

谢文杰, 刘玉兰. 基于Log-sum正则化的稀疏神经网络[J]. 广东工业大学学报, 2025, 42(6): 78–85. doi: 10.12052/gdutxb.240107.
Xie Wenjie, Liu Yulan. Sparse neural network based on Log-sum regularization[J]. Journal of Guangdong University of Technology, 2025, 42(6): 78–85. doi: 10.12052/gdutxb.240107.

基于Log-sum正则化的稀疏神经网络

谢文杰, 刘玉兰

(广东工业大学 数学与统计学院, 广东 广州 510520)

摘要: 为了训练出一个稀疏的神经网络, 本文选用Log-sum函数作为正则项, 交叉熵函数作为损失项建立模型, 随后用邻近梯度法对它进行求解, 其学习率用Meta-LR-Schedule-Net网络训练得到。在MNIST、Fashion-MNIST、CIFAR-10和CIFAR-100四个数据集上的数值实验结果表明: 在同样的学习率规则下, 用Log-sum函数作为正则项比用其他能诱导稀疏的函数, 如向量1范数、截尾向量1范数或向量1/2范数等作为正则项建立的模型能把网络训练得更稀疏; 在稀疏度近似的情况下, 用Meta-LR-Schedule-Net训练得到的学习率, 比使用固定规则训练得到的网络有更高的正确率。

关键词: 稀疏神经网络; Log-sum 函数; 邻近梯度法

中图分类号: TP301

文献标志码: A

文章编号: 1007-7162(2025)06-0078-08

Sparse Neural Network Based on Log-sum Regularization

Xie Wenjie, Liu Yulan

(School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: In order to train a sparse neural network, the Log-sum function is chosen as the regularization term and the cross entropy function as the loss function to establish the model and solved it by proximal gradient algorithm, whose learning rate was obtained by Meta-LR-Schedule-Net. The network is trained more sparsely by the model with the Log-sum regularized function than by other models with other induced sparse functions, such as the 1-norm function, transformed 1-norm function, and 1/2-norm function, according to numerical experiments conducted on four data sets: MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. Additionally, the network trained by using the Meta-LR-Schedule-Net learning rule is more accurate than those trained by using a fixed learning rate.

Key words: sparse neural network; Log-sum function; proximal gradient algorithm

深度神经网络在计算机视觉^[1]与自然语言处理^[2]等多个领域展现出广泛的应用价值。然而,随着应用场景的日益复杂,网络参数规模不断膨胀,这不仅提升了运算成本,还加剧了过拟合的风险。为了应对这些挑战,去除网络中的冗余参数被视为一种有效的解决策略。例如, Alvarez和Salzmann^[3]在训练过程中将组稀疏正则项引入到目标函数中,在保持网络准确性的同时,实现了高达80%的参数减量,揭示了深度神经网络中存在大量的冗余参数,即便去除这些参数,网络的准确性仍能得到保障。除了基于稀疏正则项的方法,还有其他多种促进神经网络稀疏性的

手段,如剪枝(Pruning)^[4]和随机失活(Dropout)^[5]等。在这些方法中,基于稀疏正则项的训练方法因其较强的可解释性,能够应用于各种神经网络框架。

利用稀疏正则项训练稀疏神经网络的关键在于选择一个能够诱导稀疏性的函数作为正则项。向量的零模范数 l_0 ,即向量非零元素的个数,被认为是最理想的稀疏正则项,它直接惩罚模型中的非零参数数量。然而, l_0 范数具有组合性质,导致 l_0 范数正则化问题是NP(NP-Hard)难问题^[6]。因此常常选用 l_0 范数的近似代理函数来替代。目前,最流行的近似代理函数是向量的1范数 l_1 ^[7]。 l_1 范数正则化问题是一个备受

收稿日期: 2024-09-10 录用日期: 2024-12-12 网络首发日期: 2025-06-03

基金项目: 广东省基础与应用基础研究基金资助项目(2023A1515012891)

作者简介: 谢文杰(1999-),男,硕士研究生,主要研究方向为最优化理论及应用,E-mail: 1137656028@qq.com

通信作者: 刘玉兰(1977-),女,教授,博士,主要研究方向为最优化理论及应用,E-mail: ylliu@gdut.edu.cn

深入研究的凸问题^[8]。然而,使用 l_1 范数来近似 l_0 范数会对较大分量施加过重的惩罚。相比之下, l_0 范数对分量的惩罚是平等的。为了克服这个缺陷,越来越多的学者开始尝试采用可以诱导稀疏的非凸的代理函数来近似 l_0 范数,从而得到无偏模型并获得更加稀疏的解。最近的研究表明,使用非凸正则化函数,如截尾向量1范数^[9]、桥接向量 p 范数^[10] ($0 < p < 1$)等,更容易获得稀疏的解。作为 l_0 范数的一种非凸近似代理函数,Log-sum函数因其具有出色的诱导稀疏特性,被用作正则项来构建稀疏神经网络模型。随后,采用梯度下降法对该模型进行求解^[11]。但随后文献^[12]借助Log-sum的一阶近似,用迭代重加权 l_1 算法求解Log-sum正则化神经网络模型。

在求解Log-sum正则化神经网络模型的过程中,文献^[11]在计算过程中选择忽略Log-sum函数在零点的不可导性,同时文献^[12]使用迭代重加权 l_1 算法求解存在精度上的问题。

为了克服以上问题,本文借助Log-sum函数的邻近算子的刻画^[13],用邻近梯度法对Log-sum正则化稀疏神经网络模型进行求解,且邻近梯度法中的学习率使用基于长短期记忆网络(Long Short-term Memory, LSTM)的Meta-LR-Schedule-Net^[14]网络训练得到。本文在多个数据集以及多个经典网络架构中进行实验,与流行的基准模型相比,本文模型的有效性得到了体现。

