OPT2020: 12th Annual Workshop on Optimization for Machine Learning

# Adaptive Hessian-free optimization for training neural networks

**Tetsuya Motokawa** S1921649@S.TSUKUBA.AC.JP
**Taro Tezuka** TEZUKA@SLIS.TSUKUBA.AC.JP
*University of Tsukuba, Ibaraki, Japan 305-0006*

## Abstract

Neural networks' weight parameters are usually updated using a linear optimization method based on stochastic gradient descent (SGD), but several limitations have been pointed out. For example, linear optimization converges slowly when the learning rate is not tuned correctly. Also, it requires many iterations, making parallelization difficult and less effective. The use of second-order and quasi-second-order optimization has been actively researched to overcome these limitations. One example is Hessian-free optimization (HF), which is considerably faster due to avoiding computation of the inverse of the Hessian matrix.

While it is prevalent to use adaptive parameter updates such as RMSProp, AdaGrad, or Adam for linear optimization, it has not been used for quasi-second-order optimization methods. This paper proposes a novel method, Adaptive HF, which adds an adaptive parameter update mechanism to Hessian-free optimization. We give a proof that guarantees the convergence of Adaptive HF. Experiments showed that Adaptive HF performs better than the original Hessian-free optimization in terms of loss and accuracy. Besides, while the original HF often failed to converge for a wide range of hyperparameter values, Adaptive HF successfully converged in most cases.

The main contributions of the paper are as follows.

- We created a new optimization algorithm for training neural networks by incorporating an adaptive mechanism to Hessian-free optimization.
- We proved a convergence property of the algorithm when the hyperparameters are set within certain ranges.
- We experimentally verified that the proposed method outperforms the original Hessian-free optimization.

## 1. Introduction

Popular optimization algorithms for neural networks such as Momentum SGD, Nesterov acceleration gradient, AdaGrad, RMSProp, and Adam are all based on the stochastic gradient descent (SGD) method, with different types of adaptive mechanisms added. A neural network's weight parameters are optimized using gradients, which are the first derivatives of a loss function. In the adaptive variants of SGD, parameters are updated by incorporating values of previous steps. These adaptive methods are all linear optimization and have several limitations. They require many iterations, making parallelization difficult and ineffective. Also, there are many hyperparameters, such as the learning rate, that are difficult to adjust. When training a neural network, training often fails if the hyperparameters are not carefully tuned. These limitations come from the fact that linear optimization uses only the first term in the Taylor expansion, and the remaining terms are ignored. There are many cases that second-order or quasi-second-order optimization is far more efficient than linear

optimization, for example, when the condition number of the Hessian matrix for the loss function is large. Since second-order and quasi-second-order methods converge in far fewer iterations than first-order methods, it can benefit from parallel processing within each iteration. Recently, many papers have proposed to use the second derivative to overcome the limitations of linear optimization [2, 10, 11]. However, these proposals focused on extending the original SGD and did not use any adaptive mechanism.

In this work, we propose a new algorithm that introduces adaptive parameter updates to quasi-second-order optimization. Such an approach has not been proposed before, possibly due to popular adaptive parameter update methods such as Adam being relatively new. The use of second-order or quasi-second-order optimization for training deep networks is even more recent [3]. One difficulty is that deep neural networks have a vast number of parameters, and second-order optimization requires intensive computation to obtain the inverse of the Hessian matrix. However, the recently proposed Hessian-free optimization, which is a type of of quasi-second-order optimization, resolves this problem by not explicitly computing the Hessian matrix but computing the Hessian-vector product [10]. In this paper, we demonstrate that Hessian-free optimization can be combined with adaptive parameter updating.[1] The benefit of introducing the adaptive mechanism comes from exploiting curvature information in its past trajectory, rather than looking only at the curvature around the current parameter value.

## 2. Related Works

### 2.1. Adaptive parameter optimization for training a neural network

Neural networks are usually trained using SGD or its adaptive variants. Momentum SGD [15, 17] and Nesterov Accelerate Gradient [13] are simple to implement and also theoretically well-founded, making them popular options. Methods that use past gradients are now widely used. Most popular among these methods are AdaGrad [5], RMSProp, and Adam [8]. In a high-dimensional parameter space, optimizers are likely to be caught at a saddle point, where the norm of the gradient is nearly zero. Experimental analyses suggest that adaptive linear optimization enables the optimizer to escape from saddle points.

Another way of moving away from the saddle point is to use the natural gradient ([1]) where the gradient is scaled using the Fisher information matrix. Several recent papers evaluate natural gradient descent for training a deep learning model ([6, 14, 16, 19]).

Recently, Yao et al. [18] proposed AdaHessian, an adaptive second-order optimization method that uses the diagonal elements of the Hessian matrix. Our approach is different from AdaHessian in that it relies on the Hessian-vector product rather than the Hessian matrix, which makes it quasi-second-order optimization. In contrast, our method does not explicitly compute the Hessian matrix. It largely reduces the cost of computation while benefitting from utilizing the curvature of the loss surface.

