Document classification using Logistic Regression Assignment 2

Udit Gupta (14454) [1]

1. Implementation details

The steps followed in predicting the class of a document are as follows:

1. The training document has 48 classes so we train 48 logistic regression classifiers
2. Perform binary classification for a document such that the label is 1 for the class it belongs to and 0 for all other classes.
3. During test time calculate the probability of the document belonging to a particular class using the parameter for that class. Select the class which has the highest probability among all classes.
4. The tf-idf entries have been used as feature vectors.
5. Logistic regression with regularization is used to learn the parameters.

2. Logistic regression Algorithm

The loss function for Logistic regression (with regularization) is given by:

$$-\frac{1}{m}\sum_{i=1}^{m} y_i log(p_i) + (1-y_i) log(1-p_i) + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

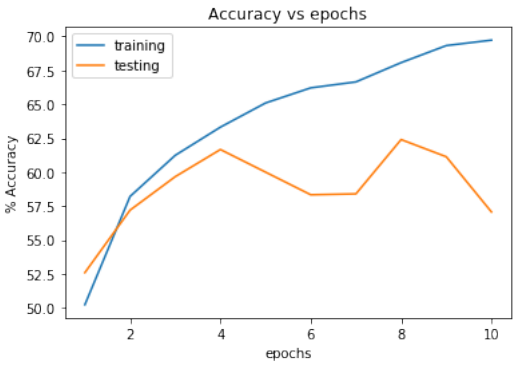
The update rule is given by:

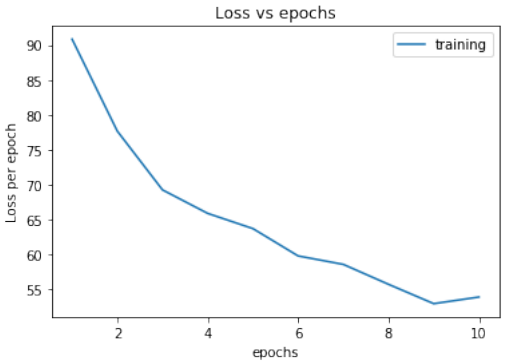
$$\theta_j -= \eta * (\frac{1}{m}\sum_{1}^{m}(p_i - y_i)x_j^i + \frac{\lambda}{m}\theta_j)$$

where η : learning rate

3. Results on local machine

3.1. Constant Learning rate





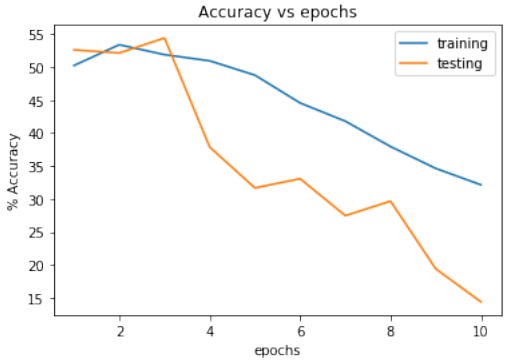
Training time per epoch: 296 seconds

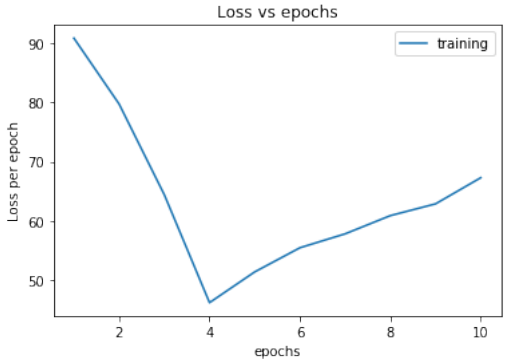
Testing time per epoch: 16 seconds

[1]Department of Computational and Data Sciences, Indian Institute of Science.

Assignment2 for the course Machine Learning with large datasets (DS-222)

