SHAMPOO: PRECONDITIONED STOCHASTIC TENSOR OPTIMIZATION

Selikhanovych Daniil
Data science
Daniil.Selihanovich@skoltech.ru

Kiseleva Elizaveta
Data science
elizaveta.kiseleva@skoltech.ru

Noskova Elizaveta
Information Science and Technology
elizaveta.noskova@skoltech.ru

December 21, 2019

ABSTRACT

Preconditioned gradient methods are the most general and powerful tools in optimization, but it requires storing and manipulating prohibitively large matrices. We have studied a new structure-aware preconditioning algorithm, called Shampoo, for stochastic optimization over tensor spaces. Shampoo maintains a set of preconditioning matrices, each of which operates on a single dimension, contracting over the remaining dimensions. We provided experiments with state-of-the-art deep learning models show that Shampoo is capable of converging considerably faster than commonly used optimizers. Although it involves a more complex update rule, Shampoo's runtime per step is comparable to that of simple gradient methods such as SGD, AdaGrad, and Adam.

1 Introduction

Our project is to learn a new structure-aware preconditioning algorithm, called Shampoo, implement it and check the results of the article [1].

The purpose of the work is to study the dependence on many hyperparameters on three datasets: IRIS, MNIST, CIFAR10. And also compare two approaches to calculating a non-integer degree of a matrix: iterative and library, which uses SVD decomposition.

2 Problem description

So, in this project we are solving functional optimization problem

2.1 Common solutions

For a functional optimization problem nowadays exist a lot of different solutions, for example:

- Gradient descent: $x_{i+1} = x_i - \lambda_i \nabla f(x_i)$(first-order method)
- Newton's method: $x_{i+1} = x_i - \left(\nabla^2 f(x_i)\right)^{-1} \nabla f(x_i)$(second-order method)
- Gradient descent with preconditioner: $x_{i+1} = x_i - \lambda_i H_i \nabla f(x_i)$ $H_i - matrix$

The idea of preconditioned gradient descent originates from Online Mirror Descent in Online Convex Optimization:

$$w_{t+1} = \text{argmin}_w \Big\{ \eta \nabla f_t(w_t)^T w + \frac{1}{2} \parallel w - w_t \parallel^2_{H_t} \Big\} \Rightarrow$$

$$w_{t+1} = w_t - \eta H_t^{-1} \nabla f_t(w_t).$$

In the last equation appears preconditioner matrix H_t

2.2 What is Shampoo?

Shampoo is an alternative approach to preconditioning, which takes into account the structure of parameter space and saves memory consumption.

2.3 How to store preconditioners

A structure-oblivious full-matrix preconditioning scheme would flatten the parameter space into an mn-dimensional vector and employ preconditioning matrices H_t of size $mn \times mn$. In contrast, Shampoo maintains smaller left $L_t \in \mathbb{R}^{m \times m}$ and right $R_t \in \mathbb{R}^{n \times n}$ matrices containing second-moment information of the accumulated gradients. On each iteration, two preconditioning matrices are formed from L_t and R_t and multiply the gradient matrix from the left and right respectively. The amount of space Shampoo uses in the matrix case is $m^2 + n^2$ instead of $m^2 n^2$. Moreover, as the preconditioning involves matrix inversion (and often spectral decomposition), the amount of computation required to construct the left and right preconditioners is $O(m^3 + n^3)$, substantially lower than full-matrix methods which require $O(m^3 n^3)$.

2.4 Shampoo algorithm for matrices

Alg $Initialize : W_1 = 0_{m \times n}; L_0 = \epsilon I_m; R_0 = \epsilon I_n$
for $t \leftarrow 1$ *to* T **do**
  $Receive\ loss\ function\ f_t : \mathbb{R}^{m \times n} \to \mathbb{R}$
  $Compute\ gradient\ G_t = f_t(W_t) \{G_t \in \mathbb{R}^{m \times n}\}$
  $Update\ preconditioners :$

$$L_t = L_{t-1} + G_t G_t^T$$
$$R_t = R_{t-1} + G_t^T G_t$$

  $Update\ parameters : W_{t+1} = W_t - \eta L_t^{-1/4} G_t R_t^{-1/4}$

ϵ and η, responsible for initialization and learning rate respectively, are hyperparameters of our algorithm.

3 Experiments

We decided to check the results of the article and train the CNN image classifier using Shampoo as the base optimizer.