### 2.2. Hessian-free optimization

Hessian-free optimization is one example of the truncated-Newton method that has been used for numerical analysis. It computes the product of the gradient and the Hessian matrix efficiently. [12] used it for training a neural network. Since neural networks have many weights, the dimension of

1. Code is made available at: https://github.com/mtkwT/adaptive-hessian-free-optimization .

the Hessian matrix tends to be quite large. The cost of computation is prohibitive in terms of both time and space. Hessian-free optimization circumvents this problem by using the conjugate gradient method (CGM). A parallelized version of HF has also been proposed in [7].

## 3. Method

We propose adaptive Hessian-free optimization (Adaptive HF), which introduces adaptive parameter update mechanisms to Hessian-free optimization. More specifically, the method introduces the adaptive mechanism used in Adam [8].

### 3.1. Adaptive HF

Adam (adaptive moment) is an adaptive linear optimization method that exploits past gradients similarly to the momentum algorithm. The main difference is that Adam is based more on statistical consideration. It computes the first and second moments from past gradients and uses them to standardize the current gradient. Although the theoretical analyses on Adam are still ongoing [4], it is widely used to its effectiveness in practice.

Adam standardizes the gradient using the weighted first and second moments of past gradients. Updates in Adam can be expressed as follows. $m_t$ and $v_t$ represent the first-order and second-order weighted moments, respectively. Like in the momentum method, $a_t$ represents the acceleration at step $t$, corresponding to the gradient in the case of linear optimization. $\beta_1$ and $\beta_2$ are hyperparameters.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t & \qquad \hat{v} &= \frac{v_t}{1-\beta_2^t} \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 & & \\ \hat{m} &= \frac{m_t}{1-\beta_1^t} & \qquad \theta_{t+1} &= \theta_t + \frac{\alpha \hat{m}}{\sqrt{\hat{v}+\mu}} \end{aligned} \tag{1}$$

Based on this mechanism, we propose to utilize the first and second moments of the Hessian-vector product. Our proposed algorithm of Adaptive HF is indicated in Algorithm 1. It introduces the mechanism of Adam [8] for updating the parameter vector $\theta_t$ using moments $m_t$ and $v_t$ of $\delta_t$, which constitutes the vector part of the Hessian-vector product $H(\theta_t)\delta_t$.

### 3.2. Convergence of Adaptive HF

We prove that Algorithm 1 converges under specific conditions regarding the loss function and the range of hyperparameters. In doing so, we introduce $L$-conjugate smoothness, a condition that is similar to Lipschitz continuity but defined using the conjugate gradient vector.

**Definition 1 ($L$-conjugate smoothness):**

A differential function $f$ is $L$-conjugate smooth if

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L}{2} \|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{2}$$

where is $\boldsymbol{g}(\boldsymbol{x})$ is the conjugate gradient vector at $\boldsymbol{x}$, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ for some $L > 0$. ■

**Algorithm 1** Adaptive Hessian-free optimization

Input: $\theta_0$, $\epsilon$, $\sigma$, $L$, $\beta_2 \in [0, 1)$, $\mu$, $\alpha$, $\lambda$
$\delta_0 \leftarrow 0$, $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 1$
$\beta_1 = \frac{\epsilon}{\epsilon+\sigma}$
**while** solution is not satisfactory **do**
  Select a mini-batch $S^{'} \subset S$ from the training set to compute the gradient
  $b \leftarrow -\nabla h(\theta_t)$ on $S^{'}$
  Select a mini-batch $S^{''} \subset S$ from the training set to compute the curvature
  Define $A \equiv (H(\theta_t) + \lambda I)\delta_t$ on $S^{''}$
  $\delta_t \leftarrow$ CG $(b, A)$ (CG: conjugate gradient)
  Update $\lambda$ using the Levenberg-Marquardt heuristics [12]
  Update $\alpha$ using Equation 3
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\delta_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)\delta_t^2$
  $\hat{m} \leftarrow \frac{m_t}{1-\beta_1^t}, \hat{v} \leftarrow \frac{v_t}{1-\beta_2^t}$
  $\theta_{t+1} \leftarrow \theta_t + \frac{\alpha \hat{m}}{\sqrt{\hat{v}+\mu}}$
  $t \leftarrow t + 1$
**end while**

**Theorem 1 (Convergence of Adaptive HF):**
Let $f : \mathcal{R}^d \rightarrow \mathcal{R}$ be $L$-conjugate smooth function, and let the norm of its conjugate gradient vector $\|\boldsymbol{g}_t\|$ does not increase for step $t$ beyond some natural number $\tau$. In other words, there exists a natural number $\tau$ and a real number $\sigma$ such that $\sigma = \sigma_t := \max_{k=1,\ldots,t} \|\boldsymbol{g}_k\|$ for all $t \geq \tau$. Assume also that $f$ has a minimum. In other words, $f(\boldsymbol{x}_*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$ for some $\boldsymbol{x}_*$.

