Investigations on Sharpness-Aware Minimization

Honam Wong, Linying Yao, Shiyi Huang

***Abstract*—In this report, we aim at gaining a better understanding of the popular optimization algorithm - sharpness-aware minimization (SAM), which is proposed to enhance the generalization ability of the model. We investigate how does the radius, metrics of the ball, and normalization affect its generalization ability, and connect this with existing theory.**

I. Introduction

Deep neural networks are powerful for various machine learning tasks. However, they frequently struggle with poor generalization, prompting recent extensive research into improving their generalization capabilities. It is widely believed that the sharpness of the loss surface and the generalization performance of neural networks are closely related. Especially, Sharpness-Aware Minimization (SAM) [1], a recently proposed optimization technique, improves the generalization performance by minimizing sharpness of loss landscape. The formula is as follows:

$$\min_x f^{\text{SAM}}(x) = \min_x \max_{\|\varepsilon\|_2 \leq \rho} f(x+\varepsilon). \tag{1}$$

where f is the vanilla loss function.

The key intuition behind SAM is to minimize the maximum value of the loss function within a small L^2 ball with radius ρ around the current parameter values, promoting the discovery of wider and flatter minima, which are known to generalize better than sharp minima. It can be derived that one iteration of SAM is a set of two-step update equations:

$$\begin{cases} y_t = x_t + \rho \frac{\nabla f(x_t)}{\|\nabla f(x_t)\|}, \\ x_{t+1} = x_t - \eta \nabla f(y_t) \end{cases} \tag{2}$$

In recent years, Sharpness-aware minimization (SAM) already has wide applications in many areas, such as computer vision[2] and natural language processing[3], it would be interesting to explore it in theory. However, many current theoretical work either assumes diminishing or very small radius ρ, or analyze SAM without normalization on $\nabla f(x_t)$ [4]. In this project, we conduct a comprehensive empirical study on understanding this algorithm. To be more specific, we aim to investigate the following research questions:

1) How does the choice of the ball radius ρ affect the optimization process and the generalization performance of SAM?
2) What is the impact of using different metrics (e.g., L_2, L_∞) of the ball in the SAM algorithm?
3) How does normalization in 2 influence the effectiveness of SAM?

By addressing these questions, we seek to gain a deeper understanding of the SAM optimization algorithm. Our findings may provide valuable insights for researchers and practitioners aiming to improve the generalization and robustness of deep neural networks across various domains and applications.

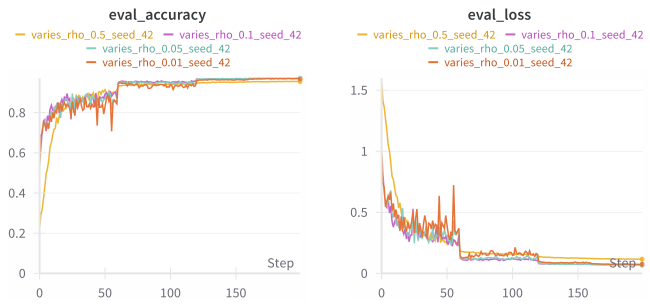
To conduct the experiments, we use https://github.com/davda54/sam as our base repository and implement most of our experiments on top of that. Unless otherwise specified, we follow their setting, we set initial learning rate as 0.1 and we use the step learning rate scheduler, where it maintains the initial rate for the first 30% of epochs, then reducing it to 20%, 4%, and 0.8% of the initial rate at 30%, 60%, and 80% of the total epochs, respectively. The momentum is set as 0.9, and weight decay is 0.0005.

II. Radius of the Ball

Recall that in eq. 1, the objective is to minimize the maximum value of the loss function within the small ball with radius ρ. We are curious about how the radius ρ affects the optimization performance of SAM. The original SAM paper [1] only considers varying rho for ablation studies, we investigate this further by plotting the loss and accuracy curve and observe its performance in more details.

A. Training details and results

We vary $\rho = \{0.01, 0.05, 0.1, 0.5\}$ and test their performance on CIFAR-10 with WideResNet [5], since they are the common combinations for testing optimization algorithm's performance [6][7]. We run the experiment for 200 steps each with three different random seeds 0, 10, 42 and observe similar phenomenon across all seeds. Our results are displayed in Fig. 1 (seed 42), other two random seeds perform similar results. Tab. I shows our results when varing different ρ values with three different random seeds.



