House Sale Price Prediction

A. 題目描述

使用回歸模型做房價預測，用 train.csv 跟 valid.csv 訓練模型，房屋交易資料包括 id, price 與 21 種房屋參數，最後將 test.csv 中的每一筆房屋參數，輸入訓練好的模型，預測其房價。

B. 使用方法說明

● 程式環境

我使用 Win 10 作業系統，安裝 Anaconda 5.0.0 的 Python 3.6 版本，接著安裝 Keras、Theano 和 Tensorflow，最後使用 Spyder 進行編譯。

● 使用方法

由於還不太熟悉 Python 的矩陣操作，因此我的預處理是使用 MATLAB，並將處理結果輸出於新的 CSV 檔，供 Spyder 中的程式讀取。操作步驟如下：

MATLAB 的程式碼為 analysis_YW.m

Spyder 的程式碼為 mainLoop.py

C. 程式流程與寫法

程式流程為先進行資料預處理，並拿這些特徵進行訓練房價預估的模型，最後則將欲 test 的特徵導入模型，得到預測結果。

● 預處理

在 MATLAB 中的預處理，我做了以下事情：

1. 創造新特徵：屋齡 = year - yr_built
2. 創造新特徵：是否翻修 = yr_renovated ≠ 0 ? 1:0

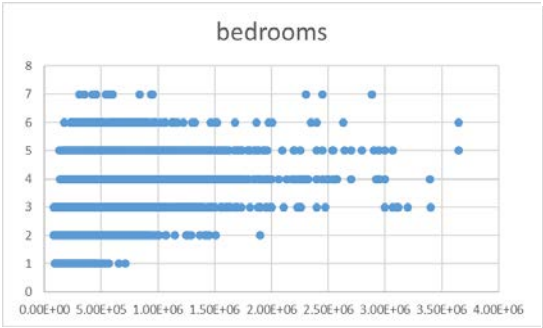
3. 創造新特徵：整合年月日 = ((year-2014)*12+(month-1))*30 + day
4. 創造新特徵：房間與廁所的總和 = bedrooms + bathrooms
5. 創造新特徵：自家與附近的居住範圍總和 = sqft_living + sqft_living15
6. 創造新特徵：自家與附近的停車範圍總和 = sqft_lot + sqft_lot15
7. 修改特徵：未翻修的房屋，該特徵值設成建造年，而非為 0
8. 修改特徵：計算 train 的平均值與標準差，將 valid 和 test 的特徵值過大或過小於界線值的，均設為界線值。上界 = mean + 4*std、下界 = mean – 3*std
9. 移除特徵：將 train 中，單坪售價大於 600 者去除，直接不考慮該筆資料
10. 移除特徵：由於已整合年月日，故只保留年，去除月和日

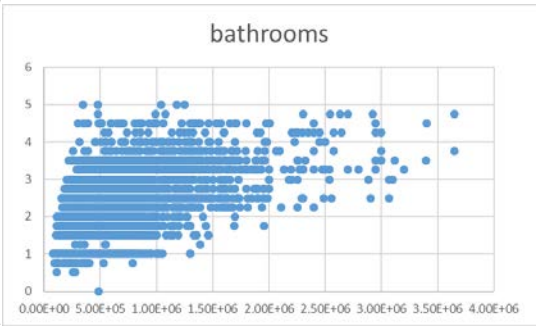
最後，得到我的 train data 特徵共有 12407 筆，每筆有 25 個特徵維度。

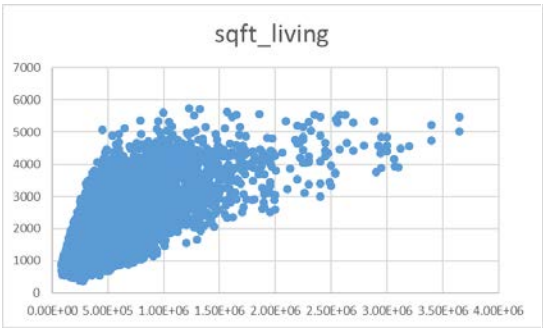
編號	名稱	內容	編號	名稱	內容
1	year	購買年	**14**	yr_built	建造年
2	days	日期先後	**15**	yr_renovated	翻修年
3	bedrooms	房間個數	**16**	zipcode	郵遞區號
4	bathrooms	廁所個數	**17**	lat	緯度
5	sqft_living	住家坪數	**18**	long	經度
6	sqft_lot	佔地坪數	**19**	sqft_living15	附近 住家坪數
7	floors	樓層數	**20**	sqft_lot15	附近 佔地坪數
8	waterfront	看的到水池	**21**	age	屋齡
9	view	多少人看過	**22**	was renovated	是否翻修
10	condition	房屋狀況	**23**	#of all room	總房間與廁所數
11	grade	評分	**24**	all living	總屋坪數
12	sqft_above	地上坪數	**25**	all lot	總佔地坪數
13	sqft_basement	地下室坪數			