Then for Algorithm 1, there exists a natural number $T$ and a sequence of step sizes $\{\alpha_{t'}\}_{t'=1,\ldots,T}$ such that $\|\boldsymbol{g}_{t'}\| \leq \epsilon$ for some $t' \leq T$, for all $\epsilon > 0$, $\beta_1 < \frac{\epsilon}{\epsilon+\sigma}$, and $\xi > \frac{\sigma^2\beta_1}{-\beta_1\sigma+\epsilon(1-\beta_1)}$. ■

A proof is provided in Appendix A. One option for hyperparameters that assures convergence is using

$$\beta_1 = \frac{\epsilon}{\epsilon + 2\sigma} \qquad \hat{\alpha}_t = \frac{4\left(\|\boldsymbol{g}_t\|^2(1-\beta_1) - \|\boldsymbol{g}_t\|(\beta_1 - \beta_1^t)\sigma\right)}{3L(1-\beta_1^t)^2\sigma}. \tag{3}$$

at step $t$. We used the above hyperparameter $\beta_1^t$ and step size $\hat{\alpha}_t$ in our implementation.

## 4. Experiments

As shown in Theorem 1, we need to set hyperparameters $L$ and $\sigma$ to assure convergence. Since they represent bounds for the shape of the loss landscape, the search range for these hyperparameters were determined by pre-running the algorithm and observing the dynamics of $\|\boldsymbol{g}(\theta_t)\|$. In Figure 1, the left graph indicates the dynamics of $\|\boldsymbol{g}(\theta_t)\|$ during pre-running. The right graph shows the dynamics of the step size during pre-running. Pre-running requires $L$ and $\sigma$ to determine the step size, so they were set using Bayesian optimization. More details are provided in Appendix B.

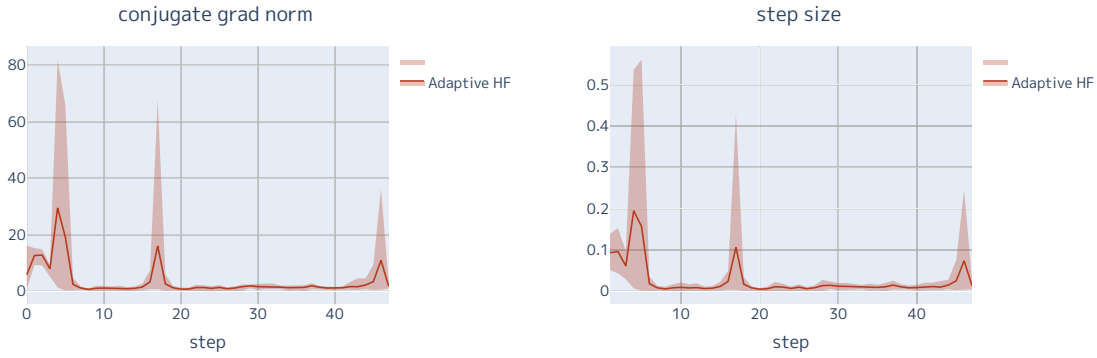


Figure 1: The norm of the conjugate gradient vector and the learning rate for different values of hyperparameters when using Adaptive HF (15 trials).

After setting up $L$ and $\sigma$, we compared the original HF and Adaptive HF using LeNet [9] and the Fashion-MNIST dataset. We optimized hyperparameters using Bayesian optimization and compared training loss and test accuracy (Figure 2). The results indicate that for many values of hyperparameters, the original HF totally fails to optimize, and therefore, the variances of the evaluation measures are much higher than Adaptive HF. We also observed that Adaptive HF converges to minima that show better generalization performance (test accuracy) on average.

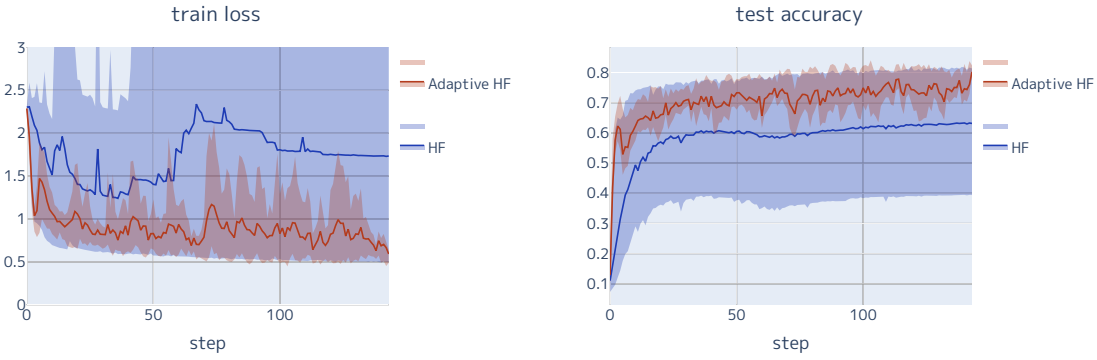


Figure 2: Train loss and test accuracy for different values of hyperparameters (70 trials).

