mathematics

MDPI

Article

Adaptive Levenberg–Marquardt Algorithm: A New Optimization Strategy for Levenberg–Marquardt Neural Networks

Zhiqi Yan, Shisheng Zhong *, Lin Lin and Zhiquan Cui *

Department of Mechanical Engineering, Harbin Institute of Technology, Harbin 150000, China; cumtmaris@163.com (Z.Y.); waiwaiyl@163.com (L.L.)
* Correspondence: zhongss@hit.edu.cn (S.Z.); xiaocui2002yan@163.com (Z.C.)

Abstract: Engineering data are often highly nonlinear and contain high-frequency noise, so the Levenberg–Marquardt (LM) algorithm may not converge when a neural network optimized by the algorithm is trained with engineering data. In this work, we analyzed the reasons for the LM neural network's poor convergence commonly associated with the LM algorithm. Specifically, the effects of different activation functions such as Sigmoid, Tanh, Rectified Linear Unit (RELU) and Parametric Rectified Linear Unit (PRLU) were evaluated on the general performance of LM neural networks, and special values of LM neural network parameters were found that could make the LM algorithm converge poorly. We proposed an adaptive LM (AdaLM) algorithm to solve the problem of the LM algorithm. The algorithm coordinates the descent direction and the descent step by the iteration number, which can prevent falling into the local minimum value and avoid the influence of the parameter state of LM neural networks. We compared the AdaLM algorithm with the traditional LM algorithm and its variants in terms of accuracy and speed in the context of testing common datasets and aero-engine data, and the results verified the effectiveness of the AdaLM algorithm.

Keywords: Levenberg–Marquardt algorithm; convergence; neural networks; local minima; optimization

Citation: Yan, Z.; Zhong, S.; Lin, L.; Cui, Z. Adaptive Levenberg–Marquardt Algorithm: A New Optimization Strategy for Levenberg–Marquardt Neural Networks. *Mathematics* **2021**, *9*, 2176. https://doi.org/10.3390/math9172176

Academic Editors: Higinio Rubio Alonso, Alejandro Bustos Caballero, Jesus Meneses Alonso and Enrique Soriano-Heras

Received: 16 August 2021
Accepted: 1 September 2021
Published: 6 September 2021

1. Introduction

When applied to real-world data interspersed with high nonlinearity and high-frequency noise, LM neural networks have irreplaceable advantages. They reduce the requirement of computing resources and guarantee high convergence speed, making their performance superior to other existing regression methods. The LM neural network is an artificial neural network with an LM optimizer. The LM algorithm is a second-order method for solving general least squares problems. As an optimizer, the LM algorithm has wide engineering applications in LM neural networks because of its fast convergence and small memory occupation [1–3].

However, the disadvantage of the LM algorithm makes LM neural networks have potential symptoms of divergence or convergence to a bad local minimum. The paper [4] reported that the LM algorithm may become stuck and fail to degrade the cost function. For a nonconvex optimization problem, such as neural network optimization, there can be many local minima. The cost function may end up in the "bad" local minima when it is difficult to guarantee the nonsingularity of the Jacobian matrix and Lipschitz continuity [5]. LM neural networks show higher errors than do regular neural networks in practical engineering application [6].

Most of the recent work have tried to help the LM algorithm find one of the "good" local minima by combining with other algorithms for better engineering applications, because it is generally acceptable in the modern neural network community that searching for a global minimum is often an unnecessary endeavor, e.g., the genetic algorithm (GA)

can be used to optimize the initial weight of LM neural networks [7,8], and Bayesian estimation can be used to calibrate model variables. In Refs. [9,10], the wavelet method was used to preprocess data and the LM neural networks method was used to analyze them. In Ref. [11], the least squares support vector machine (LSSVM) and LM were combined into a hybrid model. The above hybrid methods have been proved to be effective by experiments.

Several other research groups have looked into improving the LM algorithm. Yang [12] presented a high-order LM method that has biquadratic convergence. Chen [13] presented a fourth-order method called the accelerated modified LM method. Derakhshandeh [14] presented a new three-step LM (TSLM) algorithm based on fuzzy logic theory (FLT). In Ref. [15], an adaptive Levenberg–Marquardt (LM) algorithm-based echo state network was proposed by adding a new adaptive damping term to the LM algorithm. These works proved the validity of the modified method algorithm by a trust region technique.

However, hybrid methods indicate a lot of computation, which weakens the advantage of the LM algorithm. The problem with improved algorithms is that their performance improvement is conditionally limited. These improved algorithms prove to be effective only under specific prerequisites:

(1). $F(w)$ and $f(w)$ are nonlinearly continuous and differentiable. The $f(w)$ is a neural network model, w is a parameter of the neural network model, and $F(w)$ is the cost function of the neural network. In Refs. [12–15], the global optimization ability of the proposed algorithm based on nonlinearly continuous and differential $F(w)$ and $f(w)$ was verified. In detail, the activation functions in the neural networks are not necessarily continuous and differentiable, so the above references are only valid in the current setting without proving that the global optimization can be achieved in the neural network model.

(2). $J = \partial F(w)/\partial w$. J is a Jacobian matrix. The Jacobian matrix in Refs. [16–18] was obtained by directly solving the system of nonlinear equations. The calculation results were relatively accurate, but the calculation process was very complex. In the neural networks, the Jacobian matrix participating in the proof process was obtained by back-propagation rather than by the derivation of the objective function. Therefore, the theory of the above references may be not suitable for the field of neural networks.

Rather, a new adaptive LM algorithm is proposed for neural networks in this paper. We explain in detail the specific factors that cause the cost function to fall into bad local minima by the original LM algorithm, by analyzing the output performance of several activation functions. In view of these factors, the new algorithm makes up for the deficiency of the LM algorithm to train a network efficiently.

The remaining parts are organized as follows. Section 2 introduces the LM algorithm. Section 3 proves that the original LM algorithm has the possibility of falling into local minima in neural networks. Section 4 gives the optimization of the LM algorithm. Section 5 verifies the correctness of this theory about experiments, and the conclusions are drawn in last section.

2. Explanation and Problem of LM Algorithms

2.1. LM Algorithm

The LM algorithm was first proposed for nonlinear least squares problems by scholars Levenberg and Marquardt [19]. Before introducing the LM and carrying out research, some parameters need to be defined, which are the same as those of [20].

Definition 1. *Given a neural network model f(w), the cost function is the least squares problem:*

$$F(w) = \Sigma(y_{\text{label}} - f(w))^2 \tag{1}$$

where w represents neural network parameters and y_{label} represents label data.

Definition 2. *A model L is assumed to describe the behavior of F in the current iteration*

$$L(h) \approx F(w+h) \equiv F(w) + h^{\mathrm{T}} J^{\mathrm{T}} f(w) + {}^{1}/_{2} h^{\mathrm{T}} J^{\mathrm{T}} J h \tag{2}$$

Definition 3. *There is a method to evaluate the accuracy of model L, called gain ratio q:*

$$q = \frac{F(w) - F(w+h)}{L(0) - L(h)} \tag{3}$$

LM method minimizes $F(w)$ by iteration. h is introduced to describe the change direction of $F(w)$.

Definition 4. *h is a descent direction for F(w) at w, then F(w + h) < F(w).*

$$h = -\left(J^{\mathrm{T}} J + \mu I\right)^{-1} J r \tag{4}$$

where J is the Jacoby matrix, r is the cost function error, and μ is the damping parameter.

Given a variable v, the damping parameter μ is adjusted by v in every iteration. The μ and v updating formulas are illustrated below [20]:

(1). If $q < 0$, $\mu = \mu \times v$, $v = v \times 2$;
(2). If $q > 0$, $\mu = \mu \times \max\{{}^{1}/_{3}, 1 - (2q-1)^3\}$.

