Neural Networks and CIFAR10 Classification

B083040029 邱品諺

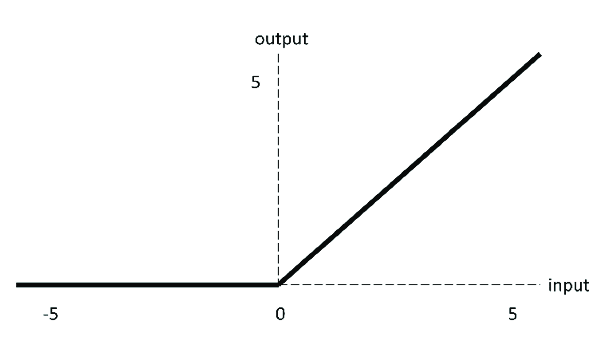
在作業4我們實作了Logistic Regression，其為單層的Network，且在作業4只利用1項Feature（也就是一個Neuron）訓練，也就是線性方程（f=Wx）再通過一個Non-linearity（Sigmoid）。但Deep Learning之所以叫Deep，表示Network通常不會只有一層，而本次作業的目的即為以CIFAR10為資料集建立一個多層的神經網路。

1. Neural Network Models
   1. Modularization

訓練多層神經網路需要透過Loss Function對Model Parameters的微分得知各參數的更新方向，以持續更新至模型收斂。由於多層的神經網路有許多Model Parameters，因此在實務操作上需要在forward的時候記錄神經網路中各個Model Parameters，以方便在Backward時方便計算Gradient。

* 1. Layer Example: Non-Linearities

除了作業4使用的Sigmoid外，ReLU為現今Deep Learning最常用的non-linearity，下圖為ReLU的函式圖：



* 1. Affine Layers

此定義了每一層Network的Forward及Backward方式。Forward主要透過以下方程式處理Neuron間的計算：

而Backward主要為計算Loss Function對各Model Parameters的微分，核心概念主要為Chain Rule，以上式為例：需先計算Loss Function對z的微分，再計算z對Model Parameters的微分。以下為計算式：

但在電腦運算中，資料多以矩陣儲存，下列為矩陣的表示方式：

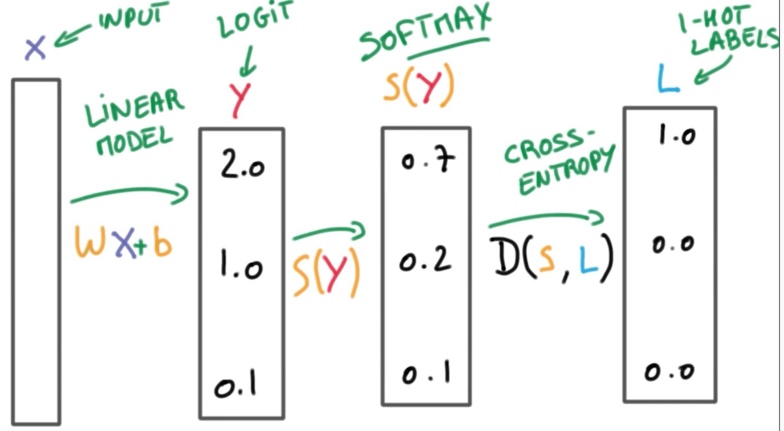
* 1. N-layer Classification Network

classification\_net此函式主要包含了Linear layer的Forward及Backward、Activation Function的Forward及Backward，並紀錄儲存每一層的Model Parameters以及Backward後Loss Function對各個Model Parameters的微分。另外，可透過輸入參數決定Input Layer、Hidden Layer、輸出類別的數量，以及神經網路的層數和Activation Function的種類。

1. CIFAR10 Dataset

CIFAR10為50000張32x32x3訓練資料的資料集，而本作業中取1%（500張）作為欲建立模型之訓練資料，且Batch大小為8，因此如果drop last的話會有62個batch，而input layer的neuron數為32x32x3=3072。

1. Cross-Entropy/Softmax Loss From Logits



首先，Softmax為將輸出結果轉換為[0,1]機率分佈的函式，先透過exponential將輸出轉換為保證>=0的數值，再透過Normalization轉換為[0,1]間的機率分佈。另外，Softmax其實是Sigmoid的multi-class、廣義版本，而Sigmoid為two-class的版本。以下為Softmax的函式：

