信息与控制　2020 年　第 49 卷　第 3 期：306 ~ 314

DOI：10.13976/j.cnki.xk.2020.9332　　文章编号：1002 - 0411(2020) - 03 - 0306 - 09

激活函数 LeafSpring 的构建及多数据集对比研究

郜天柱[1,2,3]

1. 中国科学院沈阳自动化研究所机器人学国家重点实验室，辽宁 沈阳　110016；
2. 中国科学院机器人与智能制造创新研究院，辽宁 沈阳　110016；3. 中国科学院大学，北京　100049

基金项目：中国科学院战略高技术创新基金资助项目(GQRC - 19 - 16)

通信作者：郜天柱，gaotianzhu@sia.cn　收稿/录用/修回：2019 - 06 - 12/2019 - 10 - 08/2020 - 01 - 10

摘要

针对神经网络中 ReLU 激活函数存在返回值非负及神经元未激活等问题，提出了一种全新的类 ReLU 激活函数——LeafSpring. LeafSpring 既继承了 ReLU 的优势，又可以返回负值，通用性更强. LeafSpring 函数的推导及函数性质也会被讨论. 该激活函数还引入了刚度系数 k，可以通过训练主动调节相邻两层的权重系数. 为了减少计算量，LeafSpring 可以在一定情况下简化为 ReLU 或 Softplus. 为了展现 LeafSpring 激活函数的性能，还将其与 ReLU、Softplus 及 Sigmoid 在 4 种不同类型的数据集上进行拟合精度对比. 对比结果表明，LeafSpring 在不同的数据集上均能兼顾拟合精度和收敛速度且在小网格规模可以更快、更准确地拟合复杂非线性函数.

关键词

激活函数
神经网络
线性整流函数(ReLU)
LeafSpring

中图法分类号：O175.1

文献标识码：A

Build and Multi-datasets Comparison Analysis of LeafSpring Activation Function

GAO Tianzhu[1,2,3]

1. *State Key Laboratory of Robotics*, *Shenyang Institute of Automation*, *Chinese Academy of Sciences*, *Shenyang* 110016, *China*;
2. *Institutes for Robotics and Intelligent Manufacturing*, *Chinese Academy of Sciences*, *Shenyang* 110169, *China*;
3. *University of Chinese Academy of Sciences*, *Beijing* 110049, *China*

开放科学(资源服务)标识码(OSID)

Abstract

To solve the problems of non-negative return value and non-activation of neuron in the ReLU activation function of neural network, we propose a new ReLU-like activation function called Leafspring activation function. LeafSpring not only inherits the advantages of ReLU but also returns negative values, which is more versatile. The derivation and properties of the LeafSpring activation function are discussed. In the LeafSpring activation function, we introduce the stiffness coefficient k, which can adjust the weight coefficient of two adjacent layers by training. To reduce the amount of computation, LeafSpring can be simplified to ReLU or Softplus in some cases. The performance of LeafSpring activation function is shown by comparing its fitting accuracy with that of ReLU, Softplus, and Sigmoid on four different types of datasets. Comparison results show that LeafSpring can give consideration to fitting accuracy and convergence speed in different datasets, and it can fit complex nonlinear functions faster and more accurately on a small grid scale.

Keywords

activation function;
neural network;
rectified linear unit (ReLU);
LeafSpring

0　引言

近年来，人工智能和深度学习等技术发展迅速，已经在图像处理、语音识别和自然语言处理等领域取得突破，并显著提高了原有技术的预测精度. 基于神经网络的算法，包括误差反向传播神经网络、卷积神经网络及循环神经网络等是实现人工智能和深度学习的重要算法基础[1-2]. 然而，无论是何种神经网络算法，都以感知机模型和非线性的激活函数作为基本结构. 因而构建并研究激活函数，对发展神经网络算法乃至人工智能都具有重要意义.

目前，激活函数主要以线性整流函数（ReLU）[3] 及 Sigmoid 等函数及其相关变体为主. 由于 ReLU 激活函数和导数的数学运算均较为简单，计算时间短，速度快，占用资源少，所以当面对未知数据集时，ReLU 成为了激活函数的首选类型. 然而，ReLU 也有其固有缺陷. 首先，ReLU 输出值为非负数且关于 x 轴不对称，因而其在学习包含负返回值的数据集时，速度会变得很慢. 其次，训练过程中，ReLU 常常引起神经元的永久不激活，永久不激活的神经元会占用大量的计算资源，造成浪费. 再有，ReLU 是分段线性函数，当其用于拟合光滑非线性函数时，在相同拟合精度下，采用 ReLU 作为激活函数的神经网络需要包含有更多的神经元或隐藏层.

为了弥补 ReLU 的不足，近年来出现了许多在 ReLU 的基础上进行简单修改的类 ReLU 激活函数，主要包括：Leaky ReLU，PReLU，RReLU，ELU，SELU 及 SReLU 等[4-12]. 这些类 ReLU 激活函数本质上是将 ReLU 函数预先分段叠加，形成若干组固定连接方式的 ReLU 激活函数对，从而缓解原本 ReLU 激活函数在负输入下收敛慢及神经元永久不激活等问题. 然而，理论上这些类 ReLU 函数是可以通过 ReLU 在本身的训练过程中自动拟合得到的，因而未能从根本上触及 ReLU 的本质.

此外，另一种类 ReLU 函数是 Softplus[13-19]. Softplus 是一种光滑连续函数，现对于 ReLU，其在拟合光滑连续函数时会有一定优势. 更重要的是，Softplus 在趋于正无穷和负无穷时的渐近线所组成的分段函数即为 ReLU 激活函数，因而拟合线性函数时与 ReLU 相似. Softplus 返回值恒大于零且导数依然是非零连续函数，可以防止神经元永久不激活. 然而，由于 Softplus 函数的导数通常小于 1，进而会存在梯度消失等问题. 此外，Softplus 的劣势还表现在其计算量上，由于包含指数函数等非线性运算，其每一步的计算量会显著大于 ReLU.

基于 Softplus 和 ReLU，本文构建了一种新的类 ReLU 激活函数—板簧（LeafSpring）激活函数. 在 LeafSpring 激活函数中，还引入了刚度系数 k，用于在训练过程中自适应地调节函数的光滑程度. 由于系数 k 的引入，ReLU 和 Softplus 成为了 LeafSpring 激活函数的两种特殊情况，因而 LeafSpring 可以继承前者的优势. 又由于系数 k 可以通过训练改变符号，所以 LeafSpring 可以覆盖更广泛的输出值范围. 再有，k 值的大小可以影响 LeafSpring 的光滑程度，因而可以用于拟合不同曲率的函数. 此外，通过合理简化，可以将局部的 LeafSpring 函数简化为 ReLU，或整体简化为 Softplus，从而降低计算量.

本文将从 LeafSpring 函数的构建出发，分析其与 ReLU 和 Softplus 的继承关系，并给出误差反向传播计算方法及合理简化方法. 最后，本文以 4 个不同类型的数据集对其计算性能进行评估，测试该激活函数的可行性.