(a) Accuracy (b) Loss

Fig. 1: Radius of the ball experiment with random seed 42.

	Accuracy		Loss	
ρ values	Mean	Std	Mean	Std
ρ = 0.5	0.9559	0.0007	0.1143	0.0014
ρ = 0.1	0.9710	0.0008	0.0695	0.0007
ρ = 0.05	0.9717	0.0007	0.0675	0.0011
ρ = 0.01	0.9689	0.0007	0.0729	0.0017

TABLE I: Model performance with accuracy and loss

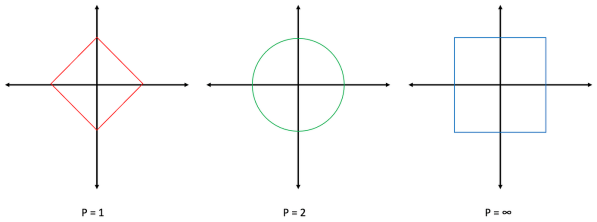


Fig. 2: Different norm L^p

B. Findings

We observe two interesting phenomenon:

First, SAM with relatively small ρ behaves similarly with some small fluctuations. This demonstrates the insensitivity of SAM over choice of hyperparameter ρ when it is relatively small, the reason is suggested to be normalization by the recent theoretical result [8]. We would also investigate into role of normalization in Section V. However, it is interesting to find that when ρ is very large i.e. 0.5 in the experiment, it would have lower convergence or even divergence. It is counter-intuitive since a larger ρ should imply it encourages a flatter minima around the point x, which is supposed to be improving generalization. Here is our hypothesis: recall that $y_t = x_t + \rho \frac{\nabla f(x_t)}{\|\nabla f(x_t)\|}$, we suspect that a larger choice of ρ would make y_t jump out of the local minimum during the optimization process, leading to slower convergence. More theoretical work would be required to explore this direction.

Most theory paper assumes vanishing ρ [4], this might also of interest to theoretically explore what will be the critical point of ρ for divergence.

In most of the runs, there are sudden drops in eval loss around 60 and 120 steps. We suspect this is related to the learning rate scheduler. By previous description, there are 200 steps, and learning rate reduces to 0.01 and 0.004 from the original 0.1 at 60 and 120 steps, which lowering learning rate in later steps is commonly known to accelerate convergence when it nears the minimum point.

III. Metrics of the ball (L^p norm)

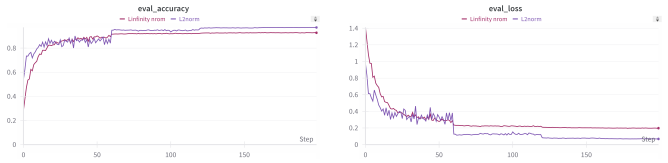
Since in SAM eq. 1 the ascent step is determined by the ball. Therefore, both of the geometry (for instance: fig 2) and radius of the ball might affect the performance of SAM. After discussing changing radius ρ, we are interested in how changing the metrics of the ball affect performance of SAM. The standard version uses L^2 metric, we try changing to L^∞ metrics and compare its performance with L^2.

The updated ascent formula would be

$$y_t = x_t + \rho \ \text{sign}(\nabla f(x_t))$$

A. Training details and results

The setup is similar to the Varies Rho experiment, but we fix $\rho = 0.05$ as suggested by the original SAM paper [1]. We modified the _grad_norm function to use $p = \infty$ instead of $p = 2$. This change calculates the gradient norm using the infinity norm, which affects how gradients are scaled during optimization.



(a) Accuracy (b) Loss

Fig. 3: Difference in $p = \infty$ and $p = 2$

B. Findings

With the experiments we find that using L^∞ norm is worse than L^2. We could not find any theoretical work discussing this. Informally speaking, we suspect the reason is for L^2 norm, $\frac{\nabla f(x_t)}{\|\nabla f(x_t)\|}$ provides more information on the geometry of the loss landscape, while $\text{sign}(\nabla f(x_t))$ can only give rough estimate of the landscape, this also worth further theoretical investigation.

