# Regularized Logisitc Regression in Local and Distributed Mode

**Karan Malhotra, M.Tech(SE), Sr No-14532**

## 1. Introduction

Logistic Regression is a discriminative classifier which models the posterior probability $p(y|x)$ using the logit function(sigmoid) in binary case and softmax function in multi class case .The following equation describes the cost function which is being optimized in multi-class L2 regularized logisitc regression:

$$W = argmin_W(-\sum_i \sum_k y_k^i log(\tilde{y^i}_k) + \lambda||W||^2)$$

$y^i$ is true probability distribution over K classes for $ith$ example.
$\tilde{y^i}$ is predicted probability distribution over K classes for $ith$ example.
$\lambda$ is regularization constant.

## 2. Dataset

The dataset provided is multi label document classification dataset and its details are as follows:
**1.** Training instances: 214998
**2.** Validation instances: 61497
**3.** Testing instances: 29997
**4.** Total Classes: 50

## 3. Logistic Regression implementation in local

In this section the implementation of logistic regression in local mode has been discussed . In the local setting the full training of the model is been performed in one machine only, but no parallelization was involved .

### 3.1. Preprocessing

**1.** All the punctuation and single length characters were removed while preprocessing.
**2.** All the words were then converted to lower case .
**3.** Porting or Stemming of the provided data sets were not performed.

### 3.2. Training procedure

**1.** The words with counts greater than 10,000 or lower than 100 were removed from the vocabulary and then all the documents were converted into bag of words feature representation .
**2.** For every document the y vector was created with distributing 1 uniformly among the classes it belong to and assigning 0 for other classes .
**3.** The Cross entropy loss with L2 regularization was used for training with parameter updates done using stochastic gradient descent algorithm with 3 types of learning rate strategies (increase,decrease,constant) .

### 3.3. Parameters

**Total Trainable Parameters:** The trainable parameters are = Vocab-size X No-of-classes=10749 X 50=537450.
**HyperParameters:** The following are the hyper parameters tuned on validation set.
**1.** Batch Size=7000.
**2.** Learning rate with initial value as 0.005 and 1.04 of exponential increase or 0.96 of exponential decrease.
**3.** L2 Regularization constant($\lambda$) = 0.001
**4.** Number of epochs =100
The below Figures shows the training performance of the three learning rate strategies for Stochastic Gradient descent. It can be observed that with decreasing learning rate the training loss was decreasing slower than as in constant but there were no transients but the increasing learning rate strategy caused a divergence in training loss after certain epochs.

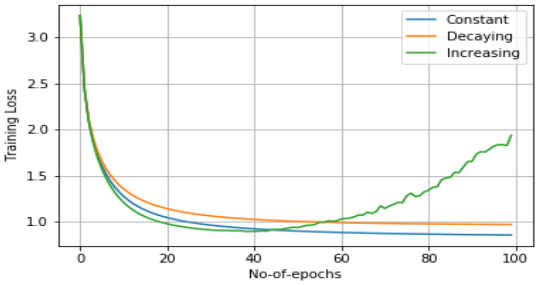


*Figure 1.* Training Loss vs No of epochs for different learning rate strategies

### 3.4. Results

The Following table summary of the results corresponding to local model implementation:

*Table 1.* Results for Local Mode

| Learning | Train Acc | Test Acc | Train time | Test time |
|---|---|---|---|---|
| **Constant** | 87.6% | 73.4% | 2300s | 1.02s |
| **Decreasing** | 86.4% | 73.6% | 2208s | 1.03s |
| **Increasing** | 79.3% | 72.1% | 2306s | 1.06s |

## 4. Distributed Logistic Regression using Tensorflow Parameter Server framework

Tensorflow Parameter server framework has been chosen for the given task as the framework provides very abstract procedure to create a cluster of tensorflow servers and then distribute the computation graphs among the several servers. It also provides various pre built classes on top of which asynchronous and synchronous mode of training can be implemented without much worrying for server communication and data distribution.
**Cluster Details(LEAP LAB,EE Cluster(GPU enabled) was used for model implementation):**
Parameter server= 10.1.1.254:2225
Worker0 = 10.1.1.253:2223
Worker1 = 10.1.1.252:2224
The **Preprocessing** step is same as done in local implementation .

### 4.1. Hyperparameters

**1.** Batch Size=7000.
**2.** Learning rate with constant value as 0.005.(As the decaying learning rate didn't provide much of performance improvement in local mode. )
**3.** L2 Regularization constant($\lambda$) = 0.001
**4.** Number of epochs =100

### 4.2. Bulk Synchronous Parallel

**Cluster Specs:** 1 parameter server and 2 workers.
The Between Graph Replication approach was utilized for training in which every worker has a copy of the computation graph.The Gradients were aggregated for both the workers and then after each step they were applied on the parameters by the parameter server. The following Figure shows the variation of the training loss with number of epochs.

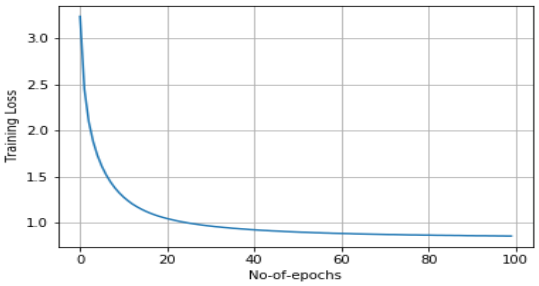


*Figure 2.* Training Loss vs No of epochs.

