OPTIMIZATION OF NEURAL NETWORKS TRAINING WITH VECTOR-FREE HEURISTIC ON APACHE SPARK

Khamitov K.G. [a], Popova N.N [b]

Lomonosov Moscow State University

E-mail: [a] berserq0123@gmail.com, [b] popova@cs.msu.su

One of the most computation complicated tasks during Neural networks development is a training process. It could be considered as high dimensional numerical optimization tasks. In the modern MapReduce systems, like Apache Spark, it's hard to efficiently implement traditional for Neural networks training gradient-based algorithms or Quasi-newton L-BFGS method, because there are too many non-linear or memory bound operations, could dramatically decrease the performance of your cluster and scalability of your training task. It's known that for L-BFGS methods there are Vector-Free heuristics which allows to reduce task complexity in terms of an amount of Map and Reduce operations, in the application for large-scale logistic regression tasks. Also, it's unclear in which types of neural networks these approaches are applicable. In this research, we applied the heuristics which reduces the amount of the memory-bound and nonlinear operations, Vector-Free heuristic, to the modern numerical optimization algorithms, like L-BFGS, Adam, AdaGrad on Spark cluster. We tested modified versions of algorithms on the different types of NNs architectures: MLP, VGG-16, LSTM, which covers popular neural network types and particular tasks for them. Also, to provide an efficient and usable environment for the computational experiment we developed a software system which could in a semi-automatic way perform a testing of these methods. It allows a researcher to measure the effect of Vector-Free or other heuristics on different platforms and neural networks architectures. Also, this system supports comparing with external data, so researcher is able to compare effectiveness and speedup with other system types like GPU's versions of the methods above. In this research, we only applied this type of heuristic to this algorithms without taking any consideration which thing results in bad performance of modified methods, and don't provide any empirical boundaries for the errors or convergence. All experiments have been performed on the Microsoft Azure cloud platform, with 16 HD12v2 nodes.

Keywords: Apache Spark, Neural networks, L-BFGS

1. Introduction

In the Big Data era, many applications require solving optimization problems with billions of variables on a huge amount of training data.

One of the most computation complicated tasks during Neural networks(NNs) development is a training process. It could be considered as high dimensional numerical optimization tasks. In the modern MapReduce environments, like Apache Spark, it's hard to efficiently implement traditional for Neural networks training gradient-based algorithms or Quasi-newton L-BFGS method, because there are too many non-linear or memory bound operations, could dramatically decrease the performance of your cluster and scalability of your training task. It's known that for L-BFGS methods there are Vector-Free heuristics which allows to reduce task complexity in terms of an amount of Map and Reduce operations, in the application for large-scale logistic regression tasks. Also, currently it's unclear in which types of neural networks these approaches are applicable, and which limits and coefficients should be used in each optimization method for each type of the method.

In this research, we determined the heuristic which reduces the amount of the memory-bound and nonlinear operations, Vector-Free heuristic, to the modern numerical optimization algorithms, which are used during NNs training process on MapReduce cluster. Also, tests on different types neural networks were performed, in this article, we take into consideration only recurrent, convolutional, and perceptron-based. For testing heuristic effectiveness the particular problems were chosen.

Also, to provide an efficient and usable environment for the computational experiment we developed a software system which could in a semi-automated way perform a testing of these methods, and modifying a set of problems to test on different architectures. It allows the researcher to measure the effect of Vector-Free or other particular heuristics on different platforms and neural networks architectures. Also, this system supports comparing with external data, so the researcher could to compare performance, the effectiveness of parallelization, and speedup with other system types like GPU versions of the chosen methods, etc.

2. Optimization algorithms

The main goals of this research are:

- Choose heuristics, which reduce the amount of memory-bound and non-linear operations on large-scale vectors in optimization methods
- Determine applicability of such heuristics for neural networks training

To validate such tasks we need to choose a proper set of methods. There are lot’s of optimization methods are used during NNs training. Most of them could be classified by different approaches. By order of techniques which are used to improve the convergence:

• Momentum-based(Adam, Nesterov)

• Adaptive-one(Adam, AdaGrad)

In this work we take into consideration only three methods which are mostly used in the packages for Neural Networks training: two SGD-like: AdaGrad, Adam and Quasi-Newton one: L-BFGS.

