9-19-2014

Large Scale Distributed Multiclass Logistic Regression

Pengtao Xie
Carnegie Mellon University

Jin Kyu Kim
Carnegie Mellon University

Eric P. Xing
Carnegie Mellon University, epxing@cs.cmu.edu

Large Scale Distributed Multiclass Logistic Regression

Pengtao Xie, Jin Kyu Kim and Eric Xing
School of Computer Science
Carnegie Mellon University

Abstract

Multiclass logistic regression (MLR) is a fundamental machine learning model to do multiclass classification. However, it is very challenging to perform MLR on large scale data where the feature dimension is high, the number of classes is large and the number of data samples is numerous. In this paper, we build a distributed framework to support large scale multiclass logistic regression. Using stochastic gradient descent to optimize MLR, we find that the gradient matrix is computed as the outer product of two vectors. This grants us an opportunity to greatly reduce communication cost: instead of communicating the gradient matrix among machines, we can only communicate the two vectors and use them to reconstruct the gradient matrix after communication. We design a Sufficient Vector Broadcaster (SVB) to support this communication pattern. SVB synchronizes the parameter matrix of MLR by broadcasting the sufficient vectors among machines and migrates gradient matrix computation on the receiver side. SVB can reduce the communication cost from quadratic to linear without incurring any loss of correctness. We evaluate the system on the ImageNet dataset and demonstrate the efficiency and effectiveness of our distributed framework.

1 Introduction

Multiclass logistic regression (MLR) [2] is an fundamental model for multiclass classification. It has been used and shows effectiveness in many small and moderate scale classification tasks. In large scale problems where the feature dimension is high, the number of classes is large and the number of data samples is numerous, performing multiclass logistic regression is very challenging. MLR learns a weight vector for each class and the dimension of the weight vector is the dimension of features. Thereby, the parameters of MLR is a matrix $\mathbf{W}$ of size $K \times D$, where K is the number of classes and D is the dimension of features. In large scale problems, the feature dimension D can be very huge. For example, web pages are usually represented with bag-of-word (BOW) vectors, whose dimensionality is equal to the vocabulary size and can be of several millions. In vision domain, the features extracted from images and videos are also in million scale. In addition to high dimensional features, the number of classes can also be enormous. For instance, the ImageNet [5] dataset has 22K classes. The Wikipedia[1] dataset has 325K classes. Such high feature dimension and large class number render the parameter matrix $\mathbf{W}$ in MLR to be extremely large. For example, in the Wikipedia dataset, supposing we choose the dimension of features (which is equivalent to the size of vocabulary) to be 1 million, together with the 325K classes, we get a matrix with 0.325 trillion entries. Storing such a matrix will floating-point format requires 1.2 terabytes memory and computing over it can take huge amount of time. Running multiclass logistic regression under such a setting is extremely hard, if not impossible.

In this paper, we build a distributed framework to do parallel multiclass logistic regression on large scale problems. We use stochastic gradient descent

[1] http://lshtc.iit.demokritos.gr/

to do distributed optimization and find that the gradient of the parameter matrix can be computed as the outer product between two vectors which are referred to as sufficient vectors. A great advantage can be taken out of this observation to reduce communication cost. Instead of communicating the gradient matrix among machines, we only communicate the two vectors and reconstruct the gradient matrix on the receiver side. This can reduce communication cost from quadratic to linear, which can greatly boost the performance of the distributed framework. To support this communication patten, we build a Sufficient Vector Broadcaster which broadcasts sufficient vectors among worker machines to collaboratively learn the parameter matrix instead of communicating the large gradient matrices as the existing centralized parameter servers [1, 4, 6] do.

The major contributions of this work are summarized as follows:

- We propose an economic communication pattern called Sufficient Vector Broadcasting (SVB) which synchronizes the parameter matrix of MLR by broadcasting the sufficient vectors among machines rather than communicating the gradient matrices. SVB can reduce the communication cost from quadratic to linear in distributed learning of MLR without incurring any loss of correctness.

