Basic:

1. 遇到的問題:
2. 在寫linear\_activation\_backward這個function時，因為對於linear\_cache, activation\_cache的用法不是很懂，將錯誤的cache傳到其他backward會用到的function時，導致答案怎麼試都不對，雖然是很小的錯誤，但也花了我蠻多時間找出來的。
3. 一開始寫的時候，有些矩陣運算(如sigmoid)地方助教提示說用一行code就可以完成，因為不是很熟悉python，所以一開始還是用for迴圈遍歷每個元素，之後上網查資料才知道如何用更便捷的方式完成。

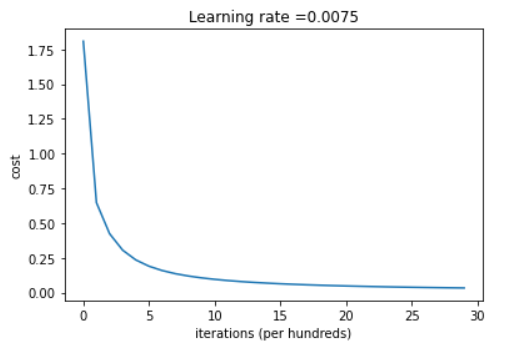
整體來說，助教在template給的提示都很清楚了，只要一步一步做基本上不會遇到很嚴重的bug。

1. 實作build the binary classifier

主要的步驟在最後implementation中的L\_layer\_model都表示得很清楚了:

1. initialize\_parameters，初始化W跟b。
2. 根據當前的parameters執行L\_model\_forward，也就是forward propagation
3. 將得到的AL輸入compute\_BCE\_cost得到cost
4. 執行L\_model\_backward得到gradiant，也就是backward propagation
5. 根據得到的gradiant去更新parameters，做完之後回到第(2)步驟，繼續下一次的iteration。
6. 結果

最後執行implementation時，layers dimension設成[4,1]，因為是binary classifier，所以結果只會是1或0，最後的node也只會有一個。至於執行function的部分，learning\_rate = 0.0075,，num\_iterations = 3000。執行的結果cost可以降到0.0329，train\_data跟validation\_data的accuracy都是1。



Bonus:

1. 遇到的問題:
2. 寫bonus時，我其實沒有遇到甚麼困難，只有在Linear\_activation backward中的softmax\_CCE\_backward寫比較久。因為我花了一些時間看懂derivative of the softmax function，看懂之後發現其實code很簡單。
3. 實作build the multi-class classifier:

跟basic不同的是，最後的節點不只一個，也就是說最後的分類選項不只一個。Bonus用的activation function是softmax，所以時做的部分只有跟basic比只多了forward propagation的softmax function, backward propagation的derivative of softmax還有Categorical cross-entropy loss這三個部分。

在L\_layer\_model中，會看classes的數量來決定要用哪一種cost的計算方式。forward跟backward propagation中也會依照classess的數量決定activation function。

1. 結果

最後執行implementation時，經過不斷的測試調整參數，layers dimension設成[64, 64, 32, 16, 4]，train\_data跟validation\_data的比例是7:2。至於執行function的部分，learning\_rate = 0.0075,，num\_iterations = 30000。執行的結果cost可以降到0.0947，train\_data的accuracy約為0.999375，validation\_data的accuracy約為0.8874。

一開始看到train\_data的accuracy可以到接近1，我當時覺得可能會有overfit的問題，但validation\_data的accuracy也很高。我推測是因為測資的分類結果只有4個，而一開始的input X有64個，所以validation的accuracy才可以到接近9成。我也有嘗試其他的layers dimension和不同的iteration，出來的cost會有些不同，但最後的train跟validation的accuracy都沒辦法再有明顯升高了。