2.2. Problem of LM Algorithm

The problem of the LM algorithm is from the gain ratio q. A positive q means $L(h)$ is a good approximation to $F(w+h)$, and vice versa. The $L(0) - L(h)$ has proved to be positive [20], and its proof is:

$$\begin{aligned}
L(0) - L(h) &= -h^{\mathrm{T}} J^{\mathrm{T}} f - \tfrac{1}{2} h^{\mathrm{T}} J^{\mathrm{T}} J h \\
&= -\tfrac{1}{2} h^{\mathrm{T}} \left(2g + (J^{\mathrm{T}} J + \mu I - \mu I) h\right) \\
&= \tfrac{1}{2} h^{\mathrm{T}} (\mu h - g) \\
&\text{where, } g = J^{\mathrm{T}} f \\
h^{\mathrm{T}} h > 0, -h^{\mathrm{T}} g &= -h^{\mathrm{T}} (J^{\mathrm{T}} f) = -\left(-\left(J^{\mathrm{T}} J + \mu I\right)^{-1} g\right)^{\mathrm{T}} g \\
&= g^{\mathrm{T}} \left(\left(J^{\mathrm{T}} J + \mu I\right)^{-1}\right)^{\mathrm{T}} g > 0 \\
L(0) - L(h) &= \tfrac{1}{2} h^{\mathrm{T}} (\mu h - g) > 0
\end{aligned} \tag{5}$$

In Definition 3, we know $q = (F(w) - F(w+h))/(L(0) - L(h))$, which indicates whether $L(h)$ is a good approximation to $F(w+h)$ dependences on $F(w) - F(w+h)$. As just said, if $q > 0$, $L(h)$ is a good approximation. If $q < 0$, the damping parameter μ in Equation (4) is adjusted and the current iteration is recalculated.

There is a situation where q will always remain negative regardless of how the damping parameter μ in Equation (4) is adjusted, while the cost function $F(w)$ in Equation (1) is still very large.

That situation indicates that iteration cannot continue, and that LM neural networks have fallen into the "bad" local minima or had divergence.

3. Analysis of Falling into "Bad" Local Minima

In the previous section, the possibility of the LM algorithm was analyzed. In this section, we focus on the case of where q would always remain negative and analyze the possible situation. If we find the condition where gain ratio $q < 0$ regardless of how the

parameters are adjusted, we can prove that LM neural networks may diverge or converge to the poor local minima.

The proof process includes three steps: First, given the inequality $f(w) + J(w) < r(w^2)$, the conditions satisfying the inequality are analyzed. Secondly, we prove that "$\exists f(w)$, when $f(w) + J(w) < r(w^2)$, in Equation (3), the gain ratio $q < 0$". The conditions leading to $q < 0$ can be obtained. Thirdly, based on the above analysis, we discuss the conditions under which LM neural networks may diverge or converge to a worse local minimum.

Definition 5. *According to the structure of the neural network, the residual function is as follows:*

$$f(w) = \sigma(wx + b) - y_{\text{label}} \tag{6}$$

where σ is an activation function, x is the input of the neural network, the weight of the neural network is w, and the bias is b.

Definition 6. *The first-order Taylor expansion for the output function f(w + h) of the neural network at h is:*

$$f(w + h) = f(w) + J(w)h + r(w^2) \tag{7}$$

where $r(w^2)$ is the remainder.

Lemma 1. *∃{w,b,h}, when the absolute values of training data x and label data y of the neural network are large enough, $f(w) + J(w) < r(w^2)$.*

Proof. According to Equation (7):

$$r(w^2) = f(w + h) - f(w) - J(w)h \tag{8}$$

Substituting Equation (8) into $f(w) + J(w) < r(w^2)$ gives:

$$f(w + h) - 2f(w) - 2J(w)h > 0 \tag{9}$$

Thus, to prove $f(w) + J(w) < r(w^2)$, you only need to prove Equation (9). By Equation (6), we obtain:

$$\begin{aligned} f(w + h) &= \sigma(wx + hx + b) - y_{label} \\ J(w) &= \sigma'(wx + b) \end{aligned} \tag{10}$$

Substituting Equation (6) and Equation (10) into Equation (9) gives:

$$\sigma(wx + hx + b) - 2\sigma(wx + b) - 2\sigma'(wx + b)h + y_{label} > 0 \tag{11}$$

Let:

$$G(w, h) = \sigma(wx + hx + b) - 2\sigma(wx + b) - 2\sigma'(wx + b)h + y_{label} \tag{12}$$

Thus, to prove $f(w) + J(w) < r(w^2)$, you only need to prove:

$$G(w, h) = \sigma(wx + hx + b) - 2\sigma(wx + b) - 2\sigma'(wx + b)h + y_{label} > 0 \tag{13}$$

In order to simplify the calculation, define:

$$w' = wx + b, h' = hx \tag{14}$$

Then:

$$\begin{aligned}
&\sigma(w') = \sigma(wx + b)\\
&\sigma(w' + h') = \sigma(wx + hx + b)\\
&\sigma'(w')h' = \sigma'(wx + b)\frac{\partial w}{\partial w'}h' = \sigma'(wx + b)\frac{h'}{x} = \sigma'(wx + b)h\\
&G(w', h') = \sigma(w' + h') - 2\sigma(w') - 2\sigma'(w')h' + y_{label} =\\
&\qquad \sigma(wx + hx + b) - 2\sigma(wx + b) - 2\sigma'(wx + b)h + y_{label}\\
&\qquad = G(w, h)
\end{aligned} \tag{15}$$

The frequently used activation functions include the sigmoid function, tanh function, ReLU function, and PReLU function. Discussion:

(1). When σ is sigmoid:

$$\begin{aligned}
&\sigma(w' + h') = \tfrac{1}{1+e^{-w'-h'}}\\
&\sigma(w') = \tfrac{1}{1+e^{-w'}}\\
&\sigma'(w') = \tfrac{e^{-w'}}{1+e^{-w'}}
\end{aligned} \tag{16}$$

Then, the function $G(w',h')$ is:

$$G(w', h') = \frac{1}{1 + e^{-w'-h'}} - \frac{2}{1 + e^{-w'}} - \frac{2he^{-w'}}{1 + e^{-w'}} + y_{label} \tag{17}$$

Due to the arbitrariness of the value of y_{label}, the first three terms of the function $G(w',h')$ are considered and expressed by $g(w',h')$:

$$g(w', h') = G(w', h') - y_{label} = \frac{1}{1 + e^{-w'-h'}} - \frac{2}{1 + e^{-w'}} - \frac{2he^{-w'}}{1 + e^{-w'}} \tag{18}$$

The image of $g(w',h')$ in Equation (18) drawn by OCTAVE is shown in Figure 1:

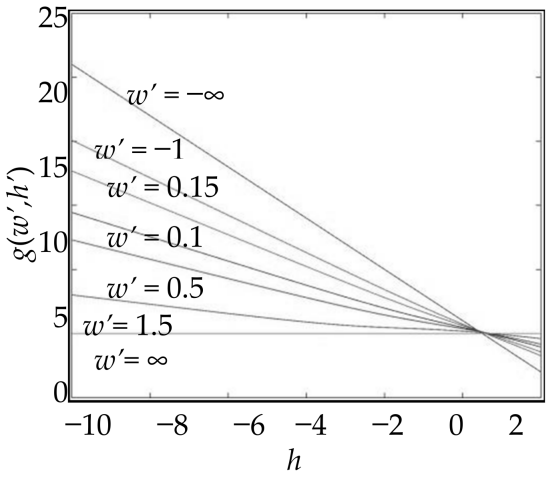


Figure 1. Regardless of the value of w', the $g(w',h')$ curve monotonically decreases and intersects the x-axis at the point (0.5, 0). This means that when the conditions (σ = sigmoid, $h' \leq 0.5$, and $y_{\text{label}} > 0$) are met, the algorithm has the possibility of divergence.

From Figure 1, it can be seen that the $g(w',h')$ curve is monotonously decreasing and converges at point ($h' = 0.5$, $g(w',h') = 0$); when $h' \leq 0.5$, $g(w',h') \geq 0$.That is:

$$\begin{aligned} &g(w',h')|_{h'\le 0.5} = G(w',h')|_{h'\le 0.5} - y_{label} \ge 0 \\ &G(w',h')|_{h'\le 0.5} \ge y_{label} \\ &G(w,h) = G(w',h')|_{h'\le 0.5} \ge y_{label} \end{aligned} \tag{19}$$

Thus, when $h' \le 0.5$ and $y_{label} > 0$, $G(w,h) > 0$. As $h' = hx$, for the arbitrariness of h, when the absolute values of x and y_{label} are large enough, we have $G(w,h) > 0$, that is, $f(w) + J(w) < r(w^2)$.