本文的主要贡献有3点:

(1) 使用邻近梯度法对带Log-sum正则项的神经网络模型进行求解。本文通过Log-sum函数邻近算子闭式解进行计算,实用性和精度更高。

(2) 采用LSTM来训练邻近梯度法的学习率,这种方法能够提升模型的性能。

(3) 在MNIST、Fashion-MNIST、CIFAR-10和CIFAR-100四个公开数据集上,对比了不同正则函数以及不同学习率选择方法的效果。实验结果表明,本文提出的模型和求解方法表现出更强的优势。

1 准备知识

1.1 正则化稀疏神经网络模型

以全连接神经网络为例。考虑用含有 N 个样本点的集合来训练一个有1个输入层, $L-1$ 个隐藏层和1个输出层的全连接神经网络。假设含第 l 层网络的神经元数量记为 $h(l)$,第 l 层的权重记 $w^{(l)} \in \mathbb{R}^{h(l+1) \times h(l)}$,偏置记为 $b^{(l)} \in \mathbb{R}^{h(l+1)}$,激活函数为 $a^{(l)}: \mathbb{R}^{h(l+1)} \rightarrow \mathbb{R}^{h(l+1)}$, ($l = 0, \dots, L-1$)。假设 $\mathcal{D} := \{x_i, y_i\}_{i=1}^N$ 为样本点集,其中

$y_i \in \mathbb{R}^{h(L)}$ 是样本点 $x_i \in \mathbb{R}^{h(0)}$ 的标签。记第 l 层的参数 $\Theta^{(l)} = (w^{(l)}, b^{(l)}) \in \mathbb{R}^{h(l+1) \times h(l)} \times \mathbb{R}^{h(l+1)}$ 。

$$\min_{\Theta^{(l)}} \mathcal{L}(f(\{\Theta^{(l)}\}_{l=0}^{L-1}), \mathcal{D}) + \lambda \sum_{l=0}^{L-1} \Omega(\Theta^{(l)}) \quad (1)$$

式中, $\mathcal{L}$ 为损失函数, Ω 为正则化函数, $\lambda > 0$ 为正则化参数, f 为神经网络的输出。用 $x^{(l)}$, ($l = 1, \dots, L$)表示上述神经网络第 l 层的输入,则对任意给定样本点 $x \in \mathbb{R}^n$ (n 与 $h(0)$ 应在数值上相等),该神经网络的输出有

$$f(x) = a^{(L-1)}(w^{(L-1)}x^{(L-1)} + b^{(L-1)}) = a^{(L-1)}(w^{(L-1)}(\dots(a^{(0)}(w^{(0)}x + b^{(0)}))\dots) + b^{(L-1)}) \quad (2)$$

常用的损失函数有均方误差损失、平均绝对误差损失和交叉熵损失等,常见的正则化函数有截尾向量1范数、桥接向量 p 范数 ($0 < p < 1$) 和 Log-sum 函数等。本文主要采用交叉熵作为损失函数,Log-sum函数作为正则化函数。对于给定的 $\epsilon > 0$, Log-sum函数 P_ϵ 定义为

$$P_\epsilon(t) = \log \left(1 + \frac{|t|}{\epsilon} \right) \quad (3)$$

1.2 Log-sum正则化的邻近算子

在邻近梯度算法中,刻画正则函数的邻近算子是关键。为此,本文先给出邻近算子的定义。任意给定函数 $g: \mathbb{R} \rightarrow \mathbb{R}$,给定参数 $\mu > 0, \lambda > 0$,其邻近算子定义为

$$\text{prox}_{\mu\lambda g}(\tau) := \arg \min_{t \in \mathbb{R}} \left\{ \frac{1}{2\mu}(t - \tau)^2 + \lambda g(t) \right\}, \forall \tau \in \mathbb{R} \quad (4)$$

特别地,Log-sum函数的邻近算子 $\text{prox}_{\mu\lambda P_\epsilon}$ 刻画如下^[13]。

引理1 对于给定的 $\epsilon > 0, \mu > 0, \lambda > 0$,对任意给定的 $\tau \in \mathbb{R}$, Log-sum函数的邻近算子 $\text{prox}_{\mu\lambda P_\epsilon}$ 定义为

$$\text{prox}_{\mu\lambda P_\epsilon}(\tau) := \arg \min_{t \in \mathbb{R}} \left\{ \frac{1}{2\mu}(t - \tau)^2 + \lambda \log \left(1 + \frac{|t|}{\epsilon} \right) \right\} \quad (5)$$

当 $\sqrt{\mu\lambda} \leq \epsilon$ 时,

$$\text{prox}_{\mu\lambda P_\epsilon}(\tau) = \begin{cases} \{0\}, & \text{若 } |\tau| \leq \frac{\mu\lambda}{\epsilon} \\ \{\text{sgn}(\tau)r_1(|\tau|)\}, & \text{若 } |\tau| > \frac{\mu\lambda}{\epsilon} \end{cases} \quad (6)$$

当 $\sqrt{\mu\lambda} > \epsilon$ 时,记 $r(\tau)$ 为

$$r(\tau) = \frac{1}{2}(\tau - \epsilon) + \sqrt{\frac{1}{4}(\tau + \epsilon)^2 - \mu\lambda} \quad (7)$$

记 τ_* 为方程

$$r(\tau) = \frac{1}{\mu\lambda}(r(\tau))^2 - \frac{2}{\mu\lambda}\tau r(\tau) + P_\epsilon(r(\tau)) \quad (8)$$

在区间 $[2\sqrt{\mu\lambda} - \epsilon, \mu\lambda/\epsilon]$ 上的根。则

$$\text{prox}_{\mu\lambda P_\epsilon}(\tau) = \begin{cases} \{0\}, & \text{若 } |\tau| < \tau_* \\ \{0, \text{sgn}(\tau)r(\tau_*)\}, & \text{若 } |\tau| = \tau_* \\ \{\text{sgn}(\tau)r(|\tau|)\}, & \text{若 } |\tau| > \tau_* \end{cases} \quad (9)$$