## 5. Conclusion

We proposed Adaptive HF, which introduces an adaptive parameter update mechanism to Hessian-free optimization. We provided proof that assures the method to converge when used with hyperparameters within a specific range. Experiments showed that our method outperforms the original Hessian-free optimization in terms of loss and accuracy. Besides, our method was more stable than the original HF for a wide range of hyperparameters.

## References


[1] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10: 251–276, 1998. URL http://dx.doi.org/10.1162/089976698300017746.

[2] Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 557–565, 2017. URL http://proceedings.mlr.press/v70/botev17a.html.

[3] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pages 2933–2941, 2014. URL http://dl.acm.org/citation.cfm?id=2969033.2969154.

[4] Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. In *ICML Workshop on Modern Trends in Nonconvex Optimization for Machine Learning*, 2018. URL https://arxiv.org/abs/1807.06766.

[5] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

[6] Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker-factored eigenbasis. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, 2018. URL https://arxiv.org/abs/1806.03884.

[7] Xi He, Dheevatsa Mudigere, Mikhail Smelyanskiy, and Martin Takác. Distributed hessian-free optimization for deep neural network, 2016. URL https://arxiv.org/abs/1606.00511.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations 2014*, 2014. URL https://arxiv.org/abs/1412.6980.

[9] Yann LeCun, Leon Bottou, Yoshua, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278 – 2324. IEEE, 1994.

[10] James Martens. Deep learning via Hessian-free optimization. *Proceedings of the 27th International Conference on Machine Learning*, pages 735–742, 2010. URL https://dl.acm.org/citation.cfm?id=3104416.

[11] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. URL https://arxiv.org/abs/1503.05671.

[12] James Martens and Ilya Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural Networks: Tricks of the Trade*, 2012.

[13] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). *Dokl. akad. nauk Sssr*, 269:543–547, 1983.

[14] Razvan Pascanu and Yoshua. Revisiting natural gradient for deep networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. URL http://arxiv.org/abs/1301.3584.

[15] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12:145–151, 1999. URL http://dx.doi.org/10.1016/S0893-6080(98)00116-6.

[16] Yang Song, Jiaming Song, and Stefano Ermon. Accelerating natural gradient with higher-order invariance. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1803.01273.

[17] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1139–1147, 2013. URL http://proceedings.mlr.press/v28/sutskever13.html.

[18] Zhewei Yao, Amir Gholami, Sheng Shen, Kurt Keutzer, and Michael W. Mahoney. ADAHESSIAN: An adaptive second order optimizer for machine learning. https://arxiv.org/pdf/2006.00719.pdf, 2020.

[19] Guodong Zhang, Shengyang Sun, David Duvenaud, and Roger Grosse. Noisy natural gradient as variational inference. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL https://arxiv.org/abs/1712.02390.

## Appendix A. Proof

### A.1. Convergence of Adaptive HF

**Definition 1 ($L$-conjugate smoothness):**

A differential function $f$ is $L$-conjugate smooth if

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2 \tag{4}$$

where is $\boldsymbol{g}(\boldsymbol{x})$ is the conjugate gradient vector at $\boldsymbol{x}$, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ for some $L > 0$. ■

**Theorem 1 (Convergence of Adaptive HF):**
Let $f : \mathcal{R}^d \to \mathcal{R}$ be a $L$-conjugate smooth function, and let the norm of its conjugate gradient

vector $\|\boldsymbol{g}_t\|$ does not increase for step $t$ beyond some natural number $\tau$. In other words, there exists a natural number $\tau$ and a real number $\sigma$ such that $\sigma = \sigma_t := \max_{k=1,\dots,t} \|\boldsymbol{g}_k\|$ for all $t \geq \tau$. Assume also that $f$ has a minimum. In other words, $f(\boldsymbol{x}_*) = \min_{\boldsymbol{x}} f(\boldsymbol{x})$ for some $\boldsymbol{x}_*$.

Then for Algorithm 1, there exists a natural number $T$ and a sequence of step sizes $\{\alpha_{t'}\}_{t'=1,\dots,T}$ such that $\|\boldsymbol{g}_{t'}\| \leq \epsilon$ for some $t' \leq T$, for all $\epsilon > 0$, $\beta_1 < \frac{\epsilon}{\epsilon+\sigma}$, and $\xi > \frac{\sigma^2\beta_1}{-\beta_1\sigma+\epsilon(1-\beta_1)}$. ■

The proof follows that of [4] up till Equation 18.

