symmetry

MDPI

Article

A Novel Learning Rate Schedule in Optimization for Neural Networks and It's Convergence

Jieun Park [1], Dokkyun Yi [1] and Sangmin Ji [2,*]

[1] Seongsan Liberal Arts College, Daegu University, Kyungsan 38453, Korea; writer2yah@daegu.ac.kr (J.P.); dkyi@daegu.ac.kr (D.Y.)

[2] Department of Mathematics, College of Natural Sciences, Chungnam National University, Daejeon 34134, Korea

* Correspondence: smji@cnu.ac.kr

Received: 6 March 2020; Accepted: 13 April 2020; Published: 22 April 2020

Abstract: The process of machine learning is to find parameters that minimize the cost function constructed by learning the data. This is called optimization and the parameters at that time are called the optimal parameters in neural networks. In the process of finding the optimization, there were attempts to solve the symmetric optimization or initialize the parameters symmetrically. Furthermore, in order to obtain the optimal parameters, the existing methods have used methods in which the learning rate is decreased over the iteration time or is changed according to a certain ratio. These methods are a monotonically decreasing method at a constant rate according to the iteration time. Our idea is to make the learning rate changeable unlike the monotonically decreasing method. We introduce a method to find the optimal parameters which adaptively changes the learning rate according to the value of the cost function. Therefore, when the cost function is optimized, the learning is complete and the optimal parameters are obtained. This paper proves that the method ensures convergence to the optimal parameters. This means that our method achieves a minimum of the cost function (or effective learning). Numerical experiments demonstrate that learning is good effective when using the proposed learning rate schedule in various situations.

Keywords: machine learning; numerical optimization; learning rate; convergence

1. Introduction

Machine learning is carried out by using a cost function to determine how accurately a model learns from data and determining the parameters that minimize this cost function. The process of changing these parameters, in order to find the optimal set, is called learning. We define a cost function to check that the learning is successful: the lower the value of the cost function, the better the learning. The cost function is also used in the learning process. Therefore, the cost function is selected and used for learning, according to the training data [1,2]. In the learning process, a gradient-based method is usually used to reduce the gradient of the cost function [3–8]. However, with larger sets of training data and more complex training models, the cost function many have many local minima, and the simple gradient descent method fails at a local minimum because the gradient vanishes at this point. To solve this problem, some gradient-based methods where learning occurs even when the gradient is zero have been introduced, such as momentum-based methods. In these methods, the rate of change of the cost function is continuously added to the ratio of an appropriate variable. Even if the rate of change of the cost function becomes zero, the learning is continued by the value of the rate of change of the previously added cost function [9]. It is not known whether the situation where learning stopped (parameters change no longer occurs) is the optimal situation for the cost function. One of the important factors in optimization using gradient-based methods is a hyperparameter called the

learning rate (or step-size). If the norm of the gradient is large, the learning will not work well. If the norm of the gradient is small, the learning will be too slow [10]. In order to solve this problem, learning rate schedules have been introduced. The usual learning rate schedule is a constant that is multiplied by the gradient in the learning process. However, since the learning rate set initially is a constant, the gradient may not be scaled during learning. To solve this problem, other methods have been developed to schedule the learning rate, such as step-based and time-based methods, where it is not a constant but a function which becomes smaller as learning progresses [11–14]. Furthermore, adaptive optimization methods with the effect of changing the learning rate have been developed, such as Adagrad and RMSProp [15–22]. Currently, adaptive optimization methods and adaptive learning rate schedules are used in combination. However, the learning rate schedules used currently only decrease as learning progresses, which may be a factor in causing the learning to not proceed further at a local minimum [23–27]. In this paper, we introduce a method to increase the learning rate, depending on the situation, rather than just decreasing as the learning progresses. For our numerical experiments, we chose the most basic Gradient Descent (GD) method, the momentum method, and the most widely known Adam method as the basic methodologies for comparison. We computed the change of numerical value according to each methodology and each learning rate schedule.

This paper is organized as follows. Section 2 explains the existing learning methods. This section discusses the basic GD method, the momentum method, learning rate methods, and the Adam method, which is a combination of them. Finally, descriptions of some more advanced methods based on Adam (i.e., Adagrad, RMSProp, and AdaDelta) are given. Section 3 explains our method and proves its convergence. Section 4 demonstrates that our proposed method is superior to the existing methods through various numerical experiments. The numerical experiments in this section consist of Weber's function experiments, to test for changes in multidimensional space and local minima, binary classification experiments, and classification experiments with several classes [28].

2. Machine Learning Method

As explained in the previous section, machine learning is achieved through optimization of the cost function. Generally, the learning data are divided into input data and output data. Here, we denote the input data by x and the output data by y. Assuming that there are l training data, we conform the following data: $(x_1, y_1), \ldots, (x_l, y_l)$. That is, when the value of x_i is entered, the value of y_i is expected. The function $y_h(x)$ is created by an artificial neural network (ANN). The most basic is constructed as follows.

$$y_h(x) = \sigma(wx + b), \tag{1}$$

where σ is called an activation function, where a non-linear function is used. We want the y_h obtained from an artificial neural network to have the same value as y, and we can define the cost function as

$$C(w,b) = \frac{1}{l}\sum_{i}^{l}(y_h(x_i) - y_i)^2 \tag{2}$$

$$= \frac{1}{l}\sum_{i}^{l}(\sigma(wx_i + b) - y_i)^2. \tag{3}$$

The process of finding the parameters (w, b) which minimize this cost function C is machine learning. More detailed information can be found in [20].

2.1. Direction Method

This section introduces ways to minimize the cost function satisfying that

$$\frac{\partial C}{\partial w} = 0. \tag{4}$$

Machine learning is completed by determining the parameters that satisfy the Equation (4) [12,25,27,28].

2.1.1. Gradient Descent Method

GD is the simplest of the gradient-based methods. This method is based on the fixed iteration method in mathematics. The update rule of this method is as follows:

$$w_{i+1} = w_i - \eta \frac{\partial C}{\partial w}(w_i), \tag{5}$$

where η is a (constant) learning rate [12,25,27,28].

2.1.2. Momentum Method

Gradient-based optimization methods were designed under the assumption that the cost function is convex [10]. However, in general, the cost function is not a convex function. This causes problems with poor learning at local minima. The momentum method was introduced to solve this problem. The update rule of this method is as follows:

$$w_{i+1} = w_i - m_i, \tag{6}$$

$$m_i = \beta m_{i-1} + (1-\beta)\frac{\partial C(w_i)}{\partial w}, \tag{7}$$

where β is a constant [9].