数值实验中,当 $|\tau| = \tau_*$ 时,取 $\text{prox}_{\mu\lambda P_\epsilon}(\tau) = \{0\}$ 。

1.3 自动化学习率选取策略

在邻近梯度算法中,学习率的选择是另一个关键问题。通常情况下,第 t 次迭代步学习率 μ_t 的选取策略有以下几种:(1) 固定学习率。给定 $\mu_0 > 0$,对任意 t 次迭代学习率 $\mu_t \equiv \mu_0$ 。(2) 指数衰减学习率。给定 $0 < \mu_0, 0 < \gamma < 1$,第 t 次迭代学习率 $\mu_t = \mu_0 \times (\gamma)^{(t-1)}$ 。(3) 带热重启的梯度下降法(Stochastic Gradient Descent with Warm Restarts, SGDR)。第 t 次迭代的学习率常选取 $\mu_t = a_{\min} + (a_{\max} - a_{\min})(1 + \cos(\pi T/T))/2$,其中学习率设定范围记为 $[a_{\min}, a_{\max}]$, T 表示总迭代次数。(4) 自适应选取学习率,例如随机梯度下降法(Stochastic Gradient Descent, SGD)的各种变体,如带动量的随机梯度下降法(Gradient Descent With Momentum, SGDM)^[15]和Adam^[16]等。

这些学习率选择策略的共同点在于它们通常在训练开始前就预先设定好了值或遵循固定的表达式。这种做法可能会导致以下两个缺陷:首先,当待训练的网络模型相对复杂时,学习率的选择方法往往缺乏足够的灵活性。其次,每次训练不同的模型时,都需要重新调整学习率的变化规则,这无疑大大增加了时间成本和计算开销。

为了解决这两个缺陷,可以采用元学习的方法来动态确定学习率。元学习的概念最早可追溯到20世纪30年代^[17],而近年来它已被应用于大规模的神经网络优化问题中^[18-19]。本文所采用的MLR-SNet正是一种元学习方法,它基于LSTM架构,通过学习梯度下降法中的学习率,使学习率能够隐式地动态调整。文献[14]的实验结果表明,通过MLR-SNet训练得到的网络有更优秀的性能。

2 算法介绍

2.1 Log-sum 正则化稀疏神经网络的邻近梯度法

$\Theta^{(l)} \in \mathbb{R}^{h(l+1) \times h(l)} \times \mathbb{R}^{h(l+1)}$, 对任意 $l = 0, \dots, L$, 本文定义

$$\Psi_\epsilon(\Theta^{(l)}) := \sum_{i=1}^{h(l+1)} \sum_{j=1}^{h(l+1)} \log \left(1 + \frac{|(\Theta^{(l)})_{ij}|}{\epsilon} \right) \quad (10)$$

本文主要考虑Log-sum 函数作为正则函数的神经网络模型,即模型(1)具体为

$$\min_{\Theta^{(l)}} \mathcal{L}(f(\{\Theta^{(l)}\}_{i=0}^{L-1}), \mathcal{D}) + \lambda \sum_{l=1}^{h+1} \Psi_\epsilon(\Theta^{(l)}) \quad (11)$$

考虑使用邻近梯度法来求解问题(11),即第 $t+1$ 步的迭代格式为

$$\Theta_{t+1}^{(l)} = \arg\min_{\Theta_t^{(l)}} \left\{ \frac{1}{2\mu} \|\Theta_t^{(l)} - (\Theta_t^{(l)} - \mu \nabla \mathcal{L}(f(\{\Theta_t^{(l)}\}), \mathcal{D}))\|_F^2 + \lambda \Psi_\epsilon(\Theta_t^{(l)}) \right\} \quad (12)$$

式中, $\nabla \mathcal{L}$ 为一个矩阵,其每个元素是损失函数 $\mathcal{L}$ 关于 $\Theta_{ij}^{(l)}$ 的导数。根据式(12),每次更新时都需要在整个训练数据集上计算梯度,这通常会导致较大的计算开销。为了降低这种开销,实践中常采用SGD作为替代^[20],即将式(12)替换为式(13)。

$$\Theta_{t+1}^{(l)} = \arg\min_{\Theta_t^{(l)}} \left\{ \frac{1}{2\mu} \|\Theta_t^{(l)} - (\Theta_t^{(l)} - \frac{\mu}{m_0} \sum_{i=1}^{m_0} \nabla \mathcal{L}(f(\{\Theta_t^{(l)}\}), \mathcal{D}))\|_F^2 + \lambda \Psi_\epsilon(\Theta_t^{(l)}) \right\} \quad (13)$$

式中, m_0 为小批量大小,从数据集 $\mathcal{D}$ 中随机抽取 m_0 个样本点组成训练数据集 $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{m_0}$ 。根据 P_ϵ 的邻近算子的定义,第 $t+1$ 步的迭代公式(13)等价于

$$(\Theta_{t+1}^{(l)})_{ij} = \text{prox}_{\mu\lambda P_\epsilon} \left(\left(\Theta_t^{(l)} - \frac{\mu}{m_0} \sum_{i=1}^{m_0} \nabla \mathcal{L}(f(\{\Theta_t^{(l)}\}_{i=0}^{L-1}), \mathcal{D}_{\text{train}}) \right)_{ij} \right) \quad (14)$$

