Adapting the Learning Rate of the Learning Rate in Hypergradient Descent

Kazuma Itakura*, Kyohei Atarashi*, Satoshi Oyama†‡ and Masahito Kurihara‡
*Graduate School of Information Science and Technology, Hokkaido University
†Global Institution for Collaborative Research and Education, Hokkaido University
‡Faculty of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan
kazuvabo03@eis.hokudai.ac.jp, {katarashi, oyama, kurihara}@ist.hokudai.ac.jp

***Abstract*—Gradient descent is a widely used optimization method. The adjustment of the learning rate is an important factor in improving its performance, and many researchers have investigated methods for automatically adjusting the learning rate. One such method, hypergradient descent, automatically adjusts the learning rate by using gradient descent. However, it introduces the "learning rate of the learning rate," and an appropriate value for the learning rate of the learning rate must be chosen in order to effectively adjust the learning rate. We investigated the use of two datasets and two optimization methods for doing this and achieved an effective adjustment of the learning rate when the objective function was convex and L-smooth.**

***Index Terms*—optimization, hypergradient descent, adjusting learning rate**

I. Introduction

Optimization methods are used in machine learning to find values for model parameters $\boldsymbol{\theta}$ that minimize loss function $f(\boldsymbol{\theta})$. A typical machine learning problem is classification, i.e., predicting an output label for input data. When training a machine learning model for classification, we search for values of parameters $\boldsymbol{\theta}$ that minimize loss function $f(\boldsymbol{\theta})$ defined by the difference between the true label and the model output label. Since the choice of the optimization method affects classification accuracy and training time, optimization methods are important to improving the performance of machine learning.

The widely used gradient descent optimization method searches for optimal parameter values by gradient updating of the parameter values:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \cdot \nabla f(\boldsymbol{\theta}_t), \tag{1}$$

where α is the learning rate. The adjustment of α is important for improving the performance of gradient descent. Choosing an appropriate value for the learning rate reduces the training time, so optimal values can be searched for more quickly. On the other hand, choosing an inappropriate value lengthens the training time, and optimal values may never be found in the worst case. Many researchers in the machine learning field have investigated methods for automatically adjusting the learning rate, and several methods based on gradient descent have been reported, including Adagrad [1], Adadelta [2], and Adam [3].

The hypergradient descent (HD) method [4] enables automatic adjustment of the learning rate by using gradient descent. In other words, HD searches for the optimal learning rate by hypergradient updating of the learning rate along with updating of the parameter values. The hypergradient is the gradient of a loss function with respect to the hyperparameters. It was proposed by Maclaurin et al. [5]. The equation for updating learning rate α,

$$\alpha_{t+1} = \alpha_t - \beta \cdot \nabla_\alpha f(\boldsymbol{\theta}_t), \tag{2}$$

follows the equation for updating $\boldsymbol{\theta}$ (Eq. (1)); $\nabla_\alpha f(\boldsymbol{\theta}_t)$ is the hypergradient with respect to α, and β is the learning rate of α. HD can be applied to not only Eq. (1) but also to equations used to update the parameters in many methods based on gradient descent by adding Eq. (2) after updating the parameter values.

However, an appropriate value for the "learning rate of the learning rate" β introduced in HD must be chosen. As it is for α, if an appropriate value for β is chosen, the learning rate is effectively adjusted. In contrast, if an inappropriate value is chosen, the convergence of the learning is slower, and learning rate α may become negative in the worst case.

We investigated methods for adapting a value for the learning rate of the learning rate β to lead to an effective adjustment of learning rate α when the loss function is convex and L-smooth. Many previous studies on HD have chosen a value for β using an experimental search method such as grid search and random search. While several researchers have suggested a specific appropriate value for β, there has been no reported research that experimentally proved that the specific value certainly leads to an effective adjustment of α. In this paper, we demonstrate the appropriateness of the method for adapting a value for β by solving classification problems.

In section 2, we describe HD and present the method for adapting a value for β. In section 3, we describe an experiment demonstrating that the β value presented in section 2 is appropriate by solving classification problems. Future work is mentioned in section 4, and the key points are summarized in section 5.

