Shampoo Optimizer

December 15, 2024

1 Introduction

This report explores the implementation of the Shampoo Optimizer which is a second-order-like optimizer that has shown good results in the training of deep neural networks like LLMs. The theory behind the optimizer is described, followed by a practical implementation in Python, using the JAX and Optax libraries. Subsequently, the performance of Shampoo is compared to popular existing optimizers: ADAM, SGD and RMSProp.

2 How does Shampoo work?

Shampoo is a second-order optimization method which means it factors in information on the curvature of the loss function to minimize it. Accounting for the curvature of the loss function generally reduces the number of iterations to convergence significantly. Typically, this curvature information is captured by the second derivative Hessian matrix but the Hessian is rather expensive to compute, especially as the number of dimensions increases. As such, Shampoo approximates the Hessian using much fewer computational resources, allowing for the benefit of faster convergence while minimizing the disadvantage of slow computation.

2.1 The Shampoo Algorithm

Initialize $W_1 = 0_{m\times n}; L_0 = \epsilon I_m; R_0 = \epsilon I_n$

1. Loss Function and Gradient:

$$\text{Receive loss function} f_t : \mathbb{R}^{m\times n} \to \mathbb{R}$$

$$\text{Compute gradient } G_t = \nabla f_t(W_t) \quad \{G_t \in \mathbb{R}^{m\times n}\}$$

2. Preconditioners Update:

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

3. Weight vector update:

$$W_{t+1} = W_t - \eta L_t^{-\frac{1}{4}} G_t R_t^{-\frac{1}{4}}$$

Where:

- L_t and R_t are the t^{th} iteration left and right preconditioners respectively.
- η is the learning rate.
- W_t is the t^{th} iteration weight vector.
- G_t is the t^{th} iteration gradient vector.

Repeat 1-3 for a set number of iterations.
Note: The Hessian is approximated by the Kronecker products of L and R although this Kronecker product is not directly used in the update rule. The definition of the Kronecker is shown below:

If

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix},$$

then

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & a_{12}b_{11} & a_{12}b_{12} \\ a_{11}b_{21} & a_{11}b_{22} & a_{12}b_{21} & a_{12}b_{22} \\ a_{21}b_{11} & a_{21}b_{12} & a_{22}b_{11} & a_{22}b_{12} \\ a_{21}b_{21} & a_{21}b_{22} & a_{22}b_{21} & a_{22}b_{22} \end{bmatrix}.$$

Note: The sizes of A and B do not need to match like in regular matrix multiplication for Kronecker products in general, but for the Shampoo Optimizer the sizes are chosen to match the gradient matrix.

2.2 Understanding the Algorithm

As seen in the algorithm, preconditioning is done using 2 matrices, L and R to pre-multiply and post-multiply the gradient matrix respectively. L contains information on the covariance between the rows of G by adding $G_t G_t^T$ in each iteration, approximating how much the loss changes in each row direction. R contains information on the covariance between the columns of G by adding $G_t^T G_t$ in each iteration, approximating how much the loss changes in each column direction. This allows L and R to account for curvature information along the rows and columns of G respectively. Subsequently, pre- and post-multiplying G conditions it in a similar way to pre-multiplying by the inverse of the Hessian, conditioning G for improved convergence.

3 Advantages of Shampoo

Contrast Shampoo's update rule with a typical second-order update rule:

$$W_{t+1} = W_t - \eta P_t^{-1} G_t$$

Where:

- P_t is the t^{th} preconditioning matrix.

One advantage Shampoo has over a typical second order method is in the use of two matrices L and R in preconditioning the gradient. Both these matrices are typically much smaller in size than the Hessian or other preconditioning methods that use a single matrix. The smaller matrices can be inverted much more quickly which allows the update to be done much faster than when using a single large preconditioning matrix like the Hessian. Furthermore, storing 2 smaller matrices also occupies less memory than when storing a single large matrix.

Shampoo's preconditioners L and R are also adaptive, taking into account the scale of the gradient in each update. This means that there is much less need for fine-tuning of the learning rate since the preconditioners can adjust the magnitude of the updates dynamically with each step. This is an advantage over methods that require fine-tuning of the learning rate, like SGD or L-BFGS.

Lastly, due to the Kronecker decomposition used in the update, parallel computing can be leveraged to speed up the algorithm execution. This is because the update and inversion of the L and R matrices can be done simultaneously on different CPU/GPU cores. This parallel computing cannot be leveraged in algorithms that only update one parameter or have update rules involving interactions between the parameters. Examples of these would be SGD and L-BFGS respectively.

4 Disadvantages of Shampoo

Firstly, on smaller problems, Shampoo may be outperformed by simpler methods like SGD or ADAM. This is due to the second order information on curvature being taken into account in the Shampoo algorithm. If the second order information is relatively constant or not significant, updating the preconditioners in each iteration would lead to unnecessary additional computations, which would slow down convergence compared to simpler methods.

