

1. (2%) 請比較實作的 **generative model** 及 **logistic regression** 的準確率，何者較佳？請解釋為何有這種情況？

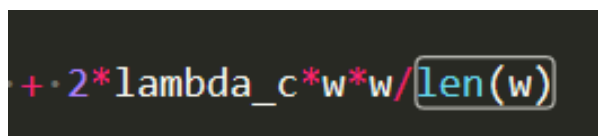
A: 透過修正不同的 **learning rate**，比較不同情況下 **generative** 和 **logistic regression** 的準確率:

Learning rate \ Accuracy	0.05	0.1	0.2
generative	0.8735818	0.8735818	0.8735818
logistic	0.8851321	0.8852959	0.8849272

**Learning rate** 的調整只對 **logistic regression** 有所影響。然而在實作這次作業的過程中，只要 **learning rate** 不是大的誇張或是小的離譜，**logistic regression** 得出的 **accuracy** 都會比 **generative model** 還要好。理由是在這次的作業中，我們擁有 2 萬多筆資料，每筆資料又包含上百筆 **feature**，因此並不算是一個缺乏資料的資料集。對 **generative model** 來說，因其數學式子的關係，可能會對資料有「腦補」的狀況，以至於對 **feature** 作出錯誤的判斷；相反的，如果在資料缺乏的情況下，利用 **generative model** 可能會有奇效。

2. (2%) 請實作 **logistic regression** 的正規化 (**regularization**)，並討論其對於你的模型準確率的影響。接著嘗試對正規項使用不同的權重 (**lambda**)，並討論其影響。(有關 **regularization** 請參考 <https://goo.gl/SSWGhf> p.35)

A: **regularization** 的做法是在 **cross\_entropy** 後面加上  $\frac{\lambda}{n} \sum_{i=1}^n (w_i)^2$ ，在 **python** 中如圖所示；並透過設定不同的 **lambda** 權重，得到以下結果：



```
+ 2 * lambda_c * w * w / len(w)
```

Lambda \ Accuracy	0.1	100	100000
Training	0.8849273	0.8735818	0.8849273
Development	0.8797321	0.8852959	0.8781791

從結果可以看到，對於 **lambda** 而言，設太大或是設太小都不是好的選擇，理由是作 **regularization** 的動機就是希望 **function** 對輸入不要太敏感，但同時也不希望太無感，因此需要通過改變 **lambda** 進行反覆的測試，找到一個折衷的數字才能得到較高的準確率。

至於有沒有做 **regularization** 與否，在我所設定的參數下，對於 **development set** 的準確率是沒有影響的。但因為不能排除測試的資料有極端的狀況出現，因此有實作 **regularization** 是較佳的選擇。

### 3. (1%) 請說明你實作的 **best model**，其訓練方式和準確率為何？

A: 在 **best model** 中，我實作的方式為 **logistic regression**。

首先，因為不能使用 **scipy** 套件，因此我自己實作了 **pearsonr** 相關係數，得出資料中 510 筆 **features** 對 **y** 的關係；透過篩選|相關係數|>0.01 者，最後刪除了 145 項 **feature**。

**Pre-processing** 中，我分別對資料進行 **regularization** 以及 **normalization**，以降低極端值誤差及確保進行 **gradient descent** 中資料能如期的往最低點前進；在 **gradient descent** 中我並沒有使用更進階的 **gradient descent** 技巧。

透過調整 **mx\_iter**、**batch\_size**、以及 **learning\_rate** 並在 **development set** 中進行測試，我認為最佳數據組合為 48、10、0.1，其結果在 **development set** 中有 0.879 的準確率。

最後將結果上傳至 **kaggle**，其 **accuracy** 為 0.89102，成功通過 **strong baseline**。

4. (1%) 請實作輸入特徵標準化 (feature normalization)，並比較是否應用此技巧，會對於你的模型有何影響。

A: 圖為 nrmalization 的程式碼：

```
...if specified_column == None:
    ...specified_column = np.arange(X.shape[1])
    ...# print(specified_column.shape)
...if train:
    ...X_mean = np.mean(X[:, specified_column], 0).reshape(1, -1)
    ...X_std = np.std(X[:, specified_column], 0).reshape(1, -1)
    ...# print(X_mean)
    ...X[:, specified_column] = (X[:, specified_column] - X_mean) / (X_std + 1e-8)
```

透過修正不同的 learning rate，比較不同情況下有無 normalization 在 development set 的準確率：

Learning rate \ Accuracy	0.01	0.02	0.03
有 normalization	0.8755990	0.8791006	0.8791006
無 normalization	0.7698120	0.7611500	0.7661261

由圖我們可以看到有沒有 normalizaion 對於資料的準確率有極大的影響：有進行 normalization 處理過的資料比沒有的資料準確率高出許多

是否應用 normalization 是項很重要的變因，通常經過 normalization 處理過後的資料都有較佳的準確率及較低的 loss；原因是我們在進行 gradient descent 時，每一次的更新都是確切往低處前進，未處理過的資料，其更新的方向則未必是朝低點前進。