3. Vector-Free Heuristics

In order to solve the first goal, we decide to choose Vector-Free heuristic as such technique. Vector-Free heuristic [1] – is heuristic which reduces the amount of memory-bound and non-linear operations on each computational step. It allows to decrease the amount of Map and even Reduce operations, that enables to improve particular performance in a MapReduce environment.

Particular techniques which are used:

• Assembling shared state(some basis, or dot-products between them) and shifting the computations between large-scale vectors to manipulation on coefficients which are used in a shared state

• Replacing non-linear operations on vectors with some finite sum of series decomposition

Originally it's used only for L-BFGS to solve large regression problems.

But such techniques could be applied not for all methods. E.g. only the second could be applied to the Adam. First technique implies the broadcasting the shared state which consists dot products between basis vectors, which could be chosen in very different approaches, e.g. in VF-LBFGS it's chosen from deltas on optimizing variables and gradients. And the resulting shift for descent is computed via reduce operation on such vector that stored in different nodes.

The main changes in methods decomposed to 3 major steps:

• distributing shared state

• compute the new update via shared state

• updating shared state via new values

The effect of heuristic depends on the number of eliminated Map and Reduce operations in the second step, in relation to amount operations to support such shared state. If this number is greater than 1, the heuristic could majorly reduce the training time. Shared update technique was brought from [1], and used only way from L-BFGS replacing the vectors in basis in a cycle of predefined length. It allows easy update of the shared state with a small number of Reduce operations. Modified versions of the AdaGrad and L-BFGS could be found in [2].

L-BFGS

Global method of non-convex optimization which is more suitable for non-convex functions which often result in the deep NNs. Given an optimization problem with d variables, BFGS requires to store a dense d by d matrix to approximate the inverse Hessian, where L-BFGS only need to store a few vectors of length d to approximate the inverse Hessian implicitly. Such method could be considered as an approximation to Newton-method, which allow solving global non-convex problems with the second degree of convergence but without direct computation of second derivatives. The ajor drawback of this method is storing a large system of each deltas(s_k, y_k), each for gradients and for prime variables, which takes a lot of space $s_k = x_k - x_{k-1}, y_k = \nabla f(x_k) - \nabla f(x_{k-1})$. The main approach, in this case, is to create matrix B with dot products between s_k, y_k . It enables to decrease the amount of Reduce operations from m per step to 3 per step.

$$
\begin{aligned}
&\mathrm{q} = \nabla f(x_k)\\
&\mathrm{for}\ \ \mathrm{i} = 1, \ \ldots, 2\mathrm{m}{+}1\\
&\qquad \delta_i = (i \geq 2m)?0 : -1\\
&\\
&\mathrm{for}\ \ \mathrm{i} = \mathrm{k}-1, \ldots, \ \mathrm{k}{-}\mathrm{m}\\
&\qquad j = i - (k - m) + 1\\
&\qquad a_i = \frac{s_i^T q}{y_i^T s_i} = \frac{\sum_{l=1}^{2m+1} \delta_l B_{l,j}}{B_{j,m+j}}\\
&\\
&\mathrm{for}\ \ \mathrm{i} = 1, \ \ldots, 2\mathrm{m}{+}1\\
&\qquad \delta_i = \frac{B_{m,2m}}{B_{2m,2m}} \delta_i\\
&\\
&\mathrm{for}\ \ \mathrm{i}{=}\mathrm{k}{-}\mathrm{m}, \ldots, \mathrm{k}{-}1\\
&\qquad j = i - (k - m) + 1\\
&\qquad b = \frac{y_i^T t}{y_i^T s_i} = \frac{\sum_{l=1}^{2m+1} \delta_l B_{m+j,l}}{B_{j,m+j}}\\
&\qquad \delta_i = \delta_i + (a_i - b)\\
&\\
&t = \textstyle\sum_i^{2m+1} \delta_i b_i\\
&x_{k+1} = x_k - \alpha_k t = x_k - \alpha_k H_{inv_k} \nabla f(x_k)
\end{aligned}
$$