Next, Shampoo, being a second order method, requires more computational resources. This is because it stores, updates and inverts the preconditioning matrices which take up additional computational resources compared to first order methods like SGD or ADAM which only store first order information. This could possibly lead to issues with lack of computational resources when dealing with extremely large datasets.

5 Implementation in Python + JAX

Now we implement the Shampoo Optimizer in Python. We take advantage of Object-Oriented Programming in Python to implement the Shampoo Optimizer by creating a Shampoo class. An instance of the Shampoo class is initialized with the learning rate and ϵ (shown in the algorithm described earlier) as well as the shape of the gradient matrix, which is used in reshaping operations to ensure matrix multiplications operate without errors and to determine the shapes of the L and R preconditionining matrices.

Our implementation of the Shampoo class contains 3 methods. The first method takes an input of a symmetric square matrix M and outputs the $-\frac{1}{4}$ power of that matrix, which will be used on the L and R matrices as shown in the update rule. This accomplished by diagonalization, as shown below:

$$M = Q\Lambda Q^T$$

Q is the matrix of eigenvectors and Λ is the diagonal matrix containing the eigenvalues.

$$\Rightarrow M^{-1/4} = Q\Lambda^{-1/4}Q^T$$

The second method takes an input of a gradient vector and updates L and R according to step 2 of the algorithm in **2.1**. The last method takes inputs of the current weight vector and a gradient function and performs an update of the weight vector as shown in step 3 of the algorithm in **2.1**. The main use of the JAX library in our implementation comes from using it to find the gradient of the loss function.

6 Shampoo's performance vs. other optimizers

Now, we compare Shampoo's performance with SGD, ADAM and RMSProp. We perform 3 different optimization tasks and observe how quickly convergence is attained using the 4 optimizers. For all tasks, we optimize the functions with the same initial point for all 4 optimizers. To avoid having to write new functions for SGD, ADAM and RMSProp, we use the Optax library which has existing functions for those optimizers. Optax is an optimization library built for JAX.

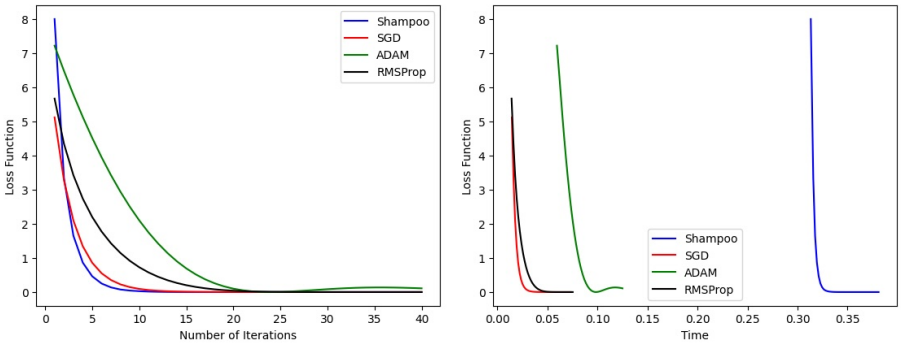
6.1 Task 1: Simple Convex Optimization

For this task, we find the minimum point the function:

$$f(x_1, x_2) = x_1^2 + x_2^2$$

This is a very simple function which would theoretically illustrate the benefits of using a simpler optimization method like SGD for smaller problems. It is obvious that the minimum point is when $(x_1, x_2) = (0, 0)$. Here are the results:

Figure 1: Loss Function vs Iterations/Time



As seen in Figure 1, Shampoo takes fewer iterations to converge compared to the others but is much slower in terms of time. This is a well-conditioned problem which results in the additional computations done to approximate the Hessian and preconditioning the gradient being less significant in improving each update. As a result, the additional computation time of each iteration outweighs the improvement in convergence per iteration, causing the overall time taken to be slower than with first-order methods like SGD, ADAM and RMSProp.

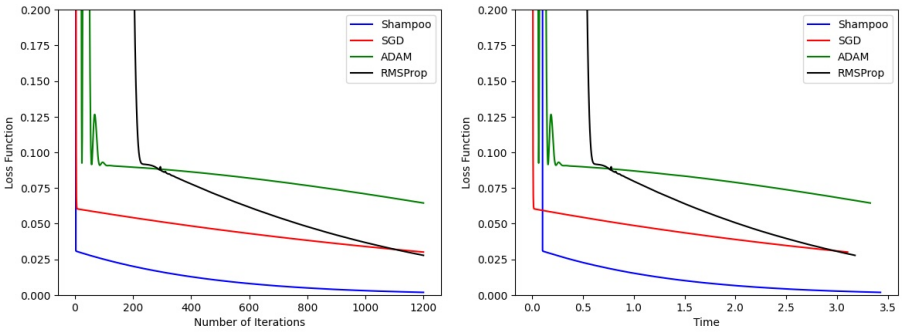
6.2 Task 2: The Rosenbrock Function

The Rosenbrock function is much more complex and non-convex, often having slow convergence when using methods like SGD due to the narrow valley where the minimum point is found. Second-order methods such as L-BFGS typically show significant improvements due to accounting for curvature information. As such, it is expected that Shampoo will outperform SGD, ADAM and RMSProp which are first-order. We use the Rosenbrock function with dimension $p = 2$ as shown below:

$$F(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

Here are the results:

Figure 2: Loss Function vs Iterations/Time



As seen in Figure 2, Shampoo converges much more quickly in terms of both time and number of iterations compared to SGD, ADAM and RMSProp. This is expected since the Rosenbrock function has a very ill-conditioned Hessian which first-order methods cannot account for, and hence struggle with optimizing. In contrast, Shampoo is able to converge more quickly as it factors in second-order information on the curvature of the function.

6.3 Task 3: Linear Regression

Lastly, we minimize the loss function in multiple linear regression with 2 independent variables. The problem is as follows:

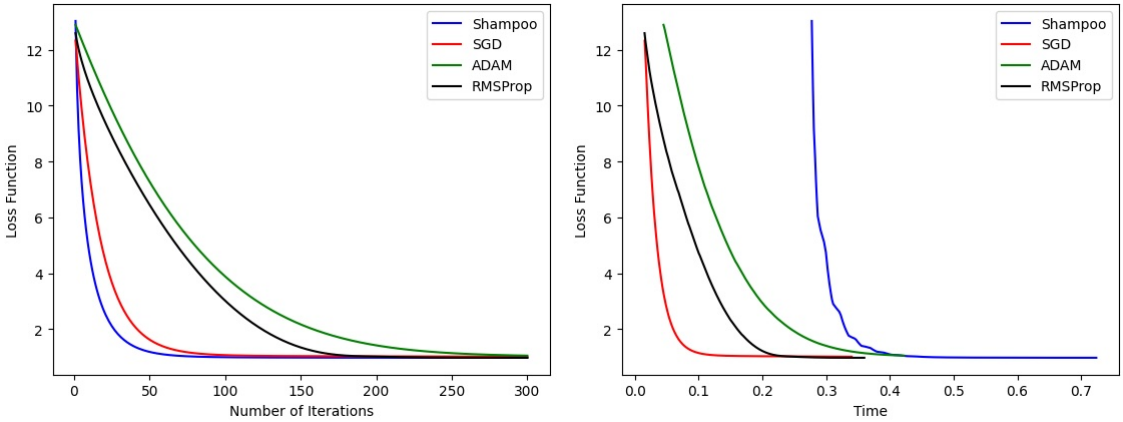
$$Y = X\beta + \epsilon$$

where:

- $\beta = [\beta_0, \beta_1, \beta_2]^T$ and β_0 corresponds to the intercept term
- For each sample, $X = [1, X_1, X_2]$ where X_1, X_2 are the independent variables
- ϵ is the error term $\sim N(0, 1)$

This is closer to a real-world problem which would illustrate the possible practical impact of using Shampoo. Here are the results:

Figure 3: Loss Function vs Iterations/Time



In Figure 3, we see similar results to Figure 1. Shampoo converges in fewer iterations compared to the others, but is much slower in terms of time. Again, this is a well-conditioned problem which means the second-order curvature information that Shampooo accounts for, does not significantly improve the update compared to first-order methods. As such, the faster computation time of the first-order methods allows them to converge more quickly in terms of time despite taking more iterations than Shampoo.

6.4 Overall results of Optimization Tasks

Overall, we can deduce that Shampoo shines in ill-conditioned problems due to its ability to account for curvature information as a second-order method. The practical implications of this would be in training deep-learning models since loss surfaces in those optimization problems are typically ill-conditioned. As such, it could have significant improvements in the speed of training deep-learning networks compared to existing popular methods like SGD, ADAM and RMSProp.

7 Conclusion

In this paper, we gave an overview of how Shampoo works and its advantages and disadvantages over other popular optimizers. We then implemented the Shampoo optimizer in Python + JAX + Optax and performed optimization in 3 separate tasks, comparing the results with other optimizers SGD, ADAM and RMSProp. We observed that Shampoo performed better in an ill-conditioned optimization problem than SGD, ADAM and RMSProp but performed worse in well-conditioned problems. Following that, we concluded that Shampoo could speed up the training of deep neural networks as they are typically ill-conditioned problems. This would be extremely useful since deep neural networks like LLMs are widely used today.

References

[GKS18] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. https://arxiv.org/abs/1802.09568, 2018. Accessed: 2024-11-15.

[MT] PhD Marco Taboga. Kronecker product. https://www.statlect.com/matrix-algebra/Kronecker-product. Accessed: 2024-10-23.

[Opt] Optax documentation. https://optax.readthedocs.io/en/latest/. Accessed: 2024-10-23.

[SB13] S. Surjanovic and D. Bingham. Rosenbrock function. https://www.sfu.ca/~ssurjano/rosen.html, 2013. Accessed: 2024-11-15.