Deep learning RNN

Prob2

在文件檔中，第 1 個檔案包含所有被 accept 的論文名稱，第 2 個檔案則是所有被 reject 的論文名稱。首先我會蒐集每個論文名稱當中所有出現過的單字，重複的單字不會出現在字典上 2 次。接著我統計每個單字出現的總次數，依照出現次數將所有單字作編號 1,2,…，但現在若使用 one-hot vector 作為 RNN 的 input，其維度實在太大，故會使用 embedding 的方式，將每個單字的維度壓縮成較小的維度來表達。這裡我使用的方法是 Skip-gram 模型，也就是利用當前這個單字，來預測他的上下文單字，上下文的範圍取決於 win 的設定。

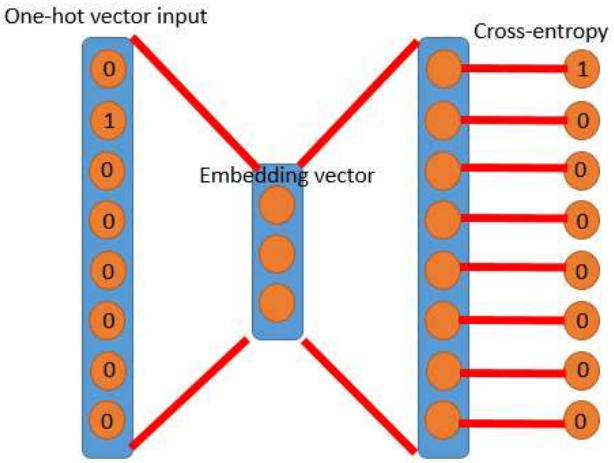
今以 ICLR_accept 論文中的第一篇例:

Minimal-Entropy Correlation Alignment for Unsupervised Deep Domain Adaptation

若我設定 win=1, 即上下文的範圍為單字的前後 1 個單字，則可以創造出訓練資料:

{ minimal, entropy }, { entropy ,minimal }, { entropy, correlation }, { correlation , entropy },{ correlation, alignment }, { alignment , correlation }, { alignment, for }, { for, alignment }, { for, unsupervised }, { unsupervised , for }, { unsupervised, deep }, { deep , unsupervised }, { deep, domain }, { domain, deep }, { Domain, Adaptation }

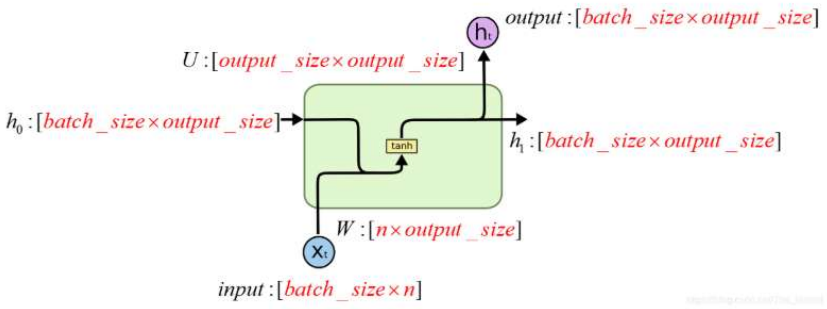
接著將每個句子所產生的訓練資料組成 one-hot vector 輸入給 embedding 的網路做訓練



