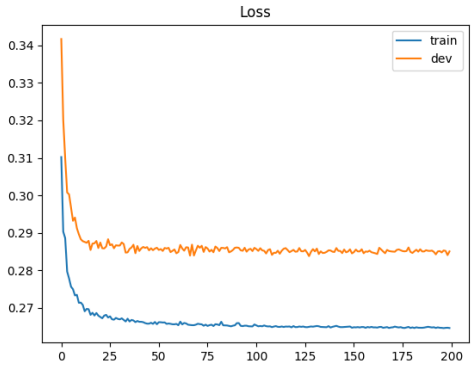
學號：R07921001　　系級：電機所二　　姓名：李尚倫

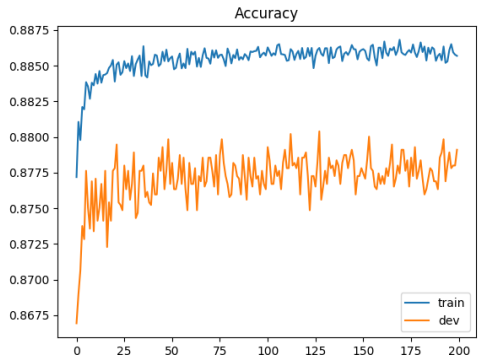
1. (2%) 請比較實作的 generative model 及 logistic regression 的準確率，何者較佳？請解釋為何有這種情況？

以 baseline model 的結果來看 generative model 的訓練結果：

```
Size of training set: 54256
Size of testing set: 27622
Dimension of data: 510
Training accuracy: 0.8725302270716603

Top 10 related data categories：
Retail trade 8.3359375
Not in universe -6.703125
34 -6.33203125
37 -5.80859375
Different county same state -5.65625
Other service -5.42578125
Abroad 4.625
Finance insurance and real estate 4.0
Same county 4.0
Other Rel 18+ never marr RP of subfamily 3.9375
```

以 baseline model 的結果來看 logistic regression model 的訓練結果(iteration=200)：





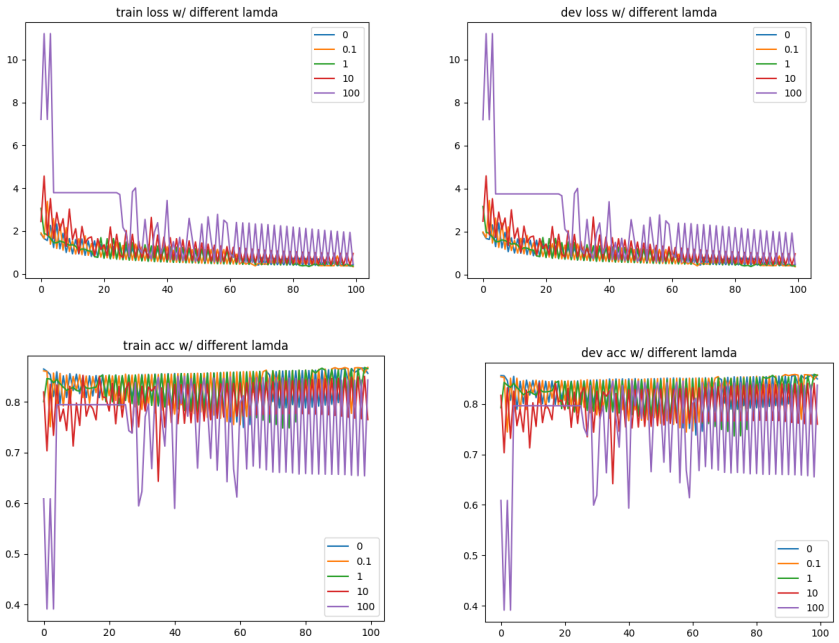

```
Training loss: 0.26462566695822753
Development loss: 0.2850628360679869
Training accuracy: 0.8856850296948597
Development accuracy: 0.879100626612606

Top 10 related data categories：
Not in universe -2.19705786183989
Spouse of householder -1.6574094289726329
Other Rel <18 never married RP of subfamily -1.4264126521048024
Child 18+ ever marr Not in a subfamily -1.3003326945908804
Other Rel <18 ever marr RP of subfamily -1.1759633817911175
Unemployed full-time 1.1158406568740264
Italy -1.1016297598508888
Vietnam -1.0827250388575935
1 0.8058595497596417
Same county -0.7573031321591495
```

