深度學習

HW5

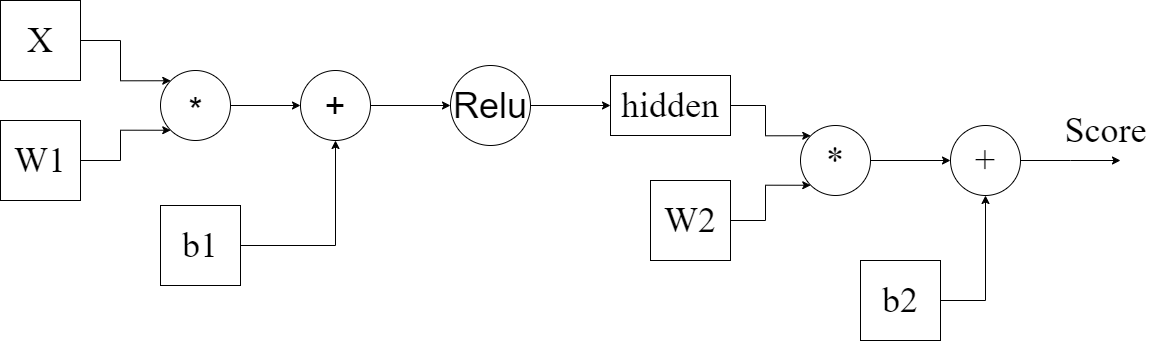
學號 : B103012002

姓名 : 林凡皓

1. Forward pass

* nn\_forward\_pass :

1. 解題思路 :

本題主要是計算神經網路對於每個類別計算出來的分數，整個神經網路的forward path如下圖

根據上圖，先利用torch.clamp( )來計算經過relu後的hidden。接著透過torch.mm( )來計算最後的score。

1. 執行結果 :

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述

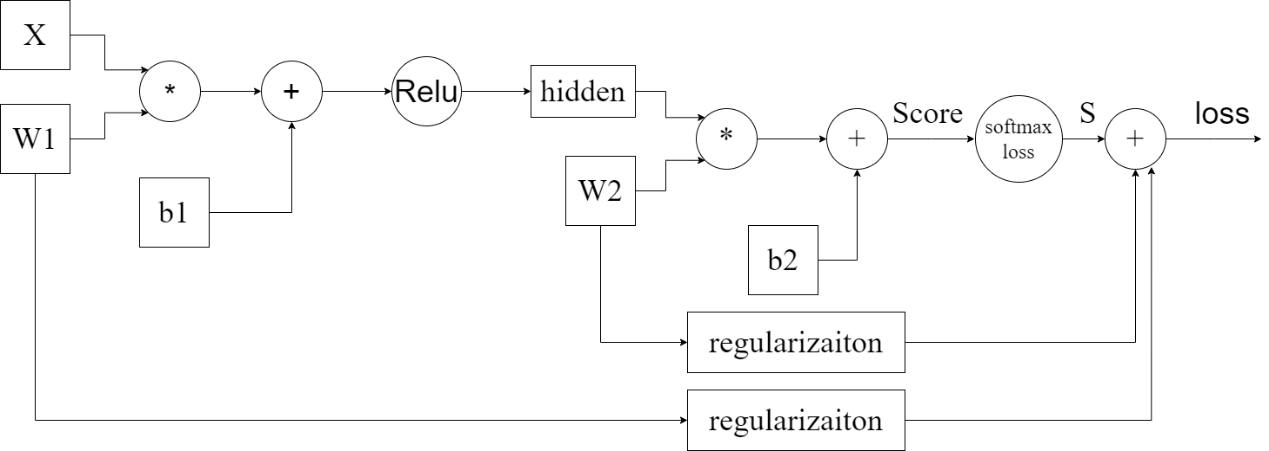
1. 額外討論 :

經過relu後，小於0的數字會被歸零，因此使用torch.clamp( )可以很好的去達到這樣的效果。torch.clamp( )的主要功能為限制tensor的範圍，使用的語法為torch.clamp(input, min, max)，其中min為tensor的最小值，max為tensor的最大值。