(2). When σ is tanh:

Similarly, let: $w' = wx + b$, $h' = hx$, then:

$$\begin{aligned} &\sigma(w'+h') = \frac{e^{w'+h'} - e^{-w'-h'}}{e^{w'+h'} + e^{-w'-h'}} \\ &\sigma(w') = \frac{e^{w'} - e^{-w'}}{e^{w'} + e^{-w'}} \\ &\sigma'(w') = \frac{4}{2 + e^{-2w'} + e^{2w'}} \end{aligned} \tag{20}$$

Then, the function $G(w',h')$ is:

$$G(w',h') = \frac{2}{1+e^{2(w'+h')}} - \frac{4}{1+e^{2w'}} - \frac{4h}{2+e^{2w'}+e^{-2w'}} - 1 + y_{label} \tag{21}$$

For the arbitrariness of the value of y_{label}, the first three terms of the function $G(w',h')$ are considered and expressed by $g(w',h')$:

$$\begin{aligned} g(w',h') &= G(w',h') - y_{label} \\ &= \frac{2}{1+e^{2(w'+h')}} - \frac{4}{1+e^{2w'}} - \frac{4h}{2+e^{2w'}+e^{-2w'}} - 1 \end{aligned} \tag{22}$$

The image of $g(w',h')$ in Equation (21) drawn by OCTAVE is shown in Figure 2:

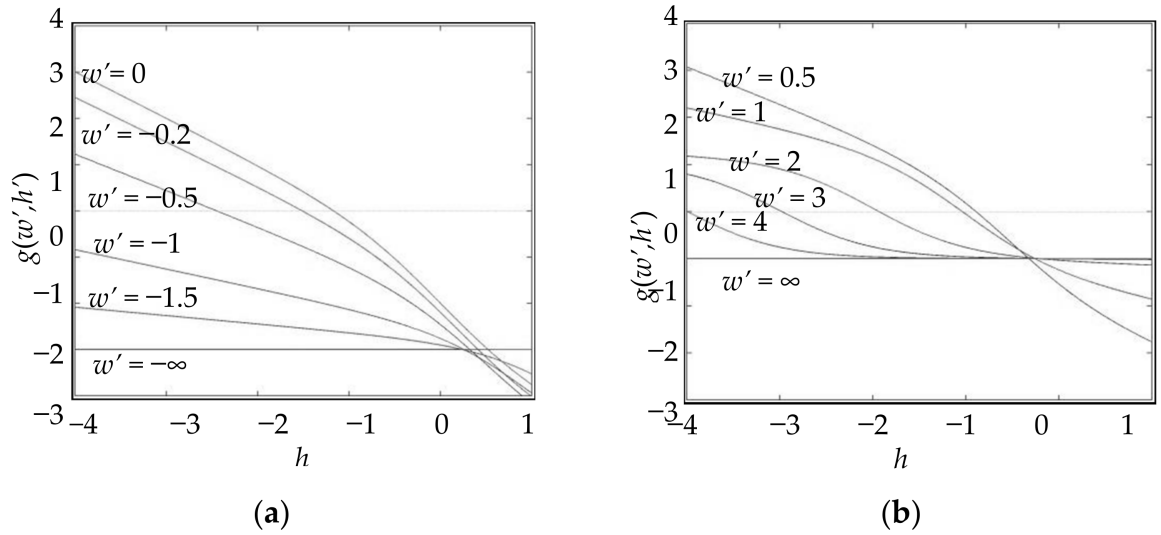


Figure 2. When σ = tanh, the curve $g(w',h')$ monotonically decreases: (**a**) when $h' < 0.3$ and $w < 0$, $\lim_{w\to\infty} g(w',h') = -3$; (**b**) when $h' < -0.3$ and $w > 0.5$, $\lim_{w\to\infty} g(w',h') = -1$. When the conditions ($\sigma$ = tanh, $h' < 0.3$, $w' < 0$, and $y_{\text{label}} > 3$) are met, the algorithm has the possibility of divergence.

In Figure 2, the curve $g(w',h')$ is also monotonically decreasing. When $h' < 0.3$, $g(w',h') \ge -3$, and so:

$$\begin{aligned} &g(w',h')|_{h'\geq -3} = G(w',h')|_{h'\geq -3} - y_{label} \geq -3 \\ &G(w',h')|_{h'\geq -3} \geq y_{label} - 3 \\ &G(w,h) = G(w',h')|_{h'\geq -3} \geq y_{label} - 3 \end{aligned} \tag{23}$$

Thus, when $h' < 0.3$ and $y_{\text{label}} > 3$, $G(w,h) > 0$. As $h' = hx$, for the arbitrariness of h, when the absolute values of x and y_{label} are large enough, we have $G(w,h) > 0$, that is, $f(w) + J(w) < r(w^2)$.

(3). When σ is ReLU:

Similarly, let: $w' = wx + b$, $h' = hx$, so:

$$\begin{aligned} &\sigma(w' + h') = \max(0, w' + h') \\ &\sigma(w') = \max(0, w') \\ &\sigma'(w') = \max(0, 1) \end{aligned} \tag{24}$$

then, $G(w',h')$ is:

$$G(w',h') = \max(0, w' + h') - \max(0, 2w') - \max(0, 2h) + y_{label} \tag{25}$$

Due to the arbitrariness of the value of y_{label}, the first three terms of the function $G(w',h')$ are considered and expressed by $g(w',h')$. The image of $g(w',h')$ is shown in Figure 3:

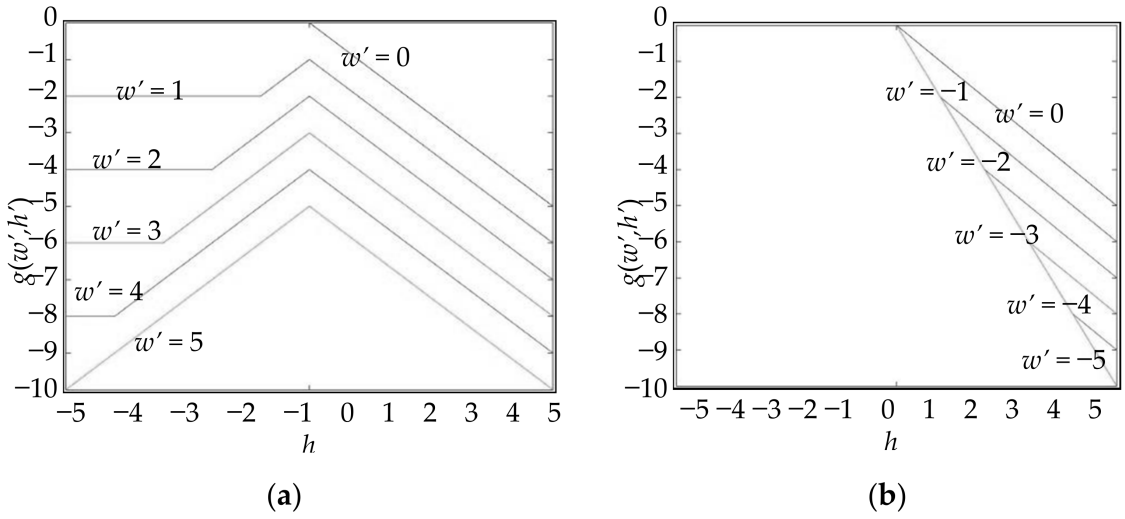


Figure 3. The curve $g(w',h')$ at σ = ReLU: (**a**) $w' > 0$; (**b**) $w' < 0$. When the conditions (σ = ReLU, $h' < 0.3$, $w' < 0$, and $y_{\text{label}} > 0$) are met, the algorithm has the possibility of divergence.

In Figure 3, when $w' < 0$ and $h' < 0$, $g(w', h') \geq 0$, and so:

$$\begin{aligned} &g(w',h')\big|_{w'<0,h'<0} = G(w',h')\big|_{w'<0,h'<0} - y_{label} \geq 0 \\ &G(w',h')\big|_{w'<0,h'<0} \geq y_{label} \\ &G(w,h) = G(w',h')\big|_{w'<0,h'<0} \geq y_{label} \end{aligned} \tag{26}$$

Thus, when $w' < 0$, $h' < 0$, and $y_{label} > 0$, $G(w,h) > 0$. As $h' = hx$ and $w' = wx + b$, for the arbitrariness of h, w, b, when the absolute values of x and y_{label} are large enough, we have $G(w,h) > 0$, that is, $f(w) + J(w) < r(w^2)$.