$P(C_1|x) = \sigma(w \cdot x + b)$

directly find **w** and b　　Find $\mu^1, \mu^2, \Sigma^{-1}$

兩者都假設擁有一樣的 covariance，用的是同一個機率的 model(擁有相同的 function candidates)，但 discriminate 是靠 data 和 regression 的方式直接求得 w 和 b，而 generative 則是基於 Gaussian distribution 的假設，和假設每個個 dimension 相互 independent(沒有 correlation)等等，在不同的情況下者兩種求解方式各有利弊，而就這次的題目來說，是 discriminate model(logistic regression)有較好的 performance，因為資料中有許多是具有相關性的，也有很多不一定會符合 Gaussian distribution，所以用 logistic regression 做出來的結果會更貼近於真實的反映一點。
因此最後選用的是 logistic model 來當 baseline model：
The best in 200iter, lr=0.2, bch=8, epoch=127
Development loss: 0.28380508846919755
Development accuracy: 0.8803907113896056
Testing accuracy(public): 0.88987 > 0.88617 = simple baseline

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。(有官 regularization 請參考 https://goo.gl/SSWGhf p.35)

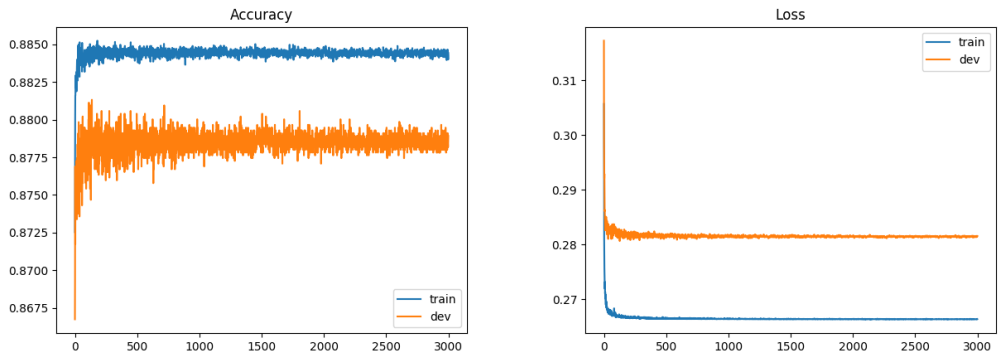


由上圖可以看到，我考慮了五種不同的 lamda(包含=0 也就是不做 regularization)，加入太大的 lamda 做 regularization 會讓 weight 的平滑度被考慮的權重比 training error 大很多，可能會使 model 收斂上困難，loss 產生震盪，而 lamda 太小則都考慮 training error 和沒加入差不多，而在這個 model 中，考慮平滑度沒有太顯著的幫助收斂效果，因此最後 strong baseline 的 model 中也未採用。

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i \right) \right)^2 + \lambda \sum (w_i)^2$$

