DS 222: ML with Large Datasets (2017)
Assignment 2

Tony Gracious [*1]

Abstract

This is the report experiments on SGD based logistic regression on Local machine and in distributed setting using parameter server.

1. Local Model Description

1.1. Training Stage

- Features used tfidf features has been used. Only the top 2000 words has been used for finding tfidf features
- The model parameters are stored in hashtable index by both the class and word. The parameters are updated as per equation in the slides and a multinomial logistic regression as a loglinear model was learned as this
- The loss function used for plotting is the normalized cross entropy function
- The regularizer was fixed $\mu = 0.01$
- training was done 30 epochs

1.1.1. Constant Learning Rate

Constant learning rate: $\lambda = 0.1$

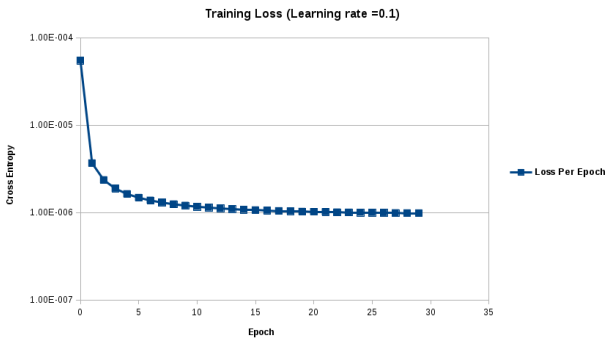

Figure 1. constant Learning rate

1.1.2. Decreasing Learning Rate

Decreasing learning rate $\frac{1}{ln(k+1)+5}$. The k is the sample counter

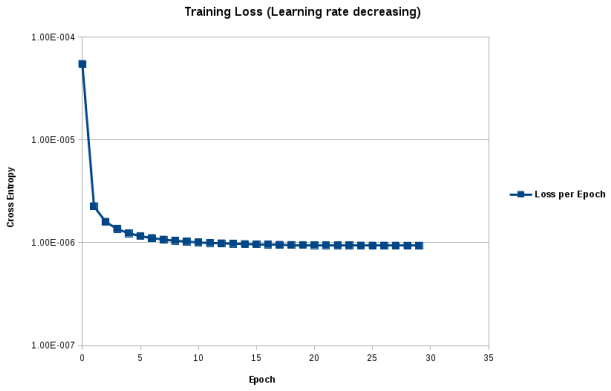

Figure 2. Decreasing Learning Rate

1.1.3. Increasing Learning Rate

Increasing Learning rate $1 - \frac{0.5}{1+log(k+1)}$. The k is the sample counter

1.1.4. Observations

- From figure 1, 2 and 3 we can infer that at initial stage learning rate is high then the learning will be faster as increasing function has the high learning rate initially so it has more decrease in the training loss.
- From figure 3 it has highest initial learning rate so the decrease in loss is higher in the initial epochs.
- So we can also conclude initial learning rate should be high for reaching convergence faster
- From figure 2 we can see as the epochs increases the training loss become constant this because the learning rate becoming small so that the gradient updates are not affecting the parameters
- From figure 3 we can see that as epoch increases the training loss decreases faster due to its larger learning

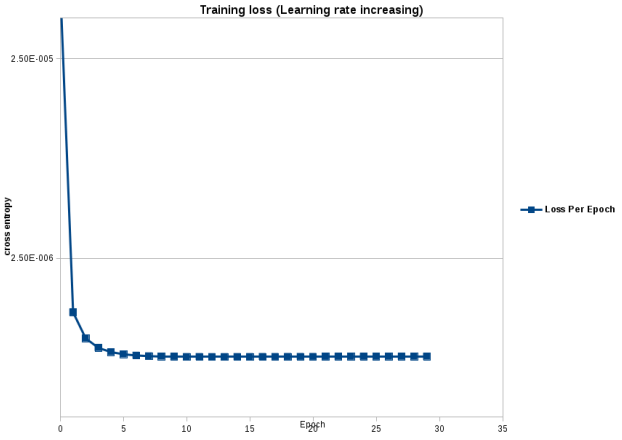

Figure 3. Increasing Learning rate

rate and reaches minimum then starts increasing. This is due over fitting on the noisy samples which are difficult to classify because they are getting more weight age due higher learning rate at increases with epoch.