2.2. Learning Rate Schedule

In gradient-based methods, one problem is that the parameters are not optimized when the size of the gradient is large. To solve this problem, we multiply the gradient by a small constant (called the learning rate) in the parameter update rule. However, in order to improve learning efficiency, methods for changing the value at each step, instead of using a constant learning rate, are used. We will introduce some learning rate schedule methods below.

2.2.1. Time-Based Learning Rate Schedule

The time-based learning rate schedule is the simplest learning rate schedule change rule. It has the form of a rational function whose denominator is a linear function:

$$\eta_{n+1} = \frac{\eta_n}{1+dn}, \tag{8}$$

where n is the iteration step, η_n is the learning rate at the n^{th} step, and d is the decay rate. This update rule changes the learning rate as learning progresses, reducing the denominator. As n starts at zero, 1 is added to prevent the denominator from being zero [11].

2.2.2. Step-Based Learning Rate Schedule

The update rule in the time-based learning rate schedule is a simple rational function. To increase its efficiency, update rules have been developed using exponential functions, as follows:

$$\eta_n = \eta_0 d^{floor\left(\frac{1+n}{r}\right)}, \tag{9}$$

where n is the iteration step, η_0 is the initial learning rate, d is the decay rate (which is set to a real number between 0 and 1), and r is the drop rate. Therefore, the step-based learning rate always decreases [13].

2.2.3. Exponential-Based Learning Rate Schedule

The step-based learning rate schedule is discrete, as it uses the floor function in its update rule. The exponential-based schedule is made continuous by using an exponential function without a floor function, as follows:

$$\eta_n = \eta_0 e^{-dn}, \tag{10}$$

where n is the iteration step, η_0 is the initial learning rate, and d is the decay rate [14]. A pseudocode version of this schedule is given in Algorithm 1.

Algorithm 1: Pseudocode of exponential-based learning rate schedule.

η_0: Initialize learning rate
d: Decay rate
$i \leftarrow 0$ (Initialize time step)
while
  $\eta_{i+1} \leftarrow \eta_0 e^{-di}$
  $i \leftarrow i+1$
end while
return η_i

2.3. Adaptive Optimization Methods

Gradient-based methods, such as those introduced in Section 2.1, only use a constant learning rate. To learn well in non-convex situations, methods such as the momentum method have been introduced, however, in order to increase the efficiency of learning, adaptive optimization methods have been developed. These methods have the effect of changing the learning rate in the optimization step. This section introduces some adaptive optimization methods.

2.3.1. Adaptive Gradient (Adagrad)

The Adagrad method is an adaptive method which changes the learning rate by dividing the learning rate by the l_2-norm of the gradient at each step. In the denominator, the square of the gradient at each step is added; therefore, as the learning progresses, the denominator becomes larger. The update rule of this method is as follows:

$$w_{i+1} = w_i - \eta \frac{g_i}{\sqrt{\sum_{j=0}^{i} g_j^2}}, \tag{11}$$

where η is the learning rate and g_i is the gradient at the i^{th} step. In this method, the learning rate in the n^{th} step η_n is $\eta / \sqrt{\sum_{j=0}^{i} g_j^2}$ [15].

2.3.2. Root Mean Square Propagation (RMSProp)

RMSProp is an adaptive method which changes the learning rate in each step by dividing the learning rate by the square root of the exponential moving average of the square of the gradient. The update rule of this method is as follows:

$$w_{i+1} = w_i - \eta \frac{g_i}{\sqrt{v_i}}, \tag{12}$$

$$v_i = \gamma\, v_{i-1} + (1-\gamma) g_i^2, \tag{13}$$

where η is the learning rate, g_i is the gradient at the i^{th} step, and γ is the exponential moving average (EMA) coefficient. In this method, the learning rate in the n^{th} step is η_n is $\eta / \sqrt{v_i + \epsilon}$ [16].

2.3.3. Adaptive Moment Estimation (Adam)

The Adam method was designed by combining Adagrad and RMSProp. It is the most widely used gradient-based method. The update rule of the Adam method is as follows:

$$w_{i+1} = w_i - \eta \frac{\hat{\mathbf{m}}_i}{\sqrt{\hat{\mathbf{v}}_i + \epsilon}}, \tag{14}$$

where

$$\mathbf{m}_i = \beta_1 \mathbf{m}_{i-1} + (1 - \beta_1) \frac{\partial C(w_i)}{\partial w}, \tag{15}$$

$$\mathbf{v}_i = \beta_2 \mathbf{v}_{i-1} + (1 - \beta_2) \left(\frac{\partial C(w_i)}{\partial w} \right)^2, \tag{16}$$

$\hat{\mathbf{m}}_i = \mathbf{m}_i / (1 - \beta_1^i)$, and $\hat{\mathbf{v}}_i = \mathbf{v}_i / (1 - \beta_2^i)$. The Adam method calculates the EMA for each of the gradients and the squares of the gradient and uses their ratios to adjust the learning rate [18–20]. The pseudocode version of this method is given in Algorithm 2.

Algorithm 2: Pseudocode of Adam method.

η: Learning rate
$\beta_1,\ \beta_2 \in [0,\ 1)$: Exponential decay rates for the moment estimates
$C(w)$: Cost function with parameters w
w_0: Initial parameter vector
$m_0 \leftarrow 0$
$v_0 \leftarrow 0$
$i \leftarrow 0$ (Initialize timestep)
while w not converged **do**
  $m_i \leftarrow \beta_1 \cdot m_{i-1} + (1 - \beta_1) \cdot \frac{\partial C}{\partial w}(w_i)$
  $v_i \leftarrow \beta_2 \cdot v_{i-1} + (1 - \beta_2) \cdot \frac{\partial C}{\partial w}(w_i)^2$
  $\hat{m}_i \leftarrow m_i / (1 - \beta_1^i)$
  $\hat{v}_i \leftarrow v_i / (1 - \beta_2^i)$
  $w_{i+1} \leftarrow w_i - \eta \cdot \hat{m}_i / (\sqrt{\hat{v}_i + \epsilon})$
  $i \leftarrow i + 1$
end while
return w_i (Resulting parameters)

3. The Proposed Method

The various algorithms introduced in Section 2 either do not change (constant) or only decrease the learning rate. These cannot be avoided when the parameter reaches a local minimum. Therefore, we introduce a new learning rate schedule using cost functions. The pseudocode version of proposed method is given in Algorithm 3.