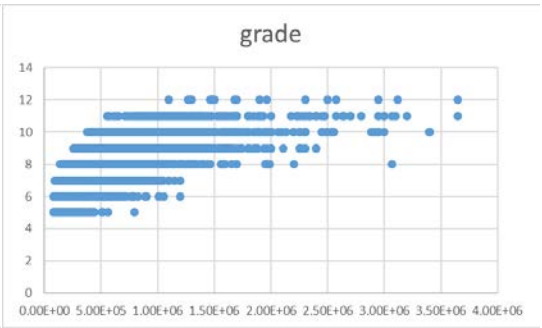
預處理還有最後一步，我在 Spyder 中進行正歸化，不同於其他同學，我並非直接除上 max(trainData)，而是將資料縮放至 [0 0.3]、[0 1]、[0 2] 三種刻度，因為直接除最大值，可能會出現資料集中在後半段，如建造年，可能會是 0.7 – 1 之間，讓資料缺乏分群的特性。

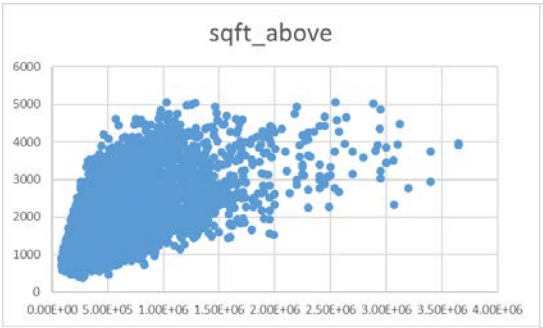
根據我的 25 種特徵與價錢的相關性，我將 bedrooms、bathrooms、sqrt_living、grade、sqrt_above、zipcode、lat、long、sqrt_living15、age、all room、all living 調整為[0 2]，將他們視為優先考慮的對象。