II. Hypergradient Descent

HD automatically adjusts the learning rate by hypergradient updating of the learning rate. In this section, we first explain the hypergradient aspect of HD. Next, we describe the

derivation of the HD update equation and present examples of HD applied to stochastic gradient descent (SGD) and Adam. Finally, we present the method for adapting a value for the learning rate of the learning rate.

A. Hypergradient

The hypergradient is the gradient of a loss function with respect to a hyperparameter and is searched for by means of automatic differentiation in reverse mode [6]. Automatic differentiation analytically searches for the value of the gradient. Specifically, we search in reverse mode for values of the gradients of intermediate variables with respect to a hyperparameter while tracing the procedure used to search for optimal parameter values through training with the hyperparameter. An intermediate variable is the result of a simple equation that is a component of the entire training procedure, which explicitly models the dependence on the hyperparameter. The product of all gradients at the intermediate variables with respect to the hyperparameter gives the hypergradient due to the chain rule of differentiation.

There are two requirements for searching for the hypergradient. First, the intermediate variables must be differentiable for the hyperparameter. For example, we can not search for the hypergradient with respect to the depth of the decision tree because the depth does not explicitly appear in the intermediate variables, and the intermediate variables are not differentiable. Second, sufficient memory is needed to hold the intermediate variables. All of the intermediate variables are used to trace the training procedure in reverse mode. The bigger the dataset for training, the greater the amount of memory needed to hold the intermediate variables and the longer the training time. In HD, the amount of memory required is reduced by searching for an approximated hypergradient.

B. Overview of Hypergradient Descent

HD iteratively updates the learning rate along with iteratively updating the parameter values using

$$\alpha_t = \alpha_{t-1} - \beta \cdot \nabla_\alpha f(\boldsymbol{\theta}_{t-1}), \tag{3}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \cdot \nabla f(\boldsymbol{\theta}_{t-1}), \tag{4}$$

where $\nabla_\alpha f(\boldsymbol{\theta}_{t-1})$ is the hypergradient of loss function $f(\boldsymbol{\theta}_{t-1})$ with respect to learning rate α, and t is the number of iterations. We can search for hypergradient $\nabla_\alpha f(\boldsymbol{\theta}_{t-1})$ because the intermediate variables are all the $\boldsymbol{\theta}$ and α values used in the $t-1$ iterations. The hypergradient explicitly depends on α.

If we searched for exactly $\nabla_\alpha f(\boldsymbol{\theta}_{t-1})$, the training time and required memory could become huge. In HD, the hypergradient is searched for as an approximate value that depends only on the previous iteration. This reduces the training time and required memory. By combining Eqs. (3) and (4), we can rewrite $\nabla_\alpha f(\boldsymbol{\theta}_{t-1})$ as

$$\nabla_\alpha f(\boldsymbol{\theta}_{t-1}) = \nabla_\alpha f\left(\boldsymbol{\theta}_{t-2} - \alpha \cdot \nabla f(\boldsymbol{\theta}_{t-2})\right). \tag{5}$$

The approximate value, which depends only on the previous iteration, does not reflect dependence with respect to the α of $\boldsymbol{\theta}_{t-2}$ and $\nabla f(\boldsymbol{\theta}_{t-2})$ on the right side of Eq. (5). In other words, $\nabla_\alpha f(\boldsymbol{\theta}_{t-1})$ is calculated as

$$\begin{aligned}\nabla_\alpha f(\boldsymbol{\theta}_{t-1}) &= \nabla f(\boldsymbol{\theta}_{t-1}) \cdot \frac{\partial \boldsymbol{\theta}_{t-1}}{\partial \alpha} \\ &= \nabla f(\boldsymbol{\theta}_{t-1}) \cdot \frac{\partial \left(\boldsymbol{\theta}_{t-2} - \alpha \cdot \nabla f(\boldsymbol{\theta}_{t-2})\right)}{\partial \alpha} \\ &= \nabla f(\boldsymbol{\theta}_{t-1}) \cdot \left(-\nabla f(\boldsymbol{\theta}_{t-2})\right). \end{aligned} \tag{6}$$

Finally, we insert Eq. (6) into Eqs. (3) and (4), so the HD update equations are expressed as

$$\alpha_t = \alpha_{t-1} + \beta \cdot \nabla f(\boldsymbol{\theta}_{t-1}) \cdot \nabla f(\boldsymbol{\theta}_{t-2}), \tag{7}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha_t \cdot \nabla f(\boldsymbol{\theta}_{t-1}). \tag{8}$$

Eqs. (7) and (8) show that only two procedures are needed to adjust the learning rate. One saves the gradients from the previous iteration $\nabla f(\boldsymbol{\theta}_{t-2})$ and the current iteration $\nabla f(\boldsymbol{\theta}_{t-1})$, and the other calculates the inner product between the two gradients.