Figure 1. pseudocode of VF-LBFGS

AdaGrad

AdaGrad implies adaptive estimation of the direction probability implying previous direction squares. It allows choosing rare directions more frequently, than frequent ones, which enables to walk through the valleys and local minimums in estimated function. $G_k = G_k + I * (\nabla_w J(w_k), \nabla_w J(w_k))$ $\delta w_k = -\eta (diag(G_k) + \epsilon I)^{-1/2} \nabla_w J(w_k) \quad 0 < \epsilon \ll 1$ The main drawbacks are storing all diagonal values of such matrix, which consume a lot of memory and computing inverse square root for such matrix. We used the same technique as in L-BFGS – introducing a shared state R, which consists of the dot-products of the bias vectors and their gradients, this matrix is used to eliminate the computations in such case we could decrease the amount of Reduce operations from 2m+1 per step to 5 per step.

$\nabla_w J(w_k) = \sum_0^m \delta_i r_i$
for i = 1, .. m
$G(i,i) = \sum_0^m \delta_m r(m,i)$
for i = 1, ... m
for j = 1... m
$R(i,j) = (r_i, r_j)$
$\Delta w_k = -\eta(\sum_0^m \delta_m r_{ij}(r_i, e_j) + eI)^{-1/2} \nabla_w J(w_k)$
if $\sum_0^m \delta_m G(m,i) > min((r_i, e_j))$
$remove\ r_0$
$r_m := \nabla_w J(w_k)$
endif
$recompute\ R(i,j)$
$w_{k+1} = w_k + \Delta w_k$

Figure 2. Pseudocode of VF-AdaGrad

4. Computational experiments

To test the applicability of such heuristic the number of computational experiments was performed. We choose three different NNs: LSTM, VGG-16, and 3-layer MLP and different problems, like time-series prediction on SILSO[3] data, image classification on the CIFAR-10 dataset, and learning Boolean functions with MLP. All training procedures were performed in the batched way, where a numbers of examples in such batch are different for different problems(e.g. from 16 examples on learning Boolean functions in MLP, to 128 in time-series prediction with LSTM). To measure weak scaling the ratio of the dataset were changed from 20% to 50%. fIn LSTM training the truncated backpropagation through time method was used with parameters(8,16).

All computations are performed on the Microsoft™ Azure© cloud platform, with 16 computational HD12v2 nodes with Apache Spark 2.0.2, and SparkNet 0.1 framework with custom TensorFlow task. All custom code was written on Scala and standard JVM configuration.

To perform testing in a semi-automated way, the custom test system was designed. It allows to use a configuration file in JSON format to store the values and parameters of different optimizers, Also, in a separate section of this file, the topology of NN was stored. The system allows perform batch training on the different computational devices with the predefined number of methods and compare results on them. Experimental results showed that with VF heuristic allow to achieve with 2.2 times speedup with AdaGrad and 3.85 times with L-BFGS on learning Boolean functions MLP problem but such problem is so small, and not used in production, and improve weak scaling of L-BFGS more than 3 times in comparison with unmodified version. In image classification problem with VGG-16, the results are at the same level (x1.1 for AdaGrad and 1.38 for L-BFGS). Also, the weak scalability of both methods is raised up to 10% with L-BFGS and up-to 5% in AdaGrad in time-series prediction problem. On time-series predictions training with LSTM, we achieve total speedup was

3.28 times for L-BFGS and 1.35 for AdaGrad, so it's the best speedup for nets which are similar to the NNs which are used in production now.

