文章编号:1007-5321(2021)06-0083-06 DOI:10.13190/j.jbupt.2021-068

多策略 MRFO 算法的卷积神经网络超参数优化

刘永利, 朱亚孟, 晁 浩
(河南理工大学 计算机科学与技术学院, 焦作 454000)

摘要: 卷积神经网络的性能与超参数配置密切相关,然而最优超参数的选择耗时耗力. 为了提高超参数选择的效率,提出了一种基于多策略的蝠鲼觅食优化算法,一方面采用半数均匀初始化策略提升种群的多样性;另一方面,融合新权重因子更新策略和分裂策略,提升收敛速度和拟合精度. 根据实数编码策略将所提算法用于卷积神经网络的超参数优化研究中,用 3 种觅食方式进行迭代,以得到最优的超参数配置. 为了评估超参数优化的有效性,与卷积神经网络超参数优化算法在手写数字和 CIFAR-10 数据集上进行了对比实验,实验结果表明,所提算法可消耗较少的资源,并获得更高的准确率.

关 键 词: 卷积神经网络; 蝠鲼觅食优化算法; 超参数优化
中图分类号: TP389.1 **文献标志码**: A

Hyperparameter Optimization of Convolutional Neural Network Based on Multi-Strategy MRFO Algorithm

LIU Yong-li, ZHU Ya-meng, CHAO Hao
(College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China)

Abstract: The performance of convolutional neural network is closely related to the configuration of hyperparameters. However, the selection of optimal hyperparameters is time-consuming and labor-consuming. In order to improve the efficiency of hyperparameter selection, a multi-strategy manta ray foraging optimization algorithm is proposed. On the one hand, half uniform initialization strategy is adopted to improve population diversity. On the other hand, it combines new weight factor update strategy and splitting strategy to improve the convergence speed and fitting accuracy respectively. According to the real coding strategy, this algorithm is applied to the research of convolutional neural network hyperparameter optimization, which can be iterated according to three foraging methods to obtain the optimal hyperparameter configuration. In order to evaluate the effectiveness of hyperparameter optimization, the proposed algorithm is compared with the mainstream convolutional neural network hyperparameter optimization algorithms on mixed national institute of standards and technology and CIFAR-10 datasets. Experimental results show that the proposed algorithm achieves higher accuracy with less resources.

Key words: convolutional neural network; manta ray foraging optimization; hyperparameter optimization

卷积神经网络(CNN, convolutional neural network)的性能与超参数配置密切相关,然而最优超参数常常靠人工配置,耗费了大量的时间和计算资源. 因此,自动优化超参数配置成为研究热点之一. 超参数配置问题属于非线性和多模态的复杂数学问题[1]. 目前,元启发优化算法是解决超参数配置问题的最佳

收稿日期: 2021-04-20
基金项目: 国家自然科学基金项目(61872126); 河南省高等学校重点科研项目(19A520004)
作者简介: 刘永利(1980—), 男, 教授, 硕士生导师, E-mail: yongli.buaa@gmail.com.

选择. 然而大多数元启发优化算法仅优化网络的第1层参数而忽略其余层对性能的影响,且若优化算法的寻优强度不足还会导致网络陷入局部最优.

蝠鲼觅食优化(MRFO, manta ray foraging optimization)算法[2]是一种模仿蝠鲼觅食行为的仿生优化算法,通过模拟蝠鲼的3种觅食习惯平衡全局搜索和局部开发,保证问题优化的高准确性. 鉴于MRFO算法具有较优的寻优强度,笔者首先提出多策略蝠鲼觅食优化(MSMRFO, multi-strategy manta ray foraging optimization)算法,采用半数均匀初始化策略初始化个体,在寻优过程中采用新的指数权重系数,并使用分裂算子改变个体位置以提高收敛精度;然后根据实数编码策略将MSMRFO算法与CNN融合,提出了多策略蝠鲼算法的CNN超参数优化(MSMRFO-CNN, multi-strategy manta ray foraging optimization-CNN)算法,根据3种觅食方式进行迭代,得到最优的CNN超参数配置.