(4). When σ is PReLU:

Similarly, let: $w' = wx + b$ and $h' = hx$, and draw the image of $g(w',h')$ as shown in Figure 4:

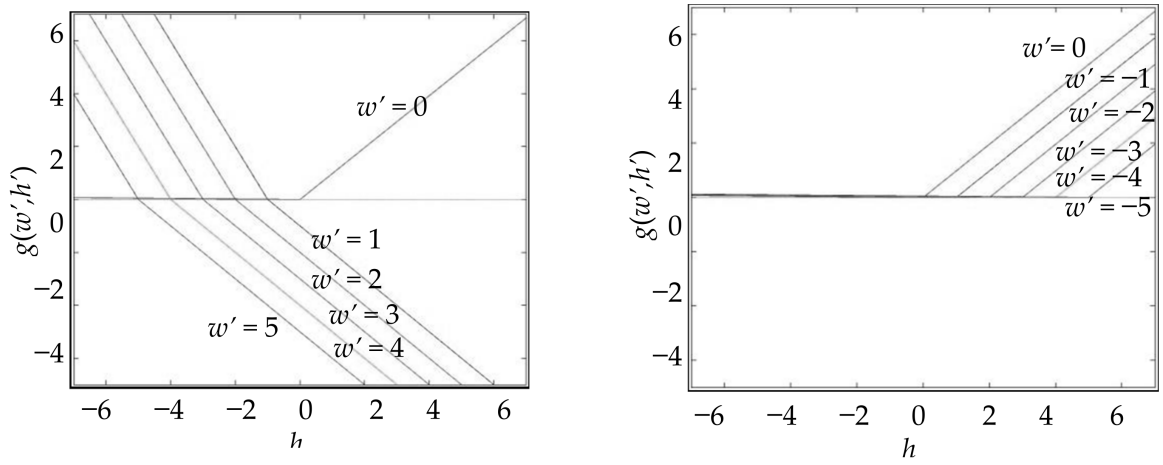


Figure 4. The curve $g(w',h')$ at σ = PReLU. (**a**) $w' > 0$; (**b**) $w' < 0$. When the conditions (σ = PReLU, $h' < 0$, $w' < 0$, and $y_{label} > 0$) are met, the algorithm has the possibility of divergence.

In Figure 4, when $w' < 0$, $g(w',h') > 0$, and so:

$$\begin{aligned} &g(w',h')\big|_{w'<0,h'<0} = G(w',h')\big|_{w'<0,h'<0} - y_{label} > 0 \\ &G(w',h')\big|_{w'<0,h'<0} > y_{label} \\ &G(w,h) = G(w',h')\big|_{w'<0,h'<0} > y_{label} \end{aligned} \tag{27}$$

Thus, when $w' < 0$ and $y_{\text{label}} > 0$, $G(w,h) > 0$. As $h' = hx$ and $w' = wx + b$, for the arbitrariness of h,w,b, when the absolute values of x and y_{label} are large enough, we have $G(w,h) > 0$, that is, $f(w) + J(w) < r(w^2)$.

To sum up, when the absolute values of x and y_{label} are large enough, there are $f(w) + J(w) < r(w^2)$. □

Lemma 2. *$\exists$ f(w), when f(w) + J(w) < r(w²), in Equation (3), the gain ratio q < 0.*

Proof. The cost function is: $F(w+h) = \frac{1}{2}\|f(w+h)\|^2$. The first-order Taylor expansion for $F(w+h)$ at w is:

$$F(w+h) = F(w) + h^{\mathrm{T}}J^{\mathrm{T}}f + \frac{1}{2}h^{\mathrm{T}}J^{\mathrm{T}}Jh + R(w) = L(h) + R(w) \tag{28}$$

As $f(w) + J(w) < r(w^2)$, the $r(w^2)$ term of the residual function $f(w+h)$ after Taylor's first-order expansion cannot be omitted. Therefore, there exists $f(w)$ that makes it impossible to omit the $R(w)$ term of Taylor's first-order expansion for $F(w+h)$. This means: $L(w) < R(w)$.

$$\because L(h) = L(0) + \Delta L \tag{29}$$

In [19], it has been proven that:

$$\begin{aligned} &\Delta L > 0, L(0) \simeq F(w) = \tfrac{1}{2}\|f(w)\|^2 > 0 \\ &\therefore \Delta L < L(h) < R(w) \\ &\therefore q = \tfrac{F(w)-F(w+h)}{L(0)-L(h)} \\ &\quad = \tfrac{F(w)-F(w)-h^T J^T f-\frac{1}{2}h^T J^T Jh-R(w)}{\Delta L} \\ &\quad = \tfrac{\Delta L-R(w)}{\Delta L} < 0 \end{aligned} \tag{30}$$

□

Lemma 3. *In the case of Lemma 1 and Lemma 2, the LM algorithm causes cost functions to fall into "bad" local minima.*

Proof. In the kth iteration, we have $q_k < 0$; in the case of Lemma 2, according to the LM algorithm in the original, when $q_k < 0$, the trust region will be reduced, that is:

$$\begin{aligned} v &= v \times 2 \\ \mu &= \mu \times v \\ h &= -\left(J^{\mathrm{T}}J + \mu I\right)^{-1} Jr \end{aligned} \tag{31}$$

At this point, the absolute value of step h decreases, and its sign remains unchanged. There is a situation where q is always negative when h is at any point in the positive or negative half-axis, and h will approach zero infinitely and the cost function cannot decrease. At this time, there is:

$$q < 0, v \to +\infty, h \to 0 \tag{32}$$

According to Equation (14), $h' = hx$, we have:

$$q < 0, v \to +\infty, \frac{h'}{x} \to 0 \tag{33}$$

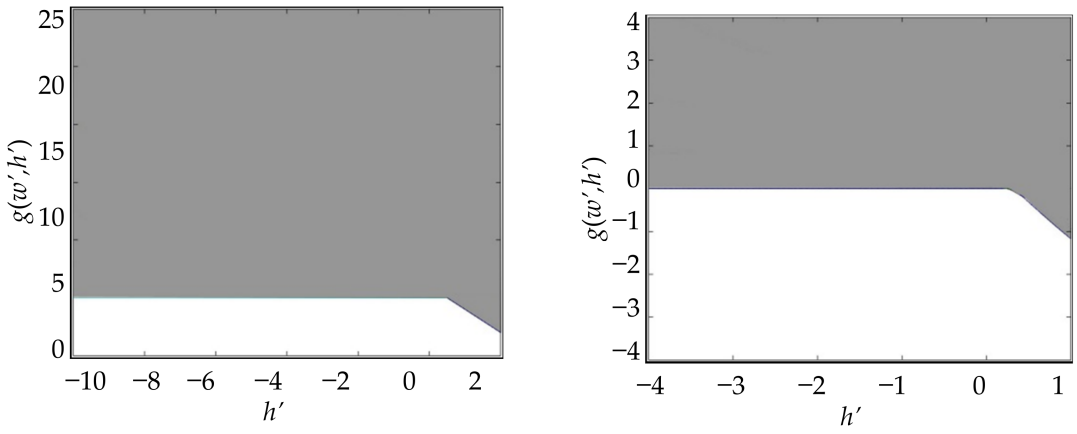
We analyze the common activation functions one by one to find out whether they have the possibility of this situation:

For the activation function σ = sigmoid, when $y_{\text{label}} > 1$, the area covered by the function is shown in Figure 5a. It can be seen that $g(w',h')$ will remain positive when h' is at any point on the negative half of the x-axis. In the case of Lemma 1 and Lemma 2, it is known that "$g(w',h')$ remains positive" means "q remains negative." According to Equation (14), $h' = hx$, so when $y_{\text{label}} > 1$ and $hx < 0$, q remains negative, and the LM algorithm will stop running, and the LM neural networks will also diverge or converge poorly.