1　LeafSpring 函数的构建

LeafSpring 函数作为一种类 ReLU 函数，同样也是从 ReLU 函数推导而来的. ReLU 函数的表达方式有很多种，这里仅以最基本的分段函数形式表示：

$$A_{\mathrm{ReLU}}(x)=\begin{cases}x, & x>0\\ 0, & x\leqslant 0\end{cases} \tag{1}$$

该表达方式简单明了，但是仍然是一种以分段形式表达的方式. 这里可以借鉴 Softplus 函数的形式，将 ReLU 改写成如式(2)所示的形式：

$$A_{\mathrm{ReLU}}(x)\Leftrightarrow \lim_{n\to+\infty}\ln(\sqrt[n]{\mathrm{e}^{nx}+1}) \tag{2}$$

由式(2)可得，当 n 趋于正无穷时，右侧函数就会无限趋近于 ReLU 函数且二者的定义域和值域均相同. 于是可以定义右侧函数为新函数 $R(n,x)$：

$$R(n,x)=\ln(\sqrt[n]{\mathrm{e}^{nx}+1}) \tag{3}$$

其中，n 为正整数. 因而有：

$$A_{\mathrm{ReLU}}(x)=R(+\infty,x)$$

若令 $n=1$，则函数 $R(n,x)$ 等价于 Softplus 函数：

$$R(1,x)=\ln(\mathrm{e}^{x}+1)=A_{\mathrm{Softplus}}(x)$$

由此可见，ReLU 函数和 Softplus 函数都是 $R(n,x)$ 函数的特殊形式. 然而，如果仅仅使用 $R(n,x)$ 函数依然无法解决激活函数返回负值的问题，因此可以将 $R(n,x)$ 函数进行变形，如式(4)所示：

$$R(n,x)=\ln(\sqrt[n]{\mathrm{e}^{nx}+1})=\frac{1}{n}\ln(\mathrm{e}^{nx}+1) \tag{4}$$

此处，n 的取值范围仍为正整数，如果将 n 的取值范围扩大到非零实数，则可以得到如式(5)所示函数，并定义为 LeafSpring 函数.

$$A_{\mathrm{LeafSpring}}(k,x)=\frac{1}{k}\ln(\mathrm{e}^{kx}+1),\ k\in\mathrm{R},\ k\neq 0 \tag{5}$$

与 $R(n,x)$ 函数类似，LeafSpring 函数同样可以表示 ReLU 函数和 Softplus 函数，但是由于 k 值可以取为负值，LeafSpring 函数可以返回负值.

根据 LeafSpring 函数的表达式，可以进而得到其在不同 k 和 x 值下的函数图像，如图 1 所示.

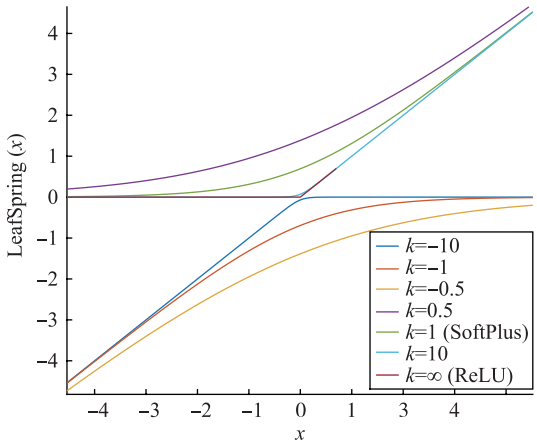

图 1　LeafSpring 激活函数曲线

Fig.1　The curve of LeafSpring activation function

根据 LeafSpring 函数的图像及其表达式，当 k 的绝对值趋于 0 时，函数趋于平滑；当 k 的绝对值趋于无穷时，函数趋于在(0，0)点导数不连续. 因而，k 的作用更像是

刚度系数对于板簧的作用，当刚度系数变化时，在相同的弯折程度下，板簧的变形曲率会随之不同. 因此，本文将上述函数命名为 LeafSpring 函数，而系数 k 命名为刚度系数.

然而，$R(n, x)$ 函数还可以被改写成如式(6)所示形式：

$$A'_{\text{LeafSpring}}(k, x) = k\ln(e^{\frac{x}{k}}+1),\ k\in\mathbf{R},\ k\neq 0 \qquad (6)$$

但是，该形式会使得当 k 随着训练的变化而跨越零点时，函数的导数发生阶跃，这样不利于其在训练过程中任意跨越零点，从而不利于其拟合任意函数.

2 LeafSpring 函数特性分析

对于 LeafSpring 函数，可以对其奇偶性、导数特性及误差反向传播方法进行分析.

2.1 奇偶性

由式(5)，可得：

$$\begin{aligned} A_{\text{LeafSpring}}(-k, -x) &= -\frac{1}{k}\ln(e^{kx}+1) \\ &= -A_{\text{LeafSpring}}(k, x) \end{aligned}$$

所以，LeafSpring 函数是关于 k 和 x 的奇函数. 由此可得，LeafSpring 函数可以随着 k 和 x 的取值不同，返回关于零点对称的值且该值的取值范围为实数.

相比于 ReLU 和 Softplus 激活函数，尽管前者关于原点对称，但是都无法解决返回值仅为非负数的问题. 通过采用 LeafSpring 函数，就可以弥补其不足.

2.2 关于 x 的偏导数特性

由式(5)，对 x 求偏导数可得：

$$\begin{aligned} \frac{\partial A_{\text{LeafSpring}}(k, x)}{\partial x} &= \frac{ke^{kx}}{k(e^{kx}+1)} \\ &= \frac{1}{1+e^{-kx}} \\ &= A_{\text{Sigmoid}}(kx) \end{aligned}$$

可见，LeafSpring 函数对 x 的偏导数为 Sigmoid 函数，而 Sigmoid(kx)的图像如图 2 所示.

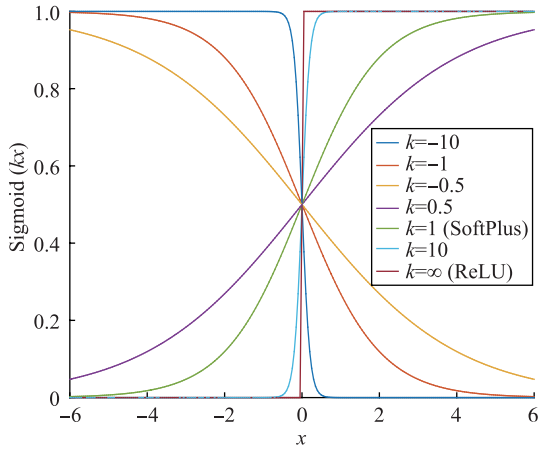

图 2 Sigmoid(kx)的函数曲线

Fig.2 The curve of Sigmoid(kx)

由图 2 可得，通过引入刚度系数 k，LeafSpring 函数的对 x 的偏导数可以覆盖更大的变化范围，偏导数随 x 的变化率也可以通过训练 k 值进行自动调节，从而可以一定程度上减缓使用 Softplus 引起的梯度消失的问题. 此外，当 $k=1$ 或 $k=+\infty$ 时，LeafSpring 函数可以兼容 ReLU 及 Sigmoid 的偏导数，从而可以继承二者在线性函数拟合方面的优势.

2.3 k 值的误差反向传播特性

由式(5)，对 k 求偏导可得：

$$\begin{aligned} \frac{\partial A_{\text{LeafSpring}}(k, x)}{\partial k} &= \frac{xe^{kx}}{k(e^{kx}+1)} - \frac{\ln(e^{kx}+1)}{k^2} \\ &= \frac{xe^{kx}}{k(e^{kx}+1)} - \frac{A_{\text{LeafSpring}}(k, x)}{k} \\ &= \frac{xe^{kx} - A_{\text{LeafSpring}}(k, x)\,e^{kA_{\text{LeafSpring}}(k, x)}}{ke^{kA_{\text{LeafSpring}}(k, x)}} \end{aligned}$$

可见，当 LeafSpring 函数对 k 求偏导数时，可以大量使用前向传播过程及返回值的误差反向传播过程中的中间数据变量，如 $A_{\text{LeafSpring}}(k, x)$ 及 $k\cdot e^{kx}$ 等. 但是，不可否认，该方法仍存在计算量提升的问题，尤其是在计算分母项时，无法通过存储之前的计算数据来简化计算. 但是，如果该方法能够在拟合精度和适应数据集范围上优于 ReLU 或 Softplus，那么该方法仍存在较大价值.