2.2 基于MLR-SNet训练学习率的算法

假设 $W_1 \in \mathbb{R}^{d \times (d+1)}$, $W_2 \in \mathbb{R}^{4d \times 2d}$, $W_3 \in \mathbb{R}^{1 \times d}$ 为MLR-SNet的可训练的参数矩阵, $[0, \gamma]$ 为学习率自动调节的范围。记 $c_t, h_t \in \mathbb{R}^d$ 为第 t 次迭代时的记忆单元和隐状态, I, F, O_t 分别为网络的输入门、遗忘门和输出门,则MLR-SNet的迭代方式为

$$\begin{pmatrix} I_{t+1} \\ F_{t+1} \\ O_{t+1} \\ g_{t+1} \end{pmatrix} := \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W_2 \begin{pmatrix} \text{ReLU} \\ \text{ReLU} \end{pmatrix} W_1 \begin{pmatrix} h_t \\ \mathcal{L}_{t+1} \end{pmatrix} \quad (15)$$

$$c_{t+1} = F_{t+1} \odot c_t + I_{t+1} \odot g_{t+1} \quad (16)$$

$$h_{t+1} = O_{t+1} \odot \tanh(c_{t+1}) \quad (17)$$

$$\mu_{t+1} = \gamma \sigma(W_3 h_{t+1}) \quad (18)$$

记 $\theta_t := \{h_{t-1}, c_{t-1}\}$, $\phi_t = (W_1, W_2, W_3)_t$, 其中 $\odot$ 表示 Hadamard 乘积。则 θ_t, ϕ_t 的迭代过程为

$$\phi_{t+1} = \phi_t + \text{Adam}(\nabla \mathcal{L}(f(\{\theta_t^{(l)}\}_{l=0}^{L-1}), \mathcal{D})) \quad (19)$$

$$\mu_{t+1} = \mathcal{A}(\mathcal{L}(f(\{\theta_t^{(l)}\}_{l=0}^{L-1}), \mathcal{D}), \theta_t, \phi_{t+1}) \quad (20)$$

式中, $\mathcal{A}$ 为 MLR-SNet 的输出, 过程如图 1 所示。结合式 (14) 可得到表达式

$$(\theta_{t+1}^{(l)} = \text{prox}_{\mu \lambda P_\epsilon}((\theta_t^{(l)} - \mathcal{A}(\mathcal{L}(f(\{\theta_t^{(l)}\}_{l=0}^{L-1}), \mathcal{D}), \theta_t, \phi_{t+1}) \times \nabla \mathcal{L}(f(\{\theta_t^{(l)}\}_{l=0}^{L-1}))_{ij})) \quad (21)$$

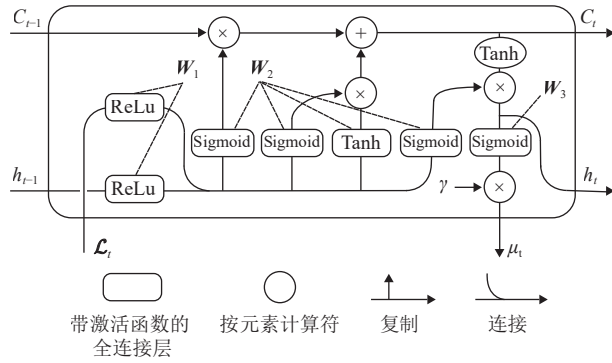


图 1 MLR-SNet 架构

Fig.1 MLR-SNet architecture

整体的算法步骤如算法 1 所示。

算法 1 自动化学习率的 Log-sum 正则化网络算法

(1) 输入: 初始学习率 μ_0 , 小批量大小 m_0 , 训练数据集 $\mathcal{D}_{\text{train}} \subseteq \mathcal{D}$, 网络层数 L , 初始网络参数 θ_0 , 迭代轮数 T , 初始参数 θ_0, ϕ_0 。

(2) 输出: 网络参数 θ_T 。

a) for $t = 1 : T$ do

b) 从 $\mathcal{D}_{\text{train}}$ 中选取 m_0 个样本 $\{(x_i, y_i)\}_{i=1}^{m_0}$ 。

c) for $l = 0 : L - 1$ do

d) 通过式 (19) 更新 ϕ_{t+1} 。

e) 通过式 (16) 和 (18) 更新 θ_{t+1} 。

f) 通过式 (20) 更新 μ_{t+1} 。

g) 通过式 (14) 更新 $\theta_{t+1}^{(l)}$ 。

h) end for

i) 移除 θ_T 中权重和偏置同时为零的神经元。

3 实验

3.1 实验数据

本文使用的数据集包括: MNIST、Fashion-MNIST、CIFAR-10 和 CIFAR-100。

(1) MNIST: 该数据集包含了大小为 28×28 的由 0~9 的共 70 000 张灰度手写数字图像。训练样本的数量为 60 000 个, 测试样本为 10 000 个。

(2) Fashion-MNIST: 该数据集包括一个包含 60 000 个样本的训练集和一个包含 10 000 个样本的测试集。每个样本是一个大小为 28×28 的灰度图像, 每个样本的内容对应 10 个类别 (T 恤、裤子、裙子等) 中的一个。

(3) CIFAR-10: 该数据集包含了 10 个类别 (飞机、汽车、鸟、猫等) 共计 60 000 张大小为 32×32 的彩色图像, 每个类别有 6 000 张图像。数据集被分成一个包含 10 000 个测试样本以及 50 000 个训练样本。

(4) CIFAR-100: CIFAR-100 是一个相比 CIFAR-10 更复杂的图像分类数据集, 其包含了 100 个不同类别的彩色图像。每个类别包含 600 张 32×32 像素的图像, 其中 500 张用于训练, 100 张用于测试。

3.2 实验设置及评价指标

本文实验使用的 CPU 为 Intel(R) Core(TM) i7 14700KF CPU @ 3.40 GHz, GPU 为 NVIDIA GeForce RTX 4070Ti 12G, 使用的 CUDA 版本为 12.1。本文实验所用的模型均通过 Python 的深度学习框架 PyTorch 实现, 所使用的版本为 2.3.1。