Algorithm 1 Stochastic gradient descent with HD

Require: $\alpha_0, f_t(\boldsymbol{\theta}), \boldsymbol{\theta}_0, \beta$
  $t, \nabla_\alpha \boldsymbol{u}_0 \leftarrow 0, \boldsymbol{0}$
  while $\boldsymbol{\theta}_t$ not converged **do**
    $t \leftarrow t + 1$
    $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$
    $h_t \leftarrow \boldsymbol{g}_t \cdot \nabla_\alpha \boldsymbol{u}_{t-1}$
    $\alpha_t \leftarrow \alpha_{t-1} - \beta \cdot h_t$
    $\boldsymbol{u}_t \leftarrow -\alpha_t \cdot \boldsymbol{g}_t$
    $\nabla_\alpha \boldsymbol{u}_t \leftarrow -\boldsymbol{g}_t$
    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \boldsymbol{u}_t$
  end while
  return $\boldsymbol{\theta}_t$

Algorithm 2 Adam with HD

Require: $\alpha_0, f_t(\boldsymbol{\theta}), \boldsymbol{\theta}, \beta, \beta_1, \beta_2, \epsilon$
  $t, \boldsymbol{m}_0, \boldsymbol{v}_0, \nabla_\alpha \boldsymbol{u}_0 \leftarrow 0, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}$
  while $\boldsymbol{\theta}_t$ not converged **do**
    $t \leftarrow t + 1$
    $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1})$
    $h_t \leftarrow \boldsymbol{g}_t \cdot \nabla_\alpha \boldsymbol{u}_{t-1}$
    $\alpha_t \leftarrow \alpha_{t-1} - \beta \cdot h_t$
    $\boldsymbol{m}_t \leftarrow \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1) \cdot \boldsymbol{g}_t$
    $\boldsymbol{v}_t \leftarrow \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \cdot \boldsymbol{g}_t^2$
    $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1-\beta_1^t)$
    $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1-\beta_2^t)$
    $\boldsymbol{u}_t \leftarrow -\alpha_t \cdot \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon)$
    $\nabla_\alpha \boldsymbol{u}_t \leftarrow -\hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon)$
    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \boldsymbol{u}_t$
  end while
  return $\boldsymbol{\theta}_t$

Fig. 1. Algorithms for SGD with HD and Adam with HD.

Furthermore, we can apply HD to many methods based on gradient descent. Eqs. (7) and (8) are the update procedure for SGD with HD. They can be replaced with equations for the update procedures of other methods. This is illustrated in Algorithms 1 and 2. The parts in red or blue are the parts to be changed when applying HD to other methods.

As mentioned in section 1, the learning rate of the learning rate β is introduced in HD, and we have to choose an appropriate value for β. Next, we present the method for adapting a value for β to lead to effective adjustment of the learning rate.

C. Adapting the learning rate of the learning rate

Rubio suggested that adapting β as $1/\left(L\left\|\nabla f(\boldsymbol{\theta}_{t-2})\right\|^2\right)$ leads to effective adjustment of the learning rate when the loss function is convex and L-smooth [7]. However, he did not describe a specific derivation of the value for β. Here we first determine whether the value for β suggested by Rubio is reasonable.

We start by rewriting the equation for updating learning rate α in HD (Eq. (5)):

$$\begin{aligned}\alpha_t &= \alpha_{t-1} - \beta \cdot \nabla_\alpha f(\boldsymbol{\theta}_{t-2} - \alpha \cdot \nabla f(\boldsymbol{\theta}_{t-2})) \\ &= \alpha_{t-1} - \beta \cdot \nabla_\alpha g_{t-1}(\alpha),\end{aligned} \tag{9}$$

where we define function $g_{t-1}(\alpha)$ as $f(\boldsymbol{\theta}_{t-2} - \alpha \cdot \nabla f(\boldsymbol{\theta}_{t-2}))$. The effective adjustment of α means that the value for loss function $f(\boldsymbol{\theta})$ is effectively reduced as $\boldsymbol{\theta}$ is updated. Moreover, when $g_{t-1}(\alpha)$ decreases as α is updated, $f(\boldsymbol{\theta})$ also decreases as $\boldsymbol{\theta}$ is updated. That is, the value for β used to reduce $g_{t-1}(\alpha)$ as α is updated leads to effective adjustment of α in HD.

