Assignment 2 -MLLD

Abhimanyu Vatta [1]

1. Problem Statement

Given DBPedia training dataset, it is required to implement and evaluate a L2-regularized logistic regression in the standalone and distributed setting using Parameter Server.

2. Pre-Processing

Preprocessing on Dataset

1. Documents were tokenized into seperate words.
2. Words were lowercased to avoid repetation due to case sensitivity.
3. Words were stripped of any punctuation marks to avoid repetation due to occurence of a word with punctuation mark.
4. Stopwords as given in NLTK library, with some additions were removed as they in general would not help determining any context attached to a document (label in this case).

3. Model Description

A simple and effective strategy employing Bag of words as features is used. Training data after pre-processing as discussed above populates a vocabulary of 9617 words. Each line is represented as a 2-dimensional matrix of size $lines * vocabulary$ of one hot like representation where every word in vocabulary occuring in a line is represented by 1 at corresponding position in matrix column. The label for each document is of size 50 (corresponding to 50 classes). If multiple labels are present, uniform probability is given to each label such that sum of each label vector is 1. Total trainable parameters = 480850(9617*50).

3.1. Local Logistic Regression

The learning algorithm performs regularized SGD wwith loss calculation as $loss = mean(cross - entopy(softmax(Wx+b), label) + 0.01 * L2norm(W))$.

[1]Indian Institute of Science, Bangalore. Correspondence to: Abhimanyu Vatta <v.manyu@gmail.com>.

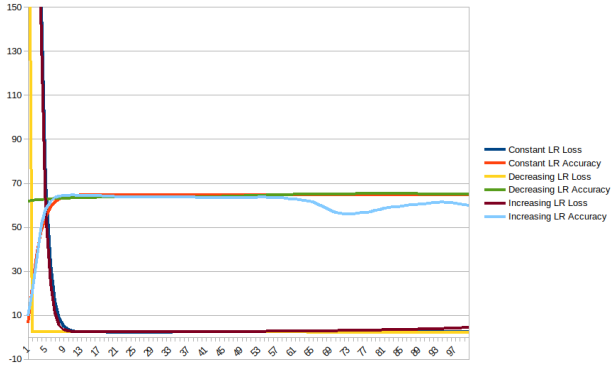

Figure 1. Loss and Accuracy for Local Logistic Regression with different Learning Rate Types

1. Hyper-Parameters

	Epoch	Batch Size	Initial LR	Dec/Inc
Const LR	100	2048	0.002	-
Dec LR	100	2048	.05	0.96
Inc LR	100	2048	0.002	1.04

2. Results

	Trg Accuracy	Test Accuracy	Trg Time(s)
Const LR	73.77	64.93	17755.33s
Dec LR	75.32	65.08	19144.55
Inc LR	57.78	59.81	19001.15

3. Graphs - Refer fig 1 to compare Loss and Accuracy for different models. Increasing Learning Rate shows overfitting towards the end and accuracy decreases as a result. Fig 2 shows the exponential variation of learning rates with epoch in increasing and decreasing learning rate models.

3.2. Distributed Logistic Regression

Framework chosen for carrying out the task is on Tensorflow. The framework is chosen due to previous experience and familiarity with tensorflow environvironment. Also, distributed graph learning can be done iteratively on Tensorflow with various requirements that is synchronous and asynchronously, and keeping in view the final project which will again be implemented on same environment. All Models make use of one parameter server and two

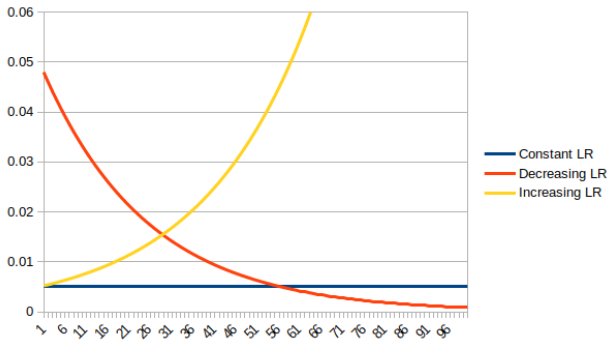

Figure 2. Varying Learning Rates of different model types in Local Logistic Regression