为了将本文提出的 Log-sum 正则化方法与对照组进行比较, 本文采用了两个指标进行定量分析: 验证正确率 (Accuracy) 和网络中使用的非零参数百分比 (简称参数使用率, Parameter Used)。验证正确率是衡量分类模型性能的基本标准, 而参数使用率则反映了模型精简的程度, 是评估稀疏引导能力的重要指标。通过这两个指标, 可以全面地评估所设置各组模型的性能。

验证正确率的计算方式为网络模型在测试集中预测正确的样本数与测试集样本总数之比, 结果保留四位小数。正确率越高, 说明模型的性能越好。参数使用率的计算方式是网络模型中取值非零的参数量与网络模型总参数量之比, 结果保留两位小数。参数使用量越低, 说明模型的性能越好。

考虑到复杂程度的网络架构会对实验结果造成的影响, 本文选择了相对简单的 LeNet-5 和相对复杂的 ResNet-18 和 WideResNet-28-10 两个著名的神经网络架构进行实验。在实验中, 使用 LeNet-5 对 MNIST 和 Fashion-MNIST 数据集进行训练, 使用 ResNet-18 和 WideResNet-28-10 对 CIFAR-10 和 CIFAR-100 数据集进行训练。同时, 使用独热码对不同图像的类别进行编码, 根据正态分布对网络权重进行随机初始化, 并选择交叉熵函数作为损失函数。在超参数的选择上, 注意到 Log-sum 函数只有两个参数, 分别为 λ 和 ϵ , 对不同的数据集进行训练时, 需要对 λ 和 ϵ 进行不同取值。

3.3 基准模型

(1) l_2 : 向量2范数(l_2 范数) 作为正则项。 l_2 范数是凸且光滑的,通常作为一种经典的正则化方法,常被用于改善模型的泛化性能。然而, l_2 范数不能诱导网络模型中的参数变为零。在实验中, l_2 函数作为正则项的网络模型通过随机梯度算法进行训练。

(2) l_1 : 以 l_1 范数作为正则项。 l_1 范数是凸且非光滑的,常作为正则项增强网络模型的稀疏性。在实验中, l_1 范数作为正则项的网络模型通过随机梯度算法进行训练。

(3) CaP: 以Capped l_1 penalty (CaP)^[21] 函数作为正则项。CaP函数是非凸且非光滑的。在实验中,CaP函数作为正则项的网络模型通过随机梯度算法进行训练。

(4) $l_{1/2}$: 以向量1/2范数($l_{1/2}$ 范数)^[22]作为正则项。 $l_{1/2}$ 函数是非凸且非光滑的,作为正则项能使网络模型有较强的稀疏性。 $l_{1/2}$ 函数作为正则项的网络模型通过随机梯度算法进行训练。

3.4 对比实验

本文提出的结合MLR-SNet的Log-sum正则化的稀疏神经网络模型与其他函数作为正则项的模型的对比结果如表1所示,不与MLR-SNet结合的对比结果如表2、表3所示,粗体表示在该实验设置下取得最优的结果。

表 1 模型在CIFAR-10和CIFAR-100数据集的对比实验结果
Table 1 Experimental results of the model in the CIFAR-10 and CIFAR-100 dataset %

模型	CIFAR-10		CIFAR-100	
	Accuracy	Parameters Used	Accuracy	Parameters Used
l_2 +MLR-SNet	94.80	100	80.44	100
l_1 +MLR-SNet	94.40	42	80.30	60
CaP+MLR-SNet	94.48	27	78.97	49
$l_{1/2}$ +MLR-SNet	94.66	41	80.22	45
Log-sum+MLR-SNet	94.75	26	80.38	45

如表1所示,本文提出的模型在正确率和参数使用量的指标上,均优于除了 l_2 外的其他基准模型,相关结果分析如下。

(1) 在与 l_2 +MLR-SNet模型的对比中, l_2 +MLR-SNet模型比Log-sum+MLR-SNet模型高0.05个百分点的正确率,这是由于 l_2 范数作为正则项时不会使网络模型变得稀疏,使得 l_2 +MLR-SNet模型保留了更多的参数,保留了有效信息。

(2) 在与 l_1 +MLR-SNet模型的对比中,Log-sum+MLR-SNet模型在CIFAR-10和CIFAR-100数据

表 2 模型在MNIST和Fashion-MNIST数据集对比实验结果
Table 2 Experimental results of the model in the MNIST and Fashion-MNIST dataset %

模型	Layer1		Layer2		Layer3		Layer4		Layer5	
	W_1	B_1	W_2	B_2	W_3	B_3	W_4	B_4	W_5	B_5
l_2	100	100	100	100	100	100	100	100	100	100
l_1	88	100	76	69	62	32	78	55	89	100
CaP	48	33	26	38	1	1	6	2	42	50
$l_{1/2}$	33	50	7	19	1	0	4	2	35	40
Log-sum	34	27	6	6	1	0	9	0	34	10

表 3 模型的结果对比
Table 3 Comparison of parameter used with different method %

模型	MNIST		Fashion-MNIST	
	Accuracy	Parameters Used	Accuracy	Parameters Used
l_2	98.67	100	90.88	100
l_1	97.76	62	90.15	76
CaP	97.44	42	90.33	64
$l_{1/2}$	97.63	13	89.99	57
Log-sum	97.77	8	90.57	51

集上的验证正确率分别高了0.35个百分点和0.08个百分点,同时参数使用量分别低了16个百分点和15个百分点。这是由于 l_1 范数作为正则项时,虽然会促进网络模型的稀疏,但是对数值较大的参数的惩罚是相对较大的,会一定程度上损失有效信息,所以难以使网络模型同时保持较高正确率和较低的参数使用量。