Eq. (9) is the same as the update equation in gradient descent for $g_{t-1}(\alpha)$ that has α as parameters. If the loss function is convex and L-smooth, learning convergence is guaranteed in gradient descent. We can prove Theorem 1 in a manner similar to that used by Bubeck [8].

Definition 1. Let C be a convex set. A function $f : C \to \mathbb{R}$ is convex if for every $\mathbf{u}, \mathbf{v} \in C$ and $\alpha \in [0, 1]$ [9, p.157],

$$f(\alpha\mathbf{u} + (1-\alpha)\mathbf{v}) \leq \alpha f(\mathbf{u}) + (1-\alpha) f(\mathbf{v}).$$

Definition 2. A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth if its gradient is β-Lipschitz; namely, for all $\mathbf{v}, \mathbf{w}$ we have [9, p.162]

$$\left\|\nabla f(\mathbf{v}) - \nabla f(\mathbf{w})\right\| \leq \beta \left\|\mathbf{v} - \mathbf{w}\right\|.$$

Theorem 1. *The gradient descent update equation for function $f(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ is*

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \cdot \nabla f(\boldsymbol{\theta}_{t-1}), \tag{10}$$

where $\boldsymbol{\theta}_t$ is the parameter for t iterations and α is the learning rate. If f is convex and L-smooth, $\alpha \leq 1/L$, and $\boldsymbol{\theta}$ is updated by gradient descent, then

$$f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}^*) \leq \frac{\left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\right\|^2}{2\alpha t},$$

where $\boldsymbol{\theta}^*$ is the optimal parameter value.

Proof: Because f is L-smooth, we have

$$f(\boldsymbol{\theta}_t) \leq f(\boldsymbol{\theta}_{t-1}) + \langle \nabla f(\boldsymbol{\theta}_{t-1}), \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} \rangle + \frac{L}{2}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2. \tag{11}$$

By rearranging Eq. (10), we obtain

$$\nabla f(\boldsymbol{\theta}_{t-1}) = \frac{\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_t}{\alpha}. \tag{12}$$

Combining Eqs. (11) and (12), because $\alpha \leq 1/L$, we obtain

$$\begin{aligned}f(\boldsymbol{\theta}_t) &\leq f(\boldsymbol{\theta}_{t-1}) + \langle \nabla f(\boldsymbol{\theta}_{t-1}), \boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1} \rangle + \frac{L}{2}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2 \\ &\leq f(\boldsymbol{\theta}_{t-1}) - \frac{1}{\alpha}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2 + \frac{L}{2}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2 \\ &\leq f(\boldsymbol{\theta}_{t-1}) - \frac{1}{2\alpha}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2.\end{aligned} \tag{13}$$

Because $f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}_{t-1}) \leq 0$, $f(\boldsymbol{\theta}_t)$ is monotone non-increasing.

Moreover, because f is convex, we have

$$f(\boldsymbol{\theta}^*) \geq f(\boldsymbol{\theta}_{t-1}) + \langle \nabla f(\boldsymbol{\theta}_{t-1}), \boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1} \rangle. \tag{14}$$

Combining Eqs. (13) and (14), because $\alpha \leq 1/L$, we have

$$f(\boldsymbol{\theta}_t) \leq f(\boldsymbol{\theta}^*) - \langle \nabla f(\boldsymbol{\theta}_{t-1}), \boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1} \rangle - \frac{1}{2\alpha}\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2. \tag{15}$$

