DS222: MLLD 2018
Assignment 2 : Logistic regression with L2 Regularization

Sourabh Balgi
Sr. No. - 14318
EE Department
M. Tech. (Systems Engineering)
sourabhbalgi@gmail.com

Abstract

Logistic regression is a widely used statistical learning model which uses logistic function to model a binary dependent variable. In this assignment, we mainly focus on implementing a multinominal logistic regression for classification of documents as the task at hand has 50 different classes. Multinominal logistic regression actually generalizes logistic regression to multiclass classification problem. Since the DBpedia dataset we are working on is a very huge dataset, we explore and compare different distributed learning versions of logistic regression to in-memory local implementation.

0.1 Dataset

We use DBPedia 2015 dataset which consists of 50 classes based on the labels obtained from mapping based type of the documents.

The columns are separated by space-tab and the list of labels are comma separated. Each line is a unique document with labels and document data.

1 Logistic regression with L2 regularization

Let (x_i, y_i) be a single instance of given data, where (x_i, y_i) denote the feature vector and it's corresponding label for the considered data instance. let $haty_{ik}$ be the predicted value of the label for the given data, where

$$\hat{y_{ik}} = softmax(W_k * x_i + b_k)$$

k represents the class index and i is represents i^{th} data instance.

The loss function used for finding the gradients and updating the weights W of the model is **Cross-entropy loss**. The **Cross-entropy loss** can be formulated as below

$$J_{y_{ik}}(\hat{y_{ik}}) := -\sum_k y_{ik} \log(\hat{y_{ik}})$$

where y_{ik} is the true label of class k, $haty_{ik}$ is the predicted label of class k.
To the above loss function, we add a L2 regularizer term $L2\_norm(W)$ to prevent over-fitting.
Final loss function is of the form,

$$J_{y_{ik}}(\hat{y_{ik}}) := \\ -\sum_k y_{ik} \log(\hat{y_{ik}}) + \lambda * L2\_norm(W)$$

where λ is a hyperparameter. For our observations, λ was tuned to 0.01 on the dev set.

Finally we find the gradients with respect to W for the above function to use gradient update rule to train our model.

$$W := (1 - lr * \lambda)W - (lr * \triangledown W)$$

2 Pre-processing of the dataset

- All the punctuation, digits and special characters were removed from the corpus.
- Since the vocabulary size observed after the step 1 was very large, The English stopwords from NLTK were removed from the corpus.
- Since the vocabulary size was still large, words with a frequency less than 100 in the entire corpus were dropped.
- After all the pre-processing, a vocabulary size of 10,428 was obtained.

3 Implementation approach

3.1 Local logistic regression with L2 regularization

The local in-memory model was trained using the below gradient update rule.

$$W := (1 - lr * \lambda)W - (lr * \triangledown W)$$

3.1.1 Configurations of learning rate

Different configurations of learning rates were tried to observe the corresponding convergence behaviors.

1. **Constant** : The learning rate was set to constant i.e. $lr = 0.02$ and the model was trained with above SGD rule.
2. **Decreasing** : The initial learning rate was set to constant i.e. $lr = 0.02$ and an exponential decay with a decay factor of 0.95 was used to train the model using SGD rule.
3. **Increasing** : The initial learning rate was set to constant i.e. $lr = 0.02$ and an exponential decay with a decay factor of 1.05 was used to train the model using SGD rule.

3.1.2 Observations

1. **Constant** : The convergence is moderate number of epochs. Moderate oscillations in the loss as the number of epochs progress.
2. **Decreasing** : The convergence is the slowest compared to all the 3 configurations. The oscillations in the loss is very less as the number of epochs progress.
3. **Increasing** : The convergence is the fastest among all the configurations. However more oscillations in the loss as the number of epochs progress.

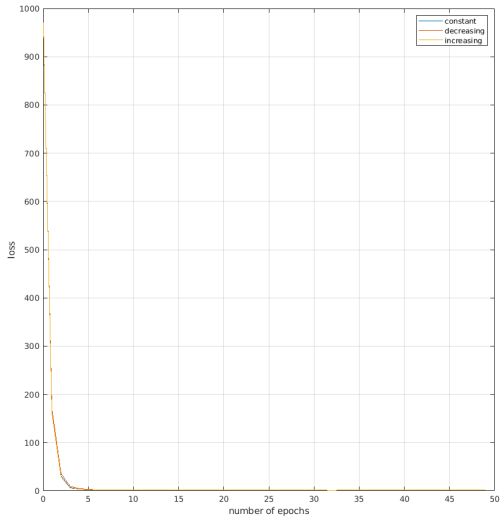

Figure 1: **Zoomed out version of the plot of loss vs epochs**

4 Distributed Tensorflow implementation

Implementation of BSP,SSP and ASP Stochastic gradient descent and observation of results.