(3) 作为同样是非凸且非光滑的函数,CaP函数和 $l_{1/2}$ 范数作为 l_0 范数的近似,分别在CaP+MLR-SNet和 $l_{1/2}$ +MLR-SNet模型表现出了较强的性能。实验结果表明,在正则项的系数取相同的值时, $l_{1/2}$ +MLR-SNet模型表现出明显更强的稀疏性。在CIFAR-100数据集上, $l_{1/2}$ +MLR-SNet与Log-sum+MLR-SNet在两个指标上的表现接近。

如表3所示,在相对更简单的MNIST和Fashion-MNIST数据集上,当Log-sum函数作为正则项时,网络模型同样有良好的表现。在MNIST数据集上,即使仅用8%的参数,也能保持与基准模型相近的正确率。而在Fashion-MNIST数据集上,参数使用量均优于其他基准模型。考虑到LeNet-5是一个相对简单的网络,这部分实验使用的是随迭代轮数增加而衰减的学习率选取策略。

表2展示了MNIST数据集上LeNet-5每层的参数使用量。其中 W_i 表示第 i 层权重的参数使用量, B_i 表示第 i 层偏置的参数使用量。正则函数在网络模型中间层产生的稀疏效果较为显著。

3.5 交叉验证实验

为了避免偶然性,更全面地评估模型的表现,本节将CIFAR-10和CIFAR-100数据集中的10类数据平均分成5份进行分层五折交叉验证。交叉验证的实验设置与对比实验相同,5次实验结果如表4所示。

表 4 Log-sum+MLR-SNet模型的交叉验证结果
Table 4 Cross validation experimental results of Log-sum+MLR-SNet model %

序号	CIFAR-10		CIFAR-100	
	Accuracy	Parameters Used	Accuracy	Parameters Used
1	94.88	26	80.22	45
2	94.77	21	80.08	40
3	94.77	20	80.44	46
4	94.92	26	80.58	48
5	94.85	27	80.14	41
Average	94.83	24	80.33	44

由表4可以看出,本文提出的结合MLR-SNet的Log-sum正则化网络模型在CIFAR-10和CIFAR-100两个数据集上的交叉验证表现与对比实验中的结果相近,证明了该模型的可靠性。

3.6 消融实验

为验证本文模型中Log-sum函数和MLR-SNet模块产生的影响,去除对应模块进行消融实验。在各个数据集上的消融实验结果如表5所示,粗体表示在该指标下取得的最优结果。其中Log-sum+MLR-SNet表示本文模型,Log-sum表示在本文模型的基础上去除自动更新的学习率策略,而改为随迭代轮数增加而衰减的学习率,MLR-SNet表示在本文模型的基础上,不使用Log-sum函数作为正则项,Base表示在本文模型的基础上同时去除MLR-SNet模块和不使用Log-sum函数作为正则项。

根据表5的数据对消融实验结果的分析如下。

(1) 与MLR-SNet模型相比,Log-sum+MLR-SNet模型在有相当接近的正确率下,在CIFAR-10数据集上仅有26个百分点的参数使用量,在CIFAR-

表 5 模型在CIFAR-10和CIFAR-100数据集的消融实验结果
Table 5 Results of ablation experiments in CIFAR-10 and CIFAR-100 dataset %

模型	CIFAR-10		CIFAR-100	
	Accuracy	Parameters Used	Accuracy	Parameters Used
Base	90.93	100	76.88	100
Log-sum	84.48	29	65.72	51
MLR-SNet	94.80	100	80.44	100
Log-sum+MLR-SNet	94.75	26	80.38	45

100数据集上仅有45个百分点的参数使用量。结果表明在使用灵活的学习率策略的情况下,使用Log-sum函数作为正则项可以有效降低参数使用量。

(2) 与Log-sum模型相比,Log-sum+MLR-SNet模型在CIFAR-10数据集上的正确率高出10.27个百分点,在CIFAR-100数据集上的正确率高出14.66个百分点。结果表明在使用Log-sum作为正则函数时,使用更灵活的学习率策略可以在保持较低参数使用量的前提下,有效提升正确率。

(3) Base模型与Log-sum+MLR-SNet的对比结果表明,在以Log-sum函数作为正则项的同时使用类似MLR-SNet的灵活的学习率策略,可以在显著提升正确率的同时,降低参数使用量。

3.7 参数敏感性实验

本文2.1节的式(11)中,正则项系数 λ 和Log-sum函数的参数 ε 是两个重要的超参数,由网格搜索获得。因此,本节对 λ 和 ε 的取值对模型性能的影响进行评估。参数敏感性实验结果分别如图2~5所示。