3.1 Datasets

- Iris: The dataset consists of 150 examples of 3 different types of 3 different types of irises' (Setosa, Versicolour, and Virginica) with 4 features
- MNIST: The database of handwritten digits, it has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image.
- CIFAR-10: The dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

3.2 Models

- Iris: Dense/ReLU + Dense/ReLU + Dense
- MNIST: Conv2D/ReLU + Conv2D/ReLU + Flatten + Dense/Softmax, $200k$ parameters
- CIFAR10: Conv2D/ReLU/MaxPooling + Conv2D/ReLU + Flatten + Dense/ReLU + Dense/Softmax, $120k$ parameters

3.3 Implementetion

We implemented Shampoo in its general tensor form in Python as a new TensorFlow optimizer. Our implementation follows almost verbatim the pseudocode shown in Algorithm. We used the built-in $tensordot$ operation to implement tensor contractions and tensor-matrix products. Matrix powers were computed simply by constructing a singular value decomposition (SVD) and then taking the powers of the singular values. These operations are fully supported in TensorFlow.

We can compute $L_T^{-1/4}$ and $R_T^{-1/4}$ using

- SVD
- stable iterations for the matrix square root [2]:

$$Y_0 = A - \text{PSD-matrix}, Z_0 = I,$$
$$k = 0, 1, \ldots \Rightarrow \begin{cases} Y_{k+1} = \frac{1}{2} Y_k (3I - Z_k Y_k) \\ Z_{k+1} = \frac{1}{2} (3I - Z_k Y_k) Z_k \end{cases}$$

Sufficient condition for convergence: $\text{diag} \parallel (A - I)^2 \parallel < 1 \Rightarrow$ quadratic convergence of Y_k to $A^{1/2}$ and Z_k to $A^{-1/2}$.

4 Results

4.1 First results

Firslt, we start on training our Shampoo optimizer on small Iris dataset to check that our code is working and that the written optimizer really minimizes the loss function. See the results on Figure 1. It displays loss and accuracy on train dataset.

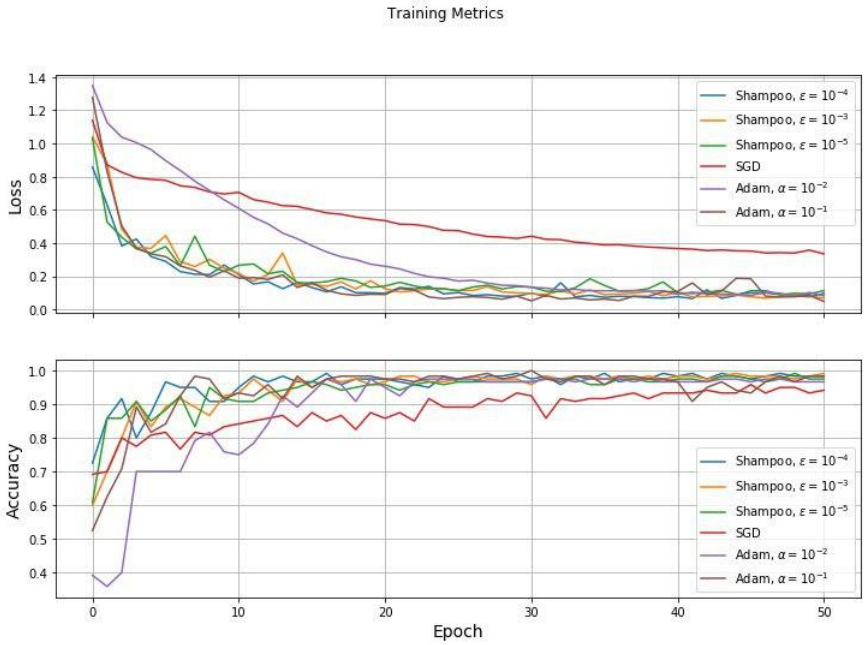


Figure 1: First training metrics

4.2 Selection of parameters for MNIST classification

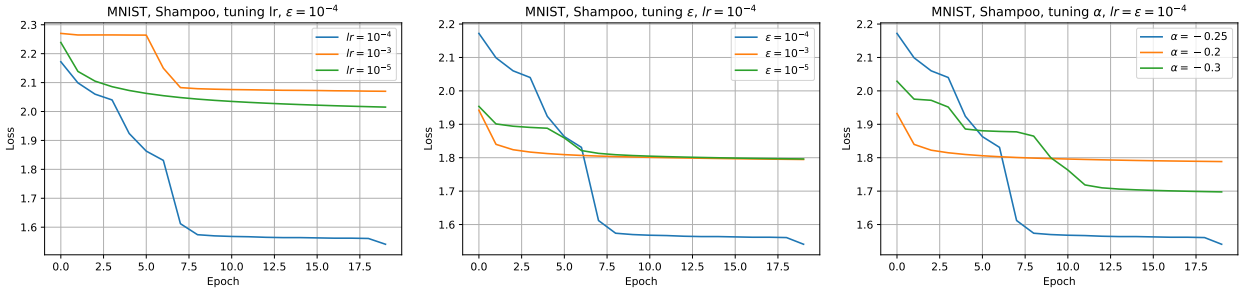
Firstly, we did search for different Shampoo hyperparameters - η, ε. Also we implement our Shampoo class with additional parameter $\alpha < 0$, which is responsible for exponent to which preconditioners L_t and R_t are raised in updating parameters rule:

$$W_{t+1} = W_t - \eta L_t^{\alpha} G_t R_t^{\alpha}.$$

Table 1: Accuracy on test, MNIST

Dataset	Shampoo	Adam	AdaGrad	SGD
MNIST	88	95	94	93
CIFAR10	91	93	91	-

Authors used in their paper value $\alpha = -0.25$ and for MNIST dataset it really was the best choise. See Figure 2.