The $\langle \nabla f(\boldsymbol{\theta}_{t-1}), \boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1} \rangle$ is rewritten as

$$\begin{aligned}&\langle \nabla f(\boldsymbol{\theta}_{t-1}), \boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1} \rangle \\ &= \frac{1}{\alpha}\langle \boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_t, \boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1} \rangle \\ &= \frac{1}{2\alpha}\Big(\left\|(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}) - (\boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1})\right\|^2 \\ &\qquad - \left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2 - \left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1}\right\|^2\Big) \\ &= \frac{1}{2\alpha}\left(\left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\right\|^2 - \left\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}\right\|^2 - \left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1}\right\|^2\right).\end{aligned} \tag{16}$$

Therefore, by combining Eqs. (15) and (16), we obtain

$$f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}^*) \leq \frac{1}{2\alpha}\left(\left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_{t-1}\right\|^2 - \left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_t\right\|^2\right). \tag{17}$$

Adding all inequalities from 1 to t in Eq. (17), we obtain

$$\begin{aligned}&\sum_{k=1}^{t}\left(f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}^*)\right) \\ &\qquad \leq \sum_{k=1}^{t}\left(\frac{1}{2\alpha}\left(\left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_{k-1}\right\|^2 - \left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_k\right\|^2\right)\right) \\ &\qquad = \frac{1}{2\alpha}\left(\left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\right\|^2 - \left\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_t\right\|^2\right).\end{aligned} \tag{18}$$

Also, because $f(\boldsymbol{\theta}_t)$ is monotone non-increasing, we have

$$f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}^*) \leq \frac{1}{t}\sum_{k=1}^{t}\left(f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}^*)\right). \tag{19}$$

Thus, by combining Eqs. (18) and (19), we obtain the target inequality:

$$\begin{aligned} f(\boldsymbol{\theta}_t) - f(\boldsymbol{\theta}^*) &\leq \frac{1}{t}\sum_{k=1}^{t}\left(f(\boldsymbol{\theta}_k) - f(\boldsymbol{\theta}^*)\right) \\ &\leq \frac{1}{2\alpha t}\left(\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\|^2 - \|\boldsymbol{\theta}^* - \boldsymbol{\theta}_t\|^2\right) \\ &\leq \frac{\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\|^2}{2\alpha t}. \end{aligned}$$

■

Next, consider the requirement for the loss function $f(\boldsymbol{\theta})$ for making $g(\alpha)$ convex and L-smooth. Because we defined $g(\alpha)$ as $f(\boldsymbol{\theta}_{t-2} - \alpha \cdot \nabla f(\boldsymbol{\theta}_{t-2}))$, Theorems 2 and 3 hold. Moreover, we can show Theorem 4 holds from Theorem 3.

Theorem 2. *Assume that $f : \mathbb{R}^d \to \mathbb{R}$ can be written as $f(\mathbf{w}) = g(\langle \mathbf{w}, \mathbf{x} \rangle + y)$ for some $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$, and $g : \mathbb{R} \to \mathbb{R}$. Then, convexity of g implies convexity of f [9, p.115].*

Theorem 3. *Let $f(\boldsymbol{w}) = g(\langle \boldsymbol{w}, \mathbf{x} \rangle + b)$, where $g : \mathbb{R} \to \mathbb{R}$ is a β-smooth function, $\mathbf{x} \in \mathbb{R}^d$, and $b \in \mathbb{R}$. Then, f is $\beta \|\mathbf{x}\|^2$-smooth [9, pp.162–163].*

Theorem 4. *Let $f(\boldsymbol{w}) = g(\mathbf{X}\boldsymbol{w} + \boldsymbol{b})$, where $g : \mathbb{R}^m \to \mathbb{R}$ is an L-smooth function, $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\boldsymbol{w} \in \mathbb{R}^n$, and $\boldsymbol{b} \in \mathbb{R}^m$. Then, f is $L\|\mathbf{X}\|_F^2$-smooth, where $\|\mathbf{X}\|_F^2$ is the square of the Frobenius norm ($\|\mathbf{X}\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2$).*

Proof: Due to the chain rule of differentiation, we have

$$\nabla f(\boldsymbol{w}) = \mathbf{X} \cdot \nabla g(\mathbf{X}\boldsymbol{w} + \boldsymbol{b}).$$

