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

Souvik Karmakar[1]

Algorithm 1 L2 regularized logistic regression

Input: features x_i, labels y_i
Initalize parameters W size [9655, 50], b size [50]
for $i = 1$ **to** $totalepochs$ **do**
  for $batch$ **in** $batches$ **do**
    loss = **cross-entropy(softmax(**$Wx + b$**))** + 0.01∗**L2(norm(**W**))**
    $W = W - learningrate * grad(loss)$
  end for
end for

1. Preprocessing

- All digits and punctuation were removed from text and text was converted to lowercase
- Words which had a frequency less than 100 were removed from corpus
- 'english' stopwords from NLTK library were removed

Using the above preprocessing techniques we obtained a vocab of size 9655. We use Bag of Words to generate features for each document. Our feature dimension is 9655, and we place 1 for a word if it is present in a document. We have used label dimension of size 50. If a document belongs to multiple classes, we give equal probability for all the classes, otherwise 1.

2. L2 regularized logistic regression standalone

The algorithm used is mentioned in algorithm 1. We use mini-batch SGD for gradient descent. The standalone version was trained in 3 settings as follows:

- Constant learning rate of 0.001 was used
- Exponentially decaying learning rate, with initial learning rate set at 0.01 decaying at a rate of 0.95 after each epoch
- Exponentially decaying learning rate, with initial learning rate set at 0.001 increasing at a rate of 1.05 after each epoch

Trainable Parameters: $9655 * 50 + 50$,
Batch Size: 4096

2.1. Observations

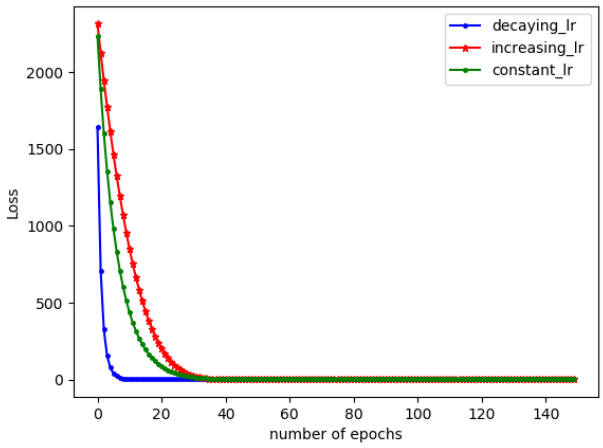

Figure 1. Comparison between various modes of learning

In Figure 1 we observe the variation of training loss with different learning rates. As expected, exponentially decaying algorithm converges first, followed by constant learning rate, followed by exponentially increasing learning rate. The final test accuracy at each setting is given in Table 1

3. L2 Regularized Logistic Regression in Distributed setting

We use Distributed Tensorflow for training and developing models in this section. We use Tensorflow's Monitored

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. 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. Comparison between various methods in standalone setting

METHOD	TRAIN ACC	TEST ACC	TRAIN TIME
FIXED LR	75.24	71.34	3453.2S
DECREASING LR	75.96	72.45	3310.8S
INCREASING LR	60.56	50.77	3356.3S

Training session for training on various worker nodes and parameter servers.

Hyperparameters:
Trainable Parameters: $9655 * 50 + 50$,
Batch Size: 4096,
Learning Rate: 0.001

We train the model in three distributed settings:

- **Asynchronous** with 2 parameter servers, and 2,3,4 worker nodes.
- **Synchronous** with 2 parameter servers, 2 worker nodes
- **Bounded Synchronous (SSP)** with 2 parameter servers, 3 worker nodes, staleness: 10, 15, 20.

3.1. Results

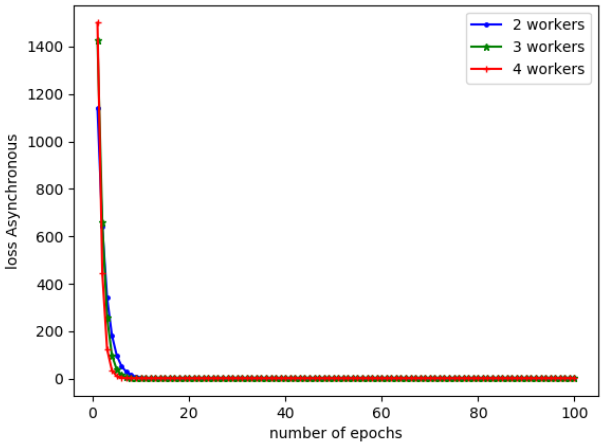

Figure 2. Asynchronous SGD 2, 3, 4 worker comparison

3.2. Analysis

Figure 2 compares the loss vs epochs when number of workers are increased in Asynchronous setting. As expected