- To support this communication pattern, we design a decentralized Sufficient Vector Broadcaster (SVB) where machines are organized into a peer-to-peer network and cooperatively learn the parameter matrix by broadcasting vector-type messages.

The rest of the paper is organized as follows. Section 2 introduces related works. Section 3 introduces the multiclass logistic regression model. Section 4 presents the distributed framework for parallel multiclass logistic regression. Section 5 gives experiments results. Section 6 concludes the paper.

2 Related Work

Designing and implementing parameter servers (PS) for distributed machine learning have been explored in several works [1, 6]. Most of these designs contain a centralized server and a collection of workers. The central server maintains the global parameter and each worker has a local copy of the parameter. Workers compute local updates of the parameter and push the updates to the central server. The central server aggregates updates from workers, applies the update to the global parameter and pushes the fresh global parameter back to workers. The existing parameter servers differ in consistency control. In Bulk Synchronous Parallel (BSP) systems like Hadoop [3], workers must wait for each other at the end of every iteration. In asynchronous parameter server [1], all workers work on their own pace and never waits. Staleness Synchronous Parallel [6] seeks a balance between BSP and ASP. Workers are allowed to see different versions of the parameter, but the difference is bounded. When the centralized parameter servers are applied to learn matrix form parameters, they can be very inefficient. At each iteration, the workers and servers need to communicate the update matrices and parameter matrices, whose size can be very huge. Transmitting these giant matrices can incur considerable communication cost and hence, render computation delay.

3 Multiclass Logistic Regression

Multiclass logistic regression (MLR) is a discriminative linear classifier. Given K classes, it uses a weight vector $\mathbf{w}_k$ to model each class k. Given a data sample with D-dimensional feature vector $\mathbf{x}$, MLR uses $\mathbf{w}_k^\mathsf{T}\mathbf{x}$ to characterize how likely $\mathbf{x}$ belongs to class k. A larger $\mathbf{w}_k^\mathsf{T}\mathbf{x}$ indicates a high possibility. Let $\boldsymbol{\pi}_k$ denote that probability that a data sample $\mathbf{x}$ belongs to class k, MLR assumes $\sum_{k=1}^K \pi_k = 1$ and π_k is defined as

$$\pi_k = \frac{\exp(\mathbf{w}_k^\mathsf{T}\mathbf{x})}{\sum_{l=1}^K \exp(\mathbf{w}_l^\mathsf{T}\mathbf{x})} \quad (1)$$

Given a set of samples $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$, where $\mathbf{x}$ are feature vectors and $\mathbf{y}$ are class labels[2], MLR learns model parameter $\{\mathbf{w}_k\}_{k=1}^{K}$ by maximizing the data likelihood $\prod_{n=1}^{N}\prod_{k=1}^{K}\boldsymbol{\pi}_k^{y_{nk}}$, which is equivalent to minimizing the negative log likelihood. We use stochastic gradient descent to optimize the negative log likelihood and the stochastic gradient of $\mathbf{w}_k$ w.r.t sample $\mathbf{x}_n$ can be computed as $\triangle\mathbf{w}_k = (\pi_k - y_{nk})\mathbf{x}_n$. If we compile weight vectors of all classes $\sum_{k=1}^{K}\pi_k = 1$ into a matrix W, where the kth row of $\mathbf{W}$ contains $\mathbf{w}_k^{\mathsf{T}}$, then the stochastic gradient $\triangle\mathbf{W}$ w.r.t $\mathbf{x}_n$ is

$$\triangle\,\mathbf{W} = (\boldsymbol{\pi} - \mathbf{y})\mathbf{x}_n^{\mathsf{T}} \tag{2}$$

where $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_K)$.

