & y_{1,49}\\ \vdots & \ddots & \vdots\\ y_{3186,1} & \cdots & y_{3186,49}\end{pmatrix}$$

經由上述的操作，在給定 X, Y 以後，就能夠順利得到矩陣 B，描述出他們之間的對應關係(函數關係)，進而預測其他的數據.

#具體方向:

最基本的想法是，把 15 種 PVT 條件都分開來看，而 a,b,c,d,e,f 等六類也分開來看，各別使用 90 個模型來訓練. 因此，我們要針對這 400 個 cell 來進行訓練. 把它們分成 85%的訓練(training)資料, 15%的測試(testing)資料. 最終，利用那 100 筆挖空答案的 cell，來輸出答案.

#實作取捨:

把 15 種 PVT 條件, a,b,c,d,e,f 六類都分開來訓練，可能會失去它們之間的關聯性. 然而, ML 模型比較適合處理單一輸出，構造簡單的問題，例如 0/1 等輸出的問題. 過於龐大而複雜的問題，不容易在 ML 模型有好的結果，故需要捨棄資訊.

而且，考量同一種 PVT 條件，同一種類別的問題，數據間的參考價值遠高於其他數據，而且單單考慮這 1/90 的資訊量，就遠遠足以訓練模型回答問題. 所以最終採取這個決定，把 15 種 PVT 條件和 a,b,c,d,e,f 等六類分開做模型.

3.實作資料結構/ 程式部分

我把整個程式分成兩個部分—

(a) 爬蟲部分: 從.tlib 檔案內的 library, 把 index_1/2 的 x 與 values 的 y 抓出.

(b) 模型部分: 使用爬蟲的 npy 檔, 與上述演算法, 進行預測.

#爬蟲部分:

首先, 注意到附圖 4 中, index1/2 的 x 資料以及 values 的 y 資料是連續排列的, 而且數據夾在兩個引號「”」之間, 且使用「一個逗號+一個空白」來分隔.

```
cell_rise (delay_template_generic_7x7) {
  index_1 ("0.002, 0.0154552, 0.062686, 0.15288, 0.293398, 0.490561, 0.75");
  index_2 ("0.001, 0.00250018, 0.00776614, 0.0178223, 0.0334892, 0.0554717, 0.0843977");
  values ( \
    "0.0680163, 0.0727861, 0.0859738, 0.105594, 0.131942, 0.166545, 0.211125", \
    "0.0725617, 0.0773088, 0.0904731, 0.110124, 0.136468, 0.171067, 0.215665", \
    "0.0874965, 0.0923036, 0.10553, 0.125197, 0.151542, 0.186189, 0.230759", \
    "0.110941, 0.115842, 0.129291, 0.149222, 0.175652, 0.210363, 0.254852", \
    "0.137285, 0.142585, 0.157353, 0.178094, 0.204979, 0.239697, 0.284081", \
    "0.161457, 0.167373, 0.183917, 0.2063, 0.23419, 0.269363, 0.313576", \
    "0.18351, 0.189786, 0.207765, 0.232736, 0.262386, 0.298369, 0.342809" \
  );
}
```

*附圖 4 為實際數據中的一項(x,y)資訊, x 為 index_1/2, y 為 values

利用這個特性, 只需要兩行程式碼, 就能把一組(x,y)給爬蟲下來.

➔ input = input[input.find('"')+1 : input.rfind('"')]

➔ input = input.split(', ')

把這兩行寫成函數(function), 再重複使用, 就能爬完所有 library 的數據了. 值得一提的是, 為了方便 python 語言的使用, 我選擇以.npy 檔的格式來儲存資料.

#模型部分:

首先, 先把一筆筆的 x 數據和 y 數據讀入, 形成矩陣. 再分堆成 trainX 和 testX, 以及 trainY 和 testY 矩陣.

