Deep Learning

Assignment 2 – Question 2

Abstract

The aim of the project is to design a computation graph with both forward propagation and backward propagation with three class of data.

Problem Statement

CIFAR10 is a public datasets with 10 classes and 6000 images each class. By default it is split into train and test datasets. The purpose of the task is to create a neural network, give the training set as the input which includes both the training and the validation sets. The network is hyper tuned to achieve maximum accuracy and finally evaluated against the test sets.

Proposed Solution

The purpose of the classification is to determine to which subset the given test image fits. Since three classes are selected, multiclass classification is implemented. In case of model design, the activation functions are sigmoid for all the layers and softmax function for the last layer as this is multiclass classification.

Multiclass classification:
The purpose of this classification model is to determine the solution of high class processing such as image processing with multiple classes. The given data belongs to either of the multiple classes. In this case three classes are obtained from the CIFAR10 data.

Model design:
Activation = sigmoid
Sigmoid function is used for activation. It combines the functions of multiplying the input vectors and the weight vectors. Then the sigmoid function is found.

Activation = softmax
The last layer uses softmax as the activation function. Softmax layers determine multiclass probabilities better and therefore are used here. It converts the output values obtained from the hidden layers to probabilities between zeros to one. This probability value is used to determine the class of the given image.

Implementation Details

Implemented using anaconda and Jupyter.
Implementation issues and processes includes:
1. Since the datasets were huge, it was necessary to increase the number of hidden layers to give better results. It was also necessary to increase the number of iterations through the training set to make the model better.
2. The task was implemented in Jupyter notebook. It was found that Jupyter stored the values of the models and at each hyper tuning the kernel was restarted.

3. The datasets were found to be huge and thus processing the data with so many epochs and hidden layers were difficult.
4. The process of running such an entire data on the local CPU was difficult.

Output obtained:

```
W = [[-0.01314412 -0.00094724 -0.00427989]
 [ 0.00305823  0.01117694  0.00056257]
 [ 0.00102789  0.00480435  0.00208372]
 ...
 [ 0.00763975 -0.00621717 -0.00254085]
 [-0.00964656 -0.01512588  0.00163176]
 [-0.01553434  0.00151929 -0.00118366]]

B = [[ 0.00319039  0.00387922 -0.00148443]]

x1 = [[170 180 198 ...  73  77  80]
 [159 102 101 ... 182  57  19]
 [164 206  84 ... 122 170  44]
 ...
 [145 161 194 ...  37  39  54]
 [189 211 240 ... 195 190 171]
 [229 229 239 ... 163 163 161]]

a0 = [[ -16.16137337   10.66272905  -28.33179919]
 [ -57.08310076   22.15582951 -102.94878474]
 [ -74.28491613  -67.49568553  -43.58044166]
 ...
 [ -59.28341437  -17.07522754 -149.44208947]
 [-124.4865879   -44.92349987  -95.56421585]
 [ -39.87055799   11.18301258  -51.50317583]]

a1 = [[9.57645309e-08 9.99976599e-01 4.96199071e-13]
 [1.61854333e-25 1.00000000e+00 1.94944447e-45]
 [5.47609327e-33 4.86402974e-30 1.18373549e-19]
 ...
 [1.79283478e-26 3.83992625e-08 1.25350243e-65]
 [8.63300130e-55 3.09009549e-20 3.14042474e-42]
 [4.83544752e-18 9.99986092e-01 4.28997674e-23]]

loss = 15411.554419467193

da1 = [[ 2.87293593e-03 -7.02015505e-01  1.48859721e-08]
 [ 4.85562998e-21 -7.16088522e-06  5.84833342e-41]
 [ 1.64282798e-28  1.45920892e-25 -3.00000000e+04]
 ...
 [ 5.37850435e-22  1.15197788e-03 -3.00000000e+04]
 [ 2.58990039e-50 -3.00000000e+04  9.42127423e-38]
 [ 1.45063426e-13 -4.17248286e-01  1.28699302e-18]]

da0 = [[ 2.75125335e-010 -1.64271412e-005  7.38640556e-021]
 [ 7.85904750e-046 -1.70927590e-015  1.14010013e-085]
```

```
 [ 8.99627926e-061  7.09763560e-055 -3.55120646e-015]
 ...
 [ 9.64276968e-048  4.42350992e-011 -3.76050729e-061]
 [ 2.23586134e-104 -9.27028648e-016  2.95868027e-079]
 [ 7.01446582e-031 -5.80312368e-006  5.52117012e-041]]

dx1 = [[ 1.55567898e-08 -1.83604263e-07 -7.89214555e-08 ...  1.02132392e-0
7
   2.48472361e-07 -2.49618012e-08]
 [ 1.61909043e-18 -1.91044672e-17 -8.21195992e-18 ...  1.06268547e-17
   2.58543071e-17 -2.59687911e-18]
 [ 1.51987789e-17 -1.99779952e-18 -7.39971394e-18 ...  9.02308149e-18
  -5.79470841e-18  4.20343155e-18]
 ...
 [-4.19011500e-14  4.94412868e-13  2.12520905e-13 ... -2.75017025e-13
  -6.69094931e-13  6.72057712e-14]
 [ 8.78116407e-19 -1.03613398e-17 -4.45377021e-18 ...  5.76349245e-18
   1.40221268e-17 -1.40842174e-18]
 [ 5.49693704e-09 -6.48611413e-08 -2.78802380e-08 ...  3.60789924e-08
   8.77773692e-08 -8.81660512e-09]]

dW = [[1.24627119e+07 8.38445975e+07 6.83158060e+06]
 [1.07849444e+07 9.83281317e+07 7.98103903e+06]
 [7.39721086e+06 1.08302439e+08 8.99178028e+06]
 ...
 [2.54745240e+07 8.14794463e+07 9.18056290e+06]
 [2.38413398e+07 8.25715914e+07 9.65992575e+06]
 [2.07932340e+07 7.62229035e+07 1.01660618e+07]]

dB = [[ 2.75125335e-010 -1.64271412e-005  7.38640556e-021]
 [ 7.85904750e-046 -1.70927590e-015  1.14010013e-085]
 [ 8.99627926e-061  7.09763560e-055 -3.55120646e-015]
 ...
 [ 9.64276968e-048  4.42350992e-011 -3.76050729e-061]
 [ 2.23586134e-104 -9.27028648e-016  2.95868027e-079]
 [ 7.01446582e-031 -5.80312368e-006  5.52117012e-041]]
```

Evaluation metrics

Initially the validation data obtained from the train data was ran across the model and finally the test data was ran across the model which was tuned with the test label.
Finally the accuracy of the model obtained was around 74%.

```
epoch[0] = 0.749711887
epoch[10] = 0.743703324
epoch[20] = 0.743196063
epoch[30] = 0.743032262
epoch[40] = 0.742953616
epoch[50] = 0.742908961
epoch[60] = 0.742876145
epoch[70] = 0.742844451
epoch[80] = 0.742815717
epoch[90] = 0.742790564
epoch[100] = 0.742768513
```

This accuracy is obtained after many hyper tunings by
1. Changing the number of layers of the model
2. Changing the epochs
3. Changing the batch size
4. Choosing the correct optimizer.

References:

1. Code given by professor
2. http://www.cs.virginia.edu/~vicente/vislang/notebooks/deep_learning_lab.html