- From these we can conclude that initial learning rates high value that satisfies descent criteria. Then as epoch increases we should decrease the learning rate such a way it shouldn't become small before the training loss reaches it minimum. The learning rate should be adaptive.

1.2. Testing Stage

As all the all the strategies used learning rate in similar range(0 to 1) the final classification ac curacies was found to be similar. Since the decreasing learning rate strategy used high initial learning rate it has better training and testing accuracies that others as it converged faster and it was also less affected by noisy samples. As the learning rate decreased as epoch increased and difficult to classify got less weight-age. The increasing learning rate has less accuracy of them both due to overfitting on the noisy data. The timing reported 2 are from three different machines but these three implementation will have similar timing performance in single machine as all of them had to run 30 epochs over the entire dataset.

All the above reported performance are on the large dataset. The training has been done train file and testing has done on the test file.

Table 1. Classification accuracy Local

METHOD	TRAINING	TESTING
LOCAL($\lambda = 0.1$)	87.79	82.233
LOCAL(λ DECREASING)	87.9	82.41
LOCAL(λ INCREASING)	86.98	80.23

Table 2. Timing Local

METHOD	TRAINING	TESTING
LOCAL($\lambda = 0.1$)	10267.25	17.15
LOCAL(λ DECREASING)	8670	14.444
LOCAL(λ INCREASING)	9010	15.16

2. Distributed SGD using Parameter server

2.1. Asynchronous SGD

Used tensorflows native parameter server was used to implement it. This was implemented with one node as parameter server and 2 nodes as worker nodes in a local machine. The update equation was the following with parameter $\lambda = 0.1$ and $\mu = 0.01$ fixed through the local implementation. The loss function evaluated is unnormalized cross entropy. For features used tfidf transformer from scikit learn. The no of epochs was 5.

$$W_{i+1} = W_i + \lambda * X_i * (Y_i - Y_{pred})^T - 2 * \lambda * \mu * W_i$$
$$Y_{pred}^T = \frac{1}{1 + e^{X_i^T * W}}$$

Where $W_i's \in R^{N_{features} * N_{Class}}$ is the parameter, is i_th sample feature $X_i \in R^{N_{features} * 1}$, $Y_i \in R^{N_{Class} * 1}$ is the one hot encoded labels and Y_{pred} is the softmax scoring for each class.

2.1.1. OBSERVATIONS

- It was observed that network communications for SGD is higher as the current implementation is sending gradients for each sample to the parameter server. Working on to improve this using data parallelism. So, training for large data set hasn't been done. The reported performance are for the small dataset on the cluster.

- It was also observed that the training loss didn't decrease but the it gave 60.7% training accuracy and 59.8% testing accuracy.

- It took 160.83 seconds to train and 2.48 seconds to test.

Table 3. Loss

EPOCH	LOSS
0	0.0199
1	0.0256
2	0.025567
3	0.025561
4	0.02553
5	0.025462

- The reason for entropy not decreasing is may due to the 2 workers being in the localhost.

Acknowledgements

My initial approach was to use spark, as it allow in memory computations, for implementing SGD which was not successful because of not getting a parameter server that is compatible with Spark. After discussing with Ram I change my to using tensorflow. I like to acknowldege the help of this distribute tensorflow tutorial the code provide them was modified and used in implementing the asynchronous SGD. I also acknowledge help of Aakash in teaching how to setup parameter server and worker nodes. I also took the help of Sonali in help interpreting how me interpret the output of summary variables of tensorflow.

References

Langley, P. Crafting papers on machine learning. In Langley, Pat (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.