Gossip training for deep learning

Michael Blot$^{(1)}$ [*] **David Picard**$^{(2)}$ **Matthieu Cord**$^{(1)}$ **Nicolas Thome**$^{(1)}$

(1) Sorbonne Universités, UPMC Univ Paris 06, LIP6, 4 place Jussieu 75005 Paris, France
(2) ETIS/ENSEA - Université Cergy-Pontoise, CNRS, Paris, France

Abstract

We address the issue of speeding up the training of convolutional networks. Here we study a distributed method adapted to stochastic gradient descent (SGD). The parrallel optimization setup uses several threads, each applying individual gradient descents on a local variable. We propose a new way to share information between different threads inspired by gossip algorithms and showing good consensus convergence properties. Our method called GoSGD has the advantage to be fully asynchronous and decentralized. We compared our method to the recent EASGD in [ZCL15]. Our experiments on CIFAR-10 show encouraging results.

1 Introduction

With deep convolutional neural networks (CNN) introduced by [Fuk80] and [LeC+98], computer vision tasks and more specifically image classification have made huge improvements last few years following [KSH12]. CNN performances benefit a lot from big databases of annotated images like [Rus+15] or [Lin+15]. They are trained by optimizing a loss function with gradient descents computed on random mini-batch. This method called stochastic gradient descent [SGD] has proved to be very efficient to train neural networks in general.

However current CNN structures are very deep like the 200 layers network ResNet of [He+16] and contains a lot of parameters (around 60M for alexnet[KSH12]) making the training on big datasets very slow. Computation on GPU accelerates the training but it is still difficult to test many architectures.

Nevertheless the mini-batch optimization seems suitable for distributing the training. Many methods have been proposed like [ZCL15] or [Dea+12]. They process SGD in parallel on different threads to optimize a local neural network. The different threads are called workers. Additionally the workers periodically exchange information with a central network. The role of this central variable is essential to mutualize the information as well as ensuring that all workers network converge toward a same local minimum. Indeed because of the symmetry property of neural networks well studied in [Cho+15] the different workers could give very different optimizations. Having a consensus is important to fully benefit from the parallelism and the information sharing. Unfortunately the proposed methods are not decentralized resulting in loss of time for synchronizing the updates of the central network. It could result of a suboptimal use of distributed computation.

A well known example of decentralized distributed algorithm is gossip averaging. As studied in [Boy+06] this method is very fast to make different agents converge toward a consensus by exchanging information in a peer to peer way. Gossip averaging has already been adapted to other machine learning algorithms such as kernel methods [Col+16] or PCA [FPG15]. This family of algorithm presents many advantages like being fully asynchronous and totally decentralized as they do not

[*]Thank to *DGA* for funding.

require a central variable. We propose here to associate this method with SGD in order to apply it to deep learning and more specifically CNN. We call the resulting optimization method GoSGD for Gossip Stochastic Gradient Descent.

The first section introduces the GoSGD algorithm. Then some experiments illustrate the good convergence properties of decentralized GoSGD.

2 Gossip Stochastic Gradient Descent

The objective is to minimize $l(x) = \mathbb{E}_{y\sim\mathcal{I}}[g(x,y)]$ where in out setup y is a couple variable (image, label) following the natural image distribution law, and x is the CNN parameters with g the error function. As used in [ZCL15] and discussed in [Boy+11] the problem can be derived in a distributed fashion as minimizing:

$$\sum_{i=1}^{M} l(x_i) + \frac{\rho}{2}||x_i - \overline{x}||_2^2 \tag{1}$$

with the x_i being worker's local variables and $\overline{x} = \frac{1}{M}\sum_{i=1}^{M} x_i$ the global consensus.
We can rewrite this loss in order to anticipate the gossip exchanges:

$$\sum_{i=1}^{M} l(x_i) + \frac{\rho}{4M}\sum_{i}^{M}\sum_{j}^{M} ||x_i - x_j||_2^2 \tag{2}$$

Finally we consider the following equivalent function in our optimization problem introducing $A = (a_{ij})_{i,j}$ a random matrix:

$$\sum_{i=1}^{M} \mathbb{E}\Big[g(x_i, y) + \sum_{j}^{M} a_{ij}||x_i - x_j||_2^2\Big] \tag{3}$$

In our gossip method the terms in the sum of (3) are sampled concurrently by different workers. a_{ij} is a random variable controlling exchanges between workers i and j with $p = \mathbb{P}(a_{ij} \neq 0)$ and $\mathbb{E}(a_{ij}) = \frac{\rho}{4M}$.

2.1 GoSGD algorithm

The GoSGD algorithm considers M independent agents called workers. Each of them hosts a CNN of the same architecture with a sets of weights noted X_i for worker i. They are all initalized with the same value. During training all workers iterativelly proceed two steps of computations described below. One consisting on local optimisation with gradient descent and the other aiming at exchanging information in order to ensure a consensus between workers:

Step 1 (Gradient update): At all iterations t a worker updates its hosted network's weights with a stochastic gradient descent on a random mini-batch. For the i-th worker the update is:

$$X_i^{t^+} = X_i^t - \eta^t v_i^t$$

Where η^t is the learning rate at iteration t and $v_i^t = \frac{1}{|b(i,t)|}\sum_{y\in b(i,t)} \nabla_X g(X_i^t, y)$ is an approximation computed on the sampled mini-batch $b(i,t)$ of the gradient of the expected error function at point X_i^t.

