Assignment 2-Logistic regression in the standalone and distributed setting using Parameter Server

Machine Learning with large Datasets-2017
Sonali Singh, Mtech-SSA, Sr no. 13253

1. L2-regularized Logistic Regression in the standalone setting

Code was written in python with and run on laptop with local memory. No GPU/cluster or any parallelism of code was used in this part.

1.1. Data Preparation

Punctuations: All english language standard punctuations and digits were removed .

Word filter on basis of length: All words with total frequency less than hundred were removed in training.

Lower case: All words were converted to lower case with blank space removals.

Note: Stopword removal was done using NLTK.

1.2. Training Algorithm

Training and Test features and labels: Bag of words were used as features . A total of 9617 size of vocabulary was prepared with training data. Feature size for each document is size of vocab. For each document if the word is present in vocab then we place a 1 at corresponding position or else zero. The label for each document is of size 50 (corresponding to 50 classes). If multiple labels are present we give uniform probability to each label such that sum of each label vector is 1.

Trainable parameters: Total trainable parameters are 9617*50 (=480850) The learning algorithm does regularized SGD:

Table 1. Training with different learning rate strategies.

STARTEGY	TRAIN-ACC	TEST-ACC	TRAIN-TIME(S)	TEST-TIME(S)
CONSTANT	75.36	71.27	2323.79	1.23
INCREASING	61.51	48.13	2078.15	1.09
DECREASING	75.47	73.20	2323.79	1.12

Algorithm::-
Initialize W matrix of size [9617,50]
Initialize b matrix of size [50]

for epoch in train epochs:
    for batch in train batch:
        loss=mean(cross-entopy(softmax(Wx+b),label)+0.01*L2 norm(W))
        W=W-learning rate*(grad W)

1.3. Training with different statergies on learning rate

Hyperparameters: Batch size **4000**
Training epochs **150**
Constant learning Rate **0.002**
Exponential decay rate of 0.96 with intial learning rate of **0.01**
Exponential increasing rate of 1.04 with intial learning rate of **0.001**

Please see **Figure 1** and **Figure 2** for variation of learning rates and loss for different learning stategies respectively. Also , **Table** 1 shows training accuracy , test accuracy , training time and testing time for all strategies.

. Correspondence to: Sonali Singh <sonalis772@gmail.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

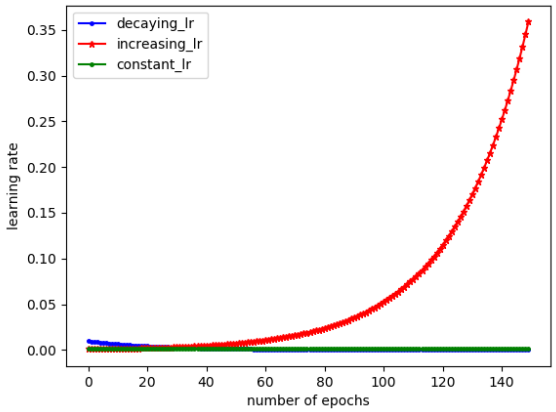

Figure 1. Plot indicating variation of learning rate with varying number of epochs for different learning strategies.

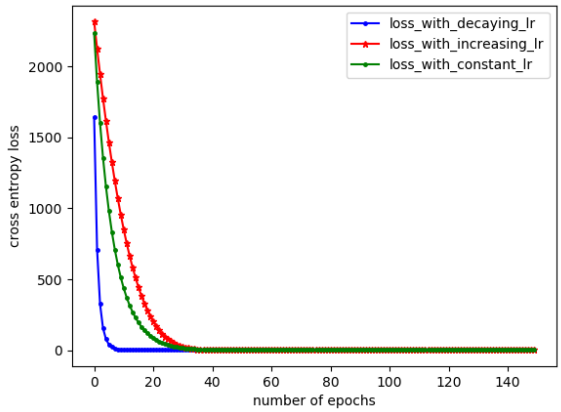

Figure 2. Plot indicating variation of total training loss with varying number of epochs for different learning strategies.