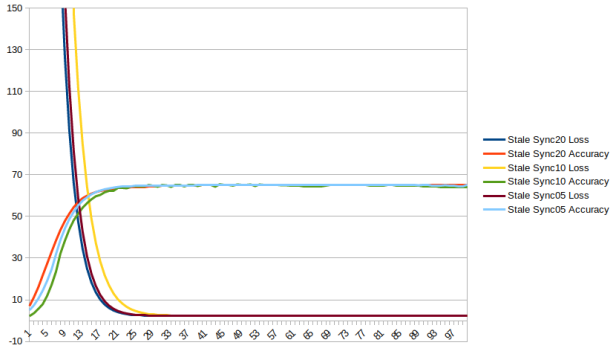

Figure 3. Stale Synchronous Model with different value for staleness parameter

worker nodes, although number worker nodes can be increased. Stale Synchronous model has been run with three difeerent values of staleness as 5, 10 and 20.

1. Hyper-Parameters - Batch size = 2048, Learning Rate = 0.002 and Epochs = 100.

2. Results

	Trg Acc	Test Acc	Trg Time(s)
Asynchronous	75.23	64.44	10693.24
Bulk Sync	75.65	65.09	12273.41
Stale Sync(20)	75.09	64.75	10818.47
Stale Sync(10)	75.2	64.00	11187.46
Stale Sync(5)	74.74	65.2	10838.90

3. Graphs - Fig 3 shows corelation between various runs of Stale Synchronous model with staleness parameter changed to different values: 5, 10 and 20. Fig 4 shows loss and accuracy comparisons between different distributed models, namely: Asynchronous, Bulk Synchronous and Stale Synchronous.

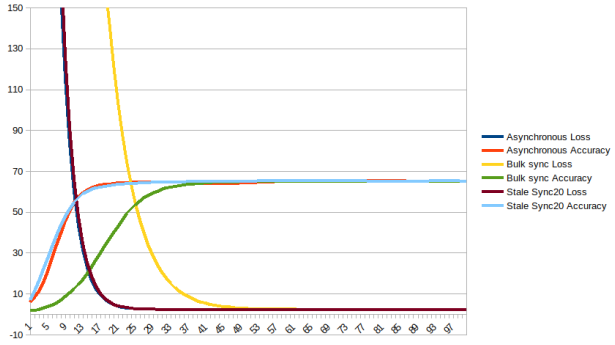

Figure 4. Comparison between different Distributed Logistic Regression Models

4. Usage

4.1. Local Logistic Regression

1. Path - abhimanyuv@turing.cds.iisc.ac.in:/home/ abhimanyuv/assignment_2/part1

2. Prepare Data - python data.py

3. Constant Learning Rate(LR) - python lr_const.py

4. Decreasing LR - python lr_dec.py

5. Increasing LR - python lr_inc.py

4.2. Distributed Logistic Regression

Data files used are loasded from path of part1 as given above. Cluster IP addresses of parameter server and workers are coded inside *.py files and can be changed as required. Thereafter following commands need to be run on respective cluster IP addresses of server and workers. Example given is for bulk synchronous model program for a single parameter server and two worker nodes. However, more worker nodes may also be defined.

1. Path - abhimanyuv@turing.cds.iisc.ac.in:/home/ abhimanyuv/assignment_2/part2

2. python bulksync.py –job_name=”ps” –task_index=0

3. python bulksync.py –job_name=”worker” –task_index=0 (on first worker IP as given inside *.py file)

4. python bulksync.py –job_name=”worker” –task_index=1 (on second worker IP as given inside *.py file)

5. Conclusion

1. Decreasing Learning Rate in local implementation gives best results with sharp decline in loss and avoiding overfitting. A similar decreasing LR can also be implemented in distributed setting, this helps in avoiding missing the global minima as LR decreases with increase in epochs and helps reduce large variation in gradients. However it is slowest among the three.

2. In Stale Synchronous Model, performance dips for staleness 10 in all aspects. Staleness of 5 and 20 perform equivalently, however a closer inspection reveals staleness of 20 performing marginally better.

3. Comparing three distributed models, it is observed that final accuracies achived are similar, however, bulk synchronous proves to be the slowest.