根據訓練資料

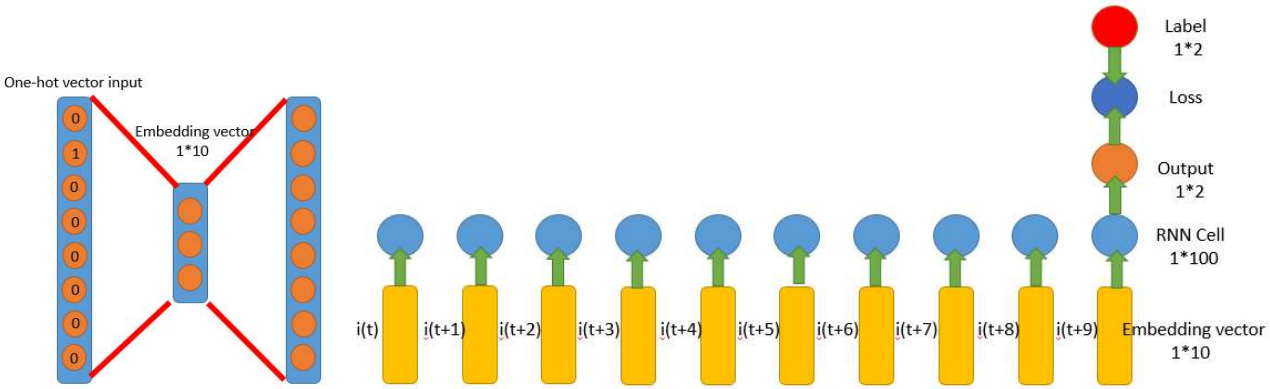
{ entropy ,minimal }，可以把 entropy 寫成 one-hot vector 為[0,1,0,0,0,…,0]，同理 minimal 可寫成 [1,0,0,0,0,…,0]，其中 entropy 為 input data，而 minimal 為 output label，將 NN 的輸出跟 minimal 做 cross-entropy 得到 Loss 來更新這個 Fully connected 網路的梯度。經過訓練幾次 epoch 後，取出這個 embedding 網路的輸入 Weight 和 Bias。假設某某單字的 one-hot vector 為 x, 則經過 embedding 後的向量即為 Weight*x+Bias，根據此結果，可以把每個單字向量轉為較少維度的 embedding vector。

接著我會將每個句子中的每個單字都轉成 embedding vector，並且每個句子只會留下個單字(根據題意)，也就是每句話會根據單字出現的順序，依序輸入給 RNN 做計算。並且 RNN 的輸出我只取在 t=t+10，也就是最終的結果來計算 Loss，也就是同時考慮了過去所有時間的輸入來計算出來的 output。



此圖片取自: 循环神经网络系列（一）Tensorflow 中 BasicRNNCell
https://blog.csdn.net/The_lastest/article/details/83544280

Case 1 配置如下:

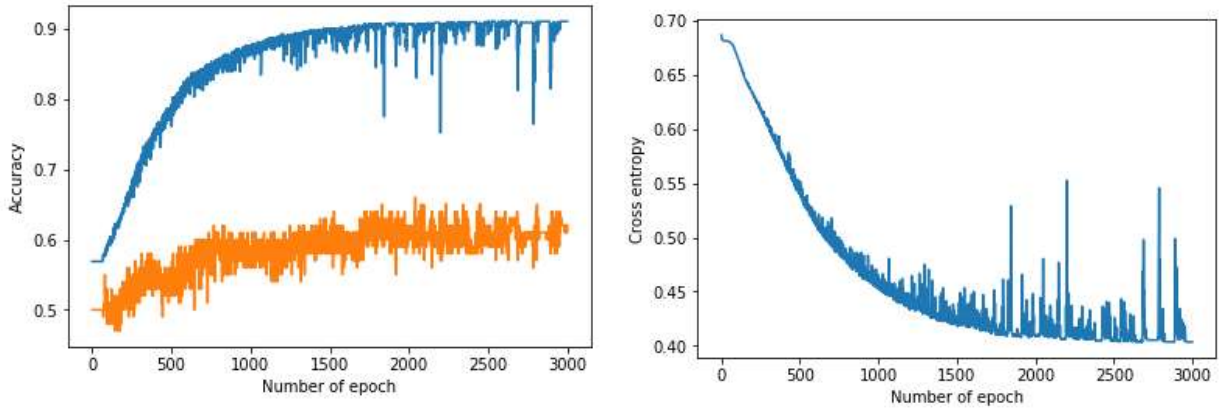


詳細規格如下:

Embedding Network:
Win size=2
Layer1: Fully connected with Weight=dictionary length*10, Bias=1*10, Output with sigmoid
Layer2: Fully connected with Weight=10*dictionary length, Bias=1* dictionary length, Output with softmax
Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:
Layer1: Fully connected with Weight=10*100, Bias=1*100, Output with sigmoid
Layer2: cell: BasicRNNCell, Input with tanh, Output dimension=100
Layer3: Fully connected with Weight=100*2, Bias=1*2, Output with softmax
Iterations=3000, Batch size=50, Learning rate=0.0001