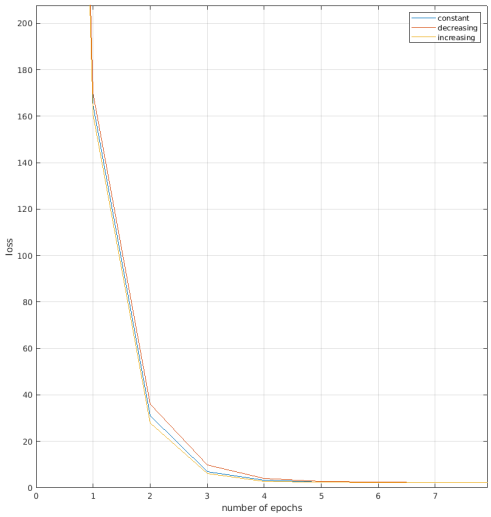

Figure 2: **Zoomed in version of the plot of loss vs epochs to observe the faster convergence**

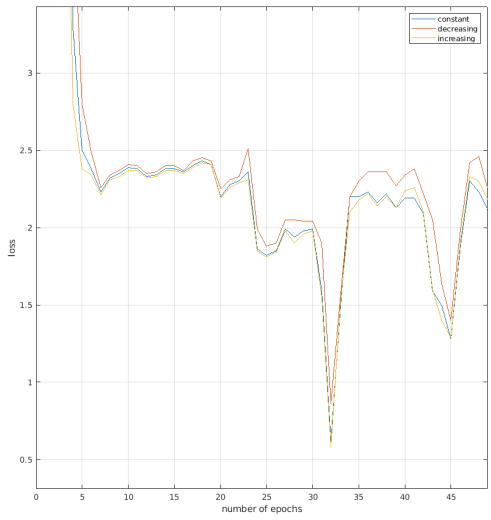

Figure 3: **Zoomed in version of the plot of loss vs epochs to observe the oscillations in loss**

Table 1: **Local in-memory SGD Accuracy**

Learning rate	**Train %**	**Test %**	**Runtime %**
Constant	52.61	58.08	448.9s
Decreasing	58.82	59.06	541.90s
Increasing	53.38	57.93	525.1s

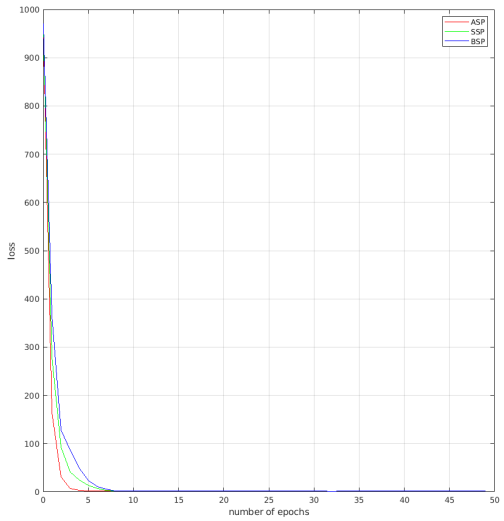

Figure 4: **Observation of convergence of different modes (BSP, SSP, ASP) of PS update methods.**

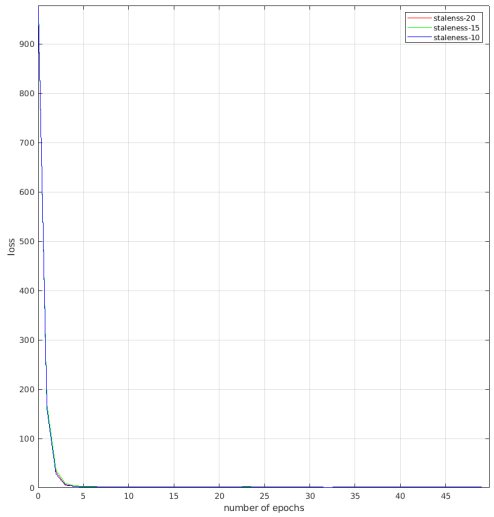

Figure 5: **Comparison of convergence of different staleness in SSP**

4.1 BSP-SGD logistic regression with L2 regularization

Distributed Tensorflow logistic regression using parameter server and bulk synchronous mode. In BSP mode, A given worker waits for all the other works to finish the computations and update the PS synchronously. Since the workers wait, the convergence is slowest of all the distributed modes.

4.2 SSP-SGD logistic regression with L2 regularization

4.2.1 SSP-SGD mode

Distributed Tensorflow logistic regression using parameter server and stale synchronous mode. In SSP mode, A given worker uses not the recent weights of the PS. Since the workers use the stale parameters, the convergence is slower as compared to ASP mode but faster than BSP mode.

4.2.2 SSP-SGD workers mode

The staleness in SSP was varied (10, 15, 20) and the convergence was noted down with the same hyperparameters.

4.3 ASP-SGD logistic regression with L2 regularization

4.3.1 ASP-SGD mode

Distributed Tensorflow logistic regression using parameter server and asynchronous mode. In ASP mode, A given worker updates the weights of the PS asynchronously. Since the workers do not wait for other workers, the convergence is the fastest among all the modes.

4.3.2 ASP-SGD workers mode

The number of workers in ASP was varied and the convergence was noted down with the same hyperparameters.

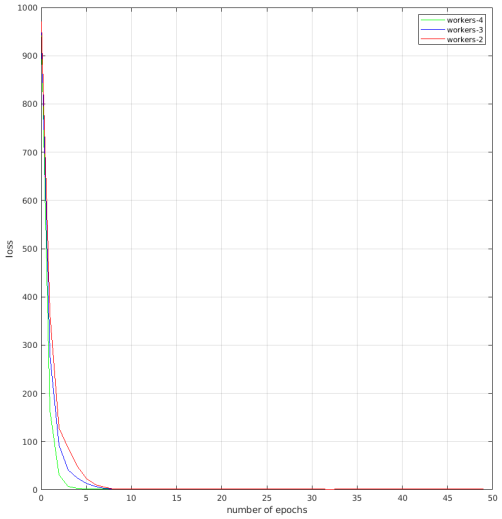

Figure 6: **Comparison of convergence of different number of workers in ASP**

5 Project Github link

DS222_assignment2

References

[1] Probabilistic Synchronous Parallel, Liang Wang and Ben Catterall and Richard Mortier, CoRR, 2017.