Skolkovo Institute of Science and Technology

MSc Data Science, 1st year, Numerical Linear Algebra

Project Proposal

Matrix Decomposition for Adaptive Optimization Regularization

Airapetyan Lusine

Chesakov Daniil

Glazov Vsevolod

Kovalev Evgeny

Matyushin Leonid

Team Leader: Matyushin Leonid

MOSCOW

2018

1 Background

Optimization is a very important discipline, especially for machine learning. So-called "Deep Learning Revolution" became possible because of the active development of optimization methods. The most simple first-order optimization method is a stochastic gradient descent. But its convergence rate could be greatly improved. There are 3 most common ways to accelerate optimization of a smooth function:

- Momentum. Example: Nesterov momentum. In this family of methods you somehow correct a step direction with the gradients of previous iterations.
- Variance reduction. Example: Importance sampling. In this family of methods you reduce variance of a stochastic gradient.
- Adaptive regularization Example: Adam. In this family of methods you choose parameters of optimization method (such as learning rate) adaptively.

In this project we are interested in the last one family of methods. First of all, let us introduce the notation and recall one of the most popular examples of such a technique — AdaGrad algorithm which is the basis of the proposed GGT method in the paper.

Suppose that we have a smooth loss function $f : \mathbb{R}^n \to \mathbb{R}$, and the following minimization problem:

$$f(x) \to \min_{x \in \mathcal{X}}$$

SGD and any methods focusing on its acceleration solve the problem iteratively: $x_0, x_1, \ldots, x_k, \ldots$

Denote $g_k \equiv \nabla f_x(x_k)$ and $\mathbf{G}_k = [g_k \; g_{k-1} \ldots \; g_1]$, where $\mathbf{G}_k \in \mathbb{R}^{n \times k}$.

In this notation the k-th step of optimization update of $\mathbf{x}_k$ in AdaGrad algorithm is represented in the following way:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\eta}{\sqrt{\mathbf{G}_k \mathbf{G}_k^T + \varepsilon \mathbf{I}}} \nabla f(\mathbf{x}_k),$$

where η is a "step size", ε is a small constant and $\mathbf{I}$ is a unit diagonal matrix.

The algorithm above is called a full-matrix AdaGrad. It is computationally impractical in high dimensions (most of modern neural networks have tens of millions of parameters) since it requires computation of the root of the matrix $\mathbf{G}_k \mathbf{G}_k^T \in \mathbb{R}^{n \times n}$. Thus we modify the update to obtain:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\eta}{\sqrt{\operatorname{diag}(\mathbf{G}_k \mathbf{G}_k^T) + \varepsilon \mathbf{I}}} \nabla f(\mathbf{x}_k)$$

Here both the inverse and root of $\operatorname{diag}(\mathbf{G}_k \mathbf{G}_k^T)$ can be computed in linear time. This algorithm is called diagonal AdaGrad. It is obvious that without this restriction on diagonal format we may have more freedom in regularization, so this construct the basis for GGT.

In the paper authors propose a GGT algorithm (mnemonic for $\mathbf{G}\mathbf{G}^T$), a practical solution of the full-matrix adaptive regularization. The main idea of this method is a beautiful way of computation of inverse square root of the low-rank second-moment matrix $\mathbf{G}\mathbf{G}^T$ of recent gradients. Authors use specific matrix decomposition. In some sense GGT generalize Adam method (Adam is mostly recovered by zeroing out the off-diagonal entries of $\mathbf{G}\mathbf{G}^T$).

2 Problem formulation

GGT uses the preconditioner from full-matrix AdaGrad, with gradient history attenuated exponentially and truncated to a window parameter r. Unlike the full-matrix AdaGrad algorithm, in GGT on each iteration we construct matrix $\mathbf{G} \in \mathbf{R}^{n \times r}$ as:

$$\mathbf{G}_k = [g_k g_{k-1} \ldots g_{k-r+1}], \quad \text{where } g_{k-t} = \beta_2^t \widetilde{\nabla} f\left(x_{k-t}\right), \text{ or } 0 \text{ if } t \geq k$$

where $\beta_2 \leq 1$ and $\widetilde{\nabla} f\left(x_{k-t}\right)$ is stochastic gradient.