4 Distributed Parallel Multiclass Logistic Regression

To do multiclass logistic regression on large scale problems, a single machine is apparently infeasible. We resort to distributed systems to parallelize the learning. Similar to [1, 6], we partition the data onto M worker machines and each worker machine holds a copy $\widetilde{\mathbf{W}}$ of the parameter $\mathbf{W}$. Each worker updates its own $\widetilde{\mathbf{W}}$ with the data it holds and different $\widetilde{\mathbf{W}}$ on different workers are synchronized globally to ensure they are as same as possible.

4.1 Sufficient Vector Broadcaster

To synchronize parameter copies among different workers, [**?**, **?**, **?**] use a a centralized server which maintains the global parameter $\mathbf{W}$. Workers compute updates $\triangle\mathbf{W}$ using the data it holds and push the updates to the central server. The central server aggregates the updates from different works and applies them to the global $\mathbf{W}$ and pushes the updated $\mathbf{W}$ back to workers. The data communicated between workers and the central server are $\mathbf{W}$ and $\triangle\mathbf{W}$, which are both matrices of size $K \times D$. In large scale problems, K and D can both be very large, rendering the matrices $\mathbf{W}$ and $\triangle\mathbf{W}$ to be very large. Communicating these large matrices among machines is very time consuming.

Note that the gradient matrix $\triangle\mathbf{W}$ in Eq.(2) is of rank one, which is essentially the outer product between two vectors: $\boldsymbol{\pi}$ of length K and $\mathbf{x}$ of length D. All the information of $\triangle\mathbf{W}$ are contained in these two vectors and the entries in $\triangle\mathbf{W}$ can be completely computed from the two vectors. We call these two vectors as sufficient vectors in the sense that they are sufficient to construct the gradient matrix. This grants us an advantage to greatly reduce communication cost. Instead of sending $\triangle\mathbf{W}$ directly among machines, we can alternatively send the two vectors $\boldsymbol{\pi}$ and $\mathbf{x}$ and compute $\triangle\mathbf{W}$ on the receiver side. Suppose worker A takes a labeled data $(\mathbf{x}_A, \mathbf{y}_A)$ and wants to do gradient update of its own parameter copy $\widetilde{\mathbf{W}}_A$. It computes $\boldsymbol{\pi}_A$ using Eq.(1) and then computes the gradient matrix $\triangle\widetilde{\mathbf{W}}_A = (\boldsymbol{\pi}_A - \mathbf{y}_A)\mathbf{x}_A^{\mathsf{T}}$. $\triangle\widetilde{\mathbf{W}}_A$ is then used to update $\tilde{\mathbf{W}}_A$. Meanwhile, worker A wants this update to be seen by other workers, namely, $\triangle\widetilde{\mathbf{W}}_A$ is to be used to update the parameter copies of other workers, say $\triangle\widetilde{\mathbf{W}}_B$ of worker B. Instead of sending $\widetilde{\mathbf{W}}_A$ to B, worker A only sends the two vectors $\boldsymbol{\pi}_A - \mathbf{y}_A$ and $\mathbf{x}_A$ to B. Worker B receives them and computes $\triangle\widetilde{\mathbf{W}}_A = (\boldsymbol{\pi}_A - \mathbf{y}_A)\mathbf{x}_A^{\mathsf{T}}$ and then apply $\triangle\widetilde{\mathbf{W}}_A$ to update its own parameter copy $\triangle\widetilde{\mathbf{W}}_B$. Sending the two vectors only needs to send $K+D$ numbers while sending the whole gradient matrix requires sending $K \cdot D$ numbers. In other words, the communication cost is reduced from quadratic to linear. While reducing communication cost, this approach does not incur any compromise of the correctness of computation. $\triangle\widetilde{\mathbf{W}}_A$ computed on the worker B is exactly the same as that on worker A.

To support the above mentioned communication pattern, we build a framework called Sufficient Vector Broadcaster (SVB) which broadcasts the sufficient vectors among workers to collaboratively learn a model rather than communicating the huge gradient and parameter matrices.