由以上分析可知，尽管 LeafSpring 函数在计算量上会大于 ReLU 和 Softplus，但是 LeafSpring 函数关于原点对称，可以返回负值，并且可以扩大导数的变化率变化范围，从而缓解梯度消失等问题，所以该方法具有通用性更强的优势，值得深入研究和广泛应用尝试.

3 LeafSpring 函数的机理分析

LeafSpring 函数无论是在值域范围还是偏导数变化率的协调性方面都优于 ReLU 和 Softplus. 其根本原因是引入了刚度系数 k. 但是，如果从另一个角度来分析，该参数之所以会起到如此不同的效果，是因为它在 LeafSpring 函数中是多次出现的. 若具体分析其在神经网络中的作用机理，可以以如图 3 所示的全连接神经网络结构为例.

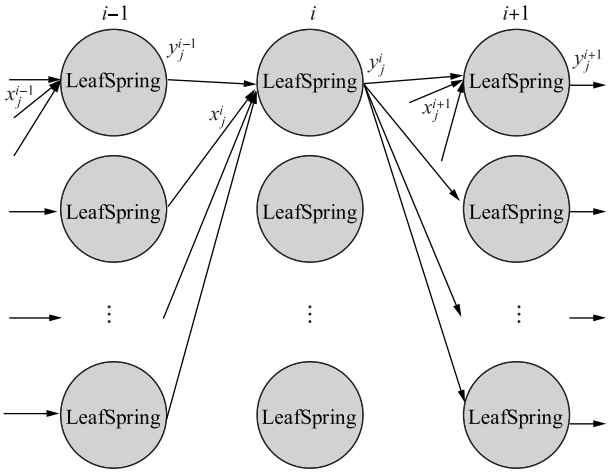

图 3 神经网络结构图

Fig.3 The structure diagram of a neural network

图 3 所示神经网络结构为多层神经网络的一部分，这里仅取其中的第 $i-1$ 层、第 i 层及第 $i+1$ 层. x_b^a 为第 a 层

的第 b 个节点的输入，y_b^a 为第 a 层的第 b 个节点的输出，由此可得第 i 层的某一神经网络节点 j 的输入 x_j^i 为

$$x_j^i = \sum_h x_{j,h}^i = \sum_h w_{j,h}^{i-1} y_h^{i-1}$$

其中，h 为前层的节点编号，为了便于分析，这里仅对节点 j 进行分析.

经过第 i 层 LeafSpring 激活函数的处理，该层的输出可以表示为

$$\begin{aligned} y_j^i &= A_{\text{LeafSpring}}(k_j^i, x_j^i) \\ &= \frac{1}{k_j^i}\ln(\mathrm{e}^{k_j^i x_j^i}+1) \\ &= \frac{1}{k_j^i}\ln(\mathrm{e}^{k_j^i \sum_h w_{j,h}^{i-1} y_h^{i-1}}+1) \end{aligned}$$

对于 $i+1$ 层的节点 p，前层节点 j 对其输入为

$$\begin{aligned} x_{p,j}^{i+1} &= w_{p,j}^i y_j^i \\ &= \bar{w}_{p,j}^i \ln\left(\mathrm{e}^{\sum_h \hat{w}_{j,h}^{i-1} y_h^{i-1}}+1\right) \\ &= \bar{w}_{p,j}^i A_{\text{Softplus}}\left(\sum_h \hat{w}_{j,h}^{i-1} y_h^{i-1}\right) \end{aligned} \tag{7}$$

其中，

$$\bar{w}_{p,j}^i = \frac{w_{p,j}^i}{k_j^i} \tag{8}$$

$$\hat{w}_{j,h}^{i-1} = k_j^i w_{j,h}^{i-1} \tag{9}$$

可见，若将式(7)的系数经由式(8)和式(9)参数代入，则式(7)可以得到进一步简化，而其简化结果与 Softplus 激活函数的作用相同.

反观式(8)和式(9)，两式都包含参数 k_j^i，而令两式相乘可得：

$$\bar{w}_{p,j}^i \hat{w}_{j,h}^{i-1} = w_{p,j}^i w_{j,h}^{i-1}$$

所以 LeafSpring 函数相对于 Softplus，最根本的区别为：通过引入刚度系数 k，在激活函数的位置同时调节该节点前后的权重系数，并保持其乘积值不变.

相比而言，以 ReLU 及 Softplus 等为激活函数的神经网络，对其权重系数的训练是逐层进行的且每层之间不存在相互协调；而 LeafSpring 函数在激活函数的位置就引入了前后权重系数的协调机制，从而可以以参数的形式自主学习层系数间的相互关系，避免个别节点由于梯度消失或未激活而形成孤立，造成资源浪费.

此外，相比于其它类 ReLU 算法，通过引入固定值的修正系数虽然可以一定程度上缓解 ReLU 算法所遇到的问题，但是这些参数的引入本质上是对个别权重系数的人为修正，存在超参数，其通用性较差. 而 LeafSpring 激活函数尽管也起到了权重系数修正的作用，但是在修正的同时还保证了各层间权重系数的乘积的不变性，且该修正程度可以由神经网络训练自主学习得到，进而提高了通用性.

4　LeafSpring 函数的合理简化

尽管从函数性质和作用机理分析，LeafSpring 激活函数都继承并扩展了 ReLU 和 Softplus 激活函数，但是 LeafSpring 函数的表达式相对较复杂，实际应用难免造成训练计算量过大的问题. 然而，由于 ReLU 和 Softplus 激活激活函数均为 LeafSpring 函数的特殊形式，如若其在实际训练过程中可以局部或整体简化为上述两种简单函数，或许可以大幅降低计算量.

4.1　简化为 ReLU 函数

由式(2)和图 1 可得，当系数 n 或刚度系数 k 趋于正无穷时，LeafSpring 函数与 ReLU 函数等价. 然而，如若 k 趋于负无穷，可以得到如(10)所示函数，记做做$A_{\text{ReLU}}^s(x)$.

$$\begin{aligned} A_{\text{LeafSpring}}(-\infty, x) &= \begin{cases} 0, & x \geqslant 0 \\ x, & x < 0 \end{cases} \\ &= \min\{x, 0\} \\ \Leftrightarrow & A_{\text{ReLU}}^s(x) \end{aligned} \tag{10}$$

$A_{\text{ReLU}}^s(x)$ 函数形式简单，计算量与 ReLU 函数相同，所以同样可以作为 LeafSpring 函数的简化目标函数.

因而，在实际的算法计算中，可以在刚度系数的值得绝对值大于某一特定常数 $\tilde{k}$ 时，将其简化为 ReLU 或 $A_{\text{ReLU}}^s(x)$ 函数，从而降低计算量，如式(11)所示：

$$A_{\text{LeafSpring}}^{\text{ReLU}}(k, x) = \begin{cases} A_{\text{ReLU}}(x), & k > \tilde{k} \\ A_{\text{LeafSpring}}(k, x), & k \in [-\tilde{k}, 0) \cup (0, \tilde{k}] \\ A_{\text{ReLU}}^s(x), & k < -\tilde{k} \end{cases} \tag{11}$$

由式(11)分析可得，当 LeafSpring 函数在训练过程中，其刚度系数的训练结果较大，则其更接近于 ReLU 函数，使用 ReLU 函数对其进行简化，可以降低计算量.