According to this notation we write down the iterative step similar to full-matrix AdaGrad as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\eta}{\sqrt{\mathbf{G}_k \mathbf{G}_k^T + \varepsilon \mathbf{I}}} \widetilde{\nabla} f(\mathbf{x}_k)$$

The key difference from the AdaGrad method is that we don't compute matrix $(\mathbf{G}_k \mathbf{G}_k^T + \varepsilon \mathbf{I})^{1/2}$ inverse directly.

The fact that the preconditioning matrix is based on a small window of gradients implies that it has low rank. GGT exploits this fact by computing the inverse square root of the empirical covariance matrix indirectly. As a result, instead of inverting a full matrix in the dimension of parameters (n), a use the special matrix structure GGT allows to invert a matrix of the window-size dimension (r).

The inversion of the large low-rank matrix $\mathbf{G}\mathbf{G}^T \in \mathbb{R}^{n \times n}$ can be performed by diagonalizing the small matrix $\mathbf{G}^T\mathbf{G} \in \mathbb{R}^{r \times r}$. The complexity of each iteration is $\mathcal{O}\left(nr^2 + r^3\right)$ by time and $\mathcal{O}(nr)$ by memory.

3 Data

Besides reproducing the results on suggested data from the original paper we are planning to use the CIFAR-10 dataset as the main one to explore possible ways to improve the method and compare times for different implementations. The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes (6,000 of images for each class). The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

4 Related work

Optimization algorithms based on the adaptive regularization saw their rise with AdaGrad [1], and Adam [2] seem to be the most popular one among the others which may be used for deep learning models training. However, it has an issue — a need to take inverses and store large matrices, so the practical application of adaptive methods may be problematic [3]. Multiple approaches were used in order to overcome it, such as matrix sketching [4, 5] and Kronecker products [6], but the former appeared to be sensitive to noise, and the latter depended on the matrix structure, therefore wasn't so universal.

5 Scope

1. Implementing GGT method on Python.
2. Reproducing the results on the data used in the original paper and comparing it to the other methods.
3. Research on applying the GGT method to the new tasks and types of data, comparison with the competitive state-of-the-art methods.
4. Analysis of the suggested GGT algorithm and its efficiency comparing to the naive algorithms.
5. Applying the idea lying at the heart of GGT algorithm to other optimization methods if possible.
6. Exploring possible ways to improve the method.

6 Evaluation

In this project we are going to compare GGT (optimization technique that we mentioned above) and its implementation with some modifications with "vanilla" SGD and its AdaGrad and Adam modifications in terms of model training speed on standard deep learning benchmarks. Moreover, we are going to show the comparison of computing the inverse square root running-time of a full matrix using the naive method and using the special matrix structure of $\mathbf{GG}^T$.

7 Team Roles

- Airapetyan Lusine - Research work on applying the GGT method to the new tasks
- Chesakov Daniil - Data preprocessing and final report preparation
- Glazov Vsevolod - Reproducing result of original paper
- Kovalev Evgeny - Implementation of GGT algorithm on Python
- Matyushin Leonid - Analysis of the suggested GGT algorithm and exploring the possible ways to improve the GGT method

References

[1] J. Duchi, E. Hazan, Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[2] D. Kingma, J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[3] A. Wilson, R. Roelofs, M. Stern, N. Srebro, B. Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pp. 4151–4161, 2017.

[4] G. Krummenacher, B. McWilliams, Y. Kilcher, J. Buhmann, N. Meinshausen. Scalable adaptive stochastic optimization using random projections. In *Advances in Neural Information Processing Systems*, pp. 1750–1758, 2016.

[5] N. Mehta, A. Rendell, A. Varghese, C. Webers. Compadagrad: A compressed, complementary, computationally-efficient adaptive gradient method. *arXiv preprint arXiv:1609.03319*, 2016.

[6] V. Gupta, T. Koren, Y. Singer. Shampoo: Preconditioned stochastic tensor optimization. *arXiv preprint arXiv:1802.09568*, 2018.