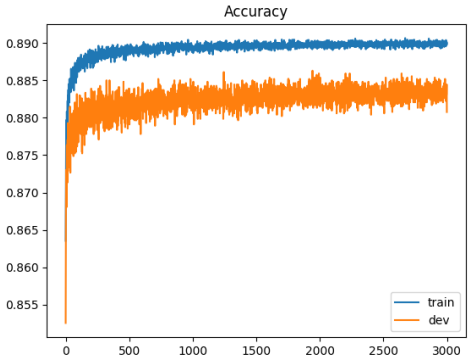
3. (1%) 請說明你實作的 best model，其訓練方式和準確率為何？

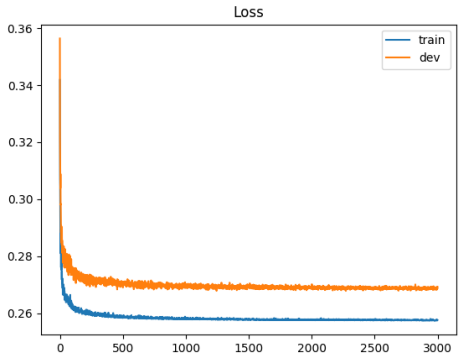
- Batch size, learning rate, iteration
  經過測試，batch size 取 1024 可以使一個 epoch 算快一點且 loss 和取 size=8 差不多。Learning rate 的部分也是經過測試後 0.2 可以在適當的時間下降到比較低的 loss，iteration 的部分統一給 3000 個 iteration 進行 training，但取其中最小 loss 的 weight 為 model 來做 prediction。
- Regularization in regression
  經過前面第二題的實驗，最後採不進行 regularization 的方式(lamda=0)。
- Vanilla gradient v.s. adagrad
  Vanilla gradient: $w^{t+1} \leftarrow w^t - \eta^t g^t \quad \eta^t = \frac{\eta}{\sqrt{t+1}} \quad g^t = \frac{\partial L(\theta^t)}{\partial w}$
  Adagrad: $w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$
  參考作業一實作 adagrad 和本次作業使用的 vanilla gradient 比較，adagrad 具有更好的收斂能力，能在比較少的 epoch 數內找到最小的 loss，故採用 adagrad。
- Filtering data
  借鏡第一次作業的經驗，在 data normalized 後把大於正負三個標準差的 data 濾掉變成等於正或負三個標準差，來減少與 mean 值相差太遠的資料，在作業一中即為雜訊，但本次作業中資料來源應為問卷，並無雜訊或過大的無意義偏差，因此加入後並未改善，最後未採用此方法。
- Normalizing data
  根據第四題實驗，將 data normalized 可以更穩定地趨近最小的 loss，故採用。
- Carefully selecting features
  根據直接觀察 dataset 和用 baseline model 取出的前十相關項目，此 dataset 中有許多無謂且多維的 feature，例如：region/state of previous residence, country of birth father/mother/self，這些資料因為 one hot encoding 會變成 57 維和 129 維的資料，雖然能使 model 變複雜，但太瑣碎，不是有意義的增加複雜度，還會因為增大維度導致計算和收斂變慢，拿掉不會有太大影響。另外或是資料分布不均衡的項目，例如：reason for unemployment 裡面大部分都是 not in universe，不看這項會比較好。
- Increasing dimension of features
  由於根據上述改進後，僅能拿到比 strong 高一點點的正確率：
  Dimension = 318
  Epoch 127, Dev Loss 0.2806616055019925, Dev Acc: 0.8809436048654626
  Testing accuracy(public): 0.89052



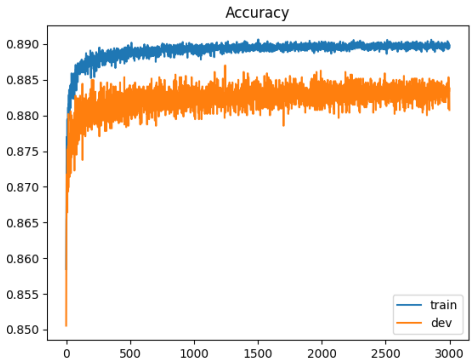
因此決定增加 model 的複查度(維度)，加入每個分類的平方項和根號項：

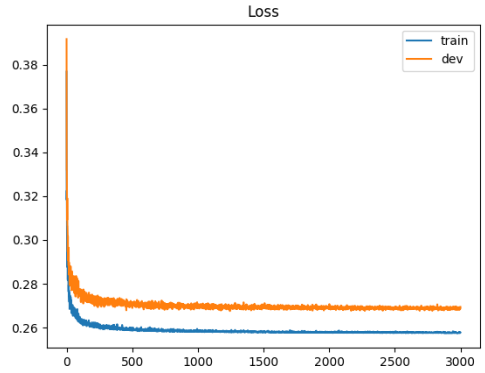
Dimension = 954
Epoch 2271, Dev Loss 0.26787734418288583, Dev Acc: 0.8846295613711759
Testing accuracy(public): 0.89406