4.2　简化为 Softplus 函数

由式(5)和图 1 可得，当 k 的值取为 1 时，LeafSpring 函数与 Softplus 函数完全相同，因而可以用 Softplus 函数代替 LeafSpring 函数，从而减少计算量. 然而，在实际使用中，往往很难保证 k 的训练结果值为 1，而且当 k 的值在以 1 为邻域的范围内变化时，其在小于 1 和大于 1 两个子邻域的取值对整体返回值的影响有较大不同，因而以此方式简化意义不大. 但是，从 LeafSpring 函数的机理进行分析，可以得到另一条简化思路，这里同样以图 3 所示神经网络结构为例.

对于第 $i+2$ 层的第 r 个节点，其输入为

$$x_{r,p}^{i+2} = \bar{w}_{r,p}^{i+1} \ln\left(\mathrm{e}^{\sum_j k_r^{i+1} \bar{w}_{p,j}^i y_j^i}+1\right)$$

其中，

$$\bar{w}_{r,p}^{i+1} = \frac{w_{r,p}^{i+1}}{k_p^{i+1}}$$

进而可得

$$\hat{w}_{p,j}^i = k_p^{i+1} \bar{w}_{p,j}^i = \frac{k_p^{i+1}}{k_j^i} w_{p,j}^i \tag{12}$$

根据式(12)，若：

$$k_p^{i+1} \approx k_j^i$$

则：

$$\hat{w}_{p,j}^i \approx w_{p,j}^i$$

因而，此时 k 作为调节两层权重系数的相互关系的作用消失了，从而刚度系数的作用仅仅体现在最后一层神经网络的返回值中，于是刚度系数的作用就可以被权重系数完全取代，在该情况下的 LeafSpring 函数可以被简化为 Softplus 函数，如式(13)所示：

$$A_{\text{LeafSpring}}^{\text{softplus}}(k,x)=\begin{cases}A_{\text{Softplus}}(x), & \forall k_1,k_2,|k_1/k_2-1|<\varepsilon\\ A_{\text{LeafSpring}}(k,x),\text{other}\end{cases} \tag{13}$$

进而，当以 LeafSpring 为激活函数的神经网络在训练过程中发生所有的刚度系数值均属于某一特定小区间，或任意两个刚度系数的商与 1 的差值的绝对值小于某一小量 ε 时，该神经网络的所有激活函数可以简化为 Softplus 函数.

通过合理简化，LeafSpring 函数可以简化为 ReLU 或 Softplus 函数，从而降低计算量. 然而，从另一个角度来分析，如若经过训练后的刚度系数值无法简化为 ReLU 或 Softplus，则说明该训练数据集具有 ReLU 或 Softplus 在此规模的网格下无法学习到的特征. 换言之，LeafSpring 激活函数可以在同等网格规模下学习到更为丰富的特征，并将该特征以可表征层间权重系数关系的刚度系数的形式储存在激活函数中.

所以，对于某一未知数据集，可以优先采用 LeafSpring 激活函数对其进行神经网络拟合. 如若训练结果倾向于 ReLU 或者 Softplus，可以再转用或直接简化为后者；如若训练结果无法简化，则 LeafSPring 激活函数或许表征了比 ReLU 和 Softplus 更丰富的特征信息.

5 多数据集下不同激活函数拟合精度对比

尽管从函数特性和作用机理分析，LeafSpring 激活函具有 ReLU 和 Softplus 等所不具备的特性，但是 LeafSpring 激活函数的神经网络拟合精度仍需要在不同的数据集下进行测试. 然而，目前可用于神经网络训练的数据集众多，应用场景各异且神经网络结构对训练结果有较大的影响. 而 LeafSpring 激活函数的优势在于其通用性，如若无法覆盖绝大多数数据集类型，则较难说明其通用性. 所以，本文采用现有数据集及按照目标函数随机生成数据集的方法，来构造不同的数据集，并通过 LeafSpring 函数和 ReLU、Softplus 及 Sigmoid 等常用激活函数进行对比，考量拟合精度，从而分析 LeafSpring 激活函数的通用性等特点.

5.1 数据集的选取及构建

对于任意映射关系，可以按照该映射关系的导数的性质划分：将其导数存在且在定义域内为常数的映射关系定义为线性函数，对其导数存在且定义域不为常数的映射关系定义为非线性函数. 如若导数不存在，则可以定义为分段函数. 因而，可以选取和构建 4 种映射关系的数据集来分析各激活函数的拟合能力：线性映射、简单非线性映射、复杂非线性映射及分段函数映射.

5.1.1 线性映射数据集的构建

首先，构建一个规模为 100 000 × 100 的由(0，1)区间均匀分布随机数组成的数组 X，作为数据集的输入值. 接着将该数组的行向量按照规则：

$$\boldsymbol{Y}_1=\boldsymbol{X}[:,:50]+\boldsymbol{X}[:,50:]$$
$$\boldsymbol{Y}=\boldsymbol{Y}_1[:,:25]+\boldsymbol{Y}_1[:,25:]$$

进行运算，从而得到维数为 100 000 × 25 的新数组，并定义为 $\boldsymbol{Y}$，作为数据输出值. 其中，$[:,:a]$，表示选取该数组的前 a 列，$[:,b:]$，表示选取该数组的 b 列[20].

由此，数组 $\boldsymbol{X}$ 经过两次线性叠加转化为了数组 $\boldsymbol{Y}$，因而可以将此函数关系定义为

$$\boldsymbol{Y}=D_{\text{Dual\_Line}}(\boldsymbol{X})$$

$D_{\text{Dual\_Line}}$ 函数由于仅包括线性元素间的线性叠加，因而可以被用作构建线性函数映射的数据集.

5.1.2 简单非线性映射数据集的构建

与线性映射数据集相同，可以构建一个均匀分布随机数生成的数组 $\boldsymbol{X}$ 作为输入，规模为 100 000 × 100，而其输出 $\boldsymbol{Y}$ 的构建规则为

$$\boldsymbol{Y}_1=\boldsymbol{X}[:,:50]*\boldsymbol{X}[:,:50]+\boldsymbol{X}[:,50:]*\boldsymbol{X}[:,50:]$$
$$\boldsymbol{Y}=\boldsymbol{Y}[:,:25]+\boldsymbol{Y}[:,25:]$$

其中，$[\cdot]*[\cdot]$表示数组的对应元素相乘[20].

由此数组 $\boldsymbol{X}$ 经过一次乘方运算，两次线性叠加运算转换为了规模为 100 000 × 25 的数组 $\boldsymbol{Y}$，这里可以将该函数定义为

$$\boldsymbol{Y}=D_{\text{Dual\_Square}}(\boldsymbol{X})$$

$D_{\text{Dual\_Square}}$ 函数包含简单的乘方运算及线性叠加运算，因而可以构建出简单非线性数据集.

5.1.3 复杂非线性映射数据集的构建

与前两种数据集构建方法类似，以规模为 100 000 × 100 的随机均匀分布数组 $\boldsymbol{X}$ 作为输入，输出数组 $\boldsymbol{Y}$ 的构建规则为

$$\boldsymbol{Y}_0[:,:20]=\sin(\boldsymbol{X}[:,:20])$$
$$\boldsymbol{Y}_0[:,20:40]=\cos(\boldsymbol{X}[:,20:40])$$
$$\boldsymbol{Y}_0[:,40:60]=3\ \text{square}(\boldsymbol{X}[:,40:60])$$
$$\boldsymbol{Y}_0[:,60:75]=3\ \text{sqrt}(\boldsymbol{X}[:,60:75])$$
$$\boldsymbol{Y}_0[:,75:90]=\exp(\boldsymbol{X}[:,75:90])$$
$$\boldsymbol{Y}_1=\boldsymbol{Y}_0[:,:50]*\boldsymbol{Y}_0[:,50:]$$
$$\boldsymbol{Y}=(\boldsymbol{Y}_1[:,:25]+\boldsymbol{Y}_1[:,25:])/5$$

其中，sin，cos，square，sqrt 及 exp 等函数分别表示对自变量数组的每一个元素进行正弦函数、余弦函数、乘方、开方及 e 指数运算[20].