Similarly, for the activation function σ = tanh, when $w \leq 0$ and $y_{\text{label}} > 3$, the area covered by the function is shown in Figure 5b. When $w > 0$ and $y_{\text{label}} > 1$, the area covered by the function is shown in Figure 5c. Thus, when $y_{\text{label}} > 3$, $x > 0$, and $hx < 0$, the LM neural network will diverge or converge poorly.

Similarly, for the activation function σ = ReLU, when $w < 0$ and $y_{\text{label}} > 0$, the area covered by the function is shown in Figure 5d. When $w < 0$, $y_{\text{label}} > 0$, $x > 0$, and $hx < 0$, the LM neural networks will diverge or converge poorly.

Finally, for the activation function σ = PReLU, when $w < 0$ and $y_{\text{label}} > 0$, the area covered by the function is shown in Figure 5d. When $y_{\text{label}} > 0$ and $w < 0$, the LM neural network will diverge or converge poorly. □



(**a**) (**b**)

Figure 5. *Cont.*

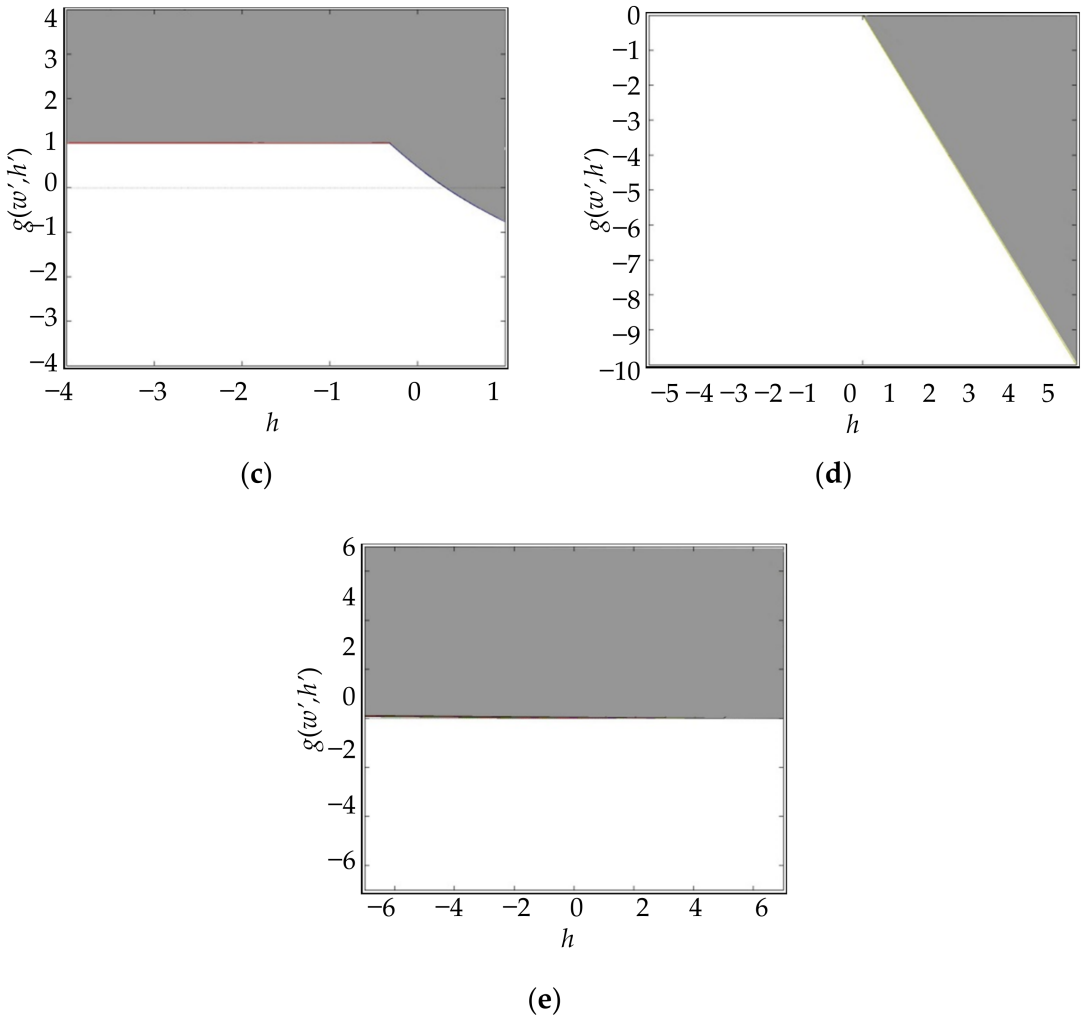


Figure 5. The gray part is the coverage area of $g(w',h')$: (**a**) The grey area covered by the $g(w',h')$ function when σ = sigmoid and $y_{\text{label}} > 1$. If $h' \leq 0.5$, $g(w',h')$ is always positive; (**b**) The grey area covered by the $g(w',h')$ function when σ = tanh, $y_{\text{label}} > 3$, and $w \leq 0$. If $h' \leq 0.3$, $g(w',h')$ is always positive; (**c**) The grey area covered by the $g(w',h')$ function when σ = tanh, $y_{\text{label}} > 1$, and $w > 0$. If $h' \leq 0$, $g(w',h')$ is always positive; (**d**) The grey area covered by the $g(w',h')$ function when σ = ReLU, $y_{\text{label}} > 0$, and $w < 0$. If $h' \leq 0$, $g(w',h')$ is always positive; (**e**) The grey area covered by the $g(w',h')$ function when σ = PReLU, $y_{\text{label}} > 0$, and $w < 0$. $g(w',h')$ is always positive.

4. The Proposed Algorithm: AdaLM

It can be seen from Section 3 that the parameter state of the model will affect the stability of the LM algorithm. In order to reduce the influence of the state of the model on the algorithm, the improved scheme starts from other algorithms to participate in the iterative operation, and then slowly transitions to the LM algorithm. Therefore, a more stable algorithm is needed, and the steepest descent method meets this requirement.

With this idea, this paper proposed the AdaLM algorithm that is close to the steepest descent method at the beginning and close to the LM algorithm after several iterations.

The core step calculation of the AdaLM algorithm can be expressed by Equation (34):

$$\begin{aligned} A &= J^{\mathrm{T}} J \\ g &= J^{\mathrm{T}} e \\ h_k &= -\tfrac{1}{1+e^{k-10}} (A + \mu I)^{-1} g - \tfrac{1}{1+e^{10-k}} \tfrac{g}{\|g\|} \end{aligned} \tag{34}$$

where A = J^TJ, α is the inertia parameter, V is the accumulated value of h, and k is the current number of iterations. The damping coefficient μ is calculated from q and v:

(1). If $q < 0$, $\mu = (\mu + 1) \times v$, $v = v \times 2$;
(2). If $q > 0$, $\mu = 0$, $v = 2$.

If AdaLM reaches the global minimum: $k \to \infty$ and $F(x_\infty) = g = 0$. Given $\varepsilon_1 > 0$, the first stop criteria can be set to:

$$\|g\|_\infty \leq \varepsilon_1 \tag{35}$$

Consistent with Reference [20], there is a second stop condition if the step size is very small:

$$\| h_k \| \leq \varepsilon_2(\|x\| + \varepsilon_2) \tag{36}$$

Finally, in all iterations, we need a protection against infinite loops, so there is a third stop condition:

$$k \geq k_{\max} \tag{37}$$

Thus, the AdaLM algorithm can be described in Algorithm 1:

Algorithm 1 AdaLM algorithm.

```
Begin
  k = 0; v = 2; x = x0; μ = 0;
  A = J^T J; g = J^T f(x);
  While (||g||∞ ≥ ε1) and (k < kmax)
  {
    k = k + 1;
    hk = −(A + μI)^−1 g/(1 + exp(k −10)) − g/||g||(1 + exp(k −10));
    if (||hk|| ≤ ε2(||x|| + ε2))
    break;
    else
    {
      xnew = x + hk;
      q = (F(x) − F(xnew))/(L(0) − L(hk));
      if (q > 0)
      {
        x = x new;
        A = J^T J; g = J^T f(x);
        if (||g||∞ ≤ ε1)
        break;
        μ =0; v = 2;
        }
      else
      {
          }μ = (μ + 1) × v; v = v × 2;
          }
}
}
End
```

5. Case Studies

5.1. Experimental Preparation

In order to evaluate the performance of AdaLM, it was arranged to compare with the original LM algorithm, as well as other popular LM variants, including High order Levenberg–Marquardt (HLM) and Three Step Levenberg–Marquardt (TSLM). Each algorithm is described in Table 1. In this paper, each algorithm and neural network were combined to build a prediction model, and the performance level of the algorithm was reflected by testing the prediction ability of the model.

