MLLD:- ASSIGNMENT 2

Dinesh Kumar
dineshk@iisc.ac.in

1 Method's Local Implementation:-

For predicting the class of a document the steps followed are:-

- the weight of a term that occurs in a document is simply proportional to the term frequency.Finding the number of times a term "t" occurs in the document "d", basically called as the 'term frequency' for a term in the document.
  $tf(t,d) = log(1 + f_{\text{t,d}})$

- Then find the 'inverse document frequency' of terms in the document. "idf" factor is incorporated which diminishes the weight of the term that occur very frequently in the document set and increases the weight of term that occur rarely.
  $idf(t,D) = N/|d \in \ D : t \in d|$

- Then tfidf(w,t,D) is calculated as:-
  $tfidf(w,t,D) = tf(w,d) * idf(w,D)$

- Now, i have as many classifiers as the number of classes. For each document in the test set, find the probability of the document belonging to a particular class by passing it over that classs classifier. the class with highest probability is the winner. Calculate accuracy accordingly.

- For a document that belongs to a particular class, its output is 1 and for the rest of the classes output is 0 , on top of it Binary classification is done[2].

- I have used n-gram model and observed that increasing n increases accuracy.

- Fig.1 shows the plot of loss against epochs and the loss function used is as follows:-
  $loss = (y-1) * log(1-p) - ylog(p)$

- The increasing and decreasing learning rates are used as follows:-
  $Increasing : lr = lr + 0.1 * lr * epoch$
  $Decreasing : lr = lr/(1.5^{\text{epoch}})$

- The decreasing function is not skipping the optimal value hence performs better than the constant and increasing ones.

2 Implementation of Parameter Server:-

Implementation of the Asynchronous SGD in Tensorflow:-

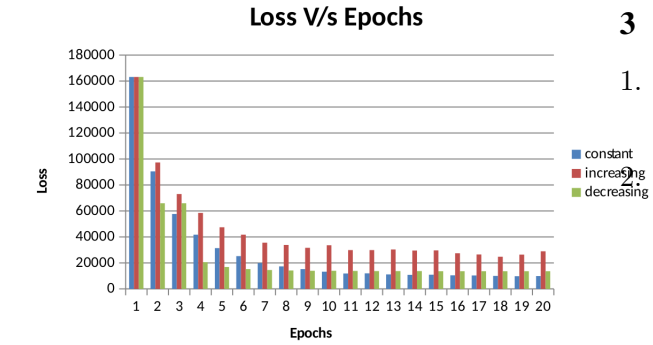
- Using the link[1] the parameter server implementation for the publicly available dataset MNIST was studied and then modified for the DBPedia dataset.

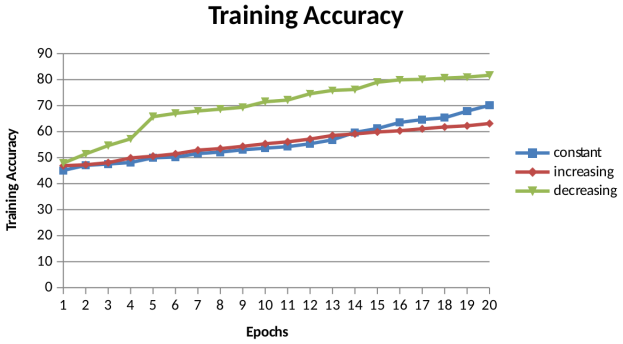
- Used GradientDescentOptimiser with cross-entropy loss.Loss is inversely proportional to the no of epochs, but accuracy didn't improved much.

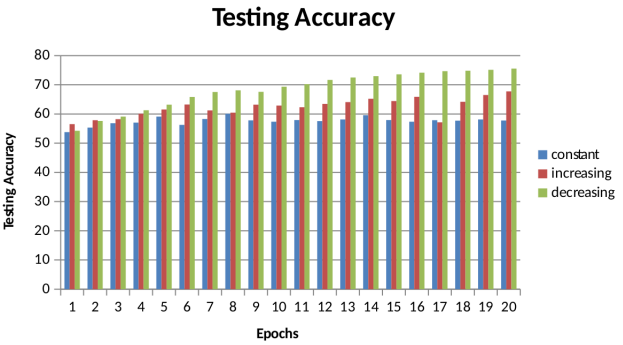
- as the no. of workers increases, the time taken for decreases and just implemented term frequency model.

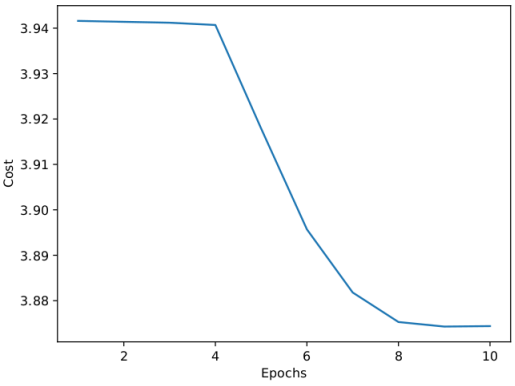
No. of workers	Time taken(seconds)
1	1428
2	720
3	580

Table 1: No. of workers v/s Total Time taken(secs)









Above last fig shows the Loss v/s Epochs in Parameter Server

3 References

1. https://github.com/ischlag/distributed-tensorflow-example

2. http://mlwiki.org/index.php/One-vs-All_Classification