對於 trainX 矩陣, 先降到 3 維做簡化, 再加上[1,…,1]的開頭行向量做平移項, 接著再利用”$\beta=(X^TX)^{-1}X^TY$”的關係式求出 β, 就能得出預測值. 並且一筆一筆地比較 testX*β 與 testY 的差異, 並計算 score. 就可得出結果.

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	49.84	49.26	47.69	43.99	49.80	48.07
升壓(10v)	49.75	48.80	49.63	48.75	50.34	49.45

*表格 6 為 BasicModel 的數據 [直接取 3 維/不用 9v]

4.模型改進—DB_BasicModel

在前一部份的結尾，我們完成了 BasicModel 的第一個版本模型，也實測出了模型的預測率，在把所有資料分成 85%的 trainset 與 15%的 testset 設定之下.

顯然地，第一版模型的結果差強人意，雖然有一定的準確度，代表我們有抓到大方向的精隨，但仍有很大的空間可以改進. 例如，善用 dummy variable 的 pin 資訊, cell 資訊，也就是使用 90 個模型之間的相關性，或者是針對準確率最低的 Case4 模型進行優化.

#引入 Alpha Stage 條件:

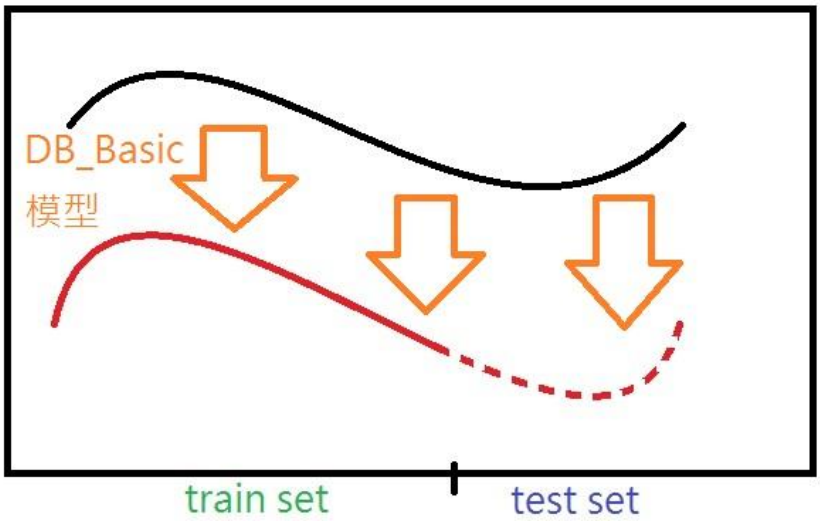
在現階段(Alpha Stage)，官方提供了普壓(9v)底下 5 個 PVT 條件的答案，希望我們利用這些條件來優化模型. 我給出了一個很直觀的優化方法:

“已知普壓(9v)的答案，想求降壓(8v)的答案，則可以把兩組一起餵入訓練”

換言之，我把 8v 與 9v 的 x 數據，混在一起變成新的 28 維 x 數據，並且把 8v 和 9v 的 y 數據相減，其差值變成新的 49 維度 y 數據. 並且把新的模型命名為”DB_BasicModel”，象徵的”double data”的意涵.

#分析 DB_Basic 模型:

在此，我舉出兩個直接相關的好處. 第一，我們有 2 倍的 x 資訊，從 14 維提升到 28 維，或者是降維後的 3 維到 6 維，這理當更能夠幫助我們抓到 y 值的變化. 第二，由於降壓(8v)和普壓(9v)之間，應該有很大的相關性，甚至很可能在一部份區間內都是平移的關係. 因此，我們可以依循普壓的答案，描繪出降壓的脈絡，進一步優化我們的答案.



*附圖 7 為直觀上, DB_Basic 模型的預測方式

#實作演算法:

基本上，我們仍舊採取一樣的做法，利用線性回歸模型與最小平方法，估計出預測的 y 值，即預測的降壓(8v)與普壓(9v)差值．假定已知答案的普壓 y 值為"y9v"，則預測的降壓答案為"y+y9v"．然而我們並不需要 y9v，就能評分好壞.