Table 1. Description table of other LM family algorithms participating in the comparative test.

Algorithm	Full Name	Description
LM	LM	$\mathrm{LM}(\Delta x_k) = [J^{\mathrm{T}}(x_k)J(x_k) + \mu I]^{-1} J(x_k)e(x_k)$
HLM	High order LM	$\mathrm{HLM}(\Delta x_k) = \mathrm{LM}(\Delta x_k) + \mathrm{LM}(\Delta x_{k+1}) + \mathrm{LM}(\Delta x_{k+2})$
TSLM	Three step LM	$\mathrm{TSLM}(\Delta x_k) = \mathrm{LM}(\Delta x_k) + \alpha\mathrm{LM}(\Delta x_{k+1}) + \mathrm{LM}(\Delta x_{k+2})$

As for the neural network model, it has been proven that the three-layer neural network can simulate any complex nonlinear mapping if there are enough neurons in the hidden layer [21–23]. Therefore, the neural network models involved in this paper were all three-layer neural networks. We found that all the models involved in this paper will converge before 50 steps of iteration. Thus, we set the maximum number of iterations of the model to 50. In this study, the mean absolute error (MAE) was used to give the accuracy of the experimental results. MAE is the average value of absolute error, which can reflect well the actual situation of the predicted value error. Its formula is as follows:

$$\mathrm{MAE}(f(x), y) = \frac{1}{n}\sum_{i=1}^{n}|f(x_i) - y_i| \tag{38}$$

The number of nodes in the model affects the quality of the neural network. The input and output nodes of the model need to be set according to the requirements of the task. The number of hidden nodes of the model can be calculated according to the empirical formula given in [24]:

$$m = \sqrt{n \times l} \tag{39}$$

where m is the number of hidden layer nodes, n is the number of input layer nodes, and l is the number of output layer nodes.

The neural network model is shown in Figure 6. The model uses the square difference formula to calculate the error between the prediction data and the label data.

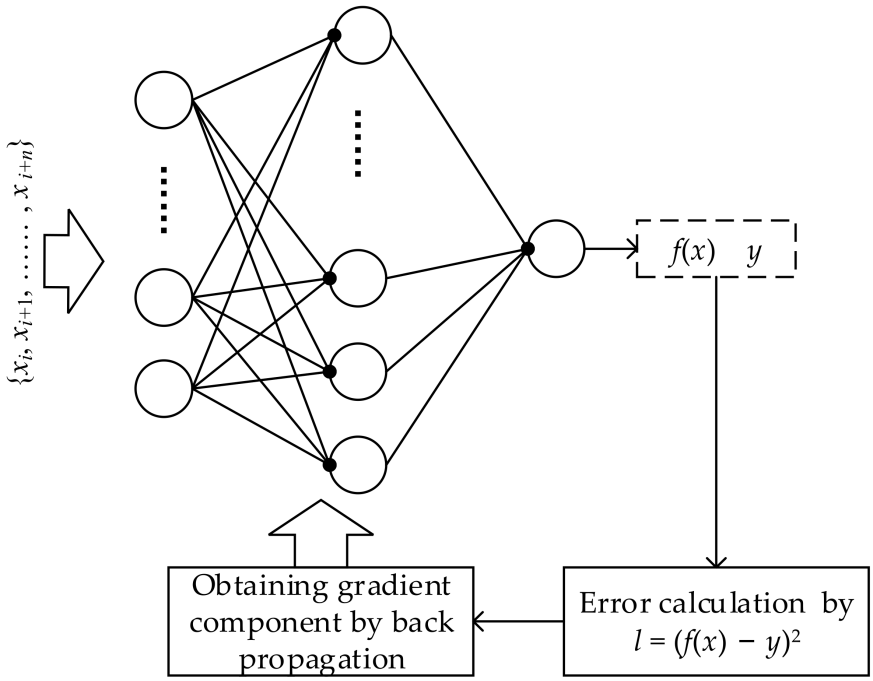


Figure 6. Neural network model.

In order to test the engineering application ability of each algorithm, the real prediction task of aero-engine performance parameters was arranged in this paper. In this paper, fuel flow data of an aero-engine were selected as training data. There is a large noise in the fuel flow data of the aero-engine due to environmental conditions, operation conditions, flight tasks, maintenance measures, etc. [25]. In Figure 7, the fuel flow data collected from the CFM56-5B engine records the fuel consumption rate of the A320 aircraft in pounds per hour.

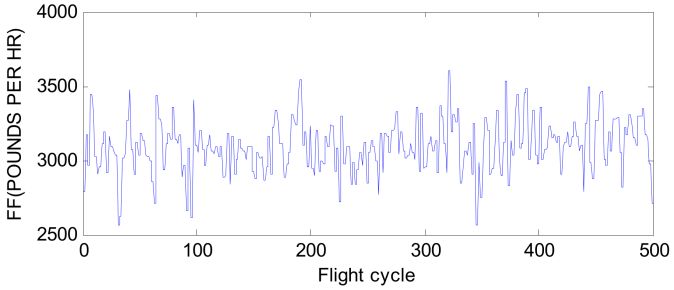


Figure 7. Aeroengine fuel flow data.

It is shown from Figure 7 that the data were highly nonlinear with many outliers. The data sample size was small. Therefore, the selection of the data can reflect the robustness and convergence of the algorithm. The specific implementation steps of the data prediction task for the fuel flow of the aeroengine are as follows:

Step 1: Obtain fuel flow data of the aeroengine. Set the total amount of data to N. Use normalization and data smoothing to preprocess the fuel flow data of the aeroengine, and then obtain the data $\boldsymbol{X} = (x_1, x_2, x_3, \ldots \ldots, x_N)$. The purpose of normalization is to avoid numerical problems such as overflow and underflow, reducing the impact of model initialization, and improving the prediction accuracy. The purpose of data smoothing is to reduce the influence of noise and outliers on the model. The data smoothing method is given by Equation (40):

$$\mathrm{MAE}(f(x), y) = \frac{1}{n}\sum_{i=1}^{n}|f(x_i) - y_i| \tag{40}$$

Step 2: Organize data into datasets. In this step, the dataset is generated according to the number of input layer nodes n of neural network, and the determination method of n is given in step 5. The fuel flow data are divided into subsequences with length of n as the input data of the model and the adjacent element of the subsequence as the label data. Because this task is a one-step forecast, the label data dimension is 1. Input data $\boldsymbol{X}$ and label data $\boldsymbol{Y}$ are, respectively:

$$\begin{gathered}
\{x_1, x_2, \ldots \ldots, x_n\} \rightarrow \{x_{n+1}\} \\
\{x_2, x_3, \ldots \ldots, x_{n+1}\} \rightarrow \{x_{n+2}\} \\
\cdots \\
\{x_i, x_{i+1}, \ldots \ldots, x_{i+n-1}\} \rightarrow \{x_{i+n}\} \\
\cdots \\
\{x_{N-n}, x_{N-n+1}, \ldots \ldots, x_{N-1}\} \rightarrow \{x_N\}
\end{gathered} \tag{41}$$

Step 3: Split the dataset $\{\boldsymbol{X},\boldsymbol{Y}\}$ into training set $\{\boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}}\}$ and test set $\{\boldsymbol{X}_{\text{test}}, \boldsymbol{Y}_{\text{test}}\}$. According to experience, the first 95% of the data are used as training sets. The rest are the test set.

Step 4: Build the neural network model and use the training set $\{\boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}}\}$ to train the model. The model uses the square difference formula to calculate the error between the prediction data and the label data.