2. Distributed Logistic Regression using Parameter Server.

Framework: Code was written in python and run on Turing cluster's different worker nodes using Distributed Tensorflow framework. This framework was chosen as distributed graph learning can be done iteratively on Tensorflow with various requirements that is synchronous and asynchronously as was asked in assignment.

2.1. Data Preparation

Data preparation is same as mentioned in section 1.1 for local logistic regression.

Hyperparameters: Batch size **4000**
Training epochs **100**
Constant learning Rate **0.002**

The hyperparameters are same for all distributed SGD techniques.

2.2. Various training strategies and Results

For all strategies I have maintained two parameter servers and two worker nodes.
I have used Between-graph replication for distributed training. In this approach, there is a separate client for each /job:worker task, typically in the same process as the worker task. An example for the job distribution in tensorflow can be seen below:-

sonalis772-node1 python asynchronous2.py –job name="ps" –task-index=0
sonalis772-node2 python asynchronous2.py –job name="ps" –task-index=1
sonalis772-node3 python asynchronous2.py –job name="worker" –task-index=0
sonalis772-node3 python asynchronous2.py –job name="worker" –task-index=1

The cluster specification for distributed training was done in the following manner:-

parameter servers = ["10.24.1.201:2225", "10.24.1.202:2225"]
workers = ["10.24.1.203:2225", "10.24.1.204:2225"]
cluster = tf.train.ClusterSpec("ps":parameter servers, "worker":workers)

2.3. Results

The **Table** 2 shown below gives results for all training stategies mainly:-
BSP - Bulk Synchronous Parallel Setting.
Asynchronous - Asynchronous Parallel Setting.
SSP - Stale Synchronous Parallel Setting.
In table SSP-10 stands for SSP with staleness of 10 i.e only 10 older gradients can be used for updation (10 older steps as compared to global step). I have tested SSP for three staleness 5 , 10 and 20.

Table 2. Training and testing results with different parallelization strategies.

METHOD	TRN-ACC	TEST-ACC	TRN-TIME(S)	TEST-TIME(
BSP	75.71	71.72	2424.67	0.92
ASYNC	75.33	71.50	1866.06	0.95
SSP-5	74.73	69.09	1881.43	0.93
SSP-10	74.90	69.41	1899.09	0.94
SSP-20	75.41	69.39	1885.37	0.93

2.4. Plots

This section gives various comparitive plots for analysis of SSP with different staleness as well as combined assessment of BSP ,SSP-10 (staleness=10) and Asysnchronous SGD.

Figure 3 gives a plot indicating variation of test accuracy with varying number of epochs for different staleness in SSP. Staleness of 10 gives best test accuracy.

Figure 4 gives a plot indicating variation of cross entropy train loss with varying number of epochs for different staleness in SSP.Since the scaling is large so graphs seems to overlap. If looked closely we can see that with staleness of 20 loss is maximum and staleness of 5 gives minimum train loss.

Figure 5 gives a plot indicating variation of accuracy with varying number of epochs for BSP ,SSP-10 and Asynchronous SGD. Train accuracy in BSP is highest but Asynchronous and SSP-10 also comes closer to BSP with increasing epochs.

Figure 6 gives a plot indicating variation of cross entropy train loss with varying number of epochs for BSP ,SSP-10 and Asynchronous SGD. Train loss on Asynchronous SGD is highest and BSP SGD is minimum initially. But with increasing epochs SSP-10 and Asynchronous SGD catches up and loss on SSP-10 seems to be minimum by end.