Thus, by combining the facts that $\|\mathbf{X}\boldsymbol{w}\| \leq \|\mathbf{X}\|_F \|\boldsymbol{w}\|$ and that g is L-smooth, we obtain

$$\begin{aligned} &\|\nabla f(\bar{\boldsymbol{w}}) - \nabla f(\boldsymbol{w})\| \\ &\quad = \|\mathbf{X}(\nabla g(\mathbf{X}\bar{\boldsymbol{w}} + \boldsymbol{b}) - \nabla g(\mathbf{X}\boldsymbol{w} + \boldsymbol{b}))\| \\ &\quad \leq \|\mathbf{X}\|_F \|\nabla g(\mathbf{X}\bar{\boldsymbol{w}} + \boldsymbol{b}) - \nabla g(\mathbf{X}\boldsymbol{w} + \boldsymbol{b})\| \\ &\quad \leq \|\mathbf{X}\|_F \cdot L\|\mathbf{X}(\bar{\boldsymbol{w}} - \boldsymbol{w})\| \\ &\quad \leq L\|\mathbf{X}\|_F^2 \|\bar{\boldsymbol{w}} - \boldsymbol{w}\|. \end{aligned}$$

Therefore, f is $L\|\mathbf{X}\|_F^2$-smooth. ■

If loss function f is convex and L-smooth, g_{t-1} is convex and $L\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2$-smooth from Theorems 2 and 3. Because of Theorem 1 and the form of the function g_{t-1}, we can effectively adjust the learning rate in HD by adapting β as $1/\left(L\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2\right)$.

Finally, we show an example loss function that is convex and L-smooth and an appropriate value for β for that function. Cross-entropy loss function E between output $\boldsymbol{y}$ of the single-layer perceptron and true label $\mathbf{t}$ is

$$\begin{aligned} \boldsymbol{y} &= \mathbf{X}\boldsymbol{\theta} + \boldsymbol{b}, \\ z_i &= \frac{\mathrm{e}^{y_i}}{\sum_{j=1}^{n}\mathrm{e}^{y_j}}, \\ E &= -\sum_{i=1}^{n}\mathrm{t}_i z_i, \end{aligned}$$

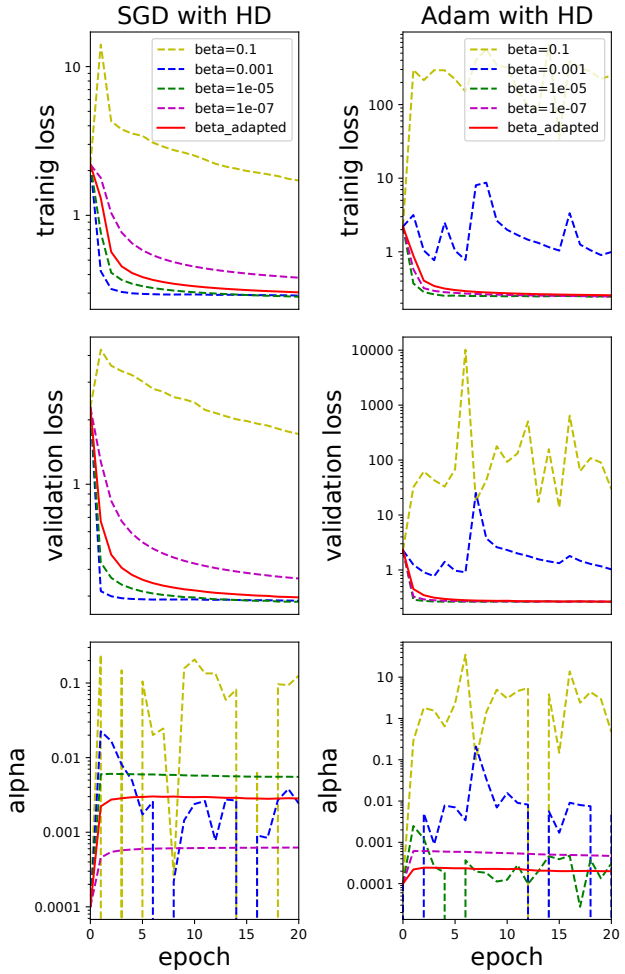

Fig. 2. Convergence of learning using Mixed National Institute of Standards and Technology (MNIST) dataset for SGD with HD and Adam with HD. Row: *top:* training loss; *middle:* validation loss; *bottom:* value of α. Color corresponding to value of β: *yellow:* 10^{-1}; *blue:* 10^{-3}; *green:* 10^{-5}; *magenta:* 10^{-7}; *red:* adapted β as $1/(L\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2)$.