* nn\_forward\_backward :

1. 解題思路 :

本題主要是利用forward propagation的方式來計算loss。延伸nn\_forward\_path時的神經網路架構，如下圖



根據softmax loss的定義，如下圖，來完成一個function。

一張含有 字型, 文字, 白色, 符號 的圖片

自動產生的描述 一張含有 字型, 印刷術, 白色, 書法 的圖片

自動產生的描述

在這題，由於網路架構有兩層，因此會有兩個權重值tensor分別為W1和W2，因此regularization term會有兩項分別為R(W1)和R(W2)。

1. 執行結果 :

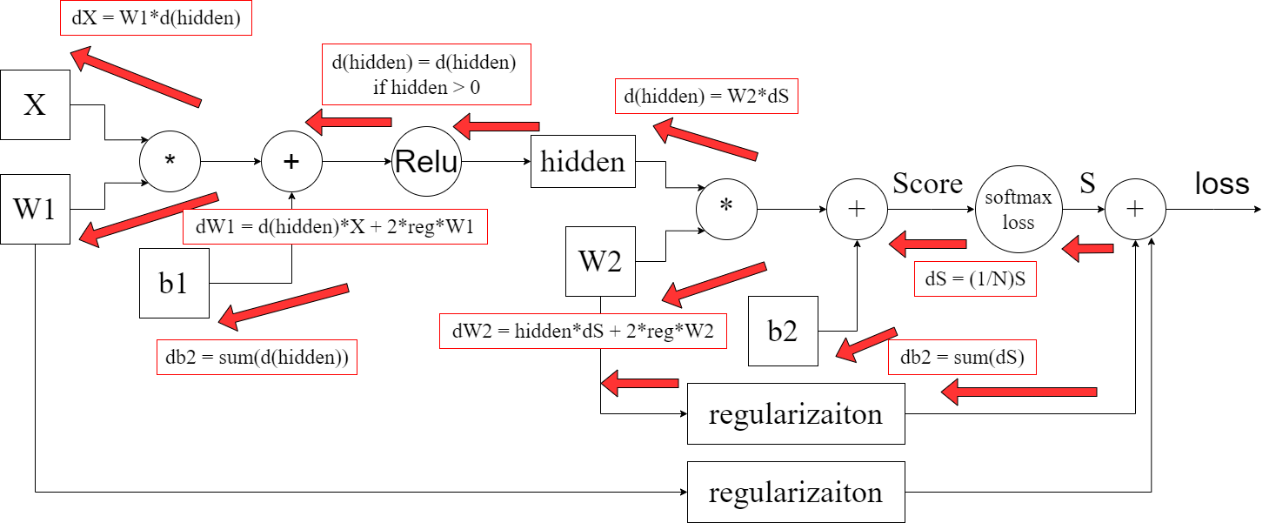
一張含有 文字, 字型, 白色 的圖片

自動產生的描述

1. Backward pass

* nn\_forward\_backward :

1. 解題思路 :

使用computation graph來實作back propagation，computational graph與計算之gradient如下

利用此計算結果加上矩陣相乘的規則(前面矩陣的column數目要等於後面矩陣的row數目)來判斷哪個矩陣要放前面，或是誰需要做轉置，來完成back propagation。

1. 執行結果 :

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

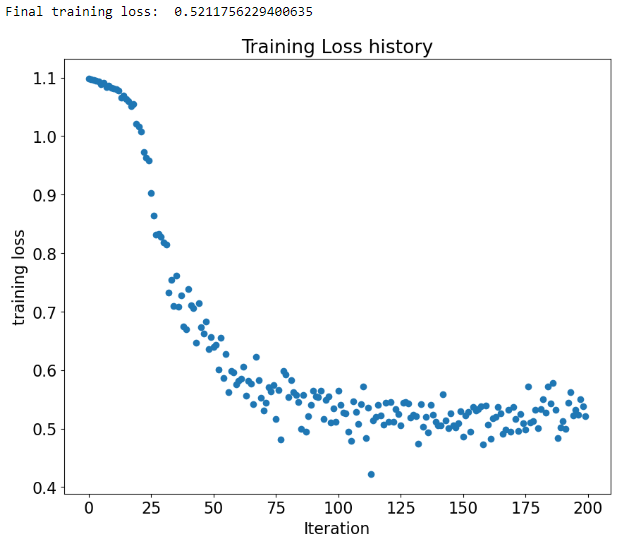
1. Train the network

* nn\_train :