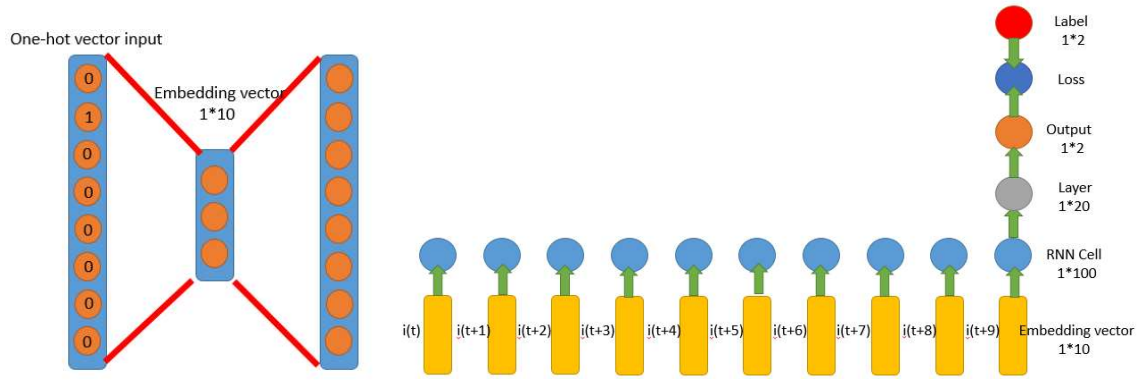
下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve



Final result: Training accuracy= 0.9100486, Testing accuracy= 0.62, Loss= 0.4035824

Training accuracy 並不高，表示神經網路並沒有完全學到訓練資料的行為，考慮是深度不夠深。

Case 2 配置如下:



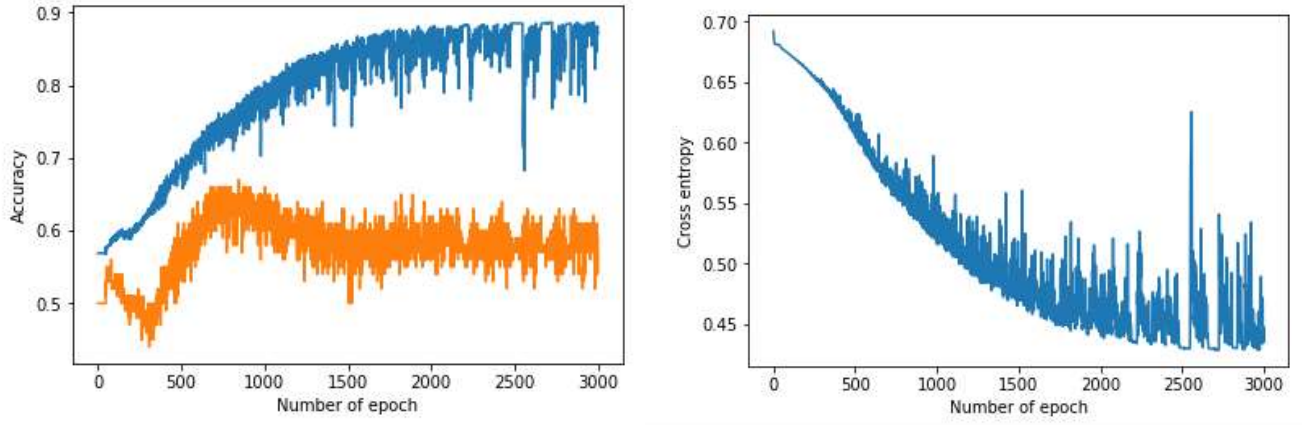
詳細規格如下:

Embedding Network:
Win size=2
Layer1: Fully connected with Weight=dictionary length*10, Bias=1*10, Output with sigmoid
Layer2: Fully connected with Weight=10*dictionary length, Bias=1* dictionary length, Output with softmax
Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:
Layer1: Fully connected with Weight=10*100, Bias=1*100, Output with sigmoid
Layer2: cell: BasicRNNCell, Input with tanh, Output dimension=100
Layer3: Fully connected with Weight=100*20, Bias=1*20
Layer4: Fully connected with Weight=20*2, Bias=1*2, Output with softmax