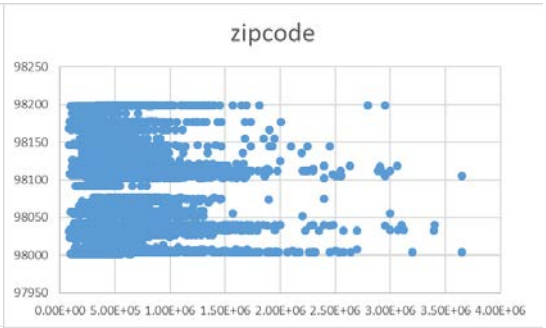


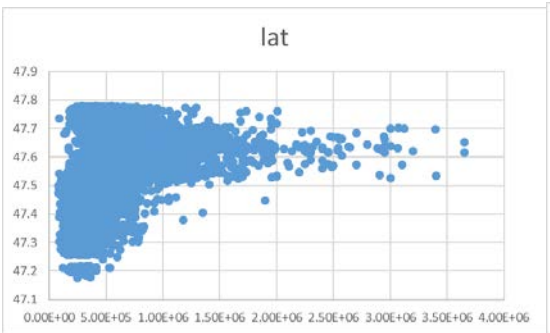


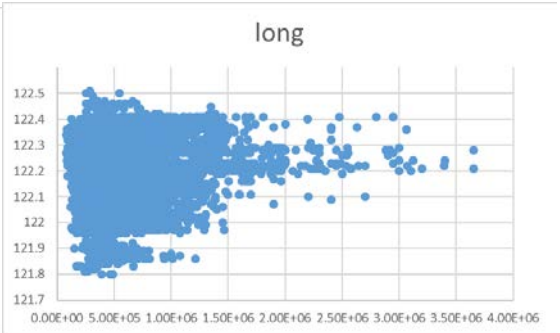


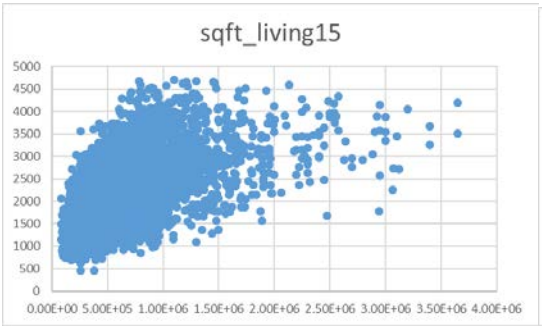


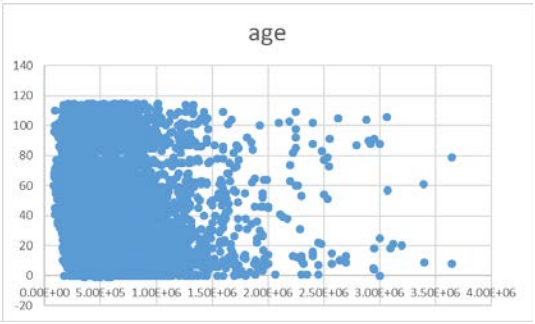


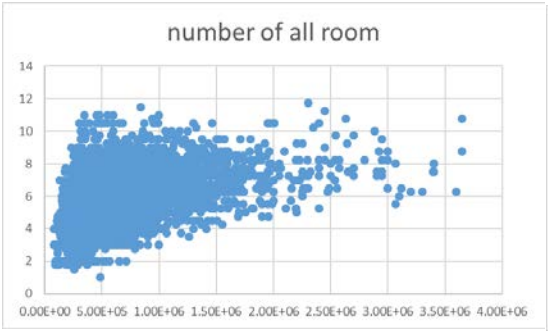


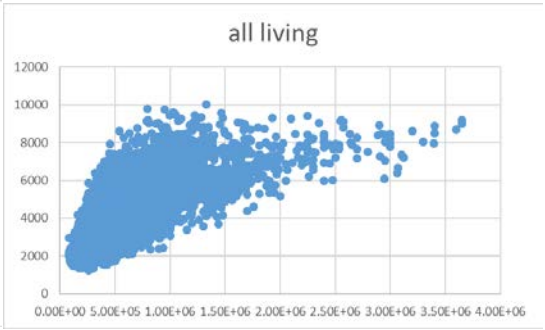