Step 5: Determine the number of nodes in the neural network model to make the model optimal. The enumeration method is used to determine the input node n of the

neural network, with the range of 5−12. In detail, steps 2 to 4 are looped 8 times, and the input node n of the neural network is set to 5 to 12 in step 2 in each loop. The number of hidden layer nodes is determined by Equation (39), and the number of output nodes is 1. The MAE error in Equation (38) is used to describe the prediction accuracy of the experimental results. The number of input nodes and the corresponding model MAE are shown in Figure 8.

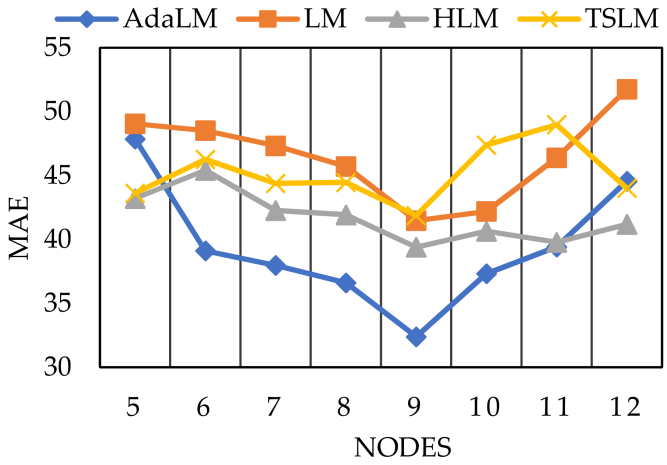


Figure 8. MAE of models with different neural network nodes.

Figure 8 shows that when the number of input nodes n is 9, the output errors of the algorithm LM reach the minimum, as well as HLM, TSLM, and AdaLM.

Step 6: Test the model. Input the test set $\boldsymbol{X}_{\text{test}}$ into the input model to obtain $f(\boldsymbol{X}_{\text{test}})$.

5.2. The Influence of Each Algorithm on Activation Function

The purpose of this section is to evaluate the impact of the four algorithms on the activation function to evaluate the performance of each algorithm. In this paper, the AdaLM algorithm was compared with LM, HLM, and TSLM. As a contrast, two frameworks were set-up in this study to investigate the performance of these algorithms under different conditions.

The test results are listed in Table 2. "√" means convergence, "good" means local minima, "○" means "bad" local minima, and "×" means divergence or the term does not exist.

Table 2. Performance comparison of different activation functions.

Algorithm	Year	Sigmoid	Tanh	ReLU	PReLU
LM	1978	√	×	×	×
HLM	2013	√	○	√	○
TSLM	2018	√	√	√	√
AdaLM	2020	√	√	√	√

Table 2 shows that the traditional LM model diverges when Tanh, ReLU, and PReLU are used as activation functions, which proves that the LM problem exists. HLM is a higher-order descent method. The HLM model could not converge when the activation functions were Tanh and PReLU, which indicates that over-high-order descent may produce negative effects. The TSLM model is an improved version of HLM. In the test, TSLM converged with four activation functions. The AdaLM algorithm proposed in this paper could change the descent direction in time and has good stability, so it achieved convergence.

5.3. Evaluation of Prediction Effect

From Figure 9a, the generated curve amplitude was very small with a large error. From the perspective of the prediction effect, the curve predicted by LM could not describe the future trend of real data. From Figure 9b,c, it could hardly reproduce the original data. After training, HLM and TSLM only achieved a line with slight fluctuation. Obviously, they could not predict the trend of future data in this test. From Figure 9d, the curves generated by AdaLM could fit the real data well. From the perspective of the prediction effect, the curve predicted by AdaLM could describe the future trend of real data.

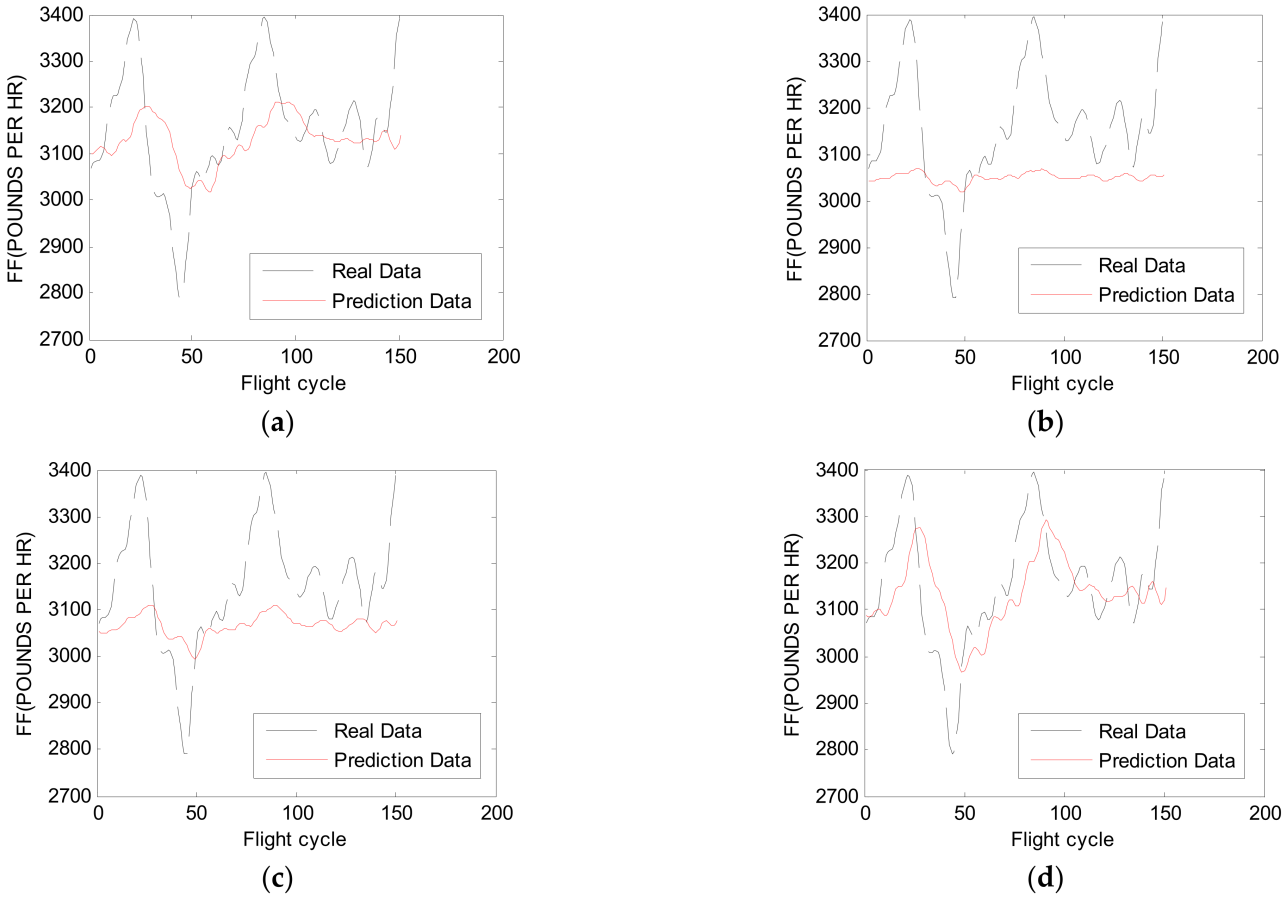


Figure 9. The prediction effect of each algorithm: (**a**) LM; (**b**) HLM; (**c**) TSLM; (**d**) AdaLM.

In order to quantify the prediction effect of each algorithm, the time consumption in 50 iterations and the prediction MAE of algorithms are shown in Table 3.

Table 3. MAE and time consumption of each algorithm.

Algorithm	LM	HLM	TSLM	AdaLM
Time/s	140	181	173	62
MAE	41.4847	38.2276	42.6715	32.9175

From Table 3, the neural network model with the original LM algorithm was trained in 140 s. The data predicted by the model after training were basically consistent with the original data, with an error of 41.4847. This shows that the original LM algorithm could basically make the neural network predict the fuel consumption data of the aeroengine, but the training took a lot of time because of the falling into a "bad" local minimum.

The neural network model with HLM algorithm completed the training in 181 s. The data predicted by the model after training fit the original data with an error of 38.2276. The HLM algorithm took the longest time to train in the prediction of fuel consumption data of an aeroengine. The HLM algorithm increased the unnecessary complexity and did not improve the performance.