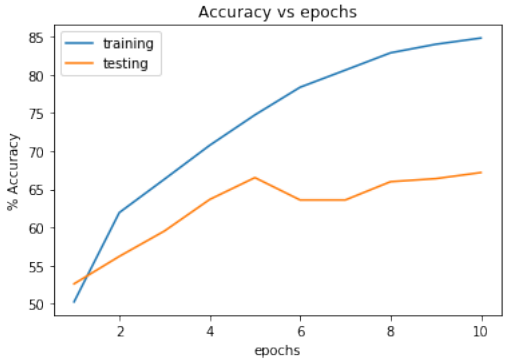
3.2. Increasing learning rate

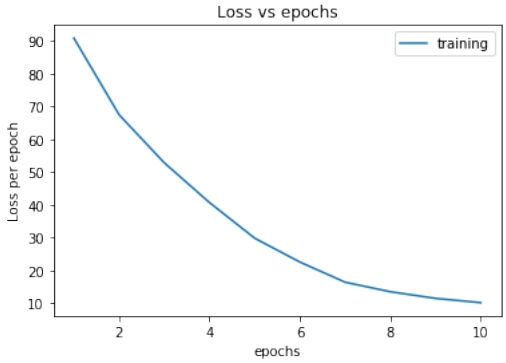




$$\eta = 0.001 * 2^{epoch}$$

3.3. Decreasing learning rate





$$\eta = 0.001/1.4^{epoch}$$

3.4. Observations

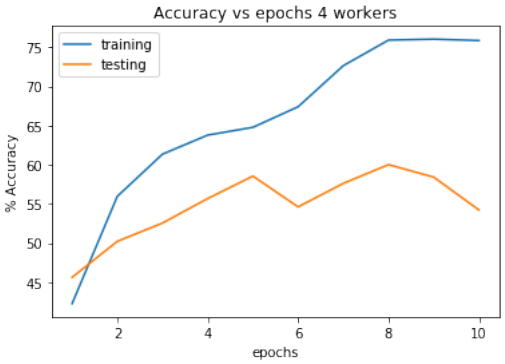
Decreasing the learning rate after each epoch gives best accuracy and also the loss gradually decreases. This approach is better than constant learning rate where it can miss the minimum. Increasing the learning rate is not recommended as the loss diverges after initial iterations.

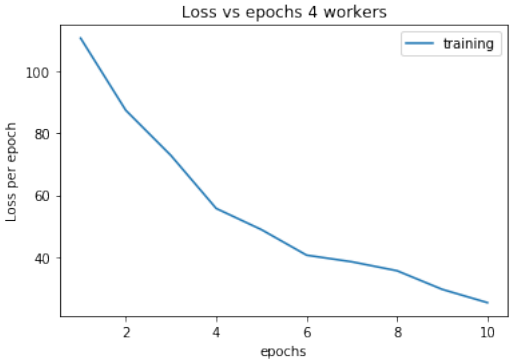
4. Parallel classifier

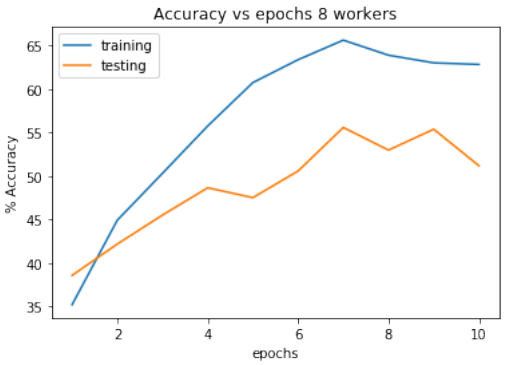
Using a constant learning rate of 0.001
Framework used: Distributed tensorflow with between-graph replication. In this approach, there is a separate client for each worker task. Each client builds a similar graph containing the parameters (pinned to a parameter server to map them to the same tasks); and a single copy of the compute-intensive part of the model, pinned to the local task in a worker.

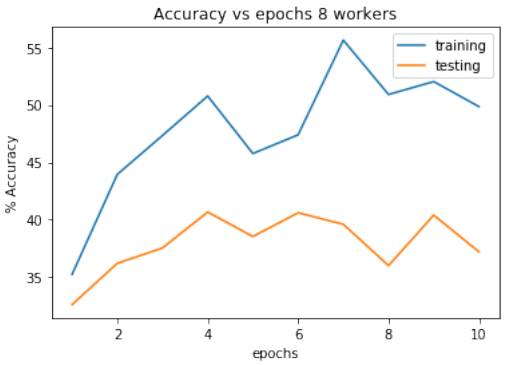
4.1. Bulk Synchronous Parallel

In this approach, all of the replicas read the same values for the current parameters, compute gradients in parallel, and then apply them together. For between-graph replication we use tf.train.SyncReplicasOptimizer.



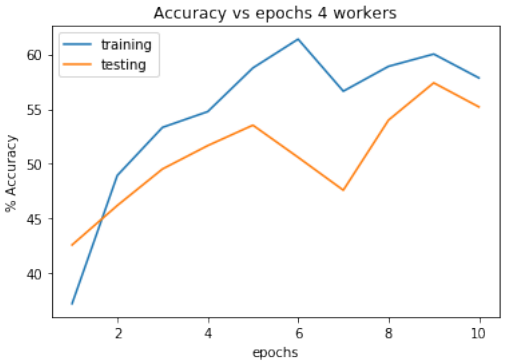






4.2. Asynchronous

Each replica of the graph has an independent training loop that executes without coordination.



4.3. Time comparison

Training time for 1 worker: 296 seconds.
Testing time for 1 worker : 16 seconds.

Type	4 workers		8 workers	
	BSP	Async	BSP	Async
Training	104	92	55	47
Testing	5.6	5	3	2.5

5. Observations

The BSP approach works better in terms of accuracy (on both train and test) but takes slightly longer to compute as compared to asynchronous approach.
There is a trade-off between accuracy and the speed of computation. If we want better accuracy and can compromise slightly on time then BSP should be preferred or if we want quicker speed of computation but can compromise slightly on accuracy then Asynchronous can be used.