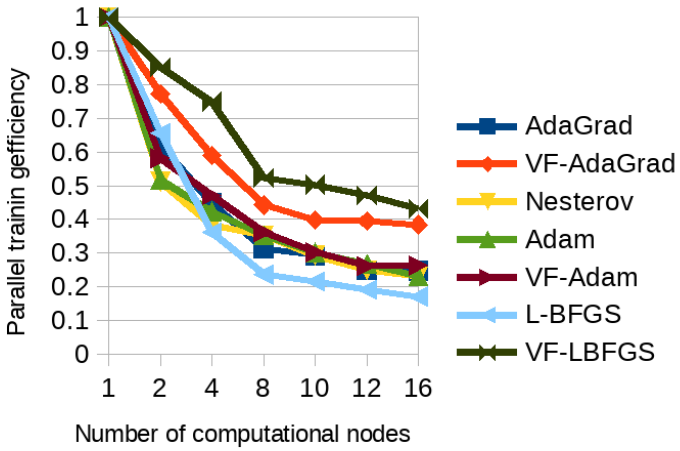

Figure 3: Efficiency of MLP training time parallelization

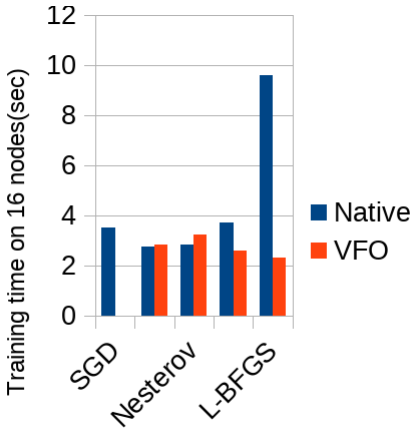

Figure 4: MLP training time comparison

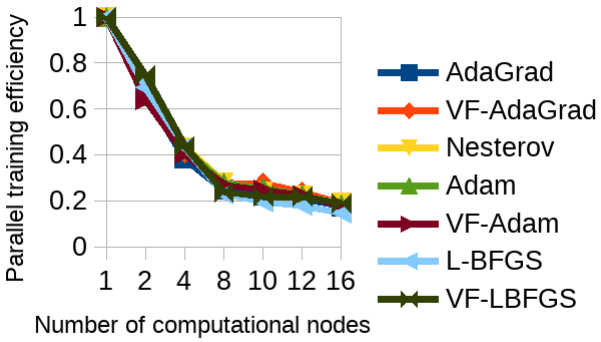

Figure 5: Efficiency VGG-16 training time parallelization

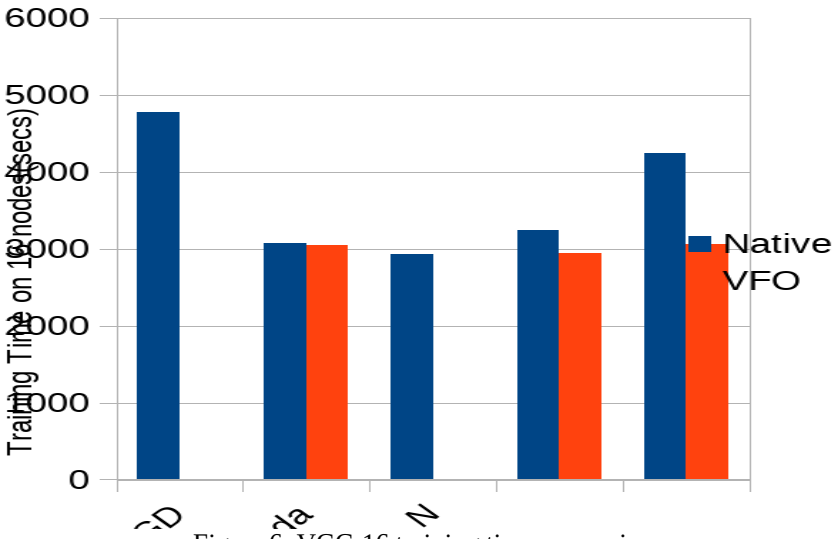

Figure 6: VGG-16 training time comparison

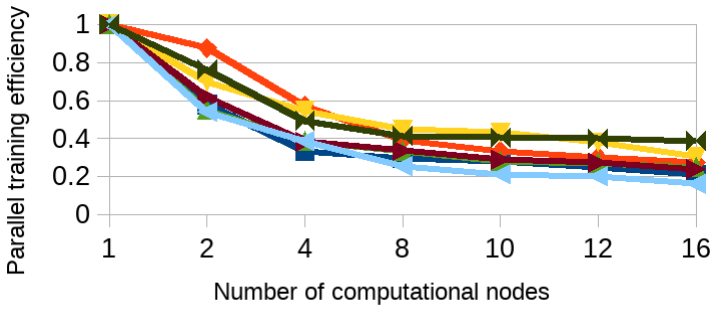

Figure 7: Efficiency of LSTM training parallelization

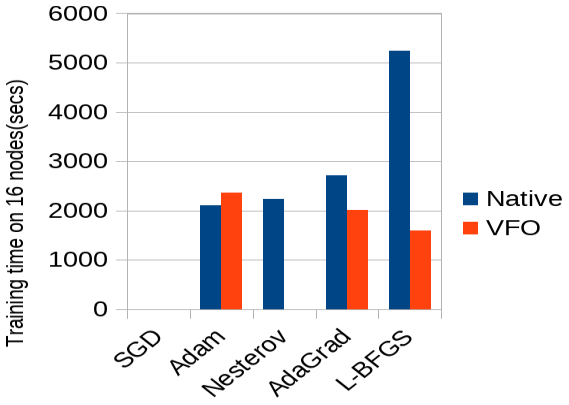

Figure 8: LSTM Training time comparison

6. Conclusion

The Vector-Free heuristic was chosen as heuristic reduce the number of non-linear and memory-bound operations. The popular optimizers for NNs training (Adam, AdaGrad, L-BFGS) were implemented on Apache Spark 2.0.2. Computational experiments demonstrated that Vector-Free heuristic improves the scalability of methods which contain a significant number of non-linear or memory-bound operations per step, like L-BFGS or AdaGrad. The maximum speedup for large nets was shown on the time series prediction problem with L-BFGS,(more than 3 times). The maximum improvement of weak-scaling Also, achieved on LSTM training. Also, the experiments showed that VF heuristic is not applicable to the methods with a significant number of linear operations, like Adam. So it proves that VF heuristic is not bound to the L-BFGS method and could be applied to