Iterations=3000, Batch size=50, Learning rate=0.0001

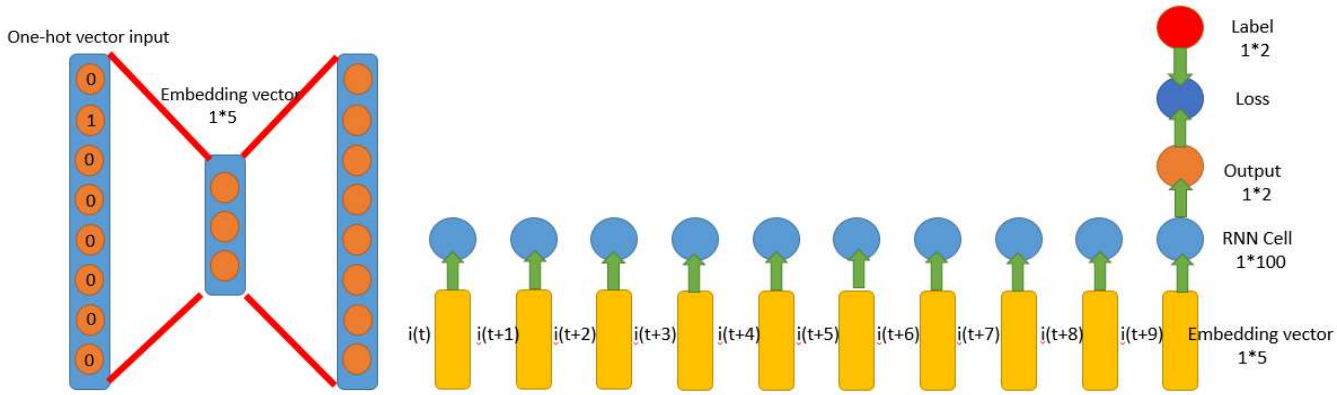
下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve



Final result: Training accuracy= 0.9051864, Testing accuracy= 0.55, Loss= 0.40840477

結果不但使得收斂速度變慢，訓練正確率沒有明顯提升，測試正確率甚至還下降，考慮到 overfitting。

Case 3 配置如下:

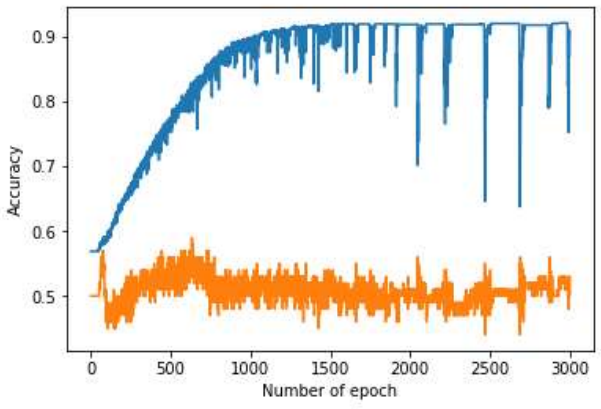


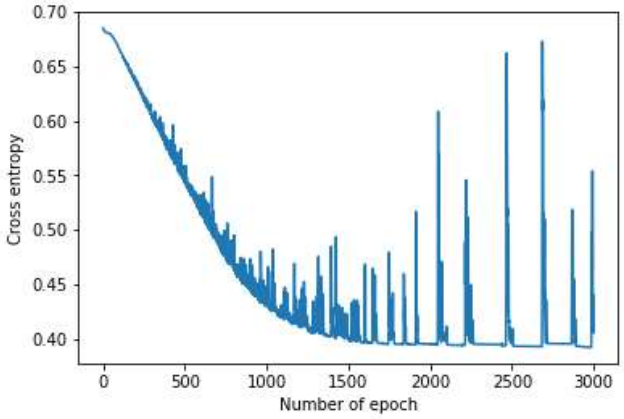
詳細規格如下:

Embedding Network:
Win size=2
Layer1: Fully connected with Weight=dictionary length*5, Bias=1*5, Output with sigmoid
Layer2: Fully connected with Weight=5*dictionary length, Bias=1* dictionary length, Output with softmax
Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:
Layer1: Fully connected with Weight=10*100, Bias=1*100, Output with sigmoid
Layer2: cell: BasicRNNCell, Input with tanh, Output dimension=100
Layer3: Fully connected with Weight=100*2, Bias=1*2, Output with softmax
Iterations=3000, Batch size=50, Learning rate=0.0001

下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve

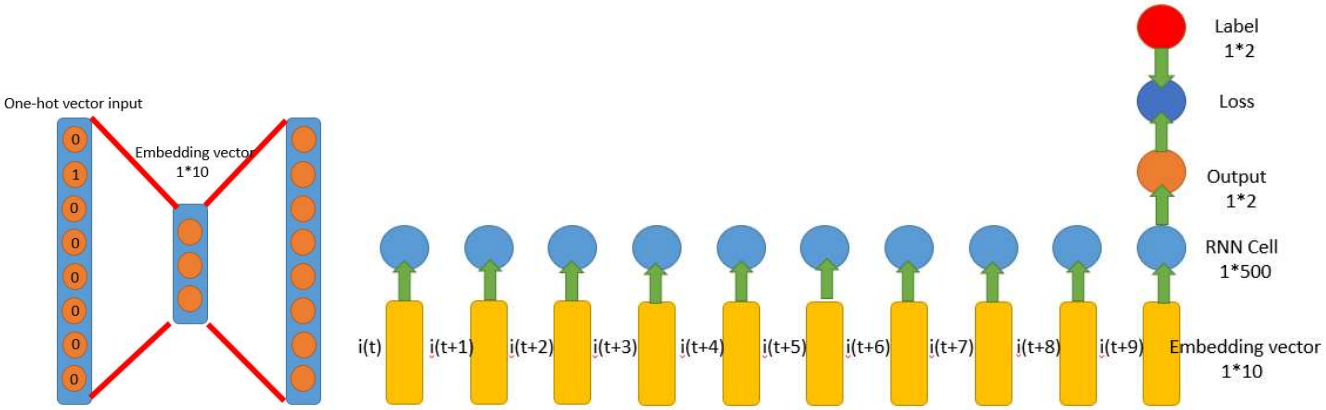




Final result: accuracy= 0.9213938, Testing accuracy= 0.52, Loss= 0.39212418

訓練正確率有上升，但測試正確率變更低了。

Case 4 配置如下:

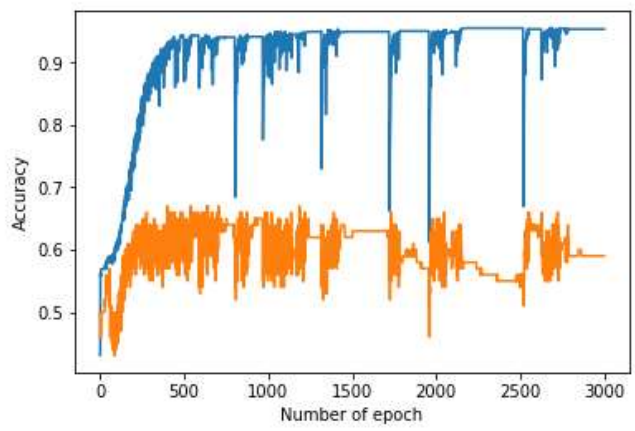


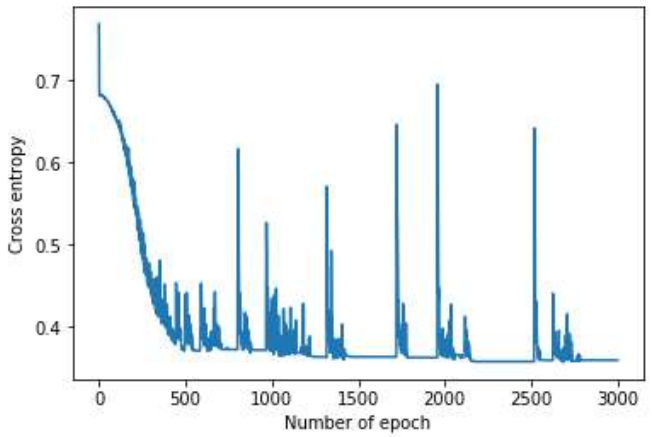
詳細規格如下:

Embedding Network:
Win size=2
Layer1: Fully connected with Weight=dictionary length*10, Bias=1*10, Output with sigmoid
Layer2: Fully connected with Weight=10*dictionary length, Bias=1* dictionary length, Output with softmax
Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:
Layer1: Fully connected with Weight=10*500, Bias=1*500, Output with sigmoid
Layer2: cell: BasicRNNCell, Input with tanh, Output dimension=100
Layer3: Fully connected with Weight=500*2, Bias=1*2, Output with softmax
Iterations=3000, Batch size=50, Learning rate=0.0001

下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve





Final result: Training accuracy= 0.9538087, Testing accuracy= 0.59, Loss= 0.3594565

加寬之後，正確率都有明顯上升了，看來問題便是神經網路的係數不夠多，當 NN 無法 fit 好資料。震動幅度感覺很大，可能是 learning rate 太大的緣故，導致在最佳解附近震盪

Case 5

詳細規格如下:

Embedding Network:

Win size=2

Layer1: Fully connected with Weight=dictionary length*10, Bias=1*10, Output with sigmoid

Layer2: Fully connected with Weight=10*dictionary length, Bias=1* dictionary length, Output with softmax

Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:

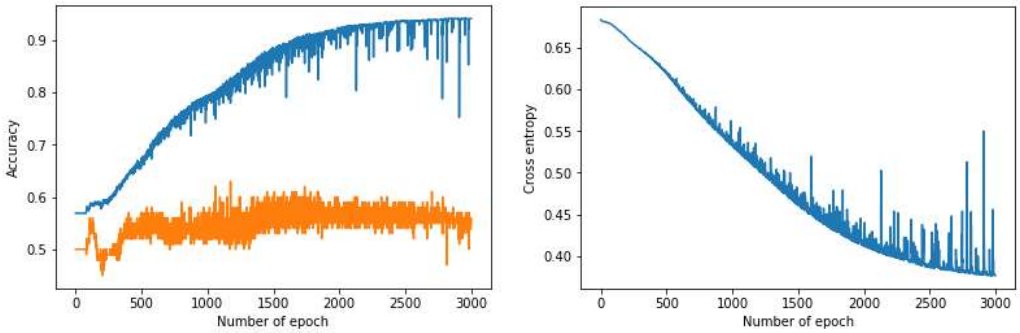
Layer1: Fully connected with Weight=10*500, Bias=1*500, Output with sigmoid

Layer2: cell: BasicRNNCell, Input with tanh, Output dimension=100

Layer3: Fully connected with Weight=500*2, Bias=1*2, Output with softmax

Iterations=3000, Batch size=50, Learning rate=0.00001

下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve



Final result: Training accuracy= 0.9408428, Testing accuracy= 0.56, Loss= 0.3766957

明顯有比之前平穩，可見 learning rate 的影響，但收斂速度也變慢了

Case 6

詳細規格如下:

Embedding Network:

Win size=2

Layer1: Fully connected with Weight=dictionary length*10, Bias=1*10, Output with sigmoid

Layer2: Fully connected with Weight=10*dictionary length, Bias=1* dictionary length, Output with softmax

Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:

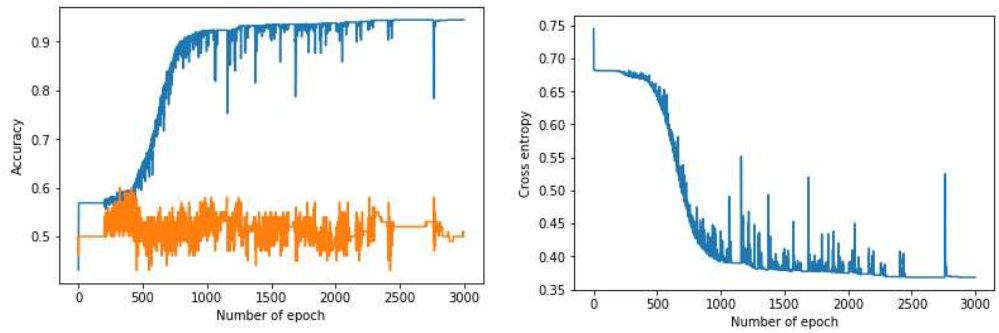
Layer1: Fully connected with Weight=10*500, Bias=1*500, Output with sigmoid

Layer2: cell: BasicLSTMCell, Input with tanh, Output dimension=100

Layer3: Fully connected with Weight=500*2, Bias=1*2, Output with softmax

Iterations=3000, Batch size=50, Learning rate=0.0001

下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve



Final result: Training accuracy= 0.9448947 , Testing accuracy= 0.51 ,Loss= 0.36837244

雖然 training accuracy 有變高，但是 testing accuracy 非常低，考慮可能是 overfitting

Case 7

詳細規格如下:

Embedding Network:

Win size=2

Layer1: Fully connected with Weight=dictionary length*10, Bias=1*10, Output with sigmoid

Layer2: Fully connected with Weight=10*dictionary length, Bias=1* dictionary length, Output with softmax

Iterations=3, Batch size=50, Learning rate=0.0001

RNN Networks:

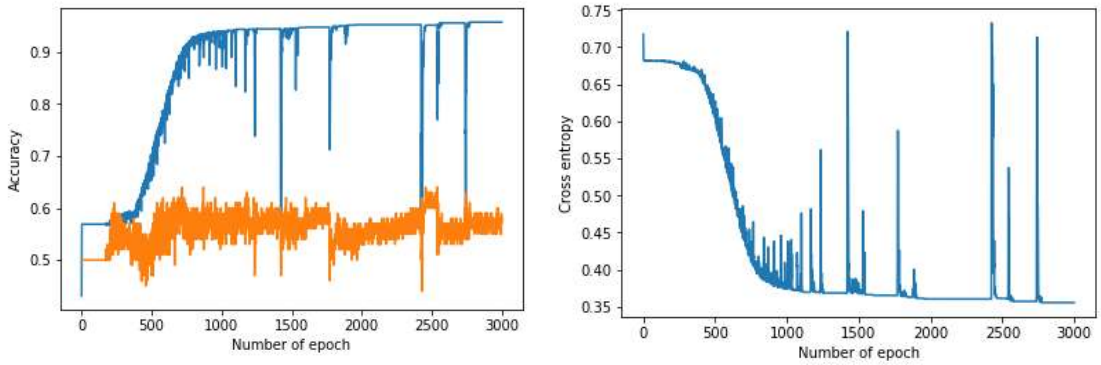
Layer1: Fully connected with Weight=10*500, Bias=1*500, Output with sigmoid

Layer2: cell: BasicLSTMCell, Input with tanh, Output dimension=100

Layer3: Input with dropout, Fully connected with Weight=500*2, Bias=1*2, Output with softmax

Iterations=3000, Batch size=50, Learning rate=0.0001

下圖為每個 epoch 後，對所有的 training data 與 testing data 做正確率計算，和 Learning curve



Final result: Training accuracy= 0.9578606 , Testing accuracy= 0.59 ,Loss= 0.3554091

LSTM 在前期(500 epoch)正確率幾乎沒什麼上升，但到中期正確率上升的比一般 RNN 更快，在 1000 epoch 以前就基本上已經收斂了。RNN 則是正確率穩定上升，上升速度均勻。

程式相關修正參數:

在 Prob2_RNN 的程式中，可修改 Embedding_length(11 行)每個單字轉成指定維度的向量。在程式 281~300 行之間，可以選擇使用的 Cell，包括: BasicRNNCell, BasicLSTMCell, MultiRNNCell。在第 305 行可以決定是否對收入採用 Dropout。在第 325~332 可以使用 learning rate decay，但要修改 363 行讓 sess2 去執行它。