We begin with an informal explanation of SVS. Assume a collection of P workers are cooperatively learning a matrix parameter $\mathbf{W}$. Each worker p has

[2]The class label is represented with 1-of-K coding. $\mathbf{y}$ is the coding vector where one component is 1 and all others are 0.

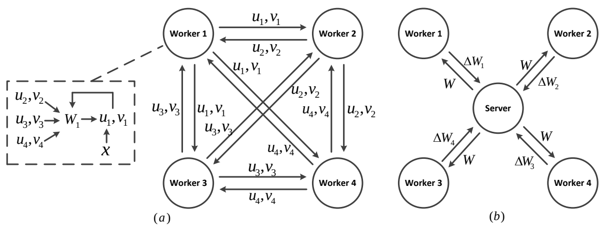

Figure 1: (a) Sufficient Vector Broadcaster (SVB). SVB consists of a collection workers, each of which maintains a local copy of the model parameter. Each worker p uses local parameter copy $\mathbf{W}_p$ and training data $\mathbf{x}$ to compute sufficient vectors $\mathbf{u}$, $\mathbf{v}$, updates $\mathbf{W}_p$ with $\mathbf{u}$, $\mathbf{v}$ and broadcasts $\mathbf{u}$, $\mathbf{v}$ to other workers. Meanwhile, it receives sufficient vectors (SV) computed by other workers and uses these SVs to update $\mathbf{W}_p$. (b) Centralized Parameter Server (CPS). CPS consists of a centralized server and a collection of workers. Central server maintains the global parameter and each worker maintains a local copy of the parameter. Each worker p computes a local update matrix $\triangle\mathbf{W}$ and sends $\triangle\mathbf{W}$ to the central server. Central server collects local updates from all workers and applies them to the global parameter matrix. The global parameter matrix is then pushed back to workers.

a local copy $\mathbf{W}_p$ of the oracle parameter $\mathbf{W}$. Suppose $\mathbf{W}^*$ is the optimal solution of $\mathbf{W}$, the goal is to make each $\mathbf{W}_p$ iteratively converges to $\mathbf{W}^*$, although the convergence path of each $\mathbf{W}_p$ is different. Each worker p holds a partition of the data and makes additive update to its local copy $\mathbf{W}_p \leftarrow \mathbf{W}_p + \triangle\mathbf{W}_p$, where the update $\triangle\mathbf{W}_p$ can be computed as the outer product of two vectors: $\triangle\mathbf{W}_p = \mathbf{u}_p\mathbf{v}_p^\mathsf{T}$ and $\mathbf{u}_p$,$\mathbf{v}_p$ are computed from $\mathbf{W}_p$ and the local data worker p holds. We call $\mathbf{u}_p, \mathbf{v}_p$ as sufficient vectors in the sense that they are sufficient to construct the update $\triangle\mathbf{W}_p$. Each worker p broadcasts the sufficient vector $\mathbf{u}_p, \mathbf{v}_p$ to other workers and also receives the sufficient vector $\mathbf{u}_q, \mathbf{v}_q$ (where $q \in \{1, \cdots, P\}/\{p\}$) sent from other workers to update its local copy $\mathbf{W}_p \leftarrow \mathbf{W}_p + \triangle\mathbf{W}_q$.

Figure 1(a) shows the graphical illustration of the Sufficient Vector Broadcaster (SVB). Worker 1 computes sufficient vectors $\mathbf{u}_1, \mathbf{v}_1$ based on the local parameter copy $\mathbf{W}_1$ and a data point $\mathbf{x}$ randomly sampled from the local data worker 1 holds. Then $\mathbf{u}_1, \mathbf{v}_1$ are used to update $\mathbf{W}_1 \leftarrow \mathbf{W}_1 + \mathbf{u}_1\mathbf{v}_1^\mathsf{T}$. Worker 1 broadcasts $\mathbf{u}_1, \mathbf{v}_1$ to worker 2, 3, 4 and receives the sufficient vectors $\mathbf{u}_2, \mathbf{v}_2$, $\mathbf{u}_3, \mathbf{v}_3$, $\mathbf{u}_4, \mathbf{v}_4$ sent from work 2, 3, 4. $\mathbf{u}_2, \mathbf{v}_2$, $\mathbf{u}_3, \mathbf{v}_3$, $\mathbf{u}_4, \mathbf{v}_4$ are used to update $\mathbf{W}_1$. The similar behavior happens on work 2, 3 and 4. In SVB, all the data communicated between workers are the sufficient vectors.