IV. Adapting the Local Geometry

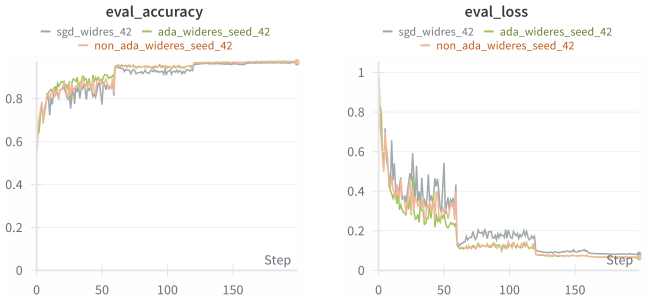
While SAM has shown promising results in improving generalization, it employs a fixed radius ball in the maximization step, which is sensitive to the model parameter re-scaling. To address this limitation, a variant called Adaptive Sharpness-Aware Minimization (ASAM)[9] has been proposed. ASAM adaptively adjusts the radius of the ball based on the curvature of the loss landscape, allowing the optimization to better navigate different regions and potentially find even flatter minima. To test its performance, since different architectures have different loss surfaces [10], we aim to compare its performance with SAM across several architectures.

A. Training details and results

In this section, we test the optimizers SAM, ASAM and SGD across several popular architecures: ResNet[11], WideResNet[5], PyramidNet[12] and VGG[13]. In detail, learning rate, momentum and weight decay are set to 0.1, 0.9 and 0.0005 respectively for SGD. For SAM, we set the hyper-parameter ρ to 0.05 while for ASAM, we set the hyper-parameter ρ to 2.0. Similarly, we test on three different random seeds. Fig. 4 shows one example of our plot results. Tab. II shows our results of the accuracy for different models.

TABLE II: Local geometry experiment results

Models	SGD	SAM	ASAM
ResNet	0.9611(0.0004)	0.9649(0.0010)	**0.9662**(0.0010)
VGG	0.9427	0.1000	0.1000
WRN	0.9669(0.0032)	0.9710(0.0005)	**0.9742**(0.0007)
PyramidNet	**0.8604**(0.0025)	0.8567(0.0056)	0.8192(0.0026)



(a) Accuracy (b) Loss

Fig. 4: ASAM, SAM and SGD comparison based on WideResNet with random seed 42.

B. Findings

We find that ASAM outperforms SAM and SAM outperforms SGD in ResNet and WideResNet as we expected. In PyramidNet, while SAM has close accuracy with SGD, ASAM performs worse. This might be caused by the initial hypermeter settings since we are using uniform hyper-parameters without tuning for each model. Another interpretation is that ASAM might not be better than SAM with PyramidNet without particular hyperparameter settings. Similar results in Table 1 of FisherSAM [14] also suggest that the difference of accuracy between SAM and ASAM is very small and SAM outperforms a little than ASAM with PyramidNet. Additionally, in VGG, we surprisingly find that both SAM and ASAM have probability to failure, which only achieves constant 10% accuracy over epochs. This interesting failure is similar to the issue addressed in the Table 2 of ASAM[9]. Since we have successful examples in the different settings while we haven't found any other paper address this issue, we guess this problem might be caused by difference in their loss surface, or some subtle undiscovered internal changes, such as the distortion of the image after channel shift.

There are also other variants of algorithms that adapt the local geometry, we put the additional discussion in Appendix A

V. Role of normalization

Recall that in SAM there is a set of two-step update equations, in which one of the steps involve normalization:

$$y_t = x_t + \rho \frac{\nabla f(x_t)}{\|\nabla f(x_t)\|}.$$

However, most of the existing theoretical research assumes no normalization is applied. We are curious about whether

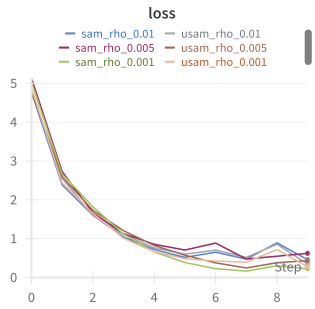


Fig. 5: Loss of USAM and SAM for $\rho = 0.001, 0.005, 0.01$

normalization is crucial, we introduce the unnormalized version of SAM (USAM) by changing the ascent step into

$$y_t = x_t + \rho \nabla f(x_t).$$

We replicate the motivating experiment of [8], we compare un-normalized SAM (USAM) with SAM in the overparametrized matrix sensing setting, which is a simplified setting to understand optimization behavior in overparametrized model [15], for details of the setting we can refer to the appendix B.