other methods, and could improve NNs training time. Effects on heuristics raise with task complexity so the major effects were observed on the LSTM training for the time-series prediction problem when the least on the MLP training for Boolean function learning. Test system which allows researcher to test the value of heuristic was implemented. The test-system doesn't bound to the Vector-Free heuristic testing. It allows the researcher developing a new optimization method to compare them on different problems for different types of NNs. The system is flexible, it allows to add custom NN architectures, custom problems for comparison. It makes easy to develop new kind of such methods. Also, it allows to add different computational devices and compare training time performance on them. The system distributes as a module on python and could be easily integrated with other python code.

References

[1] Chen Weizhu, Wang Zhenghao, Zhou Jingren. Large-scale L-BFGS Using MapReduce // Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1. NIPS'14. Cambridge, MA, USA: MIT Press, 2014. C. 1332–1340. URL:http://dl.acm.org/citation.cfm?id=2968826.2968975.

[2] Khamitov K.G., Popova N.N. Research of Vector-Free algorithms of gradient optimization for neural networks training on Apache Spark platform // Parallel Computational Technologies–XII international conference, PCT'2018, Rostov-on-Don, April 2–6 2018. Short papers and posters description. 2018. T. 3, No 1. c. 413.

[3] SILSO World Data Center. The International Sunspot Number //International Sunspot Number Monthly Bulletin and online catalogue. Royal Observatory of Belgium, avenue Circulaire 3, 1180 Brussels, Belgium, 2016. Dec. T. 1.