Logistic Regression implementation Local And Distributed
Assingment-02 DS-222
(Machine Learning with Large Dataset)

Chaikesh Chouragade
(SR No. 14358)

Abstract

Implemented Logistic Regression Classifier clasifier in local and distributed setting.Compared the accuracy and runtimes for training and testing.**TensorFlow** framework was used for distributed setting.

1. Local L2 Regularized Logistic Regression

Used python programming language and turing cluster ,single machine for local implementation.

1.1. DATA Preprocessing

Lower Case: All words converted to words with lowercase letters.
Length: All words with length less than 3 were removed.
Punctuation : All punctuation mark and alphanumerics were removed.
Stop Word : stopwords were removed using NLTK .
Thresholding : All words having word count less than 100 and greater than 8000 were excludes from the final vocabulary .

1.2. Training

Featurization: Bag of Word Feature is used with the above mentioned data preprocessing steps.Feature dimension is equal to vocab size obtained on training dataset.If particular word in vocab is present in particular document ,then the corresponding feature value to that word is set to one ,otherwise zero. Label Dimension for each document is 50.For multilabel document equal probability is assinged to each label.
Parameter: Total trainable parameter in case of logistic regression is given by vocab*#class=(8690*50)=434500

. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Table 1. Accuracy and Time Taken with three setting of learning rate

LEARNING RATE	TRAINING ACCURACY	TEST ACCURACY	TRAIN TIME	TEST TIME
CONSTANT	84.2%	70.70%	4400S	1.86S
DECAY	82.7%	71.0%	4300S	2.03S
INCRESING	81.1%	71.20%	2800S	1.98S

1.3. Experiments

Model is trained and tested under three different settings for Learning Rate.First Model is trained with constant learning rate.Second Model is trained with exponentially decaying learning rate.Thirdly model is trained with exponentially increasing learning rate.The associated hyperparameter with the model are mentioned below:

Hyperparameter:
Batch size :4000
Training epochs :100
L2 Regularization :0.001
Constant learning Rate : 0.004
Decay Rate : 0.94
Increase Rate : 1.06

1.4. Results

Figure-1 shows the convergence under three different settings of learning rate with number of epochs.
Figure-2 shows the convergence under two different settings of learning rate with number of epochs.
Figure-3 shows the test accuracy under three different settings of learning rate with number of epochs.
Table-1 shows the Train ,Test Accuracy and toatsl time taken under three different settings of learning rate.

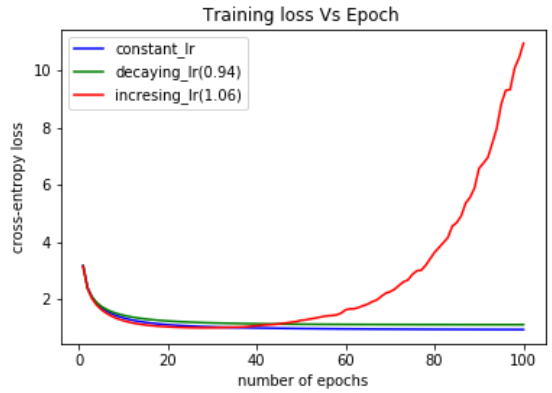

Figure 1. Number of epoch Vs loss for 3 different setting of learning rate

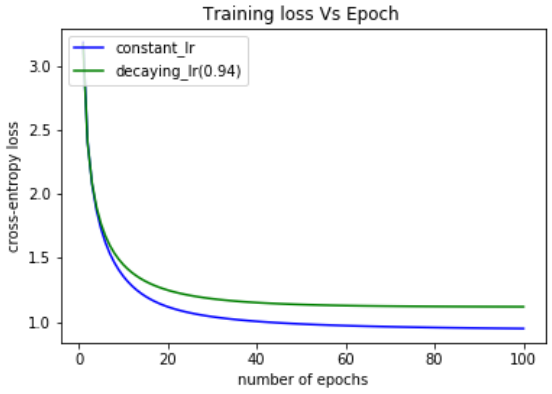

Figure 2. Number of epoch Vs loss for 2 different setting of learning rate

2. Distributed Logistic Regression

Framework:distributed Tensorflow framework is used.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.
Used LEAP LAB cluster.

2.1. DATA Preprocessing

Same as Done for Logistic Regression.