Algorithm 3: Pseudocode of cost-based learning rate schedule.

η_0: Initialize learning rate
$C(w)$: Cost function with parameters w
$i \leftarrow 0$ (Initialize time step)
while
  $\eta_{i+1} \leftarrow \eta_0 C(w_i)$
  $i \leftarrow i+1$
end while
return η_i

The cost-based learning rate schedule can be used in conjunction with other (adaptive) optimization methods. This paper proves its convergence when used with the Adam method. Equation (15) is as follows:

$$w_{i+1} = w_i - \eta \frac{\hat{\mathbf{m}}_i}{\sqrt{\hat{\mathbf{v}}_i + \epsilon}}, \tag{17}$$

$$= w_i - \eta \frac{\mathbf{m}_i / (1-\beta_1^i)}{\sqrt{\mathbf{v}_i / (1-\beta_2^i) + \epsilon}}, \tag{18}$$

$$= w_i - \eta \frac{\sqrt{(1-\beta_2^i)}}{(1-\beta_1^i)} \frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i + \epsilon}}. \tag{19}$$

As β_1 and β_2 are 0.9 and 0.999, respectively, if i is large enough, then $\sqrt{(1-\beta_2^i)}\ /(1-\beta_1^i)$ can be regarded as 1 (see Appendix A.1 for details). Therefore, our proposed method is obtained by the following equation:

$$w_{i+1} = w_i - \eta_i \frac{m_i}{\sqrt{v_i + \epsilon}}, \tag{20}$$

where

$$\eta_i = \eta_0 C(w_i) \quad (\eta_0 \text{ is a constant}), \tag{21}$$

$$m_i = (1-\beta_1) \sum_{k=1}^{i} \beta_1^{k-1} \frac{\partial C(w_{i-k+1})}{\partial w}, \quad \text{and} \quad v_i = (1-\beta_2) \sum_{k=1}^{i} \beta_2^{k-1} \left(\frac{\partial C(w_{i-k+1})}{\partial w} \right)^2. \tag{22}$$

Here, we change the learning rate from the existing η to the function $\eta_i = \eta_0 C(w_i)$, such that the change of the cost function is reflected in optimization. From Equation (20), we can obtain the sequence $\{w_i\}$. See Appendix A.2 for the process of obtaining Equation (22). In this paper, we will show that our sequence $\{w_i\}$ converges and that the limit value of the sequence $\{w_i\}$ satisfies the minimum value of the cost function.

Lemma 1. *The relationship between m_i and v_i is*

$$(m_i)^2 \leq \left(\frac{1 - \left(\frac{\beta_1^2}{\beta_2} \right)^i}{1 - \left(\frac{\beta_1^2}{\beta_2} \right)} \right) v_i. \tag{23}$$

Proof. More details can be found in [20]. □

The Equation (20) is changed as

$$\Longrightarrow \qquad |w_{i+1} - w_i| \leq \eta_i \left(\frac{1 - \left(\frac{\beta_1^2}{\beta_2}\right)^i}{1 - \left(\frac{\beta_1^2}{\beta_2}\right)} \right)^{1/2}, \tag{24}$$

by Lemma 1 (the relationship between m_i and v_i). In this equation, ϵ is intended to avoid the denominator going to zero and can be regarded as not present in the calculation. In order for the value of the right part of the Equation (24) to converge to zero, the value of β_1^2/β_2 should be smaller than 1 and the sequence $\{\eta_i\}$ converges to zero. Therefore, we must show that the cost function goes to zero under the condition of the sequence which we have defined [20].

Theorem 1. *Judgement on whether or not machine learning is well constructed depends on how small the value of the cost function is. Therefore, it is important to find the parameter whose cost function approaches zero. Evaluated at the parameter defined by our proposed method, the cost function is going to zero.*

Proof. For sufficiently large τ, sufficiently small η_i, and using Taylor's theorem, we have the following calculation:

$$C(w_{i+1}) = C(w_i) + \frac{\partial C}{\partial w}(w_i)\,(w_{i+1} - w_i) + O\left((w_{i+1} - w_i)^2\right), \tag{25}$$

where $O\left((w_{i+1} - w_i)^2\right)$ represents the second order and is ignored. By Equations (20) and (25), we have

$$\begin{aligned} C(w_{i+1}) &= C(w_i) + \frac{\partial C}{\partial w}(w_i)\,(w_{i+1} - w_i) && (26)\\ &= C(w_i) - \eta_i \frac{\partial C}{\partial w}(w_i) \left(\frac{m_i}{(v_i + \epsilon)^{1/2}} \right) && (27)\\ &= C(w_i) \left(1 - \eta_0 \frac{\partial C}{\partial w}(w_i) \left(\frac{m_i}{(v_i + \epsilon)^{1/2}} \right) \right) && (28)\\ &= C(w_i) \left(1 - \eta_0 \left(\frac{\left(\frac{\partial C}{\partial w}(w_i)\right)^2 + \beta_1 \frac{\partial C}{\partial w}(w_i) m_{i-1}}{(v_i + \epsilon)^{1/2}} \right) \right). && (29) \end{aligned}$$

As $\beta_1 < 1$, $\epsilon \neq 0$ and adjusting η_0, we have that

$$0 < \eta_0 \left(\frac{\left(\frac{\partial C}{\partial w}(w_i)\right)^2 + \beta_1 \frac{\partial C}{\partial w}(w_i) m_{i-1}}{(v_i + \epsilon)^{1/2}} \right) < 1. \tag{30}$$

Therefore, we can obtain

$$C(w_i) \leq \delta^{i-\tau} C(w_\tau), \tag{31}$$

where $\delta = supremum_{i>\tau} \left\{ 1 - \eta_0 \left(\frac{\left(\frac{\partial C}{\partial w}(w_i)\right)^2 + \beta_1 \frac{\partial C}{\partial w}(w_i) m_{i-1}}{(v_i + \epsilon)^{1/2}} \right) \right\}$ and $\lim_{i \to \infty} C(w_i) = 0$. □

In Theorem 1, it is possible that δ is less than 1. Therefore, in the following Lemma, we explain in more detail why δ is less than 1.