**Proof:**
Let us assume to the contrary that $\|\boldsymbol{g}_t\| \geq \epsilon$ for any natural number $t$. We will prove that this assumption will lead to a contradiction. From $L$-conjugate smoothness of $f$,

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) + \langle \boldsymbol{g}_t, \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \rangle + \frac{L}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2. \tag{5}$$

By defining a diagonal matrix $\boldsymbol{D} := \left(V_t^{\frac{1}{2}} + \mathrm{diag}(\xi \mathbf{1}_d)\right)^{-1}$ and substituting the update rule of Algorithm 1 to $\boldsymbol{x}_{t+1} - \boldsymbol{x}_t$,

$$f(\boldsymbol{x}_{t+1}) \leq f(\boldsymbol{x}_t) - \eta_t \langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t \rangle + \frac{L\eta_t^2}{2}\|\boldsymbol{D}\boldsymbol{m}_t\|^2 \tag{6}$$

where $\eta_t > 0$ is a dummy step length. It can be rewritten as

$$f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) \leq \eta_t \left( -\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t \rangle + \frac{L\eta_t}{2}\|\boldsymbol{D}\boldsymbol{m}_t\|^2 \right). \tag{7}$$

Since the roots of the quadratic in the right-hand side are 0 and $\frac{2\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t \rangle}{L\|\boldsymbol{D}\boldsymbol{m}_t\|^2}$, the minimum is at their midpoint. We set a candidate step length $\alpha_t^*$ to the midpoint, $\frac{\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t \rangle}{L\|\boldsymbol{D}\boldsymbol{m}_t\|^2}$. It guarrantees decrease in the loss function,

$$f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) \leq -\frac{\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t \rangle^2}{2L\|\boldsymbol{D}\boldsymbol{m}_t\|^2}. \tag{8}$$

To upper bound the right-hand side of Equation 8, we bound the numerator and the denominator separately.

**Upper bound for $\|\boldsymbol{D}\boldsymbol{m}_t\|$:**

We first construct an upper bound for $\|\boldsymbol{D}\boldsymbol{m}_t\|$ in Equation 8. By definition, the largest eigenvalue $\lambda_{\max}$ of $\boldsymbol{D}$ fulfills

$$\lambda_{\max} \leq \frac{1}{\xi + \min_{i=1,\dots,d} \sqrt{(\boldsymbol{v}_t)_i}} \tag{9}$$

since $(\boldsymbol{v}_t)_i$ is the $i$-th column vector of $V_t$. By solving the recursive definition of $\boldsymbol{v}_t$ in Algorithm 1,

$$\boldsymbol{v}_t = (1-\beta_2)\sum_{k=1}^{t}\beta_2^{t-k}\boldsymbol{g}_k^2. \tag{10}$$

Using $\epsilon_t := \min_{k=1,\dots,t}\min_{i=1,\dots,d}(\boldsymbol{g}_k^2)_i$,

$$\lambda_{\max} \leq \frac{1}{\xi + \sqrt{(1-\beta_2^t)\epsilon_t}}. \tag{11}$$

Solving the recursive definition of $\boldsymbol{m}_t$ in Algorithm 1 gives $\boldsymbol{m}_t = (1-\beta_1)\sum_{k=1}^{t}\beta_1^{t-k}\boldsymbol{g}_k$. Using the triangle inequality and $\sigma_t := \max_{i=1,,\dots,t}\|\boldsymbol{g}_i\|$ gives $\|\boldsymbol{m}_t\| \leq (1-\beta_1^t)\sigma_t$, then

$$\|\boldsymbol{D}\boldsymbol{m}_t\| \leq \frac{(1-\beta_1^t)\sigma_t}{\xi + \sqrt{\epsilon_t(1-\beta_2^t)}} \leq \frac{(1-\beta_1^t)\sigma_t}{\xi}. \tag{12}$$

**Lower bound for $\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t\rangle^2$:**

Next, we construct a lower bound for $\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t\rangle^2$ in Equation 8. Using $Q_{t,i} := \langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_i\rangle$,

$$Q_{t,i} - \beta_1 Q_{t,i-1} = \langle \boldsymbol{g}_t, \boldsymbol{D}(\boldsymbol{m}_i - \beta_1\boldsymbol{m}_{i-1})\rangle = (1-\beta_1)\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{g}_i\rangle. \tag{13}$$