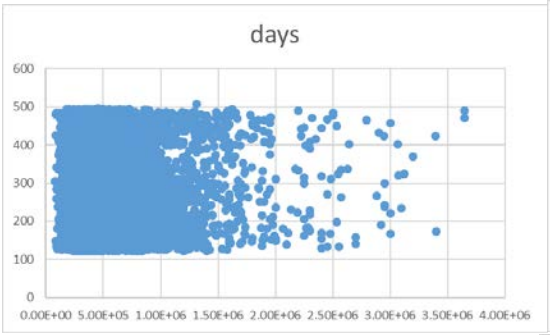


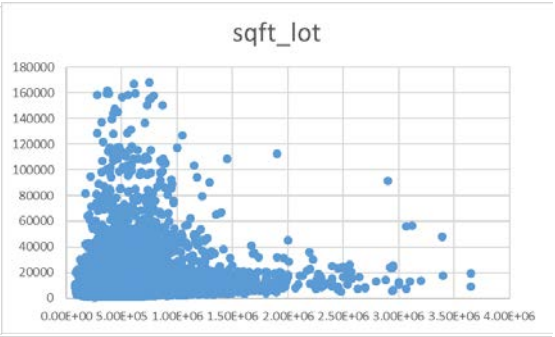


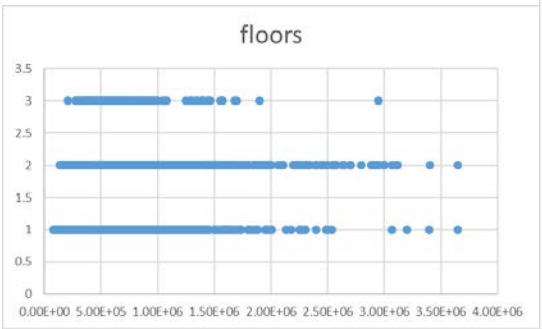


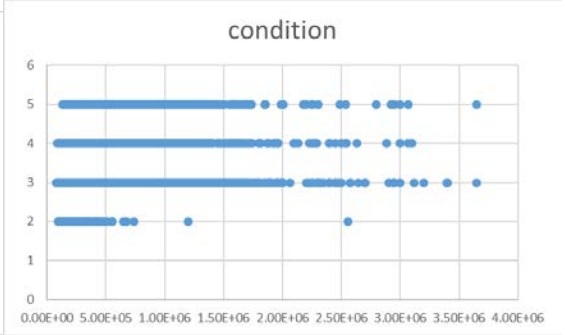


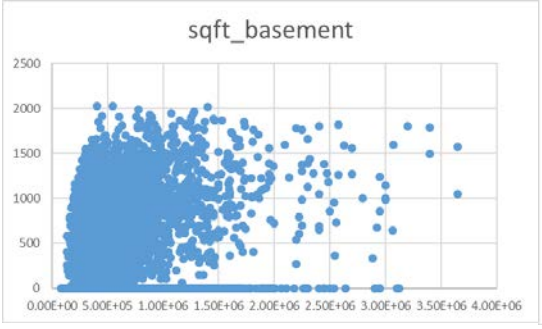
將 year、days、sqrt_lot、floors、condition、sqrt_basement、yr_built、yr_renovated、sqrt_lot15、was renocated、all lot 縮放為[0 1]，視他們為普通特徵。