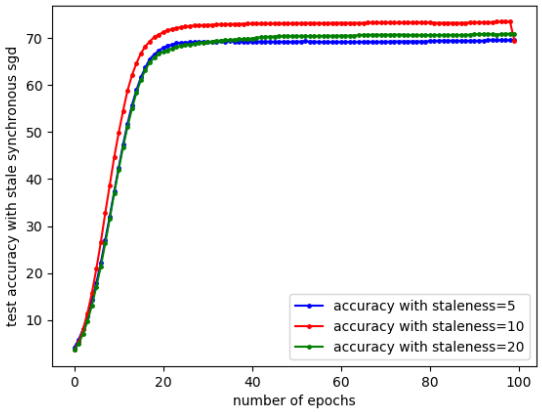

Figure 3. Plot indicating variation of test accuracy with varying number of epochs for different staleness in SSP.

3. Analysis and Conclusion

It can be seen clearly from **Figure 5** that although BSP gives better accuracy but SSP and Asynchronous takes lesser time with minor decrease in accuracy. SSP with staleness of 10 eventually becomes as good as BSP.

The time improvement in Asynchronous SGD over Synchronous SGD is approximately **25 %**. The performance reduction between SSP-10 and BSP is **3.23 %** which is minor if we compare to time consumption improvement which is **25 %**.

If we compare local and distributed setting we see a minor performance drop from **73.20 %** test accuracy on standalone system with decreasing learning rate to **70.0 %** on distributed systems. However train time tends to decrease a lot in distributed setting.

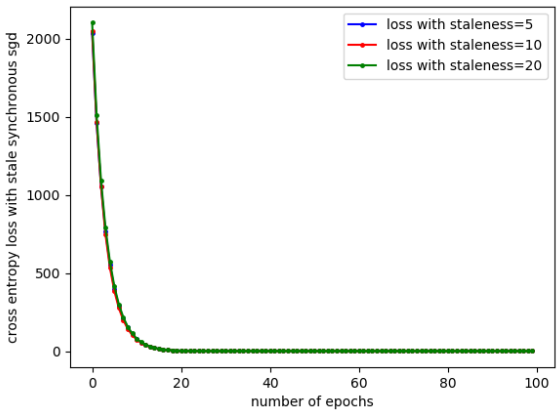

Figure 4. Plot indicating variation of cross entropy train loss with varying number of epochs for different staleness in SSP.

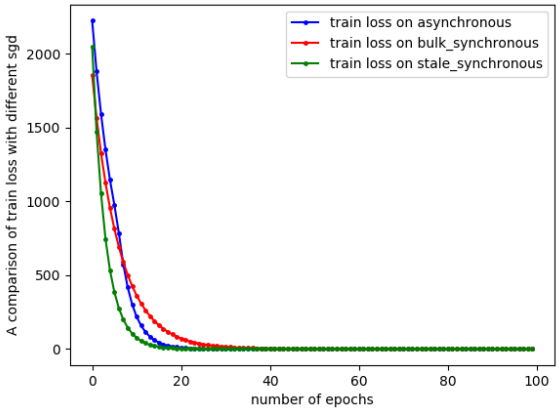

Figure 6. Plot indicating variation of train loss with varying number of epochs for BSP ,SSP-10 and Asynchronous.

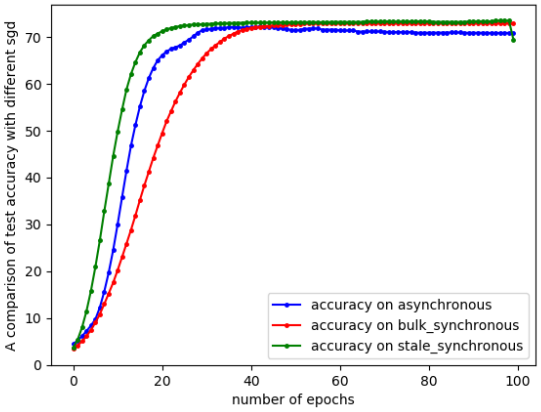

Figure 5. Plot indicating variation of accuracy with varying number of epochs for BSP ,SSP-10 and Asynchronous.