where $\mathbf{X}$ is the input data, $\boldsymbol{\theta}$ is a parameter, $\boldsymbol{b}$ is the bias term, and t_i, y_i, and z_i are the i-th element of each vector. Function E is convex and 1-smooth with respect to $\boldsymbol{y}$ [10]. Therefore, function E is convex and $\|\mathbf{X}\|_F^2$-smooth with respect to $\boldsymbol{\theta}$ because of Theorem 4. By adapting β as $1/(\|\mathbf{X}\|_F^2\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2)$, we can effectively adjust the learning rate.

III. Experiment

The experimental results show that adapting β as $1/(L\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2)$ leads to effective adjustment of the learning rate if the loss function is convex and L-smooth regardless of the datasets and optimization methods used.

A. Settings

We experimentally examined the appropriateness of the value for β by solving classification problems. We used MNIST and CIFAR10 datasets, which predict an output label for input images, to investigate learning convergence for

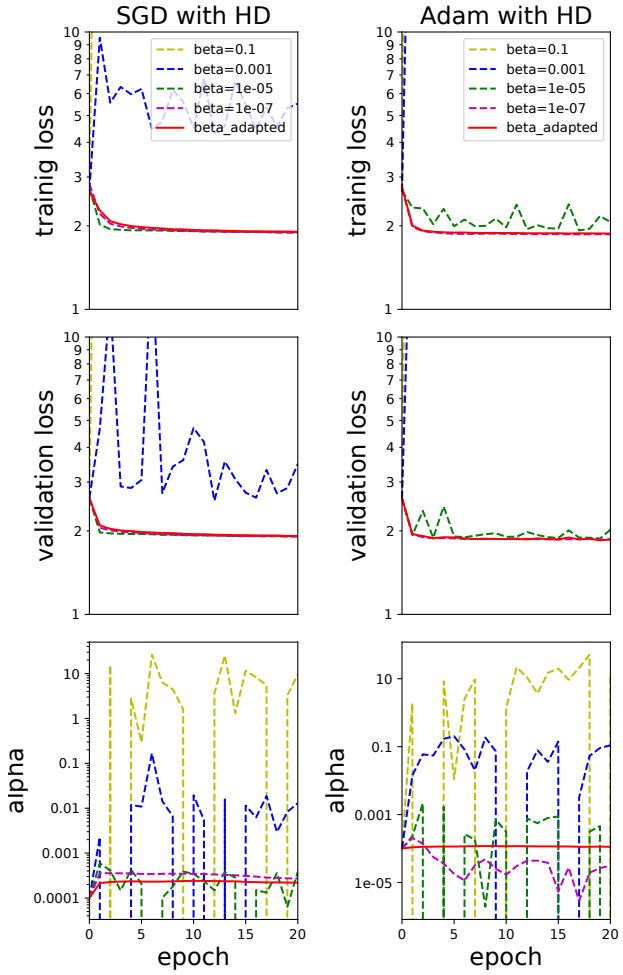

Fig. 3. Convergence of learning using Canadian Institute For Advanced Research (CIFAR)-10 database for SGD with HD and Adam with HD. Row: *top:* training loss; *middle:* validation loss; *bottom:* value of α. Color corresponding to value of β: *yellow:* 10^{-1}; *blue:* 10^{-3}; *green:* 10^{-5}; *magenta:* 10^{-7}; *red:* adapted β as $1/(L\,\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2)$

various combinations of the two datasets and two optimization methods: SGD with HD and Adam with HD (Algorithms 1 and 2, respectively). We adapt β as $1/(\|\mathbf{X}\|_F^2\,\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2)$, which we refer to as "adapted β". To utilize the value chosen for β (see section 2), we used the cross-entropy function between the output of a single-layer perceptron and the true label as the loss function, which was convex and $\|\mathbf{X}\|_F^2$-smooth. We compared learning convergence when β was fixed at a value taken from the set $\{10^{-1}, 10^{-3}, 10^{-5}, 10^{-7}\}$ and when β was "adapted β". The initial value of learning rate α was 10^{-4} regardless of the dataset and optimization method. We implemented the algorithm for this experiment by modifying the source code published by Baydin in *github*[1].