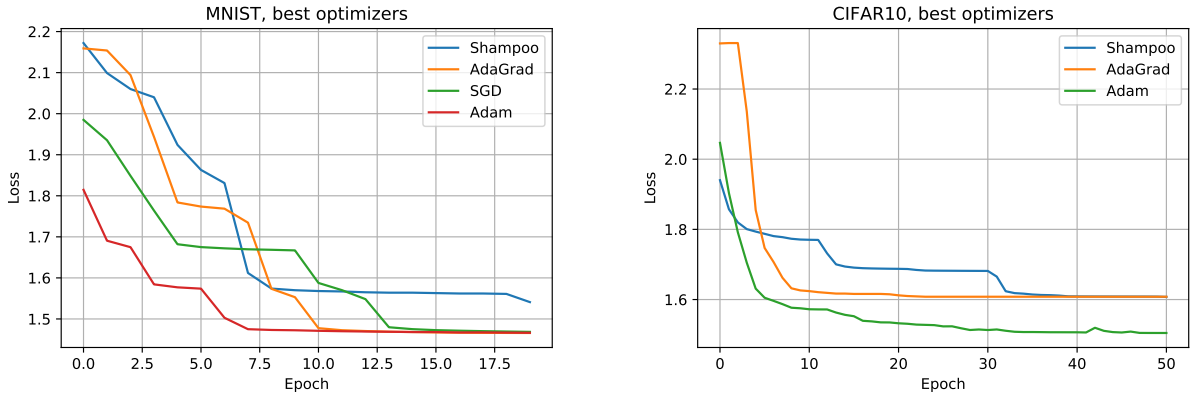


Tuning η Tuning ε Tuning α

Figure 2: Selection of optimal Shampoo optimizer parameters for MNIST classifier training

4.3 Results for best parameters

Unfortunately, as a result, on large datasets MNIST/CIFAR10, the Shampoo optimizer turned out to be not optimal on the considered set of hyperparameters. It is also worth noting that the running time of hand-written Shampoo optimizer was about one and a half times longer than standard optimizers. See Figure 3



MNIST CIFAR10

Figure 3: Best optimizers

4.4 Results on test

We provide the accuracy results of our neural networks in a test set for the best selection of parameters. It is interesting to note that Shampoo algorithm, like AdaGrad from the same preconditioner algorithm family, seems to have found the same local minimum on CIFAR data and has similar results on the test. See Table 1.

4.5 Conclusions

- Shampoo algorithm has many different hyperparameters to be selected.

- An implementation with iterative square root search in practice does not work well: it is slower than SVD and it is difficult to guess the number of nesessary iterations — small numbers of iterations lead to unstability of calculation.
- We did not get the desired Shampoo performance on these architectures (Adam, SGD and AdaGrad works better). The authors claim that on large networks such as ResNet and Inception the performance is higher. We believe that the Shampoo advantage can indeed be observed on neural network architectures with a large number of parameters. We tried to conduct a similar study on Google Colab using the VGG with 33M parameters, but our estimates showed that network training for one set of hyperparameters takes about 12 hours and we did not manage to do this part of the work.
- We can use the diagonal version of the Shampoo for tensor dimensions with the following properties: preconditioners are difficult to store in memory, there are difficulties in SVD calculations.

5 Contribution of team member

All members of team worked parallel on experiments both using our local computers or Google Colab. That's why some of the graphs were collected in parts from different $ipynb$ notebooks. On our Github project the names of all $ipynb$ notebooks display the author responsible for the corresponding experiment in this part of our work.

- Selikhanovych Daniil: Compilation the pipeline of work, numerical experiments, compilation of a report and presentation
- Kiseleva Elizaveta: Numerical experiments, compilation of a report and presentation
- Noskova Elizaveta: Numerical experiments, compilation of a report and presentation

6 Acknowledgment

We thank Daniil Merkulov for providing the idea for the project.

References

[1] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. *arXiv preprint arXiv:1802.09568*, 2018.

[2] Nicholas J Higham. Stable iterations for the matrix square root. *Numerical Algorithms*, 15(2):227–242, 1997.

Code, presentation and this report may be found at

`https://github.com/Daniil-Selikhanovych/Shampoo_optimizer`