Lemma 2. *The δ defined in Theorem 1 is less than 1.*

Proof. First of all, i is assumed to be a number such that τ is sufficiently large. Under the condition of sufficiently small η_i, the equation

$$\eta_0 \left(\frac{\left(\frac{\partial C}{\partial w}(w_i)\right)^2 + \beta_1 \frac{\partial C}{\partial w}(w_i) m_{i-1}}{(v_i + \epsilon)^{1/2}} \right),$$

can be divided into two parts, and can be computed as

$$0 < \left(\frac{\left(\frac{\partial C}{\partial w}(w_i)\right)^2}{(v_i + \epsilon)^{1/2}} \right) < 1 \tag{32}$$

and

$$\left(\frac{\beta_1 \frac{\partial C}{\partial w}(w_i) m_{i-1}}{(v_i + \epsilon)^{1/2}} \right). \tag{33}$$

The Equation (32) is obtained by an easy calculation. We must show that Equation (33) is less than 1. Under the assumption of this Lemma 2 and by the definition of m_i, we get $sign(m_i) = sign(\partial C(w_i)/\partial w)$ as $\beta_1 < 1$ and the sign does not change dramatically as $\partial C(w)/\partial w$ is a continuous function. Therefore, we have $sign(\partial C(w_i)/\partial w \;\; m_{i-1}) > 0$. By Equation (33), we get

$$\begin{aligned}
\beta_1 \frac{\partial C}{\partial w}(w_i) \frac{m_{i-1}}{(v_i + \epsilon)^{1/2}} &= \beta_1 \frac{\partial C}{\partial w}(w_i) \frac{m_{i-1}}{(v_{i-1} + \epsilon)^{1/2}} \left(\frac{(v_{i-1} + \epsilon)^{1/2}}{(v_i + \epsilon)^{1/2}} \right) && (34) \\
&\leq \beta_1 \eta_i \left| \frac{\partial C}{\partial w}(w_i) \right| \left(\frac{1 - \left(\frac{\beta_1^2}{\beta_2}\right)^{i-1}}{1 - \left(\frac{\beta_1^2}{\beta_2}\right)} \right)^{1/2} \left(\frac{(v_{i-1} + \epsilon)^{1/2}}{(v_i + \epsilon)^{1/2}} \right) && (35) \\
&\leq \beta_1 \eta_i \left| \frac{\partial C}{\partial w}(w_i) \right| \left(\frac{1}{1 - \left(\frac{\beta_1^2}{\beta_2}\right)} \right)^{1/2} < \beta_1. && (36)
\end{aligned}$$

If $sign(\partial C(w_i)/\partial w \;\; m_{i-1}) < 0$, then there is $\zeta \in [w_i, w_{i-1}]$ or $[w_{i-1}, w_i]$ such that $\partial C(\zeta)/\partial w \approx 0$. Therefore, $\partial C(w_i)/\partial w \approx 0$. Equation (33) is a small value, less than 1. By this process, we can set η_0 as a small constant and obtain the following result: $\delta = supremum_{i>\tau} \left\{ 1 - \eta_0 \left(\frac{\left(\frac{\partial C}{\partial w}(w_i)\right)^2 + \beta_1 \frac{\partial C}{\partial w}(w_i) m_{i-1}}{(v_i + \epsilon)^{1/2}} \right) \right\} < 1$. □

4. Numerical Tests

In this section, we detail the results of various numerical experiments. In each experiment, we compare the results of our proposed method with the numerical results of GD, momentum, and Adam with various learning rates. For the types of learning rates, we apply the constant, time-based, step-based, exponetial-based, and cost-based schedules described above.

4.1. Two-Variable Function Test Using Weber's Function

In this section, we look at how the parameters change (according to how we compare them) using a two-variable function, Weber's function. Weber's function is

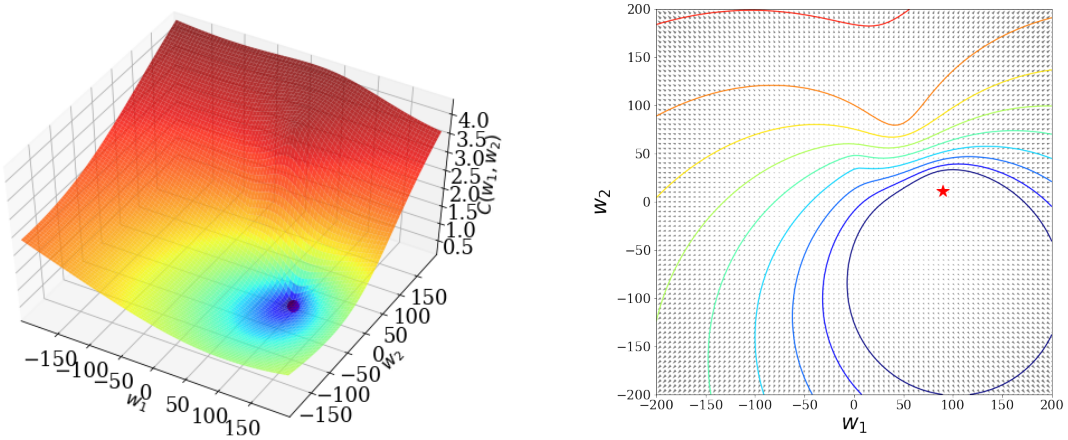
$$C(w_1, w_2) = \sum_{i=1}^{\kappa} \mu_i \sqrt{(w_1 - \nu_{i,1})^2 + (w_2 - \nu_{i,2})^2}, \tag{37}$$

where κ is integer, $\mu = (\mu_1, \cdots, \mu_\kappa)$ in $\mathbb{R}^\kappa$ such that $\sum_{i=1}^{\kappa} \mu_i > 0$, $\nu = (\nu_1, \cdots, \nu_\kappa)$, and ν_i is a vector in $\mathbb{R}^2$ [28].

4.1.1. Case 1: Weber's Function with Three Terms Added

In this case, we used Weber's function with the following variables as the cost function: $\kappa = 3$, $\mu = (2, 4, -5)$, $\nu_1 = (2, 42)$, $\nu_2 = (2, 42)$, and $\nu_3 = (43, 88)$. This Weber's function has a global minimum at $(90, 11)$.