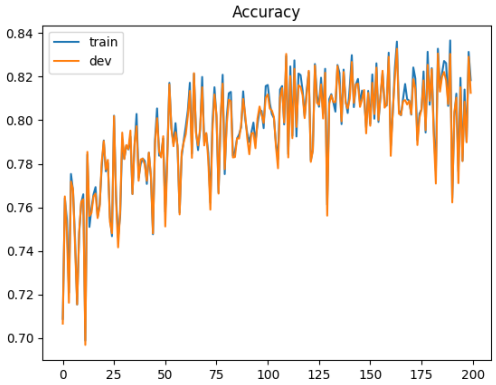


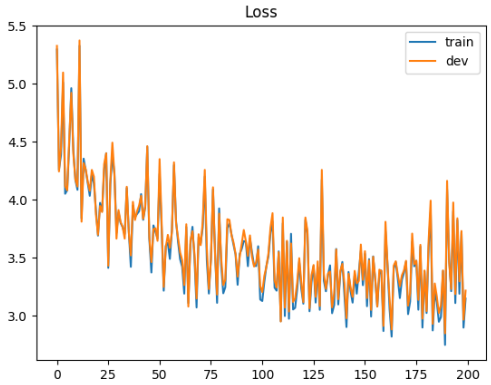
增加為度後 model 的準確率有明顯的改善，因此決定再增加維度，補上原本資料的一次項，變成 1*根號項+2*一次項+1*二次項，有些許的改善為最終定案：
Dimension = 1272
Epoch 1244, Dev Loss 0.26756170678263397, Dev Acc: 0.8870254330998895
Testing accuracy(public): 0.89457 > 0.89052 = strong baseline



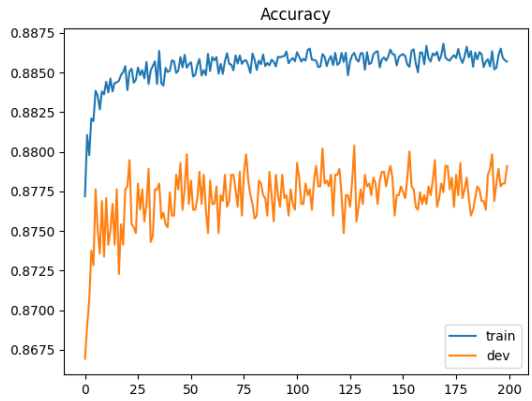


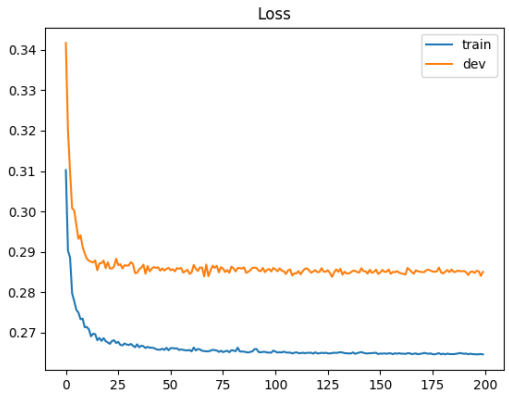
4. (1%) 請實作輸入特徵標準化 (feature normalization)，並比較是否應用此技巧，會對於你的模型有何影響。
Without normalized:
Training loss: 3.149125348988123
Development loss: 3.2175856963260223
Training accuracy: 0.8183903338111816
Development accuracy: 0.8125691116844821



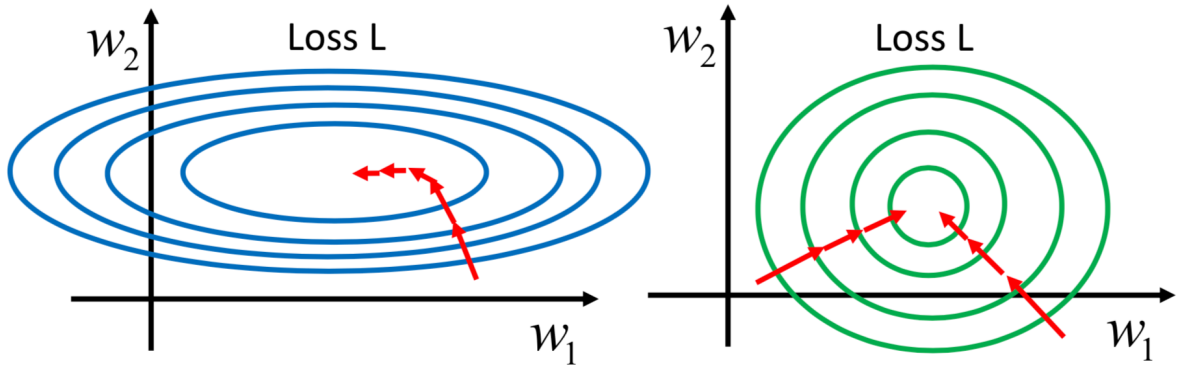


Normalized:
Training loss: 0.26462566695822753
Development loss: 0.2850628360679869
Training accuracy: 0.8856850296948597
Development accuracy: 0.879100626612606





由上面的數據可以看到，少了 data 的 normalization 會讓 model 的收斂變困難非常多，較難找到 loss 的極值，而加入 normalization 會 make different features have the same scaling



如上圖，因為我們的 feature 有 scalar 的也有 one hot encoded 的，因此做 normalization 後 loss 會好收斂很多。