B. Results and discussion

Fig. 2 plots the learning results in each setting for the MNIST dataset. The results demonstrate that the appropriate value for β depends on the optimization method. The plots for SGD with HD show that the learning converged relatively quickly when β was fixed at 10^{-3} (blue) or 10^{-5} (green). Those for Adam with HD show that the learning converged relatively quickly when β was fixed at 10^{-5} or 10^{-7} (magenta). Moreover, the learning did not converge when β was fixed at 10^{-3} using Adam with HD while they did converge using SGD with HD. In short, when β was a fixed value, the learning converged only with a specific optimization method. In contrast, when "adapted β" (red) was used, the learning converged using both SGD and Adam with HD.

Fig. 3 shows the learning results in each setting for the CIFAR10 dataset. Comparison of Figs. 2 and 3 shows that the appropriate value for β depends on the dataset. When β was fixed at 10^{-1} (yellow) or 10^{-3} in Adam using the MNIST dataset, the learning did not converge because β was too large. In contrast, in Adam using the CIFAR10 dataset, even when β was fixed at 10^{-5}, the learning did not converge because β was too large. That is, when β was a fixed value, the learning converges only for a specific dataset. In contrast, when "adapted β" (red) was used, the learning converged for both datasets.

The bottom two plots in Figs. 2 and 3 show that "adapted β" effectively adjusted the learning rate. When the learning did not converge when β was fixed, the learning rate diverged in a disorderly manner. In contrast, when "adapted β" was used, the learning rate converged to a certain value. When β was fixed, the learning rate converged to a certain value in most cases in which the learning converged.

This experiment shows that adapting β as $1/(L\,\|\nabla f(\boldsymbol{\theta}_{t-2})\|^2)$ leads to effective adjustment of the learning rate for various combinations of datasets and optimization methods. When β was a fixed value, the learning converges only in specific settings. The use of "adapted β" is feasible in multiple experimental settings.

IV. Future Work

Here we assumed that the loss function is convex and L-smooth in order to facilitate chosing the appropriate value for β. Of course, many real problems in machine learning do not meet this assumption. Thus, to solve such problems, we need an appropriate value for β when the loss function is neither convex nor L-smooth.

Moreover, because β is a learning rate, it can be adjusted using HD. In that case, β itself would be effectively adjusted in exchange for adding the learning rate of β. Future work thus includes investigating what happens when the number of learning rates to adjust increases.

V. Conclusion

Hypergradient descent enables learning rate α to be automatically adjusted using gradient descent. However, using it introduces the learning rate of the learning rate β, requiring that we choose an appropriate value for β in order to effectively adjust α. We investigated methods for adapting the value for β regardless of the dataset and optimization method under

[1]https://github.com/gbaydin/hypergradient-descent

the assumption that the loss function is convex and L-smooth. The experimental results showed that the learning converges for various combinations of datasets and optimization methods when β is adapted as $1/\left(L\left\|\nabla f(\theta_{t-2})\right\|^2\right)$. This means that an appropriate value for β can be chosen that leads to an effective adjustment of learning rate α for such loss functions.

Extension of our investigation to non-convex and/or non-L-smooth loss functions remains for future work, as does applying HD to real problems in order to improve the utility of HD.

REFERENCES

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

[2] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[3] D. P. Kingma and J. L. Ba, "Adam: a method for stochastic optimization," in *International Conference on Learning Representations*, 2015, pp. 1–13.

[4] A. G. Baydin, R. Counish, D. M. Rubio, M. Schmidt, and F. Wood, "Online learning rate adaptation with hypergradient descent," in *International Conference on Learning Representations*, 2018.

[5] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[6] A. G. Baydin and B. A. Pearlmutter, "Automatic differentiation of algorithms for machine learning," ICML AutoML Workshop, Tech. Rep., 2014.

[7] D. M. Rubio, "Convergence analysis of an adaptive method of gradient descent," *University of Oxford, Oxford, M. Sc. thesis*, 2017.

[8] S. Bubeck, "Convex optimization: Algorithms and complexity," *Foundations and Trends in Machine Learning*, vol. 8, pp. 231–357, 2015.

[9] S. S. Shwartz and S. B. David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014.

[10] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," *arXiv preprint arXiv:1704.00805*, 2017.