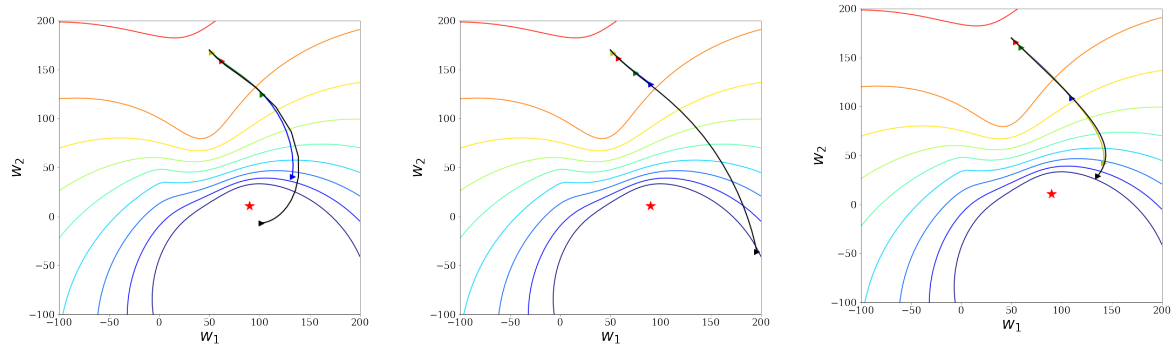
In this experiment, the initial values were $w_0 = (50, 170)$ and $\eta_0 = 5 \times 10^2$. Figure 1 shows the given Weber's function under these initial conditions. In Figure 1, the red point is its global minimum in R^3. Figure 2 visualizes the change of parameters, according to the optimization methods and the learning rates introduced in this paper. In Figure 2a, the GD method was used and the results of the changes in the learning rate are visualized. In Figure 2b, the momentum method was used and the results of the changes in the learning rate are visualized. In Figure 2c, the Adam method was used and the results of the changes in the learning rate are visualized. In particular, we stopped learning before reaching the global minimum, in order to compare the convergence rates of each method. In this experiment, there were no local minima and all the methods converged well to the global minimum. In this experiment, our cost-based learning rate schedule combined with the optimization methods showed the fastest convergence speed.



(**a**) 3D graph of the given function; global minimum presented as a red point.

(**b**) Level set of given function; global minimum presented as a red star.

Figure 1. Weber's function for given μ and ν.



(**a**) Comparison of the proposed method with other learning rate schedules using GD.

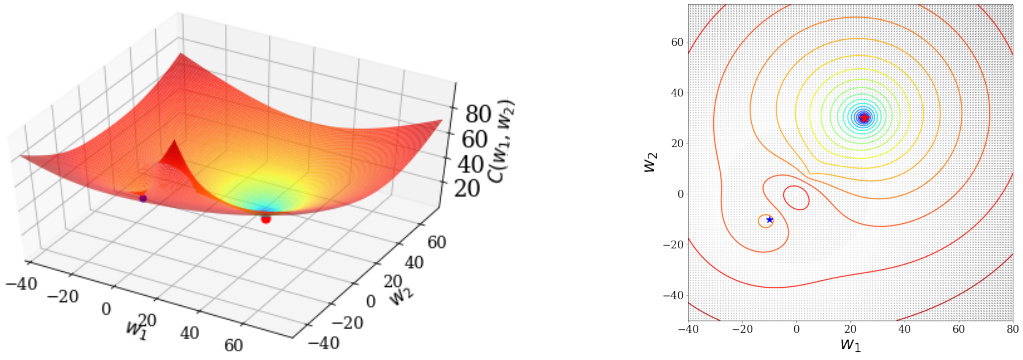
(**b**) Comparison of the proposed method with other learning rate schedules using Momentum.

(**c**) Comparison of the proposed method with other learning rate schedules using Adam.

Figure 2. Comparing methods with initial point $(50, 170)$ and initial learning rate 5×10^2. The blue line, the red line, the green line, the yellow line, and the black line represent the constant learning rate schedule, the time-based learning rate schedule, the step-based learning rate schedule, the exponential-based learning rate schedule, and the cost-based learning rate schedule, respectively.

4.1.2. Case 2: Weber's Function with Four Terms Added

In this case, we used the Weber's function with the following variables as the cost function: $\kappa = 4$, $\mu = (2, -4, 2, 1)$, $\nu_1 = (-10, -10)$, $\nu_2 = (0, 0)$, $\nu_3 = (5, 8)$, and $\nu_4 = (25, 30)$. This Weber's function had a local minimum at $(-10, -10)$ and a global minimum at $(25, 30)$. In Figure 3, the global minimum is represented by a red dot and the local minimum is represented by a blue dot. In this experiment, we wanted to see that each parameter produced by the learning rate constantly changed beyond the local minimum.

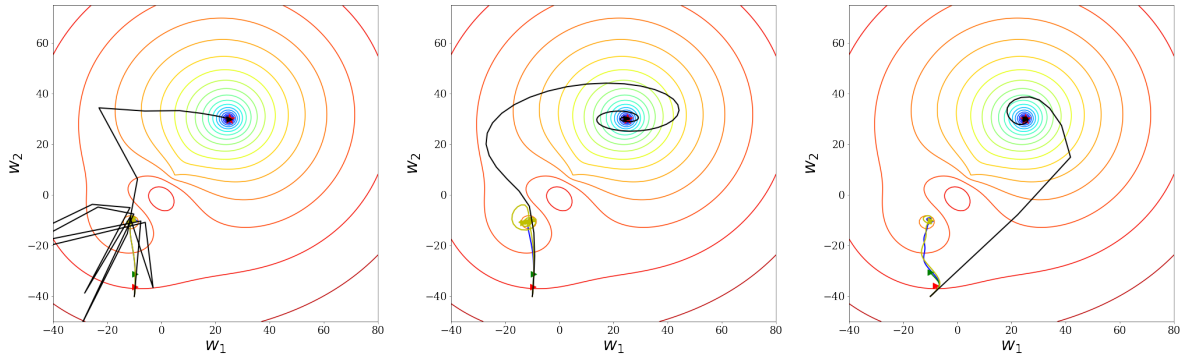


(**a**) 3D graph of given function; global minimum presented as a red point, local minimum presented as a blue point.

(**b**) Level set of given function; global minimum presented as a red star, local minimum presented as a blue star.

Figure 3. Weber's function for given μ and ν.

The initial values were $w_0 = (-10, -40)$ and $\eta_0 = 5 \times 10^{-1}$. Figure 4 visualizes the change of parameters according to the optimization methods and the learning rates introduced in this paper. In Figure 4a, the GD method was used and the results of the changes in the learning rate are visualized. In Figure 4b, the momentum method was used and the results of the changes in the learning rate are visualized. In Figure 4c, the Adam method was used and the results of the changes in the learning rate are visualized. In all optimization methods, we can see that only the cost-based learning rate schedule caused convergence beyond the local minimum to the value of the global minimum.