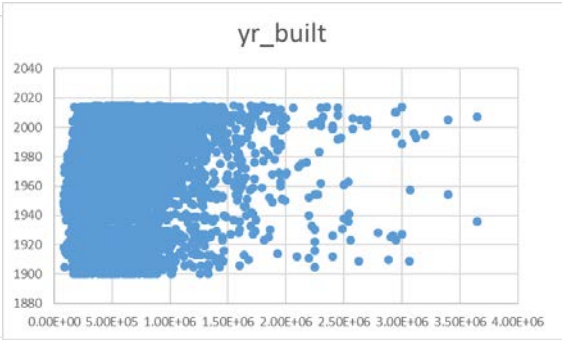


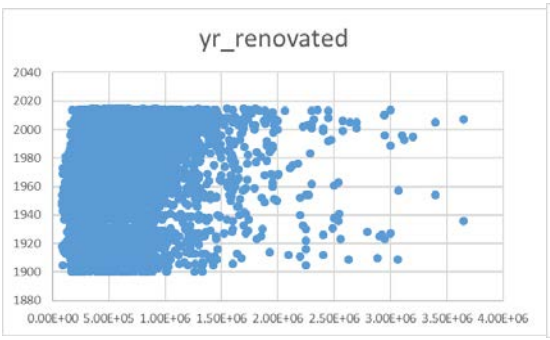


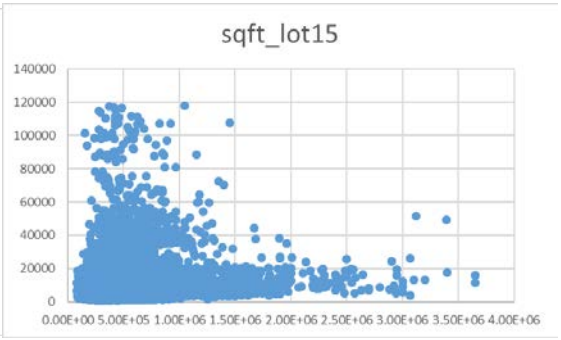


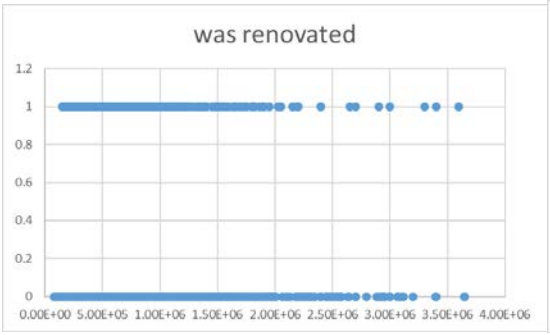


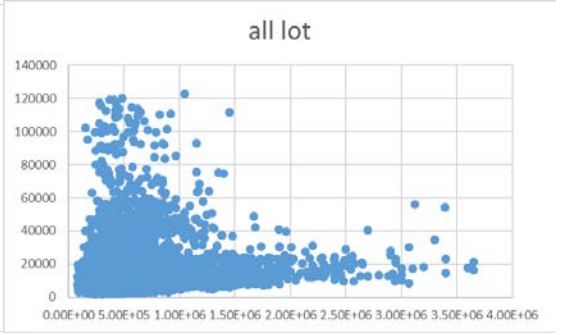




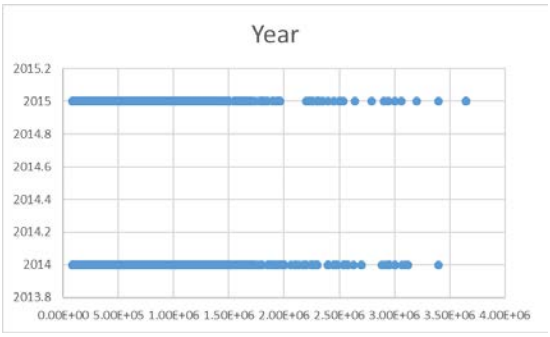


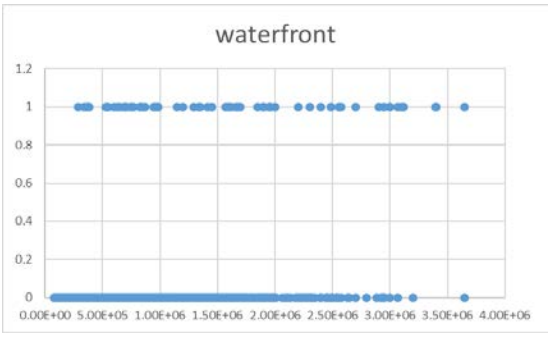


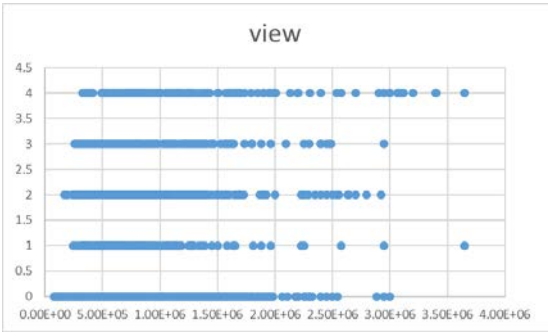




將 year、waterfront、view 縮放為[0 0.3]他們為較差的特徵。







● 訓練 DNN 模型

將所有的資料都處理好後，就能使用 DNN 架構訓練房價預估的模型，我採用以下架構，初始化均為 random_normal、函式均為 relu：

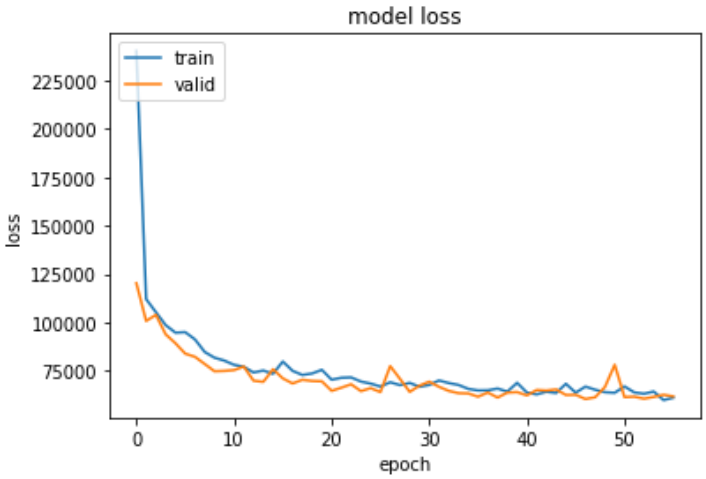
層數	神經元個數
Input layer	25
Hidden 1	128
Hidden 2	256
Hidden 3	512
Hidden 4	512
Hidden 5	512
Hidden 6	512
Hidden 7	512
Hidden 8	448
Hidden 9	384
Hidden 10	256
Hidden 11	128
Output layer	1