With $Q_t := Q_{t,t}$,

$$Q_t - \beta_1 Q_{t-1} \geq (1-\beta_1)\|\boldsymbol{g}_t\|^2\lambda_{\min} \tag{14}$$

where $\lambda_{\min}$ is the smallest eigenvalue of $\boldsymbol{D}$. Defining $\sigma_{t-1} := \max_{i=1,\dots,t-1}\|\boldsymbol{g}_i\|$ gives

$$Q_i - \beta_1 Q_{i-1} \geq -(1-\beta_1)\|\boldsymbol{g}_t\|\sigma_{t-1}\lambda_{\max} \tag{15}$$

for $i = 1, ..., t-1$. Note the following identity

$$\sum_{i=1}^{t-1}\beta_1^{t-i}(1-\beta_1) = \beta_1 - \beta_1^t. \tag{16}$$

Using Equations / Inequalities 14, 15, and 16,

$$\begin{aligned} Q_t - \beta_1^t Q_0 &= Q_t - \beta_1 Q_{t-1} + \sum_{j=1}^{t-1}\beta_1^j(Q_{t-j} - \beta_1 Q_{t-j-1}) \\ &= Q_t - \beta_1 Q_{t-1} + \sum_{i=1}^{t-1}\beta_1^{t-i}(Q_i - \beta_1 Q_{i-1}) \\ &\geq (1-\beta_1)\|\boldsymbol{g}_t\|^2\lambda_{\min} - \sum_{i=1}^{t-1}\beta_1^{t-i}(1-\beta_1)\|\boldsymbol{g}_t\|\sigma_{t-1}\lambda_{\max} \\ &= (1-\beta_1)\|\boldsymbol{g}_t\|^2\lambda_{\min} - (\beta_1 - \beta_1^t)\|\boldsymbol{g}_t\|\sigma_{t-1}\lambda_{\max}. \end{aligned} \tag{17}$$

Using Equation 10 and $\lambda_{\min} \geq \frac{1}{\xi+\sqrt{\max_{i=1,\ldots,d}(\boldsymbol{v}_t)_i}}$,

$$\lambda_{\min} \geq \frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}}. \tag{18}$$

Since $Q_0 = 0$ from the initial condition, and also because $\epsilon_t = \min_{k=1,\ldots,t} \min_{i=1,\ldots,d}(\boldsymbol{g}_k^2)_i \leq \min_{k=1,\ldots,t} \|\boldsymbol{g}_k\|^2 \leq \max_{k=1,\ldots,t} \|\boldsymbol{g}_k\|^2 = \sigma_t^2$ and $1-\beta_2^t < 1$,

$$\begin{aligned} Q_t &\geq -(\beta_1 - \beta_1^t)\|\boldsymbol{g}_t\|\sigma_{t-1}\frac{1}{\xi + \sqrt{(1-\beta_2^t)\epsilon_t}} + (1-\beta_1)\|\boldsymbol{g}_t\|^2\frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}} \\ &\geq \left(-(\beta_1 - \beta_1^t)\|\boldsymbol{g}_t\|\sigma_{t-1} + (1-\beta_1)\|\boldsymbol{g}_t\|^2\right)\frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}} \\ &\geq \frac{\left(-(\beta_1 - \beta_1^t)\|\boldsymbol{g}_t\|\sigma_{t-1} + (1-\beta_1)\|\boldsymbol{g}_t\|^2\right)}{\xi + \sigma_t} \\ &= \frac{\|\boldsymbol{g}_t\|\sigma_{t-1}(\beta_1 - \beta_1^t)}{\xi + \sigma_t}\left(\frac{(1-\beta_1)\|\boldsymbol{g}_t\|}{(\beta_1 - \beta_1^t)\sigma_{t-1}} - 1\right). \end{aligned} \tag{19}$$

Using $\theta_t := \frac{(1-\beta_1)\epsilon}{\beta_1\sigma_{t-1}} - 1$, we get $\frac{(1-\beta_1)\|\boldsymbol{g}_t\|}{(\beta_1-\beta_1^t)\sigma_{t-1}} - 1 \geq \theta_t$, since by the assumption, $\|\boldsymbol{g}_{t'}\| \geq \epsilon$ for all $t'$. As a result,

$$Q_t \geq \frac{\|\boldsymbol{g}_t\|(\beta_1 - \beta_1^t)\sigma_{t-1}\theta_t}{\xi + \sigma_t}. \tag{20}$$

**Combining the bounds:**

Substituting the upper bound in Equation 12 and the lower bound in Equation 20 to Equation 8 gives

$$\begin{aligned} f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}_t) &\leq -\frac{\langle \boldsymbol{g}_t, \boldsymbol{D}\boldsymbol{m}_t\rangle^2}{2L\|\boldsymbol{D}\boldsymbol{m}_t\|^2} \\ &\leq -\frac{Q_t^2}{2L\left(\frac{(1-\beta_1^t)\sigma_t}{\xi}\right)^2} \\ &\leq -\frac{\|\boldsymbol{g}_t\|^2(\beta_1 - \beta_1^t)^2\sigma_{t-1}^2\theta_t^2\xi^2}{2L(\xi+\sigma_t)^2(1-\beta_1^t)^2\sigma_t^2} \end{aligned} \tag{21}$$

which can be rewritten as

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}_{t+1}) \geq \frac{\|\boldsymbol{g}_t\|^2(\beta_1 - \beta_1^t)^2\sigma_{t-1}^2\theta_t^2\xi^2}{2L(\xi+\sigma_t)^2(1-\beta_1^t)^2\sigma_t^2}. \tag{22}$$