(**a**) Comparison of the proposed method with other learning rate schedule using GD.

(**b**) Comparison of the proposed method with other learning rate schedule using Momentum.

(**c**) Comparison of the proposed method with other learning rate schedule using Adam.

Figure 4. Comparing methods with initial point $(-10, -40)$ and initial learning rate 5×10^{-1}. The blue line, the red line, the green line, the yellow line, and the black line represent the Constant learning rate schedule, the time-based learning rate schedule, the step-based learning rate schedule, the exponential-based learning rate schedule, and the cost-based learning rate schedule, respectively.

4.2. Binary Classification with Multilayer Perceptron (MLP)

In this section, a Binary Classification Problem (BCP) was implemented for each method using a MLP model. The BCP was based on a 20×20 size image data set with $\circ$ and $\times$ draw. Figure 5 shows some images from the data set used in this experiment.

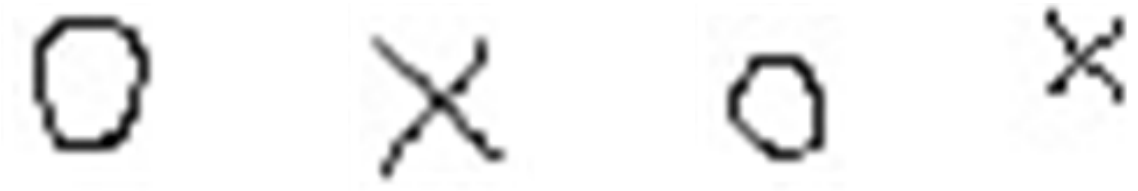
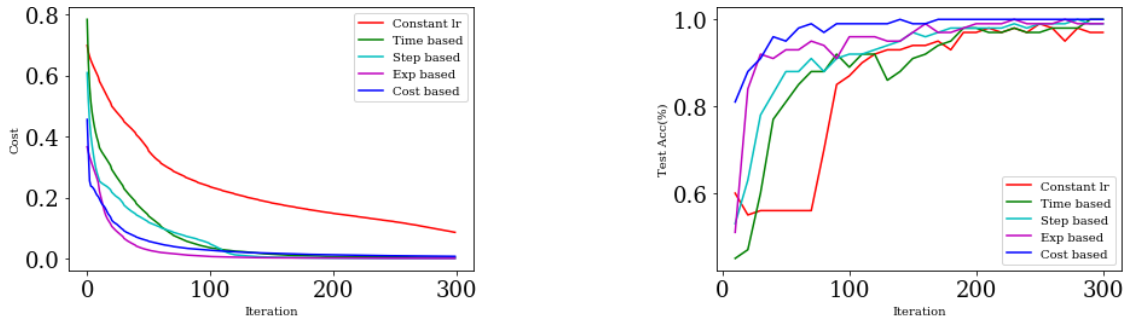


Figure 5. Examples of data set used in this experiment.

The experiment was based on the Adam method in a six-layer MLP model, with different learning rate methods. The initial learning rate was set as 0.001 and the number of iterations was 300.

Figure 6a shows the change in cost for each learning rate schedule during learning. Figure 6b shows the change in accuracy with each learning rate schedule during learning. As shown in Figure 6a, all methods quickly reduced cost (except for the constant learning rate). Unfortunately, the cost-based learning rate schedule had a high cost when learning; however, it was superior to other methods in terms of accuracy, as shown in Figure 6b.



(**a**) Training cost of each learning rate schedule.

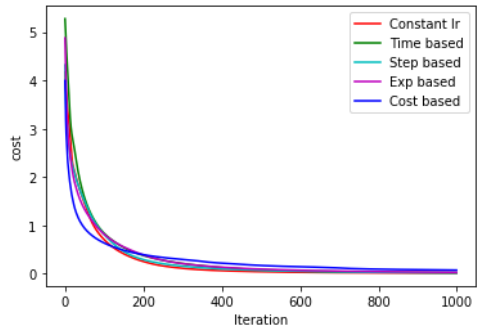
(**b**) Test accuracy of each learning rate schedule.

Figure 6. Result of binary classification with multilayer perceptron (MLP).

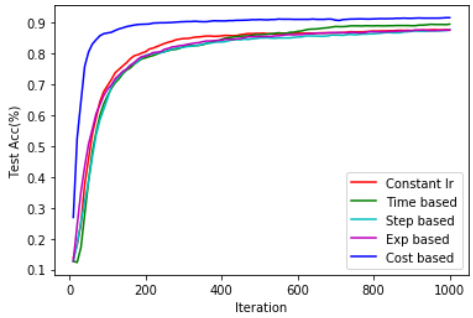
4.3. MNIST with MLP

In the next experiment, the MLP was used to solve a multi-classification problem using the MNIST data set, in order to verify the performance of each learning rate schedule. The images in the MNIST data set are grayscale numbered images (from 0 to 9) of size 32×32. The experiment was based on the Adam method in a five-layer MLP model with each learning rate schedule. The initial learning rate was set as 0.001 and the number of iterations was 1000.

Figure 7a shows the change in cost for each learning rate schedule during learning. Figure 7b shows the change in accuracy with each learning rate schedule during learning. The cost and accuracy were not much different from the results in the previous section; the cost-based learning rate schedule was the fastest and had the highest accuracy.



(**a**) Training cost of each learning rate schedule.

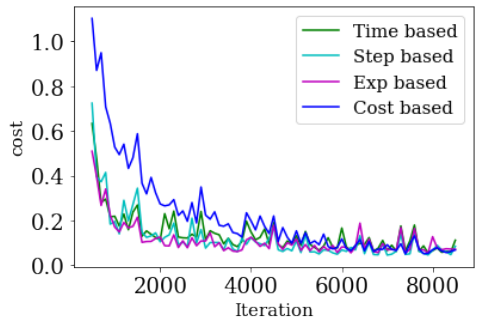


(**b**) Test accuracy of each learning rate schedule.

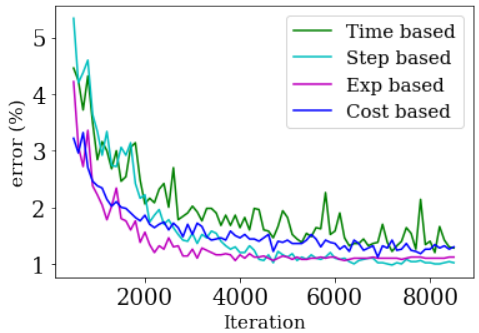
Figure 7. Result of multi-classification with MLP.