Step 2 (Mixing update): After the gradient descent each worker draws a random Bernoulli variable noted S with expectancy p. This variable decides if the worker is sharing its information with another worker which will be chosen uniformly among the others. To share the information between the update processes, we use a sum-weight gossip protocol [KDG03]. Sum-weight protocols use a sharing variable associated with each worker (noted α_i for agent i and initialized to $\frac{1}{M}$) that is updated whenever information is exchanged and defines the rate at which information is mixed. Due to their push only nature, no synchronization is required. The exchange between a worker i drawing a successful S and worker j are described in table 2.

At each iteration a worker send at most once its weights but can receive weights from several others. In this case the worker updates its weights sequentially in the reception order before performing any

Table 1: GoSGD Pseudocode of a worker

```
Input:
$x$ initialization of the weights
$p$ probability of exchange
$M$ the number of threads
$nBatch$ the number of training batch per workers
$\nu$ the learning rate
Init: $\alpha_i = \frac{1}{M}$, $x^i = x$
for $batch = 1\ to\ nBatch$ do
    $x^i = x^i - \nu g(x^i, batch)$
    if $S \sim B(p)$ then
        $alpha^i = \frac{alpha^i}{2}$
        Send($x^i,\ \alpha_i$) to $U(\{1, .., M\})$
    end if
    for $j = 1\ to\ M$ do
        if $get_j(x_j, \alpha_j)$ then
            $x^i = \frac{\alpha^i}{\alpha^i+\alpha^j} x^i + \frac{\alpha^j}{\alpha^i+\alpha^j} x^j$
            $\alpha_i = \alpha_i + \alpha_j$
        end if
    end for
end for
```

Table 2: Update sheme of a worker j receiving information from a worker i

$$\begin{aligned}
X_i^{t+1} &= X_i^{t^+} \\
\alpha_i^{t+1} &= \frac{\alpha_i^t}{2} \\
X_j^{t+1} &= \frac{\alpha_j^t}{\alpha_i^{t+1}+\alpha_j^t} X_j^{t^+} + \frac{\alpha_i^{t+1}}{\alpha_i^{t+1}+\alpha_j^t} X_i^{t^+} \\
\alpha_j^{t+1} &= \alpha_j^t + \alpha_i^{t+1}
\end{aligned}$$

gradient update. Since agent i can perform the update without waiting for an answer from j, and j performs its update in a delayed fashion, no agent is ever idling and all computing resources are always being used (either performing a gradient update or a mixing update).

Remark that these update rules are equivalent to common sum-weight gossip rules, with the main difference being that we choose not to scale X_i which results in a more complex update rule for X_j. Consequently, several key properties of sum-weight protocols are retained:

Property 1: $\alpha^t = (\alpha_i^t)_{i=1..M}$ stays a stochastic vector.

Property 2: Consensus ($\forall i, X_i^t \to 1/M \sum_j X_j^t$) is obtained at exponential speed with respect to the number of mixing updates, when there is no gradient update.

Remark: p is the only adjustable parameter of the algorithm. Obviously the bigger is p the more exchanges between threads there are and eventually the closer the workers' weights will be. In our experiment, a low p such as 0.01 already ensures a very good consensus.

2.2 Test model

The model that is evaluated on the test set is called test model. In the GoSGD method it is simply the averaging of all workers models weights:

$$\overline{X}^t = \frac{1}{M} \sum_{i=1}^{M} X_i^t$$

It is possible to show that the test model can be rewriten:

$$\overline{X}^t = \overline{X}^0 - \sum_{\tau=0}^{t-1} \nu^\tau \sum_{i=1}^{M} \lambda_i^\tau v_i^\tau$$

where $\lambda = (\lambda_i)_{i=1..M}$ is a stochastic vector.
If the workers' weights are close enough to the average consensus the value v_i^τ is very close to the gradient at point $\overline{X}^\tau$ computed on batch $b(i, \tau)$. Thus the update term $\sum_{i=1}^{M} \lambda_i^\tau v_i^\tau$ at iteration

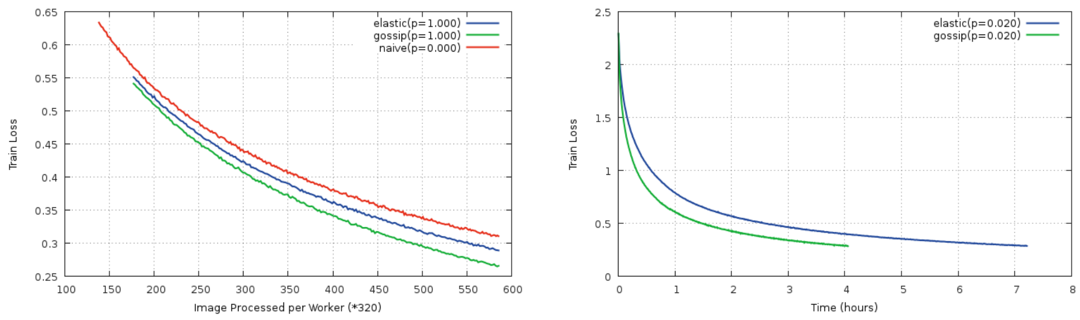

Figure 1: Evolution of the loss during training

τ approximates the gradient at point $\overline{X}^{\tau}$ on a batch of size $M \sum_{i=1}^{M} |b(i, \tau)|$. This distributed optimization scheme is then equivalent at computing a normal (single-thread) gradient descent optimization but with bigger batchs resulting on gradient with less noise.