1. 解題思路 :

Trianing的主要在做的事就是更新權重值。更新權重的方式是gradient descent，也就是往負的gradient方向去走。因此，寫法就是w = w – learning\_rate \* dw。

1. 執行結果 :



由上圖可以看出，因為gradient descent的影響，training loss會隨著訓練的次數增加而下降，直到最後趨近於飽和，代表說模型的極限已到。

* nn\_predict :

1. 解題思路 :

nn\_predict是用來當作nn\_train的pred\_func參數。將我們選擇好的參數、training data送進我們選擇的計算分數的function中，接著在計算的分數中找到最大值的index即為預測結果。

1. 執行結果 :

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述

由上圖可以看出準確度隨訓練次數增加而上升，最後趨近飽和，趨勢有點類似於loss。

1. 額外討論 :

為甚麼loss function可以用來計算分數?

關於這個問題，先看到nn\_predict傳入的loss\_func。這裡的loss\_func使用的是先前定義好的nn\_forward\_backward。在nn\_forward\_backward中可以看到有一段code是使用nn\_forward\_pass來計算分數，並在y = None時，nn\_forward\_backward會回傳score而不是loss和grad。接著看到定義nn\_pridect的地方。在定義nn\_predict的時候，我們並沒有傳入y，也就是說y是None，因此nn\_forward\_backward，就會變成是一個計算分數的函數。

1. 額外嘗試

* Gradient problem :

Gradient descent為訓練模型中很重要的一環，在訓練神經網路時，有兩個常見的gradient問題，分別為gradient vanishing和gradient exploding。接下來討論這兩個問題

1. Gradient vanishing :

一張含有 文字, 字型, 行, 圖表 的圖片

自動產生的描述

由上圖可以看到，可能影響gradient的因素有三個，分別為(預測值 – 標籤值)、權重、activation function微分。這三者中，我認為(預測值 – 標籤值)是不會對梯度造成影響的，因此會針對權重與activation function做討論。

* 權重W : 權重的初始化確實可能造成訓練上的問題，但是初始化的問題容易解決，不論是使用多次不一樣的初始化然後做平均，或是使用一些初始化策略像是Normal、uniform、truncated\_norm等，都可以有效解決初始化的問題。
* Activation function : activation function的微分會是造成梯度消失的主要原因。sigmoid function就有發生梯度消失的可能，可以在上圖的右下角處看到，sigmoid微分的最大值大約落在0.25左右而已。以三層的NN來說，gradient縮小倍率是，也就是0.0625倍。光是三層的NN就有如此可觀的下降，更何況是更加大型的NN。這也是為甚麼使用sigmoid function會發生梯度消失的問題。

了解原因後，梯度消失的解決方法如下

* 不要使用sigmoid，改用其他activation function像是relu，relu微分後只有0或是1，比起sigmoid不容易發生梯度消失的問題。
* 在NN中搭配Batch normalization。
* NN不要太深。

1. Gradient exploding :

梯度爆炸主要原因如下

* 梯度累積 : 在back propagation中，如果梯度增長太快，可能會造成權重值更新太大。
* 權重初始化 : 如果權重的初始化沒做好，這會導致靠近輸入層的權重變化很大。

對於梯度爆炸，討論的人數其實不多，這是因為梯度爆炸問題很容易解決，像是設定說如果計算出來的梯度小於-1，就讓梯度等於-1，如果梯計算出來的梯度大於1，就將梯度設為1。

* 嘗試調整training參數 :

1. 調整num\_iters :

* 將num\_iters提升到2000，結果如下

一張含有 文字, 螢幕擷取畫面, 字型, 繪圖 的圖片

自動產生的描述

可以看到說，training loss來到0.48，比num\_iters = 200時的0.52來的低一些，代表說訓練200次的模型經過更多次的訓練，仍然可以降低loss。