4.4. MNIST with Convolutional Neural Network (CNN)

In this experiment, we sought to confirm that the proposed method has good performance when used in a deep learning framework. The performance of each method was measured using a Convolutional Neural Network (CNN) model for multi-class classification problems using the MNIST data set (the same as in Section 4.3). In this experiment, a CNN model with a first convolution filter of size $5 \times 5 \times 32$, a second convolution filter of size $5 \times 5 \times 32 \times 64$, and a hidden layer of size 512 were used. In addition, the batch size was 64 and 50% dropout was used. The results are shown in Figure 8.



(**a**) Training cost of each learning rate schedule.



(**b**) Validation error of each learning rate schedule.

Figure 8. Result of multi-classification with Convolutional Neural Network (CNN).

Figure 8a shows the change in cost for each learning rate schedule during learning. Figure 8b shows the change in validation error with each learning rate schedule during learning. It can be seen that all methods had no underfitting, due to the low training cost, and no overfitting, due to low validation error. Figure 8a shows that, at a given setting, the cost did not become relatively small, even though learning was progressing well. However, as the cost-based learning rate schedule is

proportional to the cost, the learning rate is determined by the scale of the cost function when learning. Therefore, in this experiment, the cost-based learning rate schedule was slow; however, this can be solved (to some extent) by multiplying the cost function by a small constant. In this case, it was seen that the cost-based learning rate schedule was sensitive to the initial values of the cost function and the parameters.

5. Conclusions

This paper explains how the degree of learning varies with the learning rate in ANN-based machine learning. In particular, in general machine learning, local minima can cause problems in learning. In order to solve this problems, the existing methods used a monotonically decreasing learning rate at a constant rate according to the iteration time. However, these methods often lead to unexpected results that the cost function constructed by learning data is not approaching zero. In this paper, the machine learning methods worked well by use of our proposed learning rate schedule, which changes the learning rate appropriately.

By utilizing the value of the cost function in determining the change in the learning rate, we have the effect of stopping learning only at the global minimum of the cost function. The global minimum means that the cost we define is zero. Through our proposed method, the cost function converges to zero and in particular, it shows outstanding results in terms of learning accuracy. Our method has a characteristic that the learning rate is adaptive, and in particular, it has the advantage that it can be applied to existing learning methods by simply change the formula. The shortcoming of our method is if the cost of the initial parameters is high then the parameters do not learn well. However, this can be solved by multiplying the cost function by a constant smaller than 1.

Author Contributions: Conceptualization, D.Y. and S.J.; Data curation, S.J.; Formal analysis, D.Y.; Funding acquisition, D.Y.; Investigation, J.P.; Methodology, D.Y.; Project administration, D.Y.; Resources, J.P.; Software, S.J.; Supervision, J.P.; Validation, S.J.; Visualization, S.J.; Writing—original draft, J.P. and D.Y.; Writing—review & editing, J.P. and S.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the basic science research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number NRF-2017R1E1A1A03070311).

Acknowledgments: We sincerely thank anonymous reviewers whose suggestions helped improve and clarify this manuscript greatly.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1

From Equation (17), we obtain following equation

$$w_{i+1} = w_i - \eta \frac{\sqrt{(1-\beta_2^i)}}{(1-\beta_1^i)} \frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i + \epsilon}}, \tag{A1}$$

where β_1 and β_2 are constant values between 0 and 1. In general, β_1 and β_2 use 0.9 and 0.999 respectively. This implies that $lim_{i\to\infty}\beta_t = 0$, for all $t = 0, 1$. Thus,

$$\lim_{i\to\infty} \frac{\sqrt{(1-\beta_2^i)}}{(1-\beta_1^i)} = \frac{\sqrt{(1-\lim_{i\to\infty}\beta_2^i)}}{(1-\lim_{i\to\infty}\beta_1^i)}, \tag{A2}$$

$$= \frac{\sqrt{(1-0)}}{(1-0)}, \tag{A3}$$

$$= 1. \tag{A4}$$

Appendix A.2

In order to obtain Equation (22), m_i and v_i were defined as follows:

$$\mathbf{m}_i = \beta_1 \mathbf{m}_{i-1} + (1-\beta_1)\frac{\partial C(w_i)}{\partial w}, \tag{A5}$$

$$\mathbf{v}_i = \beta_2 \mathbf{v}_{i-1} + (1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2. \tag{A6}$$

where m_0 and v_0 are 0.

$$\begin{aligned}
\mathbf{m}_i &= \beta_1 \mathbf{m}_{i-1} + (1-\beta_1)\frac{\partial C(w_i)}{\partial w} && \text{(A7)}\\
&= \beta_1\left(\beta_1 \mathbf{m}_{i-2} + (1-\beta_1)\frac{\partial C(w_{i-1})}{\partial w}\right) + (1-\beta_1)\frac{\partial C(w_i)}{\partial w}, && \text{(A8)}\\
&= \beta_1^2 \mathbf{m}_{i-2} + \beta_1(1-\beta_1)\frac{\partial C(w_{i-1})}{\partial w} + (1-\beta_1)\frac{\partial C(w_i)}{\partial w}, && \text{(A9)}\\
&= \beta_1^2\left(\beta_1 \mathbf{m}_{i-3} + (1-\beta_1)\frac{\partial C(w_{i-2})}{\partial w}\right) + \beta_1(1-\beta_1)\frac{\partial C(w_{i-1})}{\partial w} + (1-\beta_1)\frac{\partial C(w_i)}{\partial w}, && \text{(A10)}\\
&= \beta_1^3 \mathbf{m}_{i-3} + \beta_1^2(1-\beta_1)\frac{\partial C(w_{i-2})}{\partial w} + \beta_1(1-\beta_1)\frac{\partial C(w_{i-1})}{\partial w} + (1-\beta_1)\frac{\partial C(w_i)}{\partial w}, && \text{(A11)}\\
&\vdots && \text{(A12)}\\
&= \beta_1^i m_0 + (1-\beta_1)\sum_{k=1}^{i-1}\beta_1^{k-1}\frac{\partial C(w_{i-k+1})}{\partial w}, && \text{(A13)}\\
&= (1-\beta_1)\sum_{k=1}^{i-1}\beta_1^{k-1}\frac{\partial C(w_{i-k+1})}{\partial w}. && \text{(A14)}
\end{aligned}$$