The neural network model of the TSLM algorithm was trained in 173 s. The data predicted by the trained model fit the original data with an error of 42.6715. TSLM algorithm has the highest error in this task, and the complexity of the algorithm affects the prediction performance.

The neural network model with the AdaLM algorithm proposed in this paper was trained in 62 s, and the data predicted by the trained model fit the original data with an error of only 32.9175, which indicates that the AdaLM algorithm took into account high performance and high speed in predicting the fuel consumption data of aeroengine, with the least time consumption and the highest accuracy.

6. Conclusions

This paper pointed out that the LM neural networks may converge poorly or diverge because the training data are large and the weight is not selected properly. This work can guide researchers to carry out some appropriate strategies such as training data preprocessing and weights intervention at the beginning of training to avoid problems in the use of the LM neural networks when they still insist on using this model. Furthermore, this work can expand the application scope of LM neural networks and make LM neural networks play its role better in more situations.

This study proposed a new solution to the problem of LM neural networks: the AdaLM algorithm. The algorithm adds the weight term and error direction on the original algorithm, which makes the algorithm close to the steepest descent method at the beginning to preliminarily adjust the weight of the neural network. Then, the AdaLM algorithm gradually approaches the LM algorithm, which improves the stability based on inheriting the efficient optimization ability of LM. This paper compared the performance of traditional LM, HLM, TSLM, and AdaLM algorithms. We found that the performance improvement of the higher-order algorithm was very limited. Among many high-order algorithms, the third-order algorithm had the better performance. The AdaLM algorithm could not only make the neural network converge to a "good" local minimum, but also greatly shorten the operation time without losing the prediction accuracy.

The application of the AdaLM algorithm focused on a small sample and highly nonlinear engineering data. It was especially effective for the analysis and prediction of engine performance parameters in the aviation field. In future research, we will collect engine data from more dimensions to comprehensively evaluate AdaLM's ability of analysis. The application scenarios of AdaLM will also be further explored.

Author Contributions: Writing—original draft preparation, Z.Y., S.Z., Z.C.; writing—review and editing, L.L., Z.C; funding acquisition, Z.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number U1733201.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Luo, G.; Zou, L.; Wang, Z.; Lv, C.; Ou, J.; Huang, Y. A novel kinematic parameters calibration method for industrial robot based on Levenberg-Marquardt and Differential Evolution hybrid algorithm. *Robot. Comput. Integr. Manuf.* **2021**, *71*, 102165. [CrossRef]
2. Kumar, S.S.; Sowmya, R.; Shankar, B.M.; Lingaraj, N.; Sivakumar, S. Analysis of Connected Word Recognition systems using Levenberg Mar-quardt Algorithm for cockpit control in unmanned aircrafts. *Mater. Today Proc.* **2021**, *37*, 1813–1819. [CrossRef]
3. Mahmoudabadi, Z.S.; Rashidi, A.; Yousefi, M. Synthesis of 2D-Porous MoS_2 as a Nanocatalyst for Oxidative Desulfuriza-tion of Sour Gas Condensate: Process Parameters Optimization Based on the Levenberg–Marquardt Algorithm. *J. Environ. Chem. Eng.* **2021**, *9*, 105200. [CrossRef]
4. Transtrum, M.K.; Machta, B.B.; Sethna, J.P. Why are nonlinear fits to data so challenging? *Phys. Rev. Lett.* **2010**, *104*, 060201. [CrossRef]
5. Amini, K.; Rostami, F. A modified two steps Levenberg–Marquardt method for nonlinear equations. *J. Comput. Appl. Math.* **2015**, *288*, 341–350. [CrossRef]
6. Kim, M.; Cha, J.; Lee, E.; Pham, V.H.; Lee, S.; Theera-Umpon, N. Simplified Neural Network Model Design with Sensitivity Analysis and Electricity Consumption Prediction in a Commercial Building. *Energies* **2019**, *12*, 1201. [CrossRef]
7. Zhao, L.; Otoo, C.O.A. Stability and Complexity of a Novel Three-Dimensional Envi-ronmental Quality Dynamic Evolution System. *Complexity* **2019**, *2019*, 3941920. [CrossRef]
8. Zhou, W.; Liu, D.; Hong, T. Application of GA-LM-BP Neural Network in Fault Prediction of Drying Furnace Equipment. *Matec. Web Conf.* **2018**, *232*, 01041. [CrossRef]
9. Jia, P.; Zhang, P. Type Identification of Coal Mining Face Based on Wavelet Packet Decomposition and LM-BP. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018.
10. Hua, L.; Bo, L.; Tong, L.; Wang, M.; Fu, H.; Guo, R. Angular Acceleration Sensor Fault Diagnosis Based on LM-BP Neural Network. In Proceedings of the 37th Chinese Control Conference, Wuhan, China, 25–27 July 2018; pp. 6028–6032.
11. Hossein, A.M.; Nazari, M.A.; Madah, R.G.H.; BehshadShafii, M.; Ahmadi, M.A. Thermal conductivity ratio prediction of Al2O3/water nanofluid by applying connectionist methods. *Colloids Surf. A Physicochem. Eng. Asp.* **2018**, *541*, 154–164.
12. Yang, X. A higher-order Levenberg–Marquardt method for nonlinear equations. *Appl. Math. Comput.* **2013**, *219*, 10682–10694. [CrossRef]
13. Chen, L. A high-order modified Levenberg–Marquardt method for systems of nonlinear equations with fourth-order convergence. *Appl. Math. Comput.* **2016**, *285*, 79–93. [CrossRef]
14. Derakhshandeh, S.Y.; Pourbagher, R.; Kargar, A. A novel fuzzy logic Leven-berg-Marquardt method to solve the ill-conditioned power flow problem. *Int. J. Electr. Power Energy Syst.* **2018**, *99*, 299–308. [CrossRef]
15. Qiao, J.; Wang, L.; Yang, C.; Gu, K. Adaptive levenberg-marquardt algorithm based echo state network for chaotic time series prediction. *IEEE Access* **2018**, *6*, 10720–10732. [CrossRef]
16. Ma, C.; Tang, J. The quadratic convergence of a smoothing Levenberg–Marquardt method for nonlinear complementarity problem. *Appl. Math. Comput.* **2008**, *197*, 566–581. [CrossRef]
17. Du, S.Q.; Gao, Y. Global convergence property of modified Levenberg-Marquardt meth-ods for nonsmooth equations. *Appl. Math.* **2011**, *56*, 481. [CrossRef]
18. Zhou, W. On the convergence of the modified Levenberg–Marquardt method with a non-monotone second order Armijo type line search. *J. Comput. Appl. Math.* **2013**, *239*, 152–161. [CrossRef]
19. Moré, J.J. The Levenberg-Marquardt algorithm: Implementation and theory. In *Numerical Analysis*; Springer: Berlin/Heidelberg, Germany, 1978; pp. 105–116.
20. Madsen, K.; Nielsen, H.B.; Tingleff, O. *Methods for Non-Linear Least Squares Problems*, 2nd ed.; Technical University of Denmark: Lyngby, Denmark, 2004.
21. Zhang, Z.; Ma, X.; Yang, Y. Bounds on the number of hidden neurons in three-layer binary neural networks. *Neural Netw.* **2003**, *16*, 995–1002. [CrossRef]
22. Liang, X.; Chen, R.C. A unified mathematical form for removing neurons based on or-thogonal projection and crosswise propagation. *Neural Comput. Appl.* **2010**, *19*, 445–457. [CrossRef]
23. Chua, C.G.; Goh, A.T.C. A hybrid Bayesian back-propagation neural network approach to multivariate modelling. *Int. J. Numer. Anal. Methods Geomech.* **2003**, *27*, 651–667. [CrossRef]
24. Sequin, C.H.; Clay, R.D. Fault tolerance in artificial neural networks. In Proceedings of the 1990 IJCNN International Joint Conference on Neural Networks, San Diego, CA, USA, 17–21 June 1990; pp. 703–708.
25. Li, Y.G.; Nilkitsaranont, P. Gas turbine performance prognostic for condition-based maintenance. *Appl. Energy* **2009**, *86*, 2152–2161. [CrossRef]