1 MRFO算法

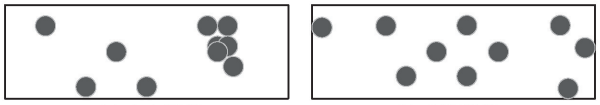
蝠鲼觅食优化算法模拟了蝠鲼的3种觅食方式,主要包括:旋风觅食、链式觅食和翻筋斗觅食. 可根据觅食方式的不同对搜索空间进行勘探和开发. 首先,在旋风觅食阶段,蝠鲼排成一个长串进行螺旋式觅食,后面的蝠鲼游向前一条,仅沿着螺旋形的路径移动,此过程有助于勘探;其次,蝠鲼认为目前找到的最优位置就是浮游生物浓度最高的位置,蝠鲼会首尾相连排成一列,形成链式觅食链,此过程有助于开发;最后,在翻筋斗觅食阶段使每个个体都能移动到新的搜索域中的任何位置,该位置位于当前位置和当前最优位置周围的对称位置之间. 随着迭代次数的增加,翻筋斗觅食的范围会自适应地减小,此过程有助于勘探.

2 MSMRFO算法

笔者从3方面对MRFO算法做了改进:1)在种群初始化方面,提出了半数均匀初始化策略,以提高种群多样性;2)提出新的指数变化权重因子,使其更好地协调全局搜索和局部开发;3)分裂算子的引入在提高种群多样性的同时进一步提升了算法收敛精度.

2.1 半数均匀初始化

在个体初始化阶段,为避免随机生成的个体过于密集,提出了一种半数均匀初始化策略. 种群中的前一半个体采用与MRFO算法相同的随机初始化方式,后一半个体在均数区间内进行初始化. 设计原理是将搜索空间人为划分为$N/2$个新的搜索子空间,然后将对应个体的初始解在对应搜索子空间内进行随机初始化. 这种初始化方式不仅保留了种群的随机性,而且可以避免初始化个体集中分布,有利于提高种群的多样性和搜索效率,并且可在一定程度上避免出现局部极值. 图1示出了同一组数据在完全随机和半数随机2种情形下的初始化结果,可见,半数随机的策略可获得更均匀的分布.



(a) 完全随机 (b) 半数随机

图1 初始化个体分布

半数初始化策略的数学模型为

$$\left.\begin{aligned} x_i(t) &= \mathrm{rand}(\boldsymbol{x}_1, \boldsymbol{x}_u) \\ x_j(t) &= \mathrm{rand}\left(x_1 + \frac{(\boldsymbol{x}_u - \boldsymbol{x}_1)}{N/2}(j-1),\right. \\ &\quad \left. x_1 + \frac{(\boldsymbol{x}_u - \boldsymbol{x}_1)}{N/2} j\right) \end{aligned}\right\} \tag{1}$$

其中:$\boldsymbol{x}_u$和$\boldsymbol{x}_1$分别代表搜索空间的上、下限区间向量,rand( )表示在限定空间内生成向量的随机函数,$i=1,2,\cdots,N/2$, $j=(N/2+1,\cdots,N)$, $\boldsymbol{x}_i(t)$, $\boldsymbol{x}_j(t)$代表第t次迭代的种群内个体初始值,N为种群大小.

2.2 指数权重系数

在链式觅食阶段,MRFO算法在迭代寻优过程中的权重系数是在固定范围内随机生成的,故算法的收敛速度因前后期的搜索步长相同而变慢. 对此,提出了一种根据迭代次数指数变化的权重系数,有

$$r = 1 - \exp\left[-\left(4\frac{T-t}{t}\right)^2\right] \tag{2}$$

其中:T为最大迭代次数,t为当前迭代次数. 算法初期r值偏大,提供了较大的步长,加速了种群个体往最优解方向逼近;算法后期r值变小,可增强局部搜索能力,保证算法的寻优性能.

2.3 分裂算子

分裂算子包括分裂条件选取、分裂个体步长阈值设置和分裂策略的定义,实行的方案包括3个方面.

1) 分裂条件. 在MRFO算法中,翻筋斗觅食阶段前期的位置更新只对适应度优于当前最优适应度的个体进行位置更新,并未改变适应度差的个体改变. 因此,在翻筋斗觅食过程中有可能出现算法空

翻的搜索范围处于固定区间，使算法陷入局部最优. 新的分裂条件为

$$f\left[x_i(t+1)\right] > f(x_{\mathrm{b}}) \tag{3}$$

其中：$f\left[x_i(t+1)\right]$为经过链式或气旋觅食更新后位置的适应度，$f(x_{\mathrm{b}})$为当前最优位置适应度.