由此，数组 $\boldsymbol{X}$ 经过 6 次复杂非线性运算及 1 次线性叠加运算，得到了规模为 100 000 × 25 的数组 $\boldsymbol{Y}$，该函数定义为

$$\boldsymbol{Y}=D_{\text{Super\_Unline}}(\boldsymbol{X})$$

$D_{\text{Super\_Unline}}$ 函数包含了复杂的非线性运算，因而可以用于构建复杂非线性映射.

5.1.4 分段函数映射数据集的选取

对于分段函数映射数据集，可以选取具有实际应用背景的，用于拟合归类关系的数据集. 现有的开源数据集 MNIST、CIFAR-10、CIFAR-100 等的数据集，为了便于对比分析，本文所采用的分段函数映射数据集为 MNIST 和 CIFAR-10.

5.2 不同激活函数拟合精度对比

利用上述构建或选取的数据集，采用两层全连接层作为隐藏层，全连接层分为稀疏(sparse)和致密(dense)两种结构，以分析隐藏层节点数量对拟合精度的影响. 隐藏层的激活函数分别采用 LeafSpring、ReLU、Softplus 及 Sigmoid 四种. 训练采用的损失函数为均方误差函数，权重系数和刚度系数的初值采用均匀分布的随机数，权重系数和刚度系数的更新采用误差反向传播方法.

前 3 个数据集规模一致，因而使用 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 的前 90 000 行数据作为训练样本，后 10 000 行用作测试样本；而分段函数映射数据集，则使用 MNIST 的开源数据集的训练数据

集和测试数据集. 对于前 3 个数据集，为了表征其拟合精度，定义测试误差函数：

$$\text{error}=\frac{\sum_{m}\sum_{n}|t_{m,n}-y_{m,n}|}{m\times n\times|t_{m,n}|}$$

其中，t 为测试数据集内的输出数据；y 为训练后的神经网络的输出数据；m 和 n 为测试数据集的行数和列数，当其为下标时，整体表示对应行和列的元素.

通过使用训练数据集进行训练，并用测试数据集进行测试，可以得到拟合精度结果.

5.2.1　线性映射数据集的拟合对比

对于线性映射数据集，分别令隐藏层节点数为 50（sparse）和 100（dense），得到如图 4 所示的拟合结果.

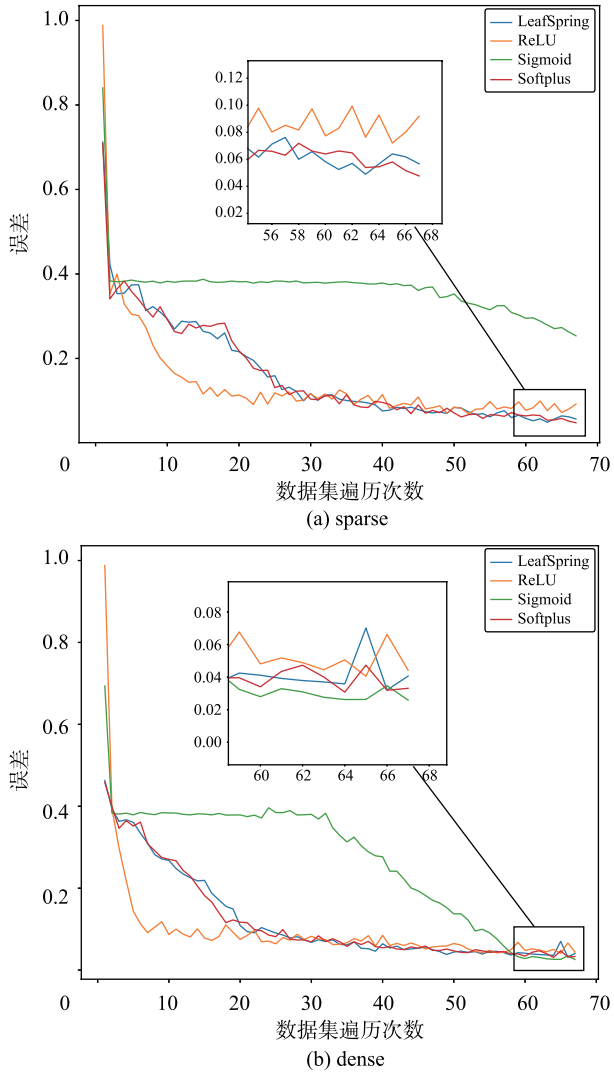

图 4　线性映射数据集拟合结果对比图

Fig.4　The fitting result compare on the dual_line database

由图 4(a)可得，当隐藏层节点数较小时，相比于 Sigmoid，ReLU、LeafSpring 及 Softplus 激活函数都可以得到较好的拟合结果，其中 ReLU 激活函数的收敛速度最快，而 LeafSpring 和 Softplus 激活函数最终的拟合误差小于 ReLU 且后两者收敛速度相同. 之所以 ReLU 激活函数具有较高的收敛速度，是因为其函数本身为分段线性的函数，当用于拟合线性函数时，可以得到更好的效果. 而最终 LeafSpring 及 Softplus 激活函数在精度上的提高，是因为二者在网格规模有限的情况下可以拟合出更为复杂的函数.

由图 4(b)可得，当隐藏层节点数较大时，4 种激活函数均能得到较好的拟合结果. 但是从收敛速度分析，ReLU 大于 LeafSpring 和 Softplus 且远大于 Sigmoid. 这主要是因为在网格规模足够的情况下，神经网络可以通过权重系数学习到足够多的信息，而激活函数对于提高精度的作用不再显著. 但是对于收敛速度，线性程度越高的函数，其收敛速度越快.

综合分析图 4(a)和图(b)，LeafSpring 激活函数在线性映射数据集上的作用与 Softplus 类似且在收敛速度可以接受的情况下，可得到较高的拟合精度.

5.2.2　简单非线性映射数据集的拟合对比

对于简单非线性映射数据集，分别令隐藏层节点数为 50（sparse）和 100（dense），可以得到如图 5 所示的拟合结果.