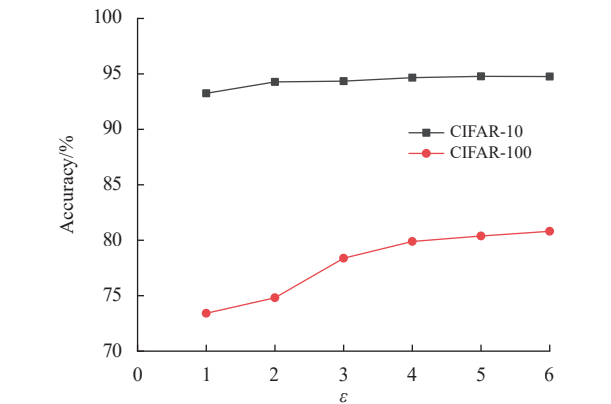


图 2 ε 的取值对模型正确率的影响
Fig.2 Effect of values of ε on model accuracy

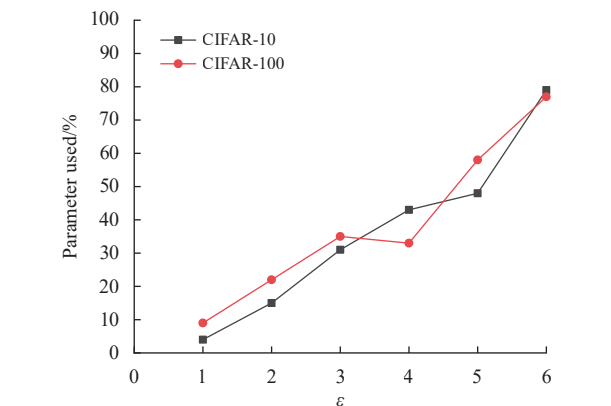


图 3 ε 的取值对模型参数使用量的影响
Fig.3 Effect of values of ε on model parameter used

从图2~5可以看出,控制其他无关变量不变的情

况下, λ 在 10^4 的数量级上、 ε 在 1 的数量级上时, 模型的性能被显著影响。多次实验结果表明, 当 λ 的取值为 10^4 , ε 的取值为 5 时, 模型在 CIFAR-10 数据集上的性能取得最优; 当 λ 的取值为 3×10^4 , ε 的取值为 5 时, 模型在 CIFAR-100 数据集上的性能取得最优。

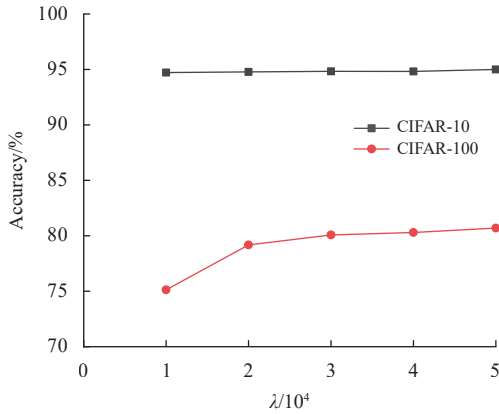


图4 λ 的取值对模型正确率的影响

Fig.4 Effect of values of λ on model accuracy

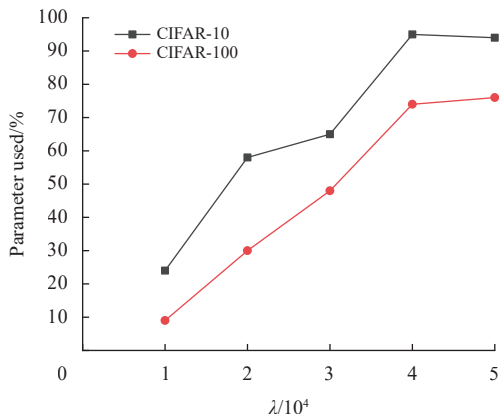


图5 λ 的取值对模型参数使用量的影响

Fig.5 Effect of values of λ on model parameter used

在训练模型时, 时间开销是一个重要考量因素。为了对比本文模型与基准模型的训练开销, 取batch size为128, 对CIFAR-10数据集进行训练, 对比10个epoch的平均训练速度, 结果如图6所示。

在图6中, 使用MLR-SNet求解学习率对模型的训练时间的影响相对较小。Base模型的平均训练速度为27.61轮/s, Log-sum模型的平均训练速度为18.74轮/s, MLR-SNet模型的平均训练速度为27.51轮/s, Log-sum+MLR-SNet模型的平均训练速度为18.70轮/s。

为了对比不同模型的前向运算速度, 分别使用模型进行10 000次前向运算, 统计所用时间, 对比结果如图7所示。

其中Base模型的10 000次前向运算时间为

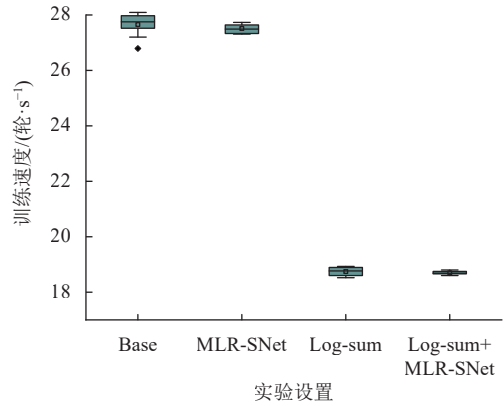


图6 模型在CIFAR数据集的训练速度

Fig.6 Comparison of training speed with different method

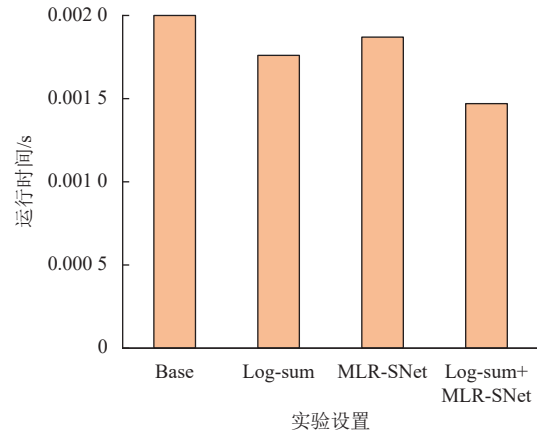


图7 模型在CIFAR数据集的训练速度

Fig.7 Comparison of running time with different method

0.002 00 s, Log-sum模型的10 000次前向运算时间为0.001 76 s, MLR-SNet模型的10 000次前向运算时间为0.001 87 s, Log-sum+MLR-SNet模型的10 000次前向运算时间为0.001 47 s。本文模型比Base模型前向运算时间的减少率为26.5个百分点。

3.8 复杂性分析

本文所使用的ResNet18设置卷积层大小为 7×7 , 步长为2, 卷积层后设置8个最大池化层, 最大池化层后有8个残差块, 简化的时间复杂度估计为

$$O_1 = O(C_{in} \times K^2 \times C_{out} \times W \times H \times L) \quad (22)$$