* 將num\_iters持續加大到10000，結果如下

一張含有 文字, 螢幕擷取畫面, 繪圖, 字型 的圖片

自動產生的描述

可以看到training loss不降反升，代表說num\_iters = 2000時的結果已經是這個模型的極限，再繼續訓練下去只會適得其反。

* 結論 : 以調整num\_iters來降低模型的loss是可行的，但是嚴格來說，嘗試不同的num\_iters並不是讓模型更加厲害，只是在尋找對於目前模型來說，最佳的表現會發生在哪裡。如果找到目前模型的最佳表現後，還希望去降低loss，那調整模型架構會是比較好的選擇。

1. 調整learning rate :

* 將learning rate降到1e-3，結果如下

一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

由上圖可以看出，loss很高，代表說現在是underfitting的情況，因此常是加大訓練次數，結果如下

一張含有 文字, 螢幕擷取畫面, 水 的圖片

自動產生的描述

我們會發現到，loss並沒有下降，代表說training的過程中，掉到了local minimum的地方，因為learning rate過小的關係，導致無論如何我們都無法跳脫local minimum的地方，所以即便增加訓練次數，對於loss基本上沒有甚麼幫助。

* 由降低learning rate的經驗可以得知，在training的過程中有可能會掉到loacal minimum中，因此不免讓我懷疑在最一開始的模型(learning rate = 1e-1)是否也有掉入local minimum的現象，因此我加大learning rate到learning rate = 1來查看結果，結果如下

一張含有 文字, 螢幕擷取畫面, 陳列, 數字 的圖片

自動產生的描述

可以看出，加大learning rate使loss下降很多，也就是說在最一開始的訓驗過程中有掉入local minimum的現象。

* 持續加大learning rate到learning rate = 1.8，結果如下

一張含有 文字, 螢幕擷取畫面, 陳列, 數字 的圖片

自動產生的描述

我們會發現loss並沒有在下降，這是因為learning rate太大導致它在loss function上左右彈跳，如下圖所示

一張含有 行, 圖表, 寫生, 繪圖 的圖片

自動產生的描述

圖片來源 : <https://ithelp.ithome.com.tw/articles/10189086>

在learning rate = 1.8的時候，只會導致左右彈跳，並不會有loss上升的問題，但如果再加大learning rate，到learning rate = 3，結果如下

一張含有 文字, 螢幕擷取畫面, 陳列, 數字 的圖片

自動產生的描述

可以看到loss有增加的情況。

* 結論 : learning rate的調整對於訓練模型來說非常重要，合適的learning rate可以讓訓練過程順利找到global minimum，而太小或太大的learning rate會讓loss無法順利下降，甚至出現不降反升的現象。

1. 心得總結

這一次作業主要的內容是實作forward propagation和back propagation，透過computation graph的幫助，讓一個原本很複雜的微分問題變得相對容易解決。

教授上課時有提到做gradient descent時會發生的問題，因此我也透過這次作業的機會來了解一下詳細內容。透過網路搜尋，我了解到梯度消失與梯度爆炸的原因與解決辦法。原本我以為過大的神經網路準確度部會太好的原因在於overfitting，但是瞭解了梯度消失與梯度爆炸後才知道，原來過大的模型有可能造成loss無法順利降低的問題，才導致準確度不如小一點的神經網路。

最後透過調整一些training的參數，成功的讓loss再一次的降低。在這過程中，我實際的看到訓練次數與learning rate對訓練模型的影響，也了解到local minimum對於訓練模型時會帶來什麼樣的困擾。

1. Reference

[1] WenWei Kang “Back-propagation” <https://medium.com/ai-academy-taiwan/back-propagation-3946e8ed8c55>

[2] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. <https://chat.openai.com/>

[3] ZZY\_dl “深度學習筆記(五): 學習率過大過小對於網路訓練有甚麼影響以及如何解決” <https://blog.csdn.net/m0_51004308/article/details/113449233>