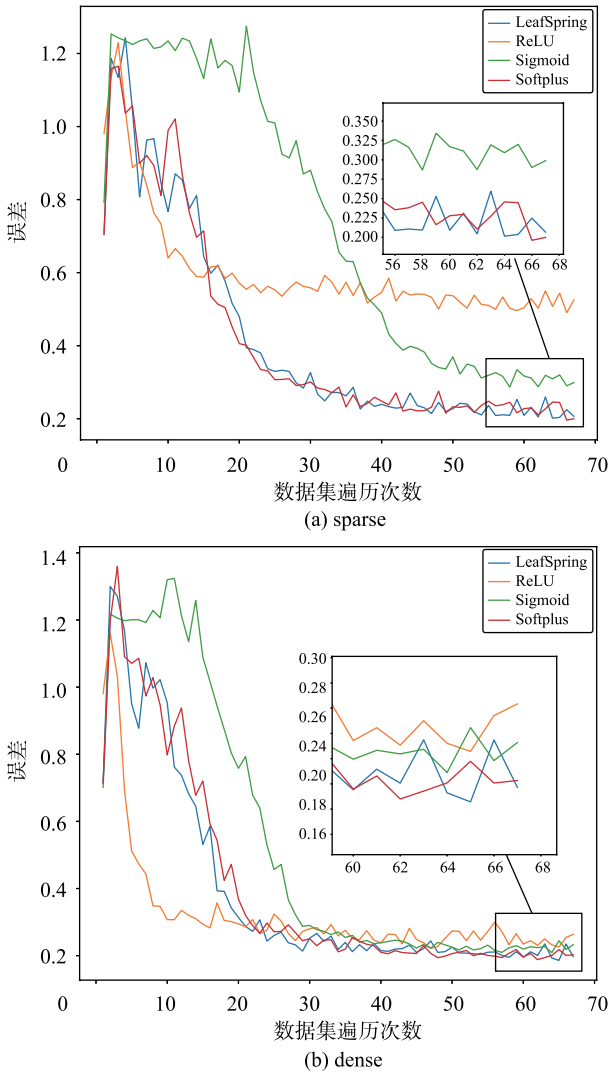

图 5　简单非线性映射数据集拟合结果对比图

Fig.5　The fitting result comparison on the simple nonlinear mapping database

由图 5(a)可得，相比于 ReLU，LeafSpring、Softplus 及 Sigmoid 都可以得到较好的拟合精度，而且 LeafSpring 和 Softplus 的拟合精度最高. 从收敛速度看，尽管 ReLU 激活函数收敛速度最快，但是最终收敛值误差较大，而 LeafSpring 激活函数和 Softplus 激活函数兼顾了收敛速度和精度. 之所以会出现这样的结果，是因为 Softplus 及 LeafSpring 激活函数本身均为非线性的光滑曲线，当其用于拟合简单的具有单一光滑曲线类型的函数时，可以在小网格规模下拟合出目标函数. 类似地，Sigmoid 函数尽管也可以拟合出目标函数，但是由于其本身具有极强的非线性，所以很难快速训练成目标函数. 而 ReLU 激活函数，由于其本身为分段线性函数，小网格规模较小的情况下，很难将复杂的光滑曲线信息学习到有限的权重系数内.

由图 5(b)可得，随着网格规模的提高，4 种激活函数都可以以较快的速度收敛到较小的误差且精度不相上下. 这主要是因为网格规模的增加使得激活函数的作用减弱，从而 4 种激活函数都可以得到较好的结果. 反观细节，ReLU 激活函数的拟合精度略低于 LeafSpring 和 Softplus 激活函数，这也能一定程度上说明 ReLU 在此类型的数据集上应用受到了限制.

综合分析图 5(a)和图 5(b)，与线性数据集类似，LeafSpring 激活函数在简单非线性映射数据集上的作用与 Softplus 相同且可以在网格规模有限的情况下得到较好的拟合精度.

5.2.3 复杂非线性映射数据集的拟合对比

对于复杂非线性映射数据集，分别令隐藏层节点数为 100 (sparse)和 200 (dense)，可以得到如图 6 所示的拟合和训练结果.

由图 6(a)和图 6(b)可得，无论是训练数据集还是测试数据集，LeafSpring 激活函数比其余 3 种激活函数收敛速度都快，拟合精度都高. 这主要是因为在复杂的非线性映射中，LeafSpring 激活函数可以通过训练刚度系数 k，在不同的节点自适应地学习不同光滑程度的信息. 而 ReLU、Softplus 及 Sigmoid，由于其不能主动调节相邻层之间的权重系数关系，很难在网格规模有限的情况下拟合出复杂的分线性映射.

由图 6(c)可得，当网格规模增大，ReLU 和 LeafSpring 激活函数都可以得到较好的拟合精度和收敛速度. 这主要是因为当网格规模足够大时，ReLU 可以用线性函数拟合出任意非线性函数，从而达到了与 LeafSpring 相当的精度.

综合分析图 6(a)～图 6(c)，LeafSpring 激活函数可以在小网格规模下以较高速度和精度拟合复杂非线性函数且优于其余 3 种激活函数.

5.2.4 分段函数映射数据集的选取的拟合对比

对于分段函数映射数据集 MINST，分别令隐藏层节点数为 20(sparse)和 50(dense)，可以得到如图 7 所示的拟合结果.