For multiclass logistic regression where $\triangle\mathbf{W}$ can be computed from two sufficient vectors $\mathbf{u}, \mathbf{v}$, the communication cost of SVB is much lower than that of Centralized Parameter Server (CPS). In SVB, all we need to communicate are the sufficient vectors. The communication cost is linear to the dimension of the sufficient vectors. CPS needs to communicate parameter matrix $\mathbf{W}$ and gradient matrix $\triangle\mathbf{W}$. The communication cost is quadratic to the dimension of the sufficient vectors.

4.2 Implementation Details

Figure 2 shows the implementation details on each worker. Each worker maintains three kinds of threads: local update thread, remote update thread and communication thread. Each work holds a local copy of the parameter matrix and a partition of the training data. It also maintains an input sufficient vector (SV) queue which stores the received SVs from other workers and an output SV queue which stores SVs to be sent to other workers. In each iteration, the local update thread randomly chooses a sample from the training data, uses the parameter matrix to compute the sufficient vectors on the training sample, updates the parameter matrix and pushes the SVs to the output SV queue. The remote update thread fetches SVs from the input SV queue and uses them to update the parameter matrix. The communication thread receives SVs from other workers and pushes them into the input SV queue and fetches SVs from the output SV queue and broadcasts them to other workers.

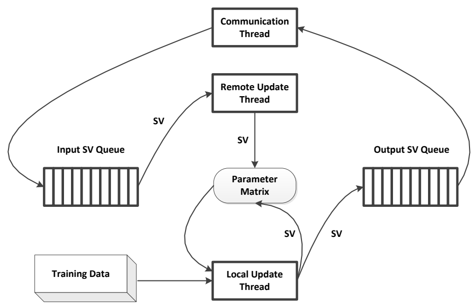

Figure 2: Implementation of workers in SVS. Each worker has a local parameter matrix and a partition of training data. It maintains an input SV queue and an output SV queue. There are three types of threads. The local update thread computes local SVs and uses local SVs to update parameter matrix. Remote update thread fetches SVs computed by other workers and updates the parameter matrix. Communication thread sends local SVs to and receives remote SVs from other machines.

4.3 Sparse Sufficient Vector Broadcasting

In many applications, the sufficient vectors can be sparse, which can be taken advantage of to further reduce communication cost and computation cost. A feature vector $\mathbf{x}$ is sparse if features are sparse. For instance, in Wikipedia text classification, the feature dimension is equal to the size of the vocabulary, which can be of several millions. However, in each document, the number of words is usually a few thousand or less. This induces great sparsity. A class probability vector $\boldsymbol{\pi}$ can be sparse due to the skewed distribution over classes. When the number of classes is large, a few components in $\boldsymbol{\pi}$ are of large values. A large portion of components are so small that they can be neglected and set to zero. The sparsity of both the feature vector and probability vector can help us to reduce communication cost and computation cost. In communication, we only need to broadcast the nonzero values of the sufficient vectors. The communication cost can be reduced linearly with the sparsity ratio. In computation, only the nonzero entries in the sufficient vectors contribute to the nonzero entries in the gradient matrix. We only need to multiply the nonzero entries. The computation cost can be reduced quadratically with the sparsity ratio.