2）分裂阈值. 将所有个体的适应度进行排序，只有符合分裂条件的个体可进行分裂，分裂阈值为

$$D=\begin{cases} P_{i,j}\mathrm{rand}\left(-\dfrac{1}{2},\dfrac{1}{2}\right), & P_{i,\mathrm{b}} \geqslant C \\ P_{i,j}+\mathrm{rand}\left(-\dfrac{P_{i,j}}{2},\dfrac{P_{i,j}}{2}\right), & \text{其他} \end{cases} \tag{4}$$

其中：$P_{i,j}$为个体 i 和 j 之间的距离，$P_{i,\mathrm{b}}$为当前个体和最优个体之间的距离，$C=\mathrm{e}^{-t/T}$为根据迭代次数指数变化的阈值，rand(-1/2,1/2)和 rand($-P_{i,j}$/2,$P_{i,j}$/2)是在[-1/2,1/2]和[$-P_{i,j}$/2,$P_{i,j}$/2]上均匀分布的随机数.

采用适应度的差值代表个体之间的距离. 算法前期 $P_{i,\mathrm{b}}$较大，随迭代次数 t 的增加，个体接近当前最优值后 $P_{i,\mathrm{b}}$减小，当 $P_{i,\mathrm{b}} \geqslant C$ 时，$D \in [-P_{i,j}/2, P_{i,j}/2]$有助于个体进行全局搜索；反之 $D \in [P_{i,j}/2, 3P_{i,j}/2]$有助于局部搜索.

3）分裂策略. 为了提高种群的多样性和算法的寻优精度，通过下面分裂策略对符合分裂的个体进行分裂：

$$x_i(t)=\frac{t}{T}x_i(t)+\left(1-\frac{t}{T}\right)x_i(t)D \tag{5}$$

寻优前期的个体位置主要由后一部分的分裂项决定，增加了种群的多样性，寻优后期的个体位置主要由个体自身位置决定.

3　MSMRFO 算法优化卷积神经网络超参数

3.1　卷积神经网络

为适应不同的分类任务，LeNet，AlexNet，VGG-Net，GoogLeNet 等许多性能较优的神经网络架构相继被提出. 其中经典的卷积神经网络结构 LeNet 包括输入层、卷积层、下采样层、全连接层和输出层.

卷积层主要用来提取特征，通过过滤器以特征图的形式保存有用的特征. 过滤器的大小和个数是影响网络架构性能的重要参数. 池化层主要用于降维，减小计算量. 常用的池化层类型有均值池化和最大值池化. 池化层的类型对于 CNN 也是一个重要的参数. 全连接层中神经元节点的个数对网络的性能有很大影响. 定义这些参数需要大量的先验知识，在调整超参数过程中也需要消耗大量的计算资源.

3.2　MSMRFO 算法优化神经网络

3.2.1　编码策略

目前常用的编码机制包括基于网络 IP 地址的编码机制[3]和基于块的编码机制[4]. 编码机制解决了元启发优化算法和神经网络结合的难点. 但编码策略并不适用于所有的启发算法，多数仅适用于粒子群优化算法和遗传优化算法. 得到的最优粒子结构还需要经过解码操作得到网络参数.

研究围绕经典的 LeNet 网络展开，其网络深度固定，所以提出了整数编码机制来实现 MSMRFO 算法和卷积的结合. 整数编码机制将需要优化的超参数定义成每个个体的分量，然后在 MSMRFO-CNN 算法中进行优化.

3.2.2　算法步骤

1）设置 MSMRFO-CNN 算法的参数，包括种群大小、最大迭代次数、维度(参数的维度由参数的优化范围决定).

2）初始化个体，将待优化参数以实数的形式编码成 MSMRFO-CNN 算法中的个体，每个个体代表一种 CNN 结构.

3）计算每个个体的适应度，这里将训练的准确率作为适应度函数值.

4）根据 MSMRFO-CNN 算法的 3 种觅食方式进行个体的更新. 在更新过程中会根据迭代次数的增加自适应地修改所提的指数权重因子，以保证算法初期个体更新时最优个体的位置有较高的权重，进而加快个体向最优超参数靠拢；算法后期权重变小，可增强个体在最优超参数附近的局部搜索能力.

5）判断是否满足迭代终止条件，满足输出最优个体即为最优 CNN 结构；否则继续执行步骤 3).

4　仿真实验

为证明改进算法的有效性，进行基准函数测试和优化神经网络超参数的测试.