2.2. Training

Featurization: Same as used for Local Logistic Regression.
Parameter: Total trainable parameter in case of logistic regression is given by vocab*#class=(8690*50)=434500

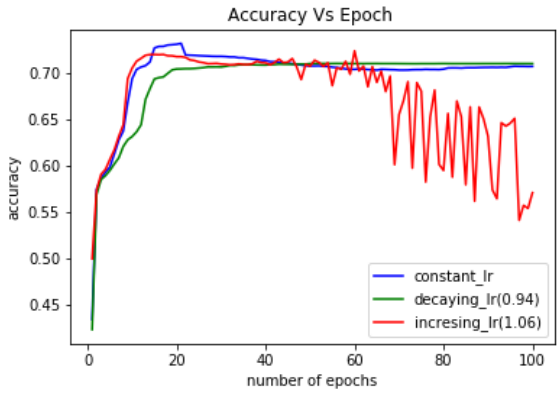

Figure 3. Number of epoch Vs test accuracy for different setting of learning rate

Hyperparameter:
Batch size :4000
Training epochs :50
L2 Regularization :0.001
Constant learning Rate : 0.004

2.3. Experiments

2.3.1. Asynchronous

In this setting each worker has entire part of graph,each worker can update the weights without any synchronization.
Parameter server=1,workers=2

2.3.2. Bulk Synchronous

In this setting each worker has entire part of graph , the gradient updates after each step is aggregated and again given to the worker.
Parameter server=1,workers=2

2.3.3. Stale Synchronous

In this setting each worker has entire part of graph,each worker can update weights provided it does not lag behind than other worker by some fixed number of updates called staleness factor.
Parameter server=1,workers=2

2.4. Results

Table-2 shows the Train ,Test Accuracy and time taken under three different settings of parallelization.
Figure-4 shows the convergence under two different values of staleness factor.
Figure-5 shows the convergence under three different

settings of parallelization.

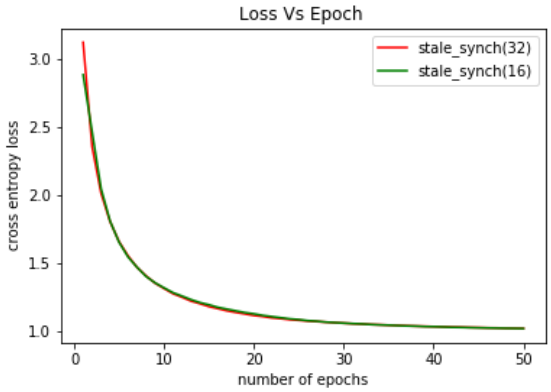

Figure 4. Number of epoch Vs Loss for different staleness factor.

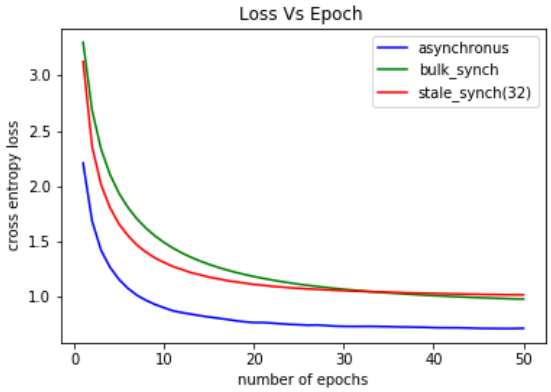

Figure 5. Number of epoch Vs Loss for 3 different settings of parallelization .

2.5. Conclusion

Local Implementation

- The important observation by experimenting with learning rate in Local Implementation is that the with exponentially increasing learning rate convergence is faster initially but then diverges abruptly.

- There is little difference between convergence rate with constant and exponentially decaying learning rate but both achieves better convergence than the exponentially increasing rate.

Table 2. Accuracy and Time Taken with three setting of parallelization

Parallelization Setting	Training Accuracy	Test Accuracy	Train Time	Test Time
Asynch	84.7%	70.3%	1050s	1.07s
Bulk Synch	84.2%	71.5%	1257s	1.03s
Stale Synch	84.3%	71.1%	980s	1.04s

- The accuracy in all three settings is nearly same if we use early stopping in case of exponentially increasing learning rate.

Distributed Implementation (Parameter Server)

- Among all the parallelization schemes asynchronous converges fastest,as it makes more frequent updates than the other two schemes.

- Convergence rate of Bulk Synchronous is slow ,which is expected as different worker have to wait if one of the worker is slow.as we need to aggregate the updates before we can move to next step.

- In case of Stale synchronous we reject the updates of worker if it lags behind by fixed staleness factor,resulting in fewer updates than asynchronous mode.Also there is some overhead task.Thus resulting in slower convergence than asynchronous mode.

- The Accuracy in all the Schemes is nearly same,which is expected as the algorithm is same only the implementation is different.

- More updates as in case of asynchronous doesn't necessarily imply better convergence as the updates may be noisy.Thus stale synchronous scheme should be prefered.