$$\begin{aligned}
\mathbf{v}_i &= \beta_2 \mathbf{v}_{i-1} + (1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2 && \text{(A15)}\\
&= \beta_2\left(\beta_2 \mathbf{v}_{i-2} + (1-\beta_2)\frac{\partial C(w_{i-1})}{\partial w}\right) + (1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2, && \text{(A16)}\\
&= \beta_2^2 \mathbf{v}_{i-2} + \beta_2(1-\beta_2)\frac{\partial C(w_{i-1})}{\partial w} + (1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2, && \text{(A17)}\\
&= \beta_2^2\left(\beta_2 \mathbf{v}_{i-3} + (1-\beta_2)\frac{\partial C(w_{i-2})}{\partial w}\right) + \beta_2(1-\beta_2)\frac{\partial C(w_{i-1})}{\partial w} + (1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2, && \text{(A18)}\\
&= \beta_2^3 \mathbf{v}_{i-3} + \beta_2^2(1-\beta_2)\frac{\partial C(w_{i-2})}{\partial w} + \beta_2(1-\beta_2)\frac{\partial C(w_{i-1})}{\partial w} + (1-\beta_2)\left(\frac{\partial C(w_i)}{\partial w}\right)^2, && \text{(A19)}\\
&\vdots && \text{(A20)}\\
&= \beta_2^i v_0 + (1-\beta_2)\sum_{k=1}^{i-1}\beta_2^{k-1}\left(\frac{\partial C(w_i)}{\partial w}\right)^2, && \text{(A21)}\\
&= (1-\beta_2)\sum_{k=1}^{i-1}\beta_2^{k-1}\left(\frac{\partial C(w_i)}{\partial w}\right)^2. && \text{(A22)}
\end{aligned}$$

By Equation (A14) and Equation (A22), we obtain Equation (A23) and Equation (A24):

$$\mathbf{m}_i = (1-\beta_1)\sum_{k=1}^{i-1}\beta_1^{k-1}\frac{\partial C(w_{i-k+1})}{\partial w}, \tag{A23}$$

$$\mathbf{v}_i = (1-\beta_2)\sum_{k=1}^{i-1}\beta_2^{k-1}\left(\frac{\partial C(w_i)}{\partial w}\right)^2. \tag{A24}$$

References

1. Bishop, C.M.; Wheeler, T. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
2. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
3. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press Cambridge: London, UK, 2016.
4. Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Le, Q.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; et al. Large scale distributed deep networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems—NIPS 2012, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1223–1231.
5. Amari, S. Natural gradient works efficiently in learning. *Neural Comput.* **1998**, *10*, 251–276. [CrossRef]
6. Hinton, G.; Deng, L.; Yu, D.; Dahl, G.E.; Mohamed, A.; Jaitly, N.; Senior, A.; Vanhoucke, V.; Nguyen, P.; Sainath, T.N. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Process. Mag.* **2012**, *29*, 82–97. [CrossRef]
7. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Neural Information Processing Systems–NIPS 2012, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
8. Pascanu, R.; Bengio, Y. Revisiting natural gradient for deep networks. *arXiv* **2013**, arXiv:1301.3584.
9. Sutskever, I.; Martens, J.; Dahl, G.; Hinton, G.E. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning—ICML 2013, Atlanta, GA, USA, 16–21 June 2013; pp. 1139–1147.
10. Forst, W.; Hoffmann, D. *Optimization—Theory and Practice*; Springer: Berlin/Heidelberg, Germany, 2010.
11. Nesterov, Y. *Introductory Lectures on Convex Optimization: A Basic Course*; Springer: Berlin/Heidelberg, Germany, 2004.
12. Bengio, Y. Practical recommendations for gradient-based training of deep architectures. *arXiv* **2012**, arXiv:1206.5533.
13. Ge, R.; Kakade, S.M.; Kidambi, R.; Netrapalli, P. The Step Decay Schedule: A Near Optimal, Geometrically Decaying Learning Rate Procedure For Least Squares. *arXiv* **2019**, arXiv:1904.12838.
14. Li, Z.; Arora, S. An Exponential Learning Rate Schedule for Deep Learning. *arXiv* **2019**, arXiv:1910.07454.
15. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
16. Tieleman, T.; Hinton, G.E. *Lecture 6.5—RMSProp, COURSERA: Neural Networks for Machine Learning*; Technical Report; University of Toronto: Toronto, ON, Canada, 2012.
17. Zeiler, M.D. Adadelta: An adaptive learning rate method. *arXiv* **2012**, arXiv:1212.5701.
18. Kingma, D.P.; Ba, J. ADAM: A method for stochastic optimization. In Proceedings of the 3rd International Conference for Learning Representations—ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
19. Reddi, S.J.; Kale, S.; Kumar, S. On the Convergence of ADAM and Beyond. *arXiv* **2019**, arXiv:1904.09237.
20. Yi, D.; Ahn, J.; Ji, S. An Effective Optimization Method for Machine Learning Based on ADAM. *Appl. Sci.* **2020**, *10*, 1073. [CrossRef]
21. Kochenderfer, M.; Wheeler, T. *Algorithms for Optimization*; The MIT Press Cambridge: London, UK, 2019.
22. Bottou, L.; Curtis, F.E.; Nocedal, J. Optimization Methods for Large-Scale Machine Learning. *SIAM Rev.* **2018**, *60*, 223–311. [CrossRef]
23. Roux, N.L.; Fitzgibbon, A.W. A fast natural newton method. In Proceedings of the 27th International Conference on Machine Learning—ICML 2010, Haifa, Israel, 21–24 June 2010; pp. 623–630.

24. Sohl-Dickstein, J.; Poole, B.; Ganguli, S. Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. In Proceedings of the 31st International Conference on Machine Learning—ICML 2014, Beijing, China, 21–24 June 2014; pp. 604–612.
25. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
26. Becker, S.; LeCun, Y. *Improving the Convergence of Back-Propagation Learning with Second Order Methods*; Technical Report; Department of Computer Science, University of Toronto: Toronto, ON, Canada, 1988.
27. Kelley, C.T. Iterative methods for linear and nonlinear equations. In *Frontiers in Applied Mathematics*; SIAM: Philadelphia, PA, USA, 1995.
28. Kelley, C.T. Iterative Methods for Optimization. In *Frontiers in Applied Mathematics*; SIAM: Philadelphia, PA, USA, 1999.