4.1　基准函数测试

为了评估 MSMRFO 算法的表现，引入鲸鱼优化算法(WOA，whale optimization algorithm)[5]、A-C 双参数鲸鱼优化算法(ACWOA，A-C parametric whale optimization algorithm)[6]、蝴蝶优化算法(BOA，butterfly optimization algorithm)[7]、哈里斯鹰优化(HHO，Harris hawks optimization)算法[8]进行了对

比实验. 对比算法均是基于种群的优化算法,具有较强的代表性和可比性. 实验中选取7个基准测试函数进行测试,函数信息如表1所示,其中$f_1 \sim f_3$为单峰函数,求解简单,主要用于对比收敛速度;$f_4 \sim f_7$为多峰函数,函数求解复杂,易陷入局部最优,主要用于评估收敛精度.

1) 优化性能分析

为了保证测试的准确性和公平性,对比算法和MSMRFO算法除了改进的初始化方式、权重系数和加入的分裂算子处不相同外,实验环境和参数全部保持一致. 实验中设置$T=500$、$N=30$、$d=30$. 每个基准测试函数独立运行30次求其平均值.

在7个基准测试函数上的对比实验结果如表1所示. 可以看出,对比算法在处理函数$f_1 \sim f_3$时均未达到理论最优值;MRFO算法的方差虽小,但f_2和f_3未达到理论最优值;MSMRFO算法在平均值和稳定性上均达到了最佳性能. 处理多峰函数$f_4 \sim f_7$时,MSMRFO算法达到的最优值均小于其他算法,方差最小且性能最稳定. 综上可知,MSMRFO算法在稳定性和寻优性能上均占优势.

表1 基准测试函数信息及结果

函数	最优值	指标	MSMRFO	MRFO	WOA	ACWOA	BOA	HHO
f_1(Sphere)	0	mean	0	0	1.60×10^{-73}	2.58×10^{-263}	1.32×10^{-11}	1.04×10^{-94}
		std	0	0	6.19×10^{-73}	0	1.08×10^{-12}	5.72×10^{-94}
f_2(Schwefel 2.22)	0	mean	0	6.60×10^{-205}	2.40×10^{-51}	5.42×10^{-138}	4.68×10^{-9}	1.01×10^{-49}
		std	0	0	8.80×10^{-51}	2.26×10^{-137}	1.40×10^{-9}	5.26×10^{-49}
f_3(Schwefel 2.21)	0	mean	0	5.60×10^{-202}	5.15×10^{1}	4.27×10^{-128}	6.03×10^{-9}	2.83×10^{-49}
		std	0	0	2.78×10^{1}	1.66×10^{-127}	4.54×10^{-10}	9.74×10^{-49}
f_4(Rosenbrock)	0	mean	9.60×10^{-1}	2.26×10^{1}	2.78×10^{1}	2.87×10^{1}	8.89×10^{1}	1.44×10^{-2}
		std	4.75×10^{-1}	5.2	4.30×10^{-1}	1.13×10^{-2}	2.58×10^{-2}	2.06×10^{-2}
f_5(Quartic)	0	mean	8.10×10^{-5}	1.40×10^{-4}	3.70×10^{-3}	8.61×10^{-5}	1.60×10^{-3}	1.13×10^{-4}
		std	1.10×10^{-4}	1.20×10^{-4}	3.50×10^{-3}	6.55×10^{-5}	5.24×10^{-4}	1.05×10^{-4}
f_6(Schwefel)	−12 569.5	mean	−12 569.3	−8 205.3	−9 979.5	−12 418.9	−3 724.6	−12 568.7
		std	9.94×10^{-1}	7.82×10^{2}	1.60×10^{3}	4.69×10^{2}	2.62×10^{2}	1.47
f_7(Penalized2)	0	mean	6.60×10^{-5}	2.71	6.15×10^{-1}	2.72×10^{-1}	2.88	2.04×10^{-4}
		std	1.40×10^{-4}	7.91×10^{-1}	2.90×10^{-1}	7.92×10^{-2}	2.14×10^{-1}	2.31×10^{-4}

2) 收敛性能分析

为了更直观地看出MSMRFO算法的收敛速度,给出30维情况下3个算法在单峰和多峰基准测试函数下的收敛曲线,如图2所示. 对于单峰函数f_1,MSMRFO算法在收敛速度上均优于其他算法,在多峰函数f_5和f_7上MSMRFO算法的收敛速度优于其他5种对比算法,证明了所提优化策略的有效性.