計分公式:

$$100 - 100 * \sqrt{\frac{1}{n}\sum_{i}^{n}\left(\min\left(1, \frac{|y_i - \hat{y_i}|}{y_i}\right)^2\right)}$$

假定降壓的答案是 y8v，模型預測答案 testY=y8v-y9v，則公式改成:

$$100 - 100 * \sqrt{\frac{1}{n}\sum_{i}^{n}\left(\min\left(1, \frac{|y8v_i - (y + y9v)_i|}{y8v_i}\right)^2\right)}$$

$$= 100 - 100 * \sqrt{\frac{1}{n}\sum_{i}^{n}\left(\min\left(1, \frac{|testY_i - y_i|}{y8v_i}\right)^2\right)}$$

因此，我們只需要 testY，與降壓的答案"y8v"，就能夠衡量出評分．基本上不需要額外修改演算法.

#實作資料結構/程式部分

在爬蟲上，既有的資料已經足夠．只要一次點開降壓和普壓的.npy 檔，把 x 值合併，把 y 值求差，再輸出，就完成了一個新的.npy 檔.

在模型上，從開頭到求出預測的 y 值為止，基本上與之前一模一樣．然而，最後計算評分時，需要使用舊的降壓.npy 檔案，作為分母．而後再輸出就結束.

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.28	76.70	70.87	59.78	75.24	71.39
升壓(10v)	82.54	80.72	71.80	57.01	78.07	72.43

*表格 8 為 DB_BasicModel 的數據[直接取 3 維/使用 9v]

5.模型再改進—DB_PCAModel

在第四段的結尾，我們給出了 DB_BasicModel 模型的實測結果．誠實地說，這確實是個還可接受的結果．在 80%的測資下，模型都能給出 75%左右的預測準確度．

這時候，考慮到先前直接取 3 個維度的手法太粗糙，而且現在至少也有 6 個維度可以使用，因此我們聯想到機器學習常用的一種工具—主成分分析 (Principal Component Analysis)．簡單來說，經由正交矩陣(orthogonal matrix) 的變換，可以把 X 轉變成一個僅有對角元素的矩陣，而且元素從大排到小．如此一來，我們就能夠取前幾大的元素，也象徵著使用前幾大重要性的 eigenvalue 來降維．

更具體而言，我們能從代數層面，使用奇異值分解(Singular Value Decomposition) 來計算出 PCA 的結果．並把這套模型命名為”DB_PCAModel”．

#分析 DB_PCA 模型:

理論上，在使用 PCA，降維 X 以後，只要取的維度是 3 以上，就蘊含著比先前更多的資訊量．合理推論，更能夠準確地預測出 y 值．預測的準確度理當有所提升．

#實作演算法/資料結構:

由於 python 已經有內建的函數，所以我們只需要使用:

➔ from sklearn.decomposition import PCA

➔ pca=PCA(n_components=3)

➔ X=pca.fit_transform(X)

短短三行的語法，就能夠達成 PCA 降維的效果．特別地是，我們還能夠調整降維的目標維度．理論上，應該大於等於先前的 3 維度，並且小於等於 rank(X)=7.

然而，如果降維後的維度仍然太大，例如維度是 6 或 7，將會造成後半段計算矩陣 β 時，反矩陣的計算超出運算範圍，產生不穩定的結果．所以我們把目標維度限定在 3, 4, 5，並且得出了實驗的數據．

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.27	76.68	70.86	59.76	75.21	71.37
升壓(10v)	82.53	80.71	71.79	57.01	78.06	72.42

*表格 9 為 DB_PCAModel 的數據 [主成分分析 dim = 3/使用 9v]

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.25	76.69	70.89	59.80	75.24	71.39
升壓(10v)	82.52	80.71	71.82	57.06	78.08	72.45

*表格 10 為 DB_PCAModel 的數據 [主成分分析 dim = 4/使用 9v]

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.32	76.71	70.99	59.83	75.27	71.44
升壓(10v)	82.55	80.76	71.88	57.06	78.14	72.48

*表格 11 為 DB_PCAModel 的數據 [主成分分析 dim = 5/使用 9v]

#結果分析:

單純就 DB_PCAModel 上，隨著 dim=3, 4, 5 變大，準確率也穩定地上升，這驗證我們的對於 PCA 的理論推測是正確的.