Define $\theta$ by substituting $\sigma$ to $\sigma_{t-1}$ in $\theta_t$. Using the existance of a natural number $\tau$ such that $\sigma = \sigma_t = \sigma_{t-1}$ for all $t \geq \tau$, we can sum over Equation 22 for $t = \max(\tau, 2), ..., T$, where $\max(\tau, 2)$ is the larger number of $\tau$ and 2.

$$\sum_{t=\max(\tau,2)}^{T}(f(\boldsymbol{x}_t)-f(\boldsymbol{x}_{t+1})) \geq \sum_{t=\max(\tau,2)}^{T}\frac{\|\boldsymbol{g}_t\|^2(\beta_1-\beta_1^t)^2\sigma_{t-1}^2\theta_t^2\xi^2}{2L(\xi+\sigma_t)^2(1-\beta_1^t)^2\sigma_t^2} \tag{23}$$
$$= \sum_{t=\max(\tau,2)}^{T}\frac{\|\boldsymbol{g}_t\|^2(\beta_1-\beta_1^t)^2\theta^2\xi^2}{2L(\xi+\sigma)^2(1-\beta_1^t)^2}.$$

The first line becomes an equality when $\max(\tau,2)\geq T$. The terms in the left-hand side cancels out and makes $f(\boldsymbol{x}_{\max(\tau,2)})-f(\boldsymbol{x}_{T+1})$. To get a lower bound for the right-hand side, define $C$ that is smaller than each term and does not depend on $t$, as

$$C=\frac{(\beta_1-\beta_1^2)^2\theta^2\xi^2}{2L(\xi+\sigma)^2}. \tag{24}$$

It holds that $C\leq\frac{(\beta_1-\beta_1^t)^2\theta^2\xi^2}{2L(\xi+\sigma_t)^2(1-\beta_1^t)^2}$ for all $t=\max(\tau,2),...,T$, since $\beta_1\geq 0$ from $1-\beta_1^t\leq 1$, and $\beta_1^2\geq\beta_1^t$ for all $t\geq 2$. Combining with $f(\boldsymbol{x}_*)=\min_{\boldsymbol{x}}f(\boldsymbol{x})$ gives

$$f(\boldsymbol{x}_{\max(\tau,2)})-f(\boldsymbol{x}_*)\geq f(\boldsymbol{x}_{\max(\tau,2)})-f(\boldsymbol{x}_{T+1})\geq\sum_{t=\max(\tau,2)}^{T}\|\boldsymbol{g}_t\|^2C \tag{25}$$

which reduces to

$$\min_{t=\max(\tau,2),...,T}\|\boldsymbol{g}_t\|^2\leq\frac{f(\boldsymbol{x}_{\max(\tau,2)})-f(\boldsymbol{x}_*)}{(T-\max(\tau,2)+1)C}. \tag{26}$$

As $T$ is increased, $T-\max(\tau,2)+1$ continues to increase, and $\min_{t=\max(\tau,2),...,T}\|\boldsymbol{g}_t\|^2$ can be made arbitrary small. It makes $\min_{t=\max(\tau,2),...,T}\|\boldsymbol{g}_t\|$ arbitrary small as well, which contradicts the original assumption that $\|\boldsymbol{g}_{t'}\|\geq\epsilon$ for all $t'$. For any $\epsilon>0$, there exists some step $t'$ such that $\|\boldsymbol{g}_{t'}\|<\epsilon$, which fulfills the convergence criterion.■

**Construction of the step size:**

In Algorithm 1, the step size $\alpha_t^*$ is determined as

$$\alpha_t^*=\frac{Q_t}{L\|D\boldsymbol{m}_t\|^2}. \tag{27}$$

In the denominator of the right-hand side, $\|D\boldsymbol{m}_t\|$ can be bounded by the upper bound obtained in Equation 12. The numerator $Q_t$ can be bounded by the second line of Equation 19. We define a step size $\alpha_t$ which fulfills $\alpha_t\leq\alpha_t^*$ as

$$\alpha_t:=\frac{\xi^2}{L(1-\beta_1^t)^2\sigma_t^2}\left(\frac{-(\beta_1-\beta_1^t)\|\boldsymbol{g}_t\|\sigma_{t-1}+(1-\beta_1)\|\boldsymbol{g}_t\|^2}{\xi+\sqrt{1-\beta_2^t}\sigma_t}\right)\leq\frac{Q_t}{L\|D\boldsymbol{m}_t\|^2}=\alpha_t^*. \tag{28}$$