訓練參數：batch_size = 128、。

我的訓練次數是當 val_loss 小於 60000 就停止，若經過 110 次都沒達到，就重新訓練，簡化的流程如下：

```
success = 0
while success == 0
    while val_loss > 60000 and epochs < 110
        ( 訓練模型 )
    if val_loss < 60000
        success = 1
```

訓練結果：epochs = 57、loss = 61846、val_loss = 59976，只花 57 次就滿足結果，因為我的步長較大。



D. 結果分析

目前在 kaggle 上的成績為 66324，暫居第 13 名，這幾個禮拜的測試經驗，valid 與 test 的距離相差約為 6000，因此在測試 NN 架構時，觀察 validation 的數值非常重要。要避免 overfitting 的方法是只要 loss 持續下降，同時 val_loss 保持不變或是越來越大，就表示要趕快中止程式。我有先確定這樣的架構有機會達到 6 萬以下，才使用無限迴圈的方式尋找那個最佳解。

我嘗試過的類神經網路組合非常多，有個簡單心得就是，神經元的數量盡量不要小於 batch_size，否則看不到效果。另外，我將 25 個特徵依照相關性縮放，效果顯著，不知道其他同學是怎麼處理這個部分。

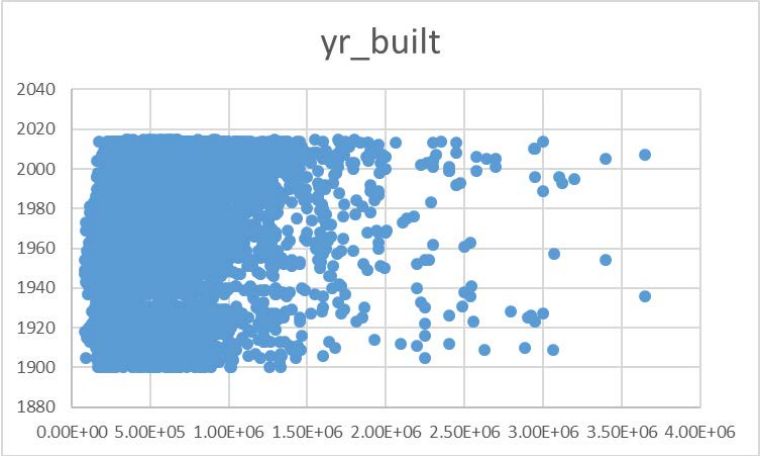
E. 討論

在這次作業的過程中，我將我的模型預測出來的 valid_Y 和真實值比較，只要是新房子，都能有 1 萬以內的誤差值而已。以下舉幾個相差大於 20 萬的案例：

- **預測較實際貴 20 萬以上**

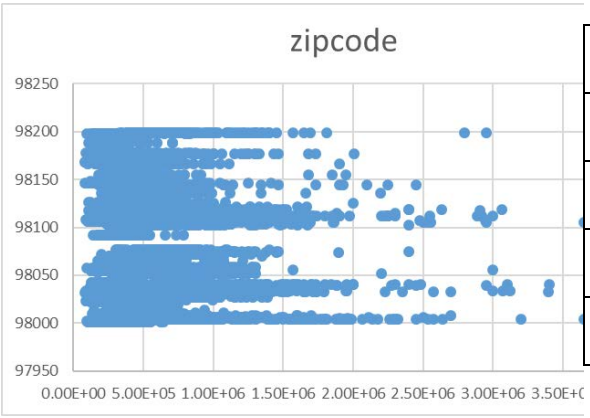
普遍出現在老房子，儘管房間數量、佔地數、評分狀況一切良好，但由於房子老舊，也未整修，事實上就造成價錢較低，但對類神經網絡的架構，這些特徵

都是平等的。如果我單方面把建造年的值縮放較大，如 [0 2]，也會為須考量到房屋的其他條件，沒辦法達到預期的成果。從下圖無法看出年代與價錢的相關性。



● 預測較實際便宜 20 萬以上

我從一般實際的特徵看不出原因，但發現可能是，他們的經緯度、郵遞區號，有許多的高單價的房屋，代表屬於好的地段，就算其他條件都普通，也會提高他的單價。要解決這個問題，或許可以把地段透過新特徵分類，而非單純靠一維度的數值大小去決定。如右下表。



	A 區	B 區	C 區	D 區
←	1	0	0	0
←	0	1	0	0
←	0	0	1	0
←	0	0	0	1