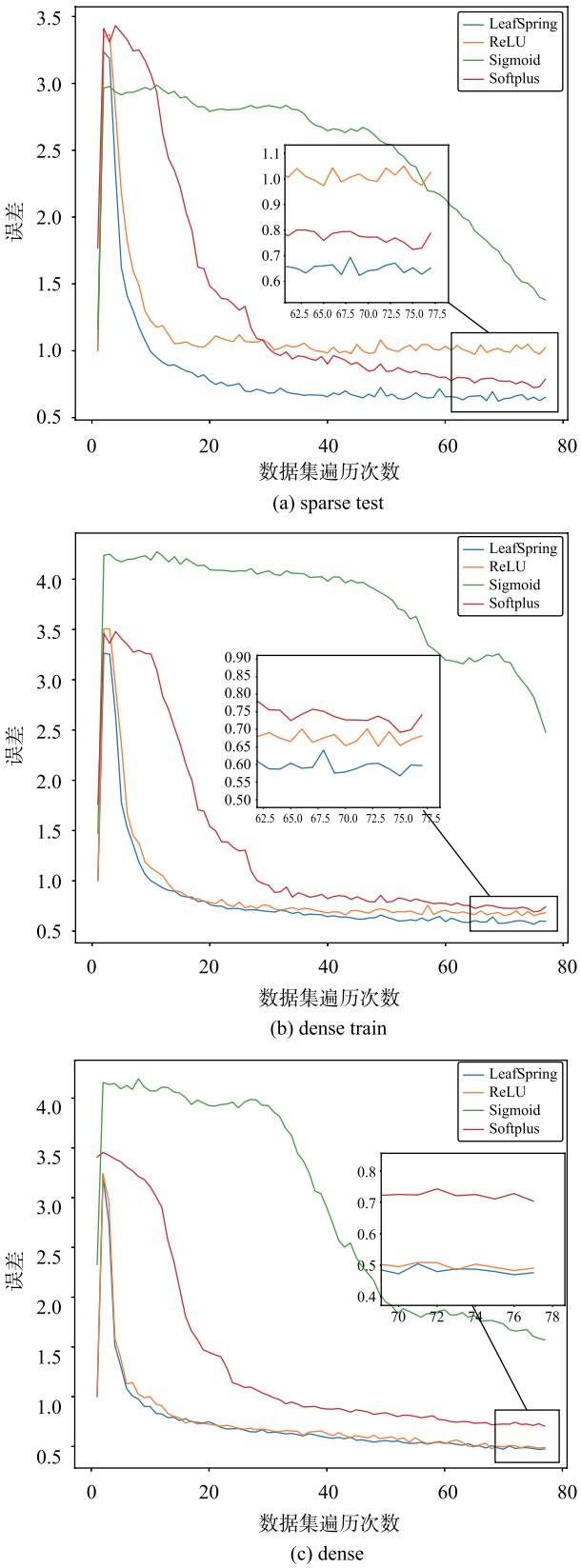

图 6 复杂非线性映射数据集拟合结果对比

Fig.6 The fitting result comparison on the complex nonlinear mapping database

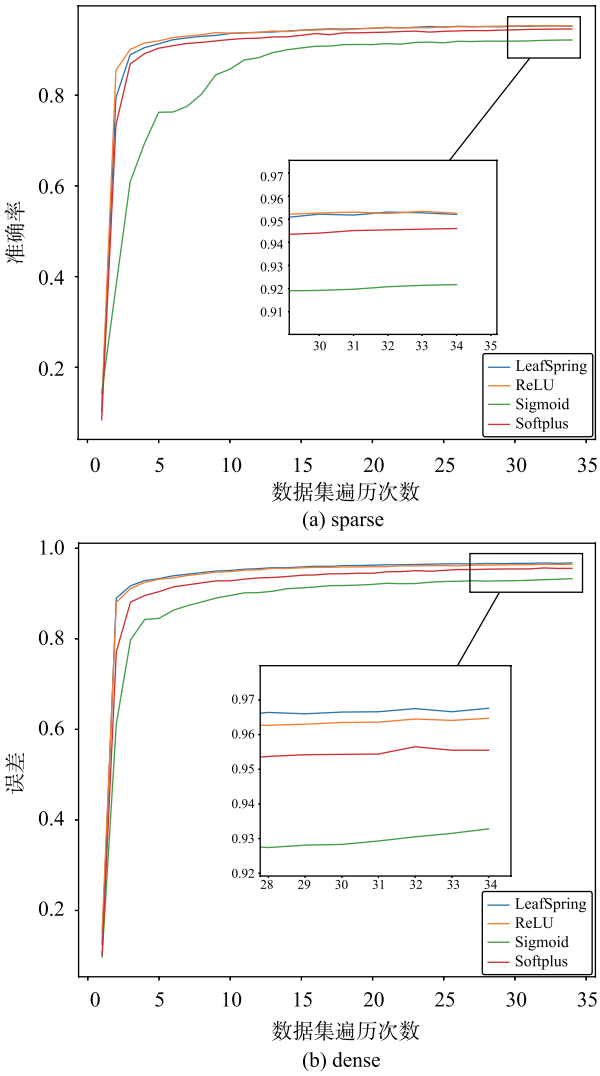

图 7　MNIST 数据集拟合结果对比

Fig.7　The fitting result comparison on the MNIST database

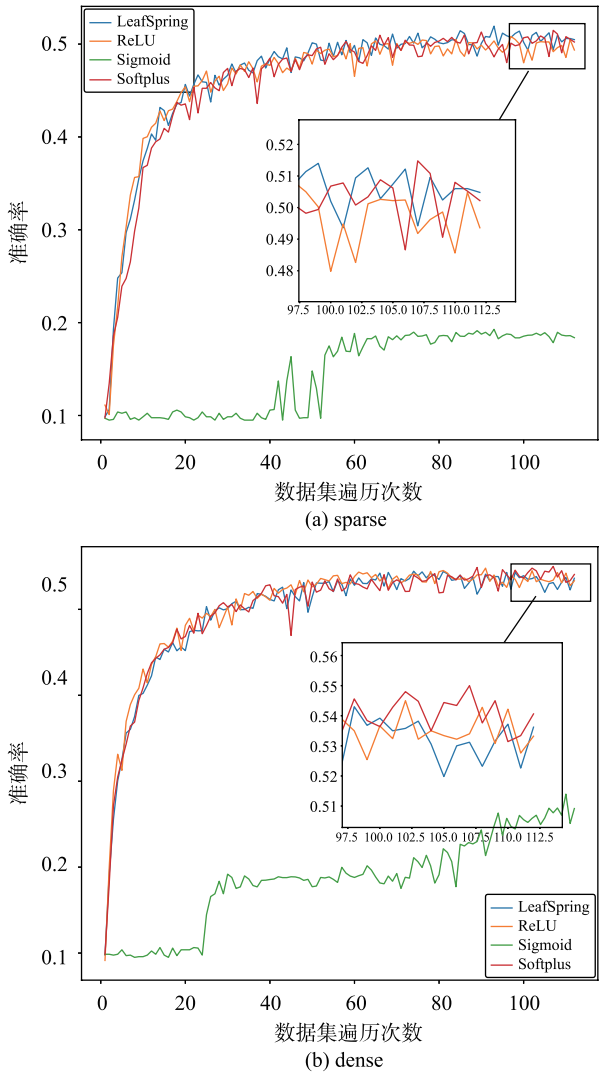

图 8　IFAR-10 数据集拟合结果对比

Fig.8　The fitting result comparison on the CIFAR-10 database

由图 7 可得，无论是在网格稀疏还是稠密的状态下，LeafSpring 和 ReLU 激活函数在收敛速度和精度上均优于其余 2 种激活函数，而且在网格数量更大的情况下，LeafSpring 激活函数可以得到略高于 ReLU 的拟合精度结果. 该结果表明，LeafSpring 激活函数在拟合 MNIST 数据集时，同样有较好的表现，并且在网格数量丰富的情况下，可以学习到更高丰富的信息.

同样，对于分段函数映射数据集 CIFAR-10，由于数据集维数较高，分别令隐藏层节点数为 50（parse）和 200（dense），并以数据集中的后 10% 作为测试样本，可以得到如图 8 所示的拟合结果.

由图 8 可得，无论是在网格稀疏还是稠密的状态下，LeafSpring、ReLU 和 Softplus 激活函数在收敛速度和精度上均优于 Sigmoid. 而且在网格数量较小的情况下，LeafSpring 激活函数可以得到略高于 ReLU 和 Softplus 的拟合精度结果. 该结果表明，当 LeafSpring 激活函数拟合 CIFAR-10 时，同样有较好的表现，并且在网格数量有限的情况下，可以得到更高的拟合精度.

综合 CIFAR-10 和 MINST 两个数据集上的表现，LeafSpring 激活函数能够在拟合分段函数时，以较快的拟合速度，得到更好的拟合效果.

综合分析多数据集下不同激活函数的拟合精度，LeafSpring 在所有的数据集下均表现出了优异的拟合能力. 无论是在拟合速度还是最终拟合精度上，LeafSpring 激活函数均可以以较小的网格规模得到较好的拟合结果. 而且在复杂的非线性数据集下，LeafSpring 激活函数的拟合精度和收敛速度均优于其余 3 种激活函数. 所以，LeafSpring 激活函数的通用性更强，更适于在未知数据集上优先尝试使用.

6　结论

本文从 ReLU 激活函数出发，借鉴 Softplus 与 ReLU 的关系，构建了 LeafSpring 激活函数，并引入刚度系数的概念. 通过分析其函数特性，发现该函数关于原点对称，可以返回负值，并且可以扩大导数的变化率范围，从而缓解

梯度消失等问题，进而通用性更强. 通过分析其作用机理，发现该函数可以在保证相邻两层权重系数乘积不变的情况下，调节权重系数的大小起到权重系数修正的作用. 此外，为了简化计算，本文分析了 LeafSpring 激活函数和 ReLU 和 Softplus 激活函数的转化关系，并给出简化方法. 最后，通过将 LeafSpring、ReLU、Softplus 及 Sigmoid 等激活函数在线性、简单非线性、复杂非线性及分段函数数据集上进行拟合训练，得到了不同激活函数在多种数据集上的拟合误差或精度结果. 结果表明，LeafSpring 激活函数在多种数据集下收敛速度和精度均较好，且通用性更强，并发现其在小网格规模下可以更好地拟合出复杂非线性函数.

诚然，本文所对比的激活函数类型及数据集种类较少，尚不能绝对证明 LeafSpring 激活函数对于所有类型的数据集，在所有结构的神经网络算法中都可以取得较好效果. 此外，本文仅讨论了激活函数本身对拟合速度和精度的影响，未涉及神经网络结构、算法优化对结果的影响. 未来，会对 LeafSpring 进行更加深入的研究，以进一步解决上述问题.