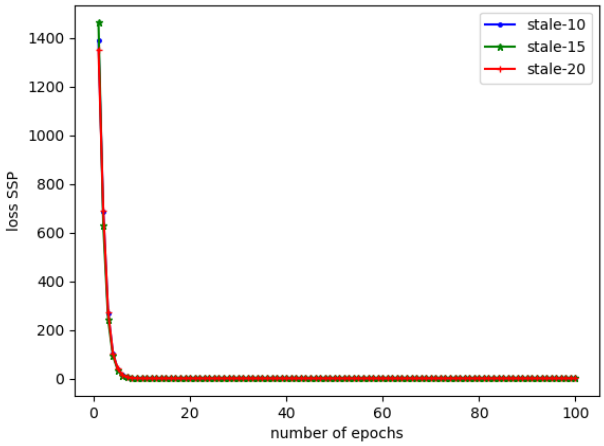

Figure 3. SSP SGD staleness 10, 15, 20

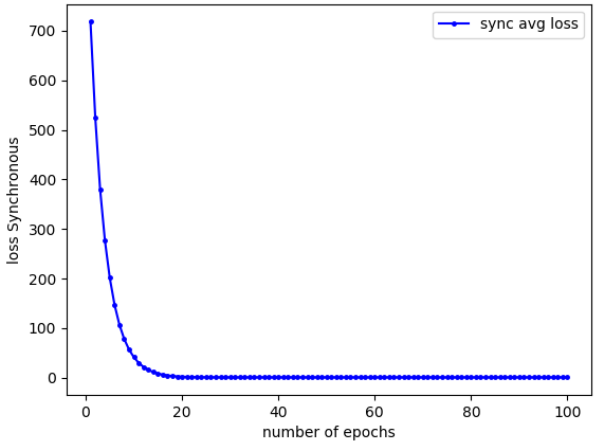

Figure 4. Synchronous SGD

Table 2. Comparison between Asynchronous, Synchronous, SSP distributed setting

METHOD	TRAIN ACC	TEST ACC	TRAIN TIME
ASYNCHRONOUS	75.12	68.94	2754.3S
BSP	75.45	69.22	3573.2S
STALE-10	74.39	67.88	2767.5S
STALE-15	74.53	68.14	2790.3S
STALE-20	74.65	68.23	2803.8S

the asynchronous mode with 4 workers converges faster, because the number of updates increases with the increase in number of workers. Figure 3 compares loss vs epochs

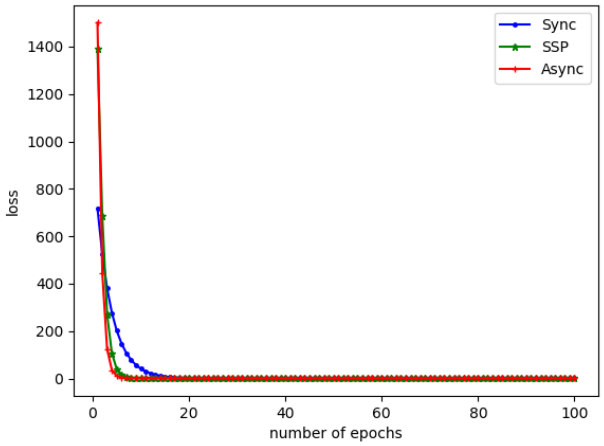

Figure 5. Synchronous Asynchronous SSP comparison

in SSP setting for different values of staleness. Here we do not observe much of a difference between various values of staleness. Figure 5 compares the Asynchronous, Synchronous and SSP modes. As expected Synchronous converges last followed by SSP followed by Asynchronous. This is because Synchronous waits for other workers for updates, BSP drops some updates after threshold and hence coverges slower.

Table 2 compares loss between various modes of distributed setting. The test accuracy are comparable for all the modes, with the Synchronous mode taking the largest time to converge. One general observation is that the accuracy in the distributed setting is lower than the standalone mode.

References

[1] https://www.tensorflow.org/deploy/distributed

[2] https://www.oreilly.com/ideas/distributed-tensorflow