另外，Softmax會有Numerical Stability的問題，也就是當輸入至Softmax函式的數值太大時進行exponential運算時會Overflow，因此需要減去所有的輸入減去輸入中的最大值，以確保Softmax不會Overflow。

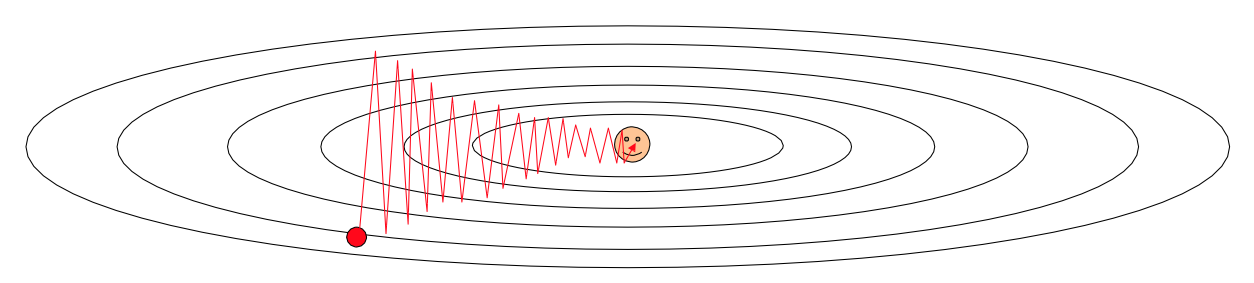
接著，Logit為進入Softmax前的layer，與其他layer不同的地方為其沒有通過Activation Function。最後，Cross-Entropy為一種計算分類問題的Loss Function，E透過計算Cross-Entropy可得知模型輸出的預測結果與目標相差多少，以下為Cross-Entropy的函式：

1. Optimization
   1. Gradient Descent vs Stochastic Gradient Descent

如果以Gradient Descent的方式更新Network，將以整個資料集作為輸入進行訓練，這樣進行訓練雖然可得到很好的結果，但是訓練的速度十分緩慢且需佔用非常大的記憶體空間，因此不是一個具有效益的方法。

* 1. SGD

SGD由於其每次輸入進Model的資料為隨機取點或mini-batch以進行訓練，因此可透過平行化改善Gradient Descent訓練速度緩慢的問題，但訓練結果可能較為noisy，由於資料的隨機性因此可能造成在梯度下降時的方向可能都不一樣而導致下圖的情形，而我們想要的是一個既訓練速度快又可以穩定向最低點下降的方式。

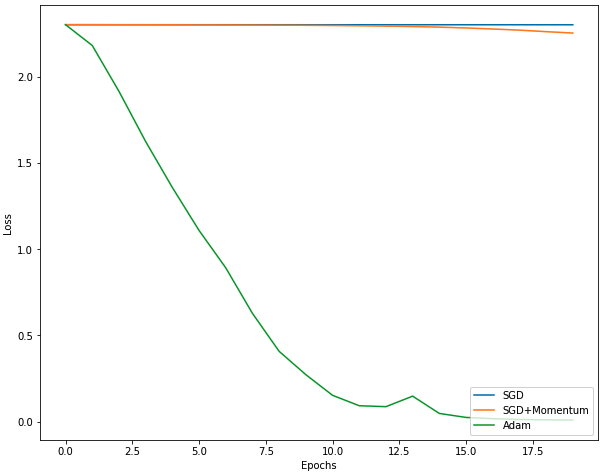


* 1. SGD + Momentum

增加動量的方式改善了上述問題，因為在每次更新時多考慮了前一次的下降速度，如此不僅可修正下降的方向外，又可保留前一次的速度以適應當時的地形進行不同的更新方式。以下為SGD + Momentum的函式：

* 1. Adam

Adam主要結合了1st 及2nd order的momentum。在訓練執行20個epochs後，相較於上述的更新方式，Adam有較快的更新速度，由下圖可見更新速度的差異：



1. 最後，Neural Network更新的一個epoch為下列步驟：
   1. Affine layer通過每一層的Activation Function直到Logit層
   2. Softmax轉換成[0, 1]機率分佈
   3. Cross Entropy計算Data Loss
   4. 由Weights計算Regularization Loss
   5. 由Total Loss對Model Parameters進行更新