We vary $\rho = 0.001, 0.005, 0.01, 0.1$ to train SAM and USAM in 10 steps and observe their performance: For small $\rho = 0.001, 0.005, 0.01$, both of them manage to converge:

However, for $\rho = 0.1$, SAM manages to converge, but USAM diverges to extremely large value in $\rho = 0.1$. We list the value of loss function here for illustration:

steps	Loss of SAM	Loss of USAM
Step 1	4.714	4.872
Step 2	2.81	22.873
Step 3	1.748	70595465
Step 4	1.199	1.545e+65

TABLE III: SAM and USAM's loss

A. Findings

This demonstrates that normalization stabilizes SAM towards choice of ρ. We find that the theoretical result [8] also supports this claim, which shows in the toy case - single-neuron linear net $L(x, y) = l(x, y)$, USAM had $y_\infty^2 >> 0$ while SAM has $y_\infty^2 = o(1)$.

VI. Summary

In this report, we find that SAM is robust to choice of ρ when it is relatively small, but when it gets larger it exhibits worse performance. We also find that using L^∞ metric performs worse than using L^2 metric. Moreover, we find that the performance of Adaptive SAM and SAM might vary in different network model architectures while the overall performance of Adaptive SAM is better than SAM. In the last section, we compare unnormalized SAM and vanilla SAM in the simplified matrix sensing setting, which unveils the importance of normalization in SAM. All these empirical investigations are either well connected with theory, or might be interesting to explore further.

REFERENCES

[1] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," 2021.

[2] L.-H. Li and R. Tanone, "Compact convolutional transformer based on sharpness-aware minimization for image classification," in *2023 12th International Conference on Awareness Science and Technology (iCAST)*. IEEE, 2023, pp. 129–135.

[3] C. Na, S. V. Mehta, and E. Strubell, "Train flat, then compress: Sharpness-aware minimization learns more compressible models," *arXiv preprint arXiv:2205.12694*, 2022.

[4] D. Si and C. Yun, "Practical sharpness-aware minimization cannot converge all the way to optima," 2023.

[5] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[6] S. Cong and Y. Zhou, "A review of convolutional neural network architectures and their optimizations," *Artificial Intelligence Review*, vol. 56, no. 3, pp. 1905–1969, 2023.

[7] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[8] Y. Dai, K. Ahn, and S. Sra, "The crucial role of normalization in sharpness-aware minimization," 2023.

[9] J. Kwon, J. Kim, H. Park, and I. K. Choi, "Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5905–5914.

[10] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," 2018.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[12] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] M. Kim, D. Li, S. X. Hu, and T. M. Hospedales, "Fisher sam: Information geometry and sharpness aware minimisation," 2022.

[15] Y. Li, T. Ma, and H. Zhang, "Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations," 2019.

[16] T. Truong, H.-P. Nguyen, T. Pham, M.-T. Tran, M. Harandi, D. Phung, and T. Le, "RSAM: Learning on manifolds with riemannian sharpness-aware minimization," 2024. [Online]. Available: https://openreview.net/forum?id=u6Ux5OCGmW

APPENDIX

We use RTX4090 cards to run our experiments, the experiments are logged using wandb, and can be replicated by running the '.sh' files in the repository, details can be found on 'README.md'.

A. Additional Discussion on local geometry

To better adapt to the local geometry of the loss landscape, [14], [16] are proposed to utilize the information geometry of the parameter space, by replacing SAM's euclidean balls with ellipsoids induced by Fisher information. It would be interesting to investigate performance of these algorithms in different carefully designed loss landscape in future work, which might further inspire better optimization algorithm.

B. Overparametrized Matrix Sensing Setup

Our setup is very similar to the Appendix A of [8], as follows:

1) Generate the true matrix by sampling each entry of $U^* \in \mathbb{R}^{d\times r}$ independently from a standard Gaussian distribution and let $X^* = U^*(U^*)^T$.
2) Normalize each column of $U^\star$ to unit norm so that the spectral norm of $U^\star$ is close to one.
3) For every sensing matrix A_i $(i = 1, 2, \ldots, m)$, sample the entries of A_i independently from a standard Gaussian distribution. Then observe $b_i = \langle A_i, X^\star\rangle$.

In particular, for the experiments, we chose $r = 5$, $d = 100$, and $m = 5dr$.

We pick the largest learning rate for SGD to converge i.e. 0.5, which adheres with the practical learning rate selection.