参考文献

[1] 斋藤康毅. 深度学习入门：基于 Python 的理论与实现[M]. 第 1 版. 北京：人民邮电出版社，2018：1－285.
Saito K. Introduction to deep learning：Theory and implementation based on Python[M]. 1st ed. Beijing：People′s post and Telecommunications Press，2018：1－285

[2] 刘宇晴，王天昊，徐旭. 深度学习神经网络的新型自适应激活函数[J]. 吉林大学学报(理学版)，2019，57(4)：1－3.
Liu Y Q，Wang T H，Xu X. A new adaptive activation function of deep learning neural network[J]. Journal of Jilin University (Science Edition)，2019，57 (4)：1－3

[3] Nair V，Hinton G E. Rectified linear units improve restricted Boltzmann machines[C]//27th International Conference on Machine Learning. Haifa，Israel. Madison，WI，USA：Omnipress，2010：807－814.

[4] He K M，Zhang X Y，Ren S Q，et al. Delving deep into rectifiers：Surpassing human-level performance on ImageNet classification[C]//2015 IEEE International Conference on Computer Vision. Piscateway，NJ，USA：IEEE，2015：1026－1034.

[5] Peng J T，Chen L，Iqbal Muhammad Ather，et al. Extreme Learning Machine based on Rectified Nonlinear Units[C]//2nd Workshop on Advanced Research and Technology in Industry Applications. Paris，France：Atlantis Press，2016：1525－1530.

[6] Maas A L，Hannun A Y，Ng A Y. Rectifier nonlinearities improve neural network acoustic models[C/OL]//30th International Conference on Machine Learning.[2019－02－26]. http://ai.stanford.edu/～awni/papers/relu_hybrid_icml2013_final.pdf.

[7] He K，Zhang X，Ren S，et al. Delving deep into rectifiers：Surpassing human-level performance on imagenet classification[C]//IEEE International Conference on Computer Vision. Piscateway，NJ，USA：IEEE，2015：1026－1034.

[8] Xu B，Wang N，Chen T et al. Empirical evaluation of rectified activations in convolutional network[J/OL]. arXiv：1505.00853.(2015－11－27)[2019－05－21]. https://arxiv.org/abs/1505.00853.

[9] Clevert D A，Unterthiner T，Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs)[J/OL]. arXiv：1511.07289.(2016－01－22)[2019－05－26]. https://arxiv.org/abs/1511.07289.

[10] Petersen P，Voigtlaender F. Optimal approximation of piecewise smooth functions using deep ReLU neural networks[J]. Neural Networks，2018，108：296－330.

[11] Konstantin Eckle，Johannes Schmidt-Hieber. A comparison of deep networks with ReLU activation function and linear spline-type methods[J]. Neural Networks，2019，111：232－242.

[12] Liu Y，Wang X，Wang L，et al. A modified leaky ReLU scheme (MLRS) for topology optimization with multiple materials[J]. Applied Mathematics and Computation，2019，352：188－204.

[13] Zhao H Z，Liu F X，Li L Y，et al. A novel Softplus linear unit for deep convolutional neural networks[J]. Applied Intelligence，2017，48(4)：1－14.

[14] Oliver Y. Riemannian metrics for neural networks I：Feedforward networks[J/OL]. arXiv：1303.08.18，2015.(2015－01－03)[2019－05－26]. https://arxiv.org/abs/1303.0818? context = cs.LG.

[15] Shi P，Li G H，Yuan Y M，et al. Prediction of dissolved oxygen content in aquaculture using Clustering-based Softplus extreme learning machine[J]. Computers and Electronics in Agriculture，2019，157：329－338.

[16] 郜丽鹏，郑辉. 基于 PReLUs-Softplus 非线性激励函数的卷积神经网络[J]. 沈阳工业大学学报，2018，40(1)：54－59.
Gao L P，Deng H. Convolution neural network based on PReLUs-Softplus nonlinear excitation function[J]. Journal of Shenyang University of technology，2018，40(1)：54－59.

[17] 孙艳丰，杨新东，胡永利，等. 基于 Softplus 激活函数和改进 Fisher 判别的 ELM 算法[J]. 北京工业大学学报，2015，41(9)：1341－1348.
Sun Y F，Yang X D，Hu Y L，et al. ELM algorithm based on Softplus activation function and improved Fisher discriminant[J]. Journal of Beijing University of Technology，2015，41(9)：1341－1348.

[18] Senior A，Lei X. Fine context，low-rank，Softplus deep neural networks for mobile speech recognition[C/OL]//2014 IEEE International Conference on Acoustics，Speech and Signal Processing. Piscateway，NJ，USA：IEEE.(2014－07－14)[2019－05－26]. https://ieeexplore.ieee.org/document/6855087.

(下转第 322 页)

na, 2016, 23(11): 1714 – 1718.
[11] 王修中, 岳红, 高东杰. 二阶加滞后连续模型的直接辨识[J]. 自动化学报, 2001, 37(5): 728 – 731.
Wang X Z, Yue H, Gao D J. Direct Identification of Continuous Second-order plus Dead-time Model[J]. Acta Automatica Sinica, 2001, 37(5): 728 – 731.
[12] 严晓久, 周爱国, 林建平, 等. 基于辅助变量法的系统参数辨识[J]. 机床与液压, 2006(12): 180 – 181, 184.
Yan X J, Zhou A G, Lin J P, et al. Parametric system identification based on instrumental variable method[J]. Machine Tool & Hydraulics, 2006(12): 180 – 181, 184.
[13] Ling X, Lei C, Xiong W. Parameter estimation and controller design for dynamic systems from the step responses based on the Newton iteration [J]. Nonlinear Dynamics, 2015, 79(3): 2155 – 2163.
[14] Salim A. Identification from step response-the integral equation approach[J]. The Canadian Journal of Chemical Engineering, 2016(94): 2243 – 2256.
[15] Chris C, Tindle J, Burn, K. A comparison of software-based approaches to identifying FOPDT and SOPDT model parameters from process step response data[J]. Applied Mathematical Modelling, 2016(40): 100 – 114.
[16] 王浩宇, 张云生, 张果. 系统辨识及自适应控制系统算法仿真实现[J]. 控制工程, 2008(S2): 77 – 80.
Wang H Y, Zhang Y S, Zhang G. Implementation on system identification and adaptive control system simulation algorithm[J]. Control Engineering of China, 2008(S2): 77 – 80.
[17] 靳其兵, 刘子宜. 基于阶跃响应的带纯滞后闭环辨识新方法[J]. 系统仿真学报, 2010, 22(9): 2168 – 2172.
Jin Q B, Liu Z Y. Novel identification method with time delay from step response[J]. Journal of System Simulation, 2010, 22(9): 2168 – 2172.
[18] Fedele G. Frequency response estimation from impulse or step-like response by virtual experiments[J]. Asian Journal of Control, 2016, 18(4): 1289 – 1298.
[19] 徐江华, 孙荣, 邵惠鹤. 大滞后过程的 PI 控制器整定[J]. 控制与决策, 2004, 19(1): 99 – 101.
Xu J H, Sun R, Shao H H. PI controller tuning for large dead-time processes[J]. Control and Decision, 2004, 19(1): 99 – 101.

作者简介

李　闯(1994 –), 男, 硕士生. 研究领域为系统辨识, 先进过程控制等.
王亚刚(1967 –), 男, 博士, 教授. 研究领域为系统辨识, 先进过程控制等.

(上接第 314 页)

[19] Shi P, Li G, Yuan Y, et al. Prediction of dissolved oxygen content in aquaculture using clustering-based Softplus extreme learning machine [J]. Computers and Electronics in Agriculture, 2019, 157: 329 – 338.
[20] 埃里克·马瑟斯. Python 编程从入门到实践[M]. 第 1 版. 北京: 人民邮电出版社, 2018: 1 – 459.
Eric M. Python programming from introduction to practice[M]. 1st ed. Beijing: People's post and Telecommunications Press, 2018: 1 – 459.

作者简介

郜天柱(1993 –), 男, 博士生. 研究领域为水下推进技术, 水下推进器控制等.