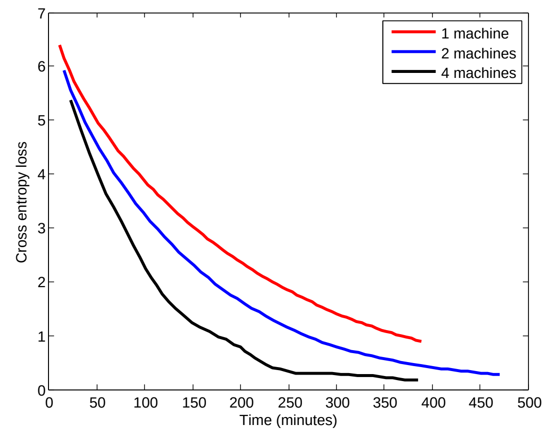

Figure 3: Convergence curves of cross entropy loss on ImageNet

5 Evaluation

We evaluate our framework on the ImageNet [5] dataset. We represent images with locality-constrained linear coding (LLC) [7] features whose dimension is 21K. The number of classes are 1K. The number of images are 63K. We run MLR on 1 machine, 2 machines and 4 machines and compare their convergence speed. Figure 3 shows the convergence curves of the cross entropy loss. The x-axis is time measured in minutes and the y-axis is the objective value. Figure 4 shows the convergence curves of accuracy. The x-axis is time measured in minutes and the y-axis is the classification accuracy. As can be seen from these two figures, MLR running on 2 machines converges faster than that on one machine. And MLR running on 4 machines converges faster than that one 2 machines. This demonstrates that our framework scales well with the number of machines.

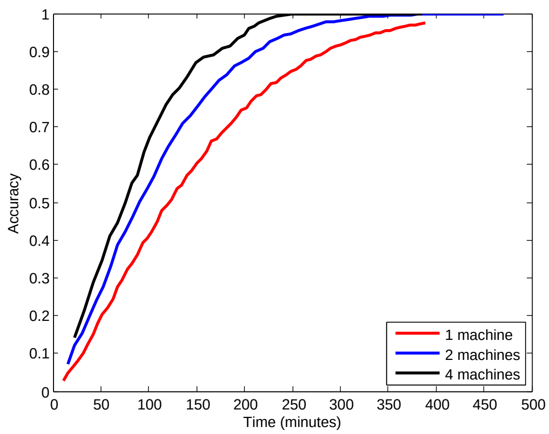

Figure 4: Convergence curves of accuracy on ImageNet

6 Conclusion

In this paper, we present a distributed framework for large scale multiclass logistic regression. Observing that the gradient matrix of MLR can be computed as the outer product of two vectors, we propose a communication pattern called Sufficient Vector Broadcasting to reduce communication cost. Instead of communicating the large gradient matrices, SVB broadcasts sufficient vectors among machines and reconstructs the gradient matrices on the receiver side, which can reduce communication cost from quadratic to linear. We implement a Sufficient Vector Broadcaster to support this communication pattern. Evaluation on ImageNet demonstrates the efficiency of our framework.

References

[1] Ahmed, A., Aly, M., Gonzalez, J., Narayanamurthy, S., & Smola, A. J. (2012, February). Scalable inference in latent variable models. In Proceedings of the fifth ACM international conference on Web search and data mining (pp. 123-132). ACM.

[2] Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 1, p. 740). New York: springer.

[3] J. Dean, and S. Ghemawat, MapReduce: simplified data processing on large clusters, In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6 (OSDI'04), Vol. 6. USENIX Association, Berkeley, CA, USA, 10-10.

[4] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., ... & Ng, A. Y. (2012). Large scale distributed deep networks. In Advances in Neural Information Processing Systems (pp. 1223-1231).

[5] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE. Chicago

[6] Ho, Q., Cipar, J., Cui, H., Lee, S., Kim, J. K., Gibbons, P. B., ... & Xing, E. (2013). More effective distributed ml via a stale synchronous parallel parameter server. In Advances in Neural Information Processing Systems (pp. 1223-1231).

[7] Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010, June). Locality-constrained linear coding for image classification. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 3360-3367). IEEE.