By definition, $\|\boldsymbol{g}_t\|>\epsilon$ until the convergence criterion is fulfilled. Using $\beta_1<\frac{\epsilon}{\epsilon+\sigma}$,

$$(1-\beta_1)\|\boldsymbol{g}_t\| - (\beta_1 - \beta_1^t)\sigma_{t-1} \;\geq\; (1-\beta_1)\|\boldsymbol{g}_t\| - (\beta_1 - \beta_1^t)\sigma \;>\; (1-\beta_1)\epsilon - (\beta_1 - \beta_1^t)\sigma \tag{29}$$
$$> \left(\frac{\sigma}{\epsilon+\sigma}\right)\epsilon - (\beta_1 - \beta_1^t)\sigma \;=\; \left(\frac{\epsilon}{\epsilon+\sigma} - \beta_1 + \beta_1^t\right)\sigma \;>\; \beta_1^t\sigma \;\geq\; 0.$$

After facoring out $\|\boldsymbol{g}_t\|$ from Equation 28, the left-hand side of Equation 29 can be substituted. Since other factors of Equation 28 are positive, $\alpha_t > 0$. It means that until the convergence criterion $\|\boldsymbol{g}_t\| \leq \epsilon$ is fulfilled, $\alpha_t$ will not be negative nor zero. In other words, $0 < \alpha_t \leq \alpha_t^*$.

The reason for using $\alpha_t$ instead of $\alpha_t^*$ is to avoid the computation of the inverse matrix. Since $0 \leq \alpha_t \leq \alpha_t^*$, using $\alpha_t$ prohibits the loss function to increase up to quadratic approximation.

**Approximations to $\alpha_t$ :**

Define $\tilde{\alpha}_t$ by setting $\beta_2 = 0$ in $\alpha_t$. Then

$$\tilde{\alpha}_t = \frac{\xi^2}{L(1-\beta_1^t)^2\sigma_t^2}\left(\frac{-(\beta_1 - \beta_1^t)\|\boldsymbol{g}_t\|\sigma_{t-1} + (1-\beta_1)\|\boldsymbol{g}_t\|^2}{\xi + \sigma_t}\right). \tag{30}$$

$\beta_1 = \frac{\epsilon}{\epsilon+2\sigma}$ fulfills the condition $\beta_1 < \frac{\epsilon}{\epsilon+\sigma}$. Substituting it to the condition $\xi > \frac{\sigma^2\beta_1}{-\beta_1\sigma+\epsilon(1-\beta_1)}$ simplifies to $\xi > \sigma$.

From $\beta_2 < 1$, $\tilde{\alpha}_t < \alpha_t$. Define $\hat{\alpha}_t$ by subsituting $\sigma = \sigma_t = \sigma_{t-1}$ and $\xi = 2\sigma$ to $\tilde{\alpha}_t$.

$$\hat{\alpha}_t = \frac{4\left(\|\boldsymbol{g}_t\|^2(1-\beta_1) - \|\boldsymbol{g}_t\|(\beta_1 - \beta_1^t)\sigma\right)}{3L(1-\beta_1^t)^2\sigma}. \tag{31}$$

$\hat{\alpha}_t$ and $\tilde{\alpha}_t$ approximates $\alpha_t$ and are easier to compute. However, since they set $\beta_2 = 0$, it maybe a little less efficient than $\alpha_t$.

## Appendix B. Experimental setup

Programs were implemented using Tensorflow. All activation functions in any layers were ReLU, and LeNet is trained with the cross-entropy. The batch size for the Fashion-MNIST dataset is set to 128 in all runs. All reported values were evaluated on the whole training and test sets of sizes 50,000 and 10,000, respectively. We describe the detailed setup of the experiments we presented in Section 4. Code is made available at: https://github.com/mtkwT/adaptive-hessian-free-optimization .

### B.1. Plotting error bars

The error bars in Figures 2 and 1 are centered at the mean value of the observed quantity. The bars indicate the standard error around the mean in positive and negative directions.

### B.2. Range of hyperparameter search

We evaluated various values of hyperparameters using Bayesian optimization. The ranges used for the hyperparameters search are presented below.

The search range of hyperparameter $L$ that should fulfill $L$-conjugate smoothness criterion (Definition 1) was determined by pre-running. Since Equation 2 can be transformed to

$$L \geq \frac{2(f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle)}{\|\boldsymbol{y} - \boldsymbol{x}\|^2}, \tag{32}$$

we computed the right-hand side to determine the range of hyperparameter optimization.

- HF
  - learning rate $\alpha$ from $\{1.0^{-3}, 1.0\}$
  - damping $\lambda$ from $\{1, 30\}$
- Adaptive HF
  - $\beta_2$ from $\{1.0^{-10}, 1.0^{-4}\}$
  - damping $\lambda$ from $\{1, 30\}$
  - $\epsilon$ from $\{1.0^{-5}, 0.1\}$
  - $\sigma$ from $\{30, 100\}$
  - $L$ from $\{50, 300\}$