**Training Accuracy:**87.6 %
**Test Accuracy:** 72.8%
**Training time:** 2104 secs
**Test time:** 0.830 secs
The training time and test time is improved compared to local approach but test accuracy got degraded by around 1%.

### 4.3. Asynchronous Parallel

**Cluster Specs: Case A:** 1 parameter server and 2 workers.
**Case B:** 1 parameter server and 3 workers.
The Between Graph Replication approach was utilized for training in which every worker has a copy of the computation graph.The **Gradients were applied on the parameters as soon as they were computed** by the worker machines independently without any synchronization.
The following Figure shows the variation of the training loss with number of epochs as number of workers are increased .

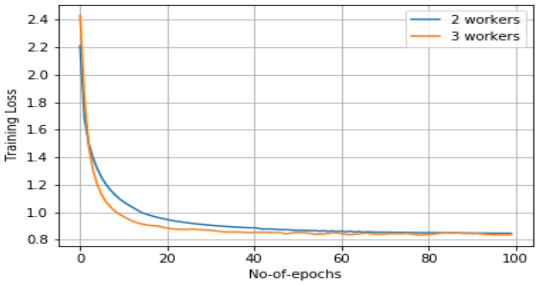


*Figure 3.* Training Loss vs No of epochs for different number of workers

**Case A: 2 Workers**
**Training Accuracy:** 87.5%
**Test Accuracy:** 72.9%
**Training time:** 1951 sec
**Test time:** 0.717 sec

**Case B: 3 Workers**
**Training Accuracy:** 87.6%
**Test Accuracy:** 72.9%
**Training time:** 1836 sec
**Test time:** 0.636 sec
Due to computation constraints , the training was not parallelized beyond 3 workers but the time improvements were being observed when number of workers were increased from 2 to 3.

### 4.4. Stale Synchronous Parallel

**Cluster Specs:**1 parameter server and 2 workers.
The Between Graph Replication approach was utilized for training in which every worker has a copy of the computation graph.The Gradients were applied on the parameters as soon as they were computed by the worker machines but if any of the two workers go faster than the **other by a certain number of steps (stale factor)** then its gradients are dropped . Stale factor is basically difference between the number of updates made by the faster worker than the slower worker.
The following Figure shows the variation of the training loss with different number of stale parameter value used .

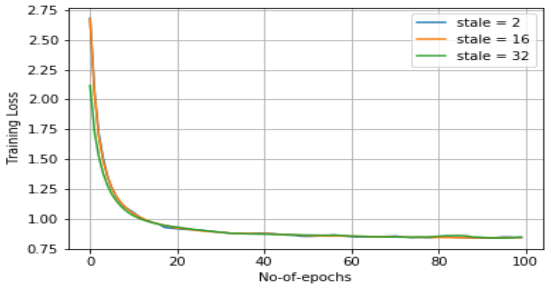


*Figure 4.* Training Loss vs No of epochs for different number of stale value.

**Stale=2**
**Training Accuracy:** 86.5%
**Test Accuracy:** 73.2 %
**Training time:** 1917 secs
**Test time:** 0.810secs
**Stale=16**
**Training Accuracy:** 87%
**Test Accuracy:** 73.1 %
**Training time:** 1957 secs
**Test time:** 0.816secs
**Stale=32**
**Training Accuracy:** 86.8%
**Test Accuracy:** 73.2 %
**Training time:** 1955 secs
**Test time:** 0.810secs

### 4.5. Comparison of different distribution setting

The Following figure shows the comparison of different distributed learning settings with stale taken 32 steps for stale synchronous and 2 workers taken for all cases .

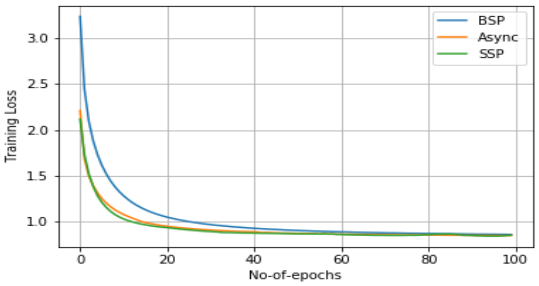


*Figure 5.* Training Loss vs No of epochs for different distribution setting

## 5. Observations & Conclusions

**1.** For the **Local implementation** it was observed that the increasing learning rate strategy provided faster training loss convergence but after certain epochs the loss started diverging .Moreover, as the initial learning rate was chosen to be very low in itself the decreasing strategy does not showed much difference in performance as compared to constant learning rate .

**2.** In **Asynchronous parallel** strategy it was observed that increasing number of workers from 2 to 3 gave faster convergence of training loss and train time was reduced from **1951 secs to 1836 secs** but the test accuracy got reduced as the updates get more noisy due to difference in update speed of workers.

**3.** For **Stale Synchronous Parallel** when the stale factor was increased from 2 to 32 steps, the training loss convergence become faster but not with much difference and it can be explained as the number of workers used were only two therefore much of noise is not induced due to speed differences between workers.

**4.** Among the three distributed approach it is being observed that the **SSP** approach gives the faster convergence and the best **test accuracy of 73.2%** for same number of workers.

**5.** When SSP is compared to local implementation it is seen that the best test accuracy get reduced to **73.2% from 73.4%** but the training and testing becomes much

faster.

**Local Train time**: 2300 secs
**SSP Train time**: 1917 secs
**Local Test time**: 1.02 secs
**SSP Test time**: 0.810 secs
**The local performance is being compared to SSP as best test accuracy is being achieved by SSP approach in Distributed mode**.