HW1 neural network reports

電機系碩士班111061562 林淞祐

1. **model architecture**

**討論:**

在整個dataset中，有60000traing data 還有10000testing data。在這裡，我使用training : validation為9 : 1，代表有54000個 training samples以及6000個validation samples。這組資料主要是在描述10種”流行服飾”，每組圖片擁有28\*28特徵資料，因為沒有要使用convolution，所以直接對資料實施flatten預處理。

因為最後使用cross\_entropy來計算Loss，所以必須先用one-hot encoding來對labels做處理，又因為總共有”10”種流行服飾，也就將label變成每個大小為(1 x 10)的資料，在為true的index標註為”1”，其餘的標註為”0”。

最後選擇的model模型為左圖:

1.Epoch = 50

2.Batch\_size = 32

3.Accuracy = 88.85%

Back-propagation:

利用softmax.backward算出的derivatives作為gradient，依照微分的chain rule，順序與forward相反，經由full\_connected & ReLU 的backward，修正weights。

Full\_connected.backward : 配合momentum 將原始權重減掉(forward時的權重．gradient)\*learning rate，以此更新權重。

ReLU.backward : 依照forward時取得資料中小於0的值的idx，帶入輸入的gradient，使其idx的值也為0，以此更新gradient。

Loss

Cross\_entropy

(10,1)

(128,1)

(128,1)

Softmax

Full\_connected

W : (128 x 10)

Bias : (10 x 1)

Relu

(256,1)

Full\_connected

W : (256 x 128)

Bias : (128 x 1)

Relu

(256,1)

Full\_connected

W : (784 x 256)

Bias : (256 x 1)

(784,1)

(28,28,1)

Input data

Flatten\_Layer

**2.3. overfitting discusion & hyperparameters comparison**

**Accuracy**

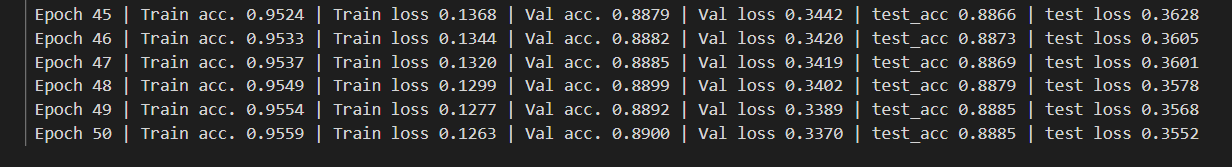
**討論:**

由上圖，可以發現與epoch=20比較，在epoch=50的時候，accuracy 和 Loss已經接近收斂完成的趨勢(88.8%的準確率)，繼續的增加迭代數並不會對accuracy(testing)造成影響，反而在validation samples 的 epoch=20開始，模型開始出現overfitting的現象(traing\_acc – test\_acc > 3%)，因此將learning\_rate在第20次迭代之後開始利用線性遞減做調整，計算每次遞減大小所用的公式為 : distance = (LR – LR/10) / (epoch - 20)。同時可以觀察到validation的準確率收斂程度可以反應testing的收斂程度，這點可以透過上圖來得知。由Batch\_size的比較可以看出，能夠影響Loss最後的收斂空間能夠繼續改善。所以最後的選擇的參數為 : epoch = 50 ; batch\_size = 32。

|  |  |  |
| --- | --- | --- |
|  | **Epoch = 20** | **Epoch = 50** |
| **Batch\_size = 32** |  |  |
| **Batch\_size = 50** |  |  |

**Loss**

|  |  |  |
| --- | --- | --- |
|  | **Epoch = 20** | **Epoch = 50** |
| **Batch\_size = 32** |  |  |
| **Batch\_size = 50** |  |  |

****

**(epoch = 50 , batch size = 32 , activation function = ReLU)**

**4. compare with deep NN**

Loss

Cross\_entropy

(10,1)

(128,1)

(128,1)

Softmax

Full\_connected

W : (128 x 10)

Bias : (10 x 1)

Relu

(256,1)

Full\_connected

W : (256 x 128)

Bias : (128 x 1)

Relu

(256,1)

Full\_connected

W : (784 x 256)

Bias : (256 x 1)

(784,1)

(28,28,1)

Input data

Flatten\_Layer

Relu

**討論:**

多加了兩層的Full\_connected layers，這兩層皆不改變data的維度，純粹加深模型的複雜度。可以發現模型的表現相當差，可見NN越深，不見得越好，因為NN太深時，會無法將資料的feature留住，導致data消散過大，因此深度應該視dataset的複雜度而做出相對應的設計。

Full\_connected

W : (256 x 256)

Bias : (256 x 1)

Full\_connected

W : (128 x 128)

Bias : (128 x 1)

Relu

****

|  |  |
| --- | --- |
|  | **Deep NN (epoch = 50 , batch\_size = 32)** |
| **Accuracy** |  |
| **Loss** |  |

**5. compare to other two activation functions(sigmoid , tanh)**

Condition : Epoch = 50 , batch\_size = 32

**討論:**

由上圖可得，activation function 為tanh時，accuracy=87.5%，為sigmoid時，accuracy=84.7%，兩者相較於ReLU來說，表現較差。因為sigmoid function有 vanishing gradient problem,導致淺層的權重變化越來越少，且函數收斂的速度也比較慢，造成相同condition下，accuracy相對低於另外兩者。而tanh相較於sigmoid，在zero center的地方擁有較陡峭的斜率，但在back propagation時一樣會有梯度消失的問題，只是會較輕微，另外tanh函式因為包含大量的Exponetial計算，需耗費大量的運算資源。相反的，ReLU function並不會有梯度消失的問題，且因為擁有固定的斜率，在微分之後的計算相對容易，因此收歛速度比另外兩者都還要快。

|  |  |  |
| --- | --- | --- |
|  | **tanh** | **sigmoid** |
| **Accuracy** |  |  |
| **Loss** |  |  |