3 Experiments

We compare the convergence speed of GoSGD (gossip) with EASGD (elastic). The version of EASGD we used is the version with momentum (=0.99) namely MEASGD with parameters $\alpha = 0.887$ as suggested by the author. The parameter τ is equivalent to the inverse of our parameters p and controls the frequency of exchange for one worker. The experiments have been done on CIFAR-10, see [Kri09] for detailed presentation. The network used is the same as in [ZCL15] and describe in [Wan+13]. For the data sampling (loading and augmentation) we used the same protocol as in [ZCL15]. During the training the learning rate is kept constant equal to 0.01 and the weight decay to 10^{-4} and all batch are composed of 128 images. We used eight workers. The only parameters that can vary are the probability p controlling the frequency of exchange. We implemented both methods in torch framework [Col+] and we used 4 Titan X GPU.

We have represented on Figure 1 the evolution of the train losses during training for the different methods. As baseline we displayed a "Naive" scheme correponding of a train without any exchange between workers. The train loss depicted is an averaging of the train loss of the last 50 batches taken regardless of the workers.

3.1 Interpretation

The first curve on the left of Figure 1 is in function of the number of images processed by worker. For clarity we have zoomed at the end of the curve. It studies the benefits of exchanging information regardless of the communication time, thus to maximize the exchanges we set p to 1. We can see that GoSGD do a better use of the exchanges than EASGD. It can signify that the gossip strategy imply a better consensus during training.

The second graph represents the evolution of the loss with time in hours. We used a small p (0.02) as it seems to give a good compromise between communication costs and consensus both for GoSGD and EASGD. We can see that GoSGD is a lot faster than EASGD. Our strategy is converging in about 4 hours when EASGD needs more than 7 hours to reach the same train loss score. This show that distributing SGD can benefit a lot from decentralized strategies.

4 Discussion

In this paper, we have introduced a new learning scheme for deep architectures based on Gossip: GoSGD. We have experimentally validated our approach. Anyway there are two theoretical aspects interesting to discuss: For the first point it is important to exhibit a consensus despite the gradient descent operations in addition to the gossip exchanges. We have discussed this requirement for our model in property 2 in section 2. For the second point as many distributed method can be interpreted as a way to compute gradients with less noise, it could be interesting to quantify this property and compare the different methods with this insight.

References

[Fuk80] K. Fukushima. “Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics, 36(4):193–202* (Dec. 1980).

[LeC+98] Y. LeCun et al. “Gradientbased learning applied to document recognition. Proceedings”. In: *IEEE, 86(11):2278–2324* (Nov. 1998).

[KDG03] David Kempe, Alin Dobra, and Johannes Gehrke. “Gossip-based computation of aggregate information”. In: *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*. IEEE. 2003, pp. 482–491.

[Boy+06] Stephen Boyd et al. “Randomized Gossip Algorithms”. In: *IEEE transaction on information theory* (June 2006).

[Kri09] A. Krizhevsky. “Learning multiple layers of features from tiny images”. PhD thesis. Computer Science Department University of Toronto, 2009.

[Boy+11] S Boyd et al. “Distributed optimization and statistical learning via the alternating direction method of multipliers”. In: *Found. Trends Mach. Learn., 3(1):1–122* (June 2011).

[Dea+12] Jeffrey Dean et al. “Large Scale Distributed Deep Networks”. In: *NIPS* (June 2012).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NIPS* (2012).

[Wan+13] L Wan et al. “Regularization of neural networks using dropconnect”. In: *ICML* (June 2013).

[Cho+15] Anna Choromanska et al. “The Loss Surfaces of Multilayer Networks”. In: *AISTATS* (Dec. 2015).

[FPG15] Jerome Fellus, David Picard, and Philippe-Henri Gosselin. “Asynchronous gossip principal components analysis”. In: *Neurocomputing, Elsevier, 2015* (2015).

[Lin+15] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *arxiv* (May 2015).

[Rus+15] Olga Russakovsky* et al. “ImageNet Large Scale Visual Recognition Challeng”. In: *IJCV* (2015).

[ZCL15] Sixin Zhang, Anna Choromanska, and Yann LeCun. “Deep learning with Elastic Averaging SGD”. In: *NIPS* (Nov. 2015).

[Col+16] Igor Colin et al. “Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions”. In: *ICML* (2016).

[He+16] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CVPR* (June 2016).

[Col+] Ronan Collobert et al. *torch 7*. `http://torch.ch/`.