式中, C_{in} 为输入的通道数, C_{out} 为输出的通道数, K 为卷积核的大小, W 为输入的宽度, H 为输入的高度, L 为总层数。

本文用于求解学习率的MLR-Snet为一个长短期记忆网络, 简化的时间复杂度估计为

$$O_2 = O(T \times (4 \times D^2 + 4 \times D \times I)) \quad (23)$$

式中, T 为序列长度, D 为隐藏层维度, I 为输入维度。

以CIFAR-10数据集为例,由于训练集中有50 000个图像,假设batch size为128,则每个epoch的批次数约为390。记总时间复杂度为 O_{total1} ,则有

$$O_{\text{total1}} = 390 \times 200 \times (O_1 + O_2) \quad (24)$$

记ResNet18的空间复杂度估计为 O_3 ,则有

$$O_3 = O\left(\sum_{l=1}^L (K_l^2 \times C_{\text{in},l} \times C_{\text{out},l} + C_{\text{out},l})\right) \quad (25)$$

式中, K_l 为第 l 层卷积核的大小, $C_{\text{in},l}$ 为第 l 层的输入通道数, $C_{\text{out},l}$ 为第 l 层的输出通道数。

与式(23)的记号相同,MLR-SNet的空间复杂度估计为 O_4 ,则

$$O_4 = O(T \times D) \quad (26)$$

记模型总空间复杂度为 O_{total2} ,则有

$$O_{\text{total2}} = O_3 + O_4 \quad (27)$$

4 结语

本文借助Log-sum函数的邻近算子的刻画,使用邻近梯度法对Log-sum函数作为正则化项的神经网络模型进行求解,其学习率用Meta-LR-Schedule-Net训练得到。数值实验结果表明,训练得到神经网络具有较好的稀疏性,较高的正确率。

参考文献:

- [1] FUKUSHIMA K, MIYAKE S, ITO T. Neocognitron: a neural network model for a mechanism of visual pattern recognition[J]. IEEE Trans on System, Man, and Cybernetics, 1983, 13(5): 826-834.
- [2] COLLOBERT R, WESTON J. A unified architecture for natural language processing: deep neural networks with multitask learning[C]//Proceedings of the 25th International Conference on Machine Learning, 2008 (ICML 2008). Helsinki, Finland: ACM, 2008: 160-167.
- [3] ALVAREZ J M, SALZMANN M. Learning the number of neurons in deep networks[C]//Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2016: 29.
- [4] SONG H, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural networks[C]//Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2015: 28.
- [5] SRIVASTAVA N, HINTON G, KRIZHEVSKY A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014, 15(1): 1929-1958.
- [6] MA R R, MIAO J Y, NIU L F, et al. Transformed ℓ_1 regularization for learning sparse deep neural networks[J]. Neural Networks, 2019, 119: 286-298.
- [7] QIAO C, SHI Y, DIAO Y X, et al. Log-sum enhanced sparse deep neural network[J]. Neurocomputing, 2020, 407: 206-220.
- [8] CHEN C H, MANGASARIAN O L. Hybrid misclassification minimization[J]. Advances in Computational Mathematics, 1996, 5(1): 127-136.
- [9] AGHASI A, ABDI A, NGUYEN N, et al. Net-Trim: convex pruning of deep neural networks with performance guarantee[C]//Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2017: 30.
- [10] ZHANG J S, JI N N, LIU J M, et al. Enhancing performance of the backpropagation algorithm via sparse response regularization[J]. Neurocomputing, 2015(153): 20-40.
- [11] FOUCART S, LAI M J. Sparsest solutions of underdetermined linear systems via ℓ_q -minimization for $0 < q \leq 1$ [J]. Applied and Computational Harmonic Analysis, 2009, 26(3): 395-407.
- [12] WANG H, YANG X Y, SHI Y M, et al. A proximal iteratively reweighted approach for efficient network sparsification[J]. IEEE Transactions on Computers, 2020(99): 185-196.
- [13] PRATER-BENNETTE A, SHEN L X, TRIPP E E. The proximity operator of the log-sum penalty[J]. Journal of Scientific Computing, 2022, 93(3): 67.
- [14] SHU J, ZHU Y W, ZHAO Q, et al. MLR-SNet: transferable LR schedules for heterogeneous tasks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2022, 45(3): 3505-3521.
- [15] TSENG P. An incremental gradient(-projection) method with momentum term and adaptive stepsize rule[J]. SIAM Journal on Optimization, 1998, 8(2): 506-531.
- [16] KINGMA D, BA J. Adam: a method for stochastic optimization[C]//International Conference on Learning Representations (ICLR). Washington DC: ACM, 2015: 7-9.
- [17] WARD L B. Reminiscence and rote learning[M]//Psychological Monographs. London: AP Association, 1937: 1-64.
- [18] ANDRYCHOWICZ M, DENIL M, GOMEZ S, et al. Learning to learn by gradient descent by gradient descent[C]//Advances in Neural Information Processing Systems. Cambridge: MIT Press, 2016: 29.
- [19] CHEN Y T, HOFFMAN M W, GÓMEZ C S, et al. Learning to learn without gradient descent by gradient descent[C]//International Conference on Machine Learning. Madison, Wis: International Machine Learning Society, 2017: 748-756.
- [20] BOTTOU L. Stochastic gradient learning in neural networks[C]//Proceedings of Neuro-Nimes. Oxford: Pergamon Press, 1991: 12.
- [21] BIAN W, CHEN X J. A smoothing proximal gradient algorithm for nonsmooth convex regression with cardinality penalty[J]. SIAM Journal on Numerical Analysis, 2020, 58(1): 858-883.
- [22] TANG A D, NIU L F, MIAO J Y, et al. Training compact DNNs with $\ell_{1/2}$ regularization[J]. Pattern Recognition, 2023(136): 109206.

(责任编辑: 杨耀辉 英文审核: 熊荣斌)