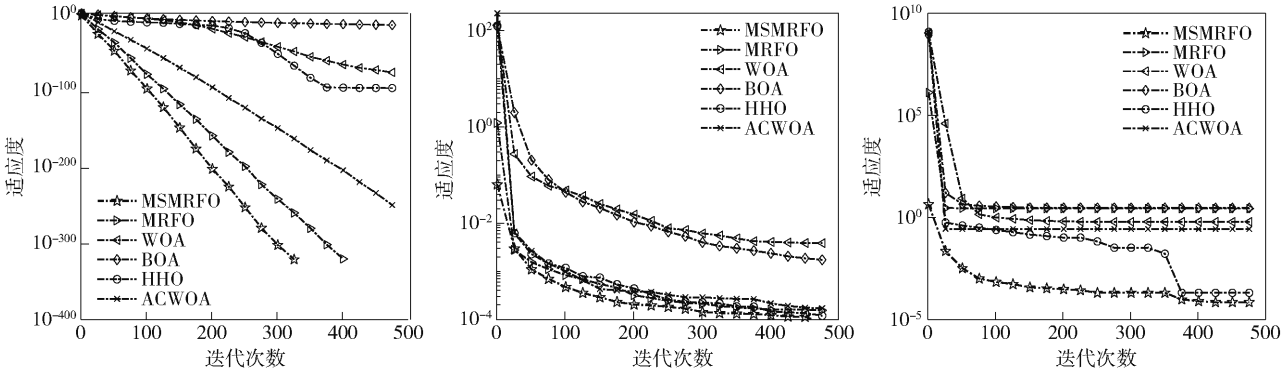
4.2 优化神经网络超参数

笔者采用的MNIST(mixed national institute of standards and technology)数据集和CIFAR-10数据集是图像识别领域中的2个基准数据集. 选取标准CNN、基于遗传算法的CNN超参数优化(GA-CNN, genetic algorithm-CNN)算法[9]、基于性粒子群算法的CNN超参数优化(LDWPSO-CNN, linearly decreasing weight particle swarm optimization-CNN)算法[10]、基于指数衰减分裂算子粒子群算法的CNN超参数优化(EDWPSO-CNN, exponential decay weight particle swarm optimization-CNN)算法和基于蝠鲼优化算法的CNN超参数优化(MRFO-CNN, manta ray foraging optimization-CNN)算法进行对比实验. 优化网络的超参数优化范围如表2所示. 为保证实验结果的准确性和公平性并减少随机性影响,实验在同一电脑上运行10次求其平均值.

4.2.1 MNIST数据集实验结果

MNIST手写数据集由10类单通道灰度图像组成,其中6万张图片用于训练,1万张图片用于测试. 通过实验主要验证经所提算法优化得到的超参数对网络性能的影响,因此无需对图像进行任何特殊化处理. 在实验中,使用优化算法对每5个历元进行参数优化,并且基于优化后的参数进行学习.

标准CNN参数和通过MSMRFO-CNN算法优化



(a) f_1测试函数收敛曲线　(b) f_5测试函数收敛曲线　(c) f_7测试函数收敛曲线

图 2　30 维测试函数寻优的收敛曲线

表 2　参数优化范围及 MSMRFO-CNN 算法的优化结果

分量坐标	超参数	参数优化范围	标准 CNN 参数	MSMRFO-CNN 算法优化后参数	
				MNIST 数据集	CIFAR-10 数据集
x_1	优化器	[Adam, SGD]	SGD	SGD	SGD
x_2	训练批量大小	[10, 101]	10	11	16
x_3	第 1 层卷积核个数	[4, 101]	6	50	98
x_4	第 1 层卷积核大小	[3×3, 5×5, 7×7]	5×5	3×3	5×5
x_5	第 1 层激活函数类型	[Sigmoid, Relu, Tanh]	Sigmoid	Sigmoid	Relu
x_6	第 2 层池化类型	[max-pooling, Average-pooling]	Average-pooling	Average-pooling	Average-pooling
x_7	第 3 层卷积核个数	[4, 101]	16	99	65
x_8	第 3 层卷积核大小	[3×3, 5×5, 7×7]	5×5	3×3	5×5
x_9	第 3 层激活函数类型	[Sigmoid, Relu, Tanh]	Sigmoid	Tanh	Relu
x_{10}	第 4 层池化类型	[max-pooling, Average-pooling]	Average-pooling	Average-pooling	Average-pooling
x_{11}	第 5 层神经元个数	[4, 201]	120	200	60
x_{12}	第 5 层激活函数类型	[Sigmoid, Relu, Tanh]	Sigmoid	Relu	Relu
x_{13}	第 6 层神经元个数	[4, 201]	84	40	115
x_{14}	第 6 层激活函数类型	[Sigmoid, Relu, Tanh]	Sigmoid	Relu