另一方面，以 DB_PCAModel 在 dim=5 上的數據，也確實比 DB_BasicModel 的準確度還要高.

然而，不論是從變異數的觀察，或者是實際數據上的差異，都顯現出對 X 做 PCA 降維的作用，與直接取 3 維，差異並不大. 這或許是因為，資料在給定時，就已經有進行前處理，做過一次 PCA 的操作.

此外，類似於 DB_BasicModel 存在的問題，在 80%的測資上，我的 Model 都有很優秀的表現. 但是在其餘的 20%測資上，卻有意外糟糕的表現. 接下來，或許可以朝向少部分特定函數的再優化，例如 Case4 的預測函數等等.

6.最終的結果表格

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	49.84	49.26	47.69	43.99	49.80	48.07
升壓(10v)	49.75	48.80	49.63	48.75	50.34	49.45

*表格 6 為 BasicModel 的數據 [直接取 3 維/不用 9v]

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.28	76.70	70.87	59.78	75.24	71.39
升壓(10v)	82.54	80.72	71.80	57.01	78.07	72.43

*表格 8 為 DB_BasicModel 的數據[直接取 3 維/使用 9v]

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.27	76.68	70.86	59.76	75.21	71.37
升壓(10v)	82.53	80.71	71.79	57.01	78.06	72.42

*表格 9 為 DB_PCAModel 的數據 [主成分分析 dim = 3/使用 9v]

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.25	76.69	70.89	59.80	75.24	71.39
升壓(10v)	82.52	80.71	71.82	57.06	78.08	72.45

*表格 10 為 DB_PCAModel 的數據 [主成分分析 dim = 4/使用 9v]

	Case1	Case2	Case3	Case4	Case5	Totally
降壓(8v)	78.32	76.71	70.99	59.83	75.27	71.44
升壓(10v)	82.55	80.76	71.88	57.06	78.14	72.48

*表格 11 為 DB_PCAModel 的數據 [主成分分析 dim = 5/使用 9v]

7.再優化的想法與點子

```
Case = 1, Char = a, answer = 90.5115999027955
Case = 1, Char = b, answer = 94.4016168376762
Case = 1, Char = c, answer = 86.32664663956416
Case = 1, Char = d, answer = 94.3136274229306
/home/asfhiol/anaconda3/lib/python3.6/site-package
tered in double_scalars
Case = 1, Char = e, answer = 81.60231702066588
Case = 1, Char = f, answer = 53.60633801714735
In Case 1, answer = 78.32189471638495

Case = 2, Char = a, answer = 89.04807573087277
Case = 2, Char = b, answer = 92.00025597315621
Case = 2, Char = c, answer = 81.36238047717889
Case = 2, Char = d, answer = 93.34601931646894
Case = 2, Char = e, answer = 82.0927784278924
Case = 2, Char = f, answer = 51.30811156724807
In Case 2, answer = 76.71392858878284

Case = 3, Char = a, answer = 70.3657649456969
Case = 3, Char = b, answer = 81.65287362123948
Case = 3, Char = c, answer = 59.943567689696465
Case = 3, Char = d, answer = 82.54328174045568
Case = 3, Char = e, answer = 82.54771796550699
Case = 3, Char = f, answer = 59.48880847355817
In Case 3, answer = 70.99269779935766

Case = 4, Char = a, answer = 53.627259730947515
Case = 4, Char = b, answer = 66.50702545338976
Case = 4, Char = c, answer = 41.33572944337639
Case = 4, Char = d, answer = 69.68732111253678
Case = 4, Char = e, answer = 82.10657744380009
Case = 4, Char = f, answer = 57.97761869079017
In Case 4, answer = 59.838094970604836

Case = 5, Char = a, answer = 82.77843485542786
Case = 5, Char = b, answer = 86.78039616528426
Case = 5, Char = c, answer = 72.22852797698397
Case = 5, Char = d, answer = 89.69201093864059
Case = 5, Char = e, answer = 82.40016562107887
Case = 5, Char = f, answer = 55.003316677404115
In Case 5, answer = 75.27531064947325

Totally, answer = 71.44166030566412
```

*附圖 12 為 DB_PCAModel 在 dim = 5, 對 8v 所預測的各項準確度.

#問題分析:

在最終的 DB_PCAModel 上, 有兩個最明顯的問題:

(a)在 Case1, 2, 3, 5, 唯獨 Char f 上的準確率非常低, 大約只有 55%左右, 拖垮平均.

(b)在 Case4, Char a, b, c, d, f, 準確率都不佳, 大約是 50~70%之間.

換言之, 在 80%*83%=66%的模型上, 我的線性回歸模型, 都有接近 80~90%的準確率. 然而, 在另外的 33%模型上, 或許可以考慮採用其他的模型, 或是加入 dummy variable 的 pin, cell 資訊在線性回歸模型上進行優化.

#解決方案 1: 加入 dummy variable

舉例而言，假設要加入 pin 的資訊，可以編碼 pin(A)為(1,0,0,0), pin(B)為(0,1,0,0), pin(C)為(0,0,1,0), pin(D)為(0,0,0,1). 這樣就可以解決變數非 0~1 之間實數的問題. 類似地，相同的手法也能用在其餘的 dummy variable 上. 通過逐步地增加 dummy variable，可以幫助我們了解哪些資訊更有用於預測 y 值.

#解決方案 2: 拋棄線性的假設

基本上，上述的所有模型，都是建立在 X,Y 是線性關係的假設上. 或許，在那其餘的 33%情況下，模型根本不是線性的，導致我們的預測失準.

然而，採取邏輯分析(Logistic Regression)應該不會是一個好的主意. 基本上，邏輯分析與線性回歸分析的差異，在於前者多了一層 output-layer，把 0~1 之間的輸出值，印射到 0 或 1，進行簡單的 0/1 預測. 這顯然無助於此.

或許，我們可以參考一些 ML 模型的做法，加入一些非線性的層(Activation Function)，比照三層神經網路，並進行監督式學習，嘗試得到更好的結果.

#解決方案 3: 三重模型(Triple Model)

就 Case4 的情況來說，唯獨它的準確率特別低，而且其他的準確率高達 8~9 成. 或許可以考慮，在 Case1, 2, 3, 5 預測完畢後，使用這些預測的數據，來輔助 Case4 一起再做預測. 由於 Case4 的準確率只有 60%附近，相信提升預測率應該是不太困難. 但大概不會再提升太多的準確率，畢竟早先的 4 個 Case 也存在誤差.

Reference

1. 積體電路電腦輔助設計軟體製作競賽 (ICCAD Contest 2020):
http://iccad-contest.org/2020/tw/

2. 附圖 4 擷取自:
http://iccad-contest.org/2020/tw/Problem_D/20200414_base_nom_0p8v.7z
[ICCAD 競賽所提供測資, “lib1_ff0p88v125c_base_400.tlib” in “base_nom_0p8v”]

3. 附圖 5 擷取自 wiki—“迴歸分析”頁面:
https://zh.wikipedia.org/wiki/%E8%BF%B4%E6%AD%B8%E5%88%86%E6%9E%90

4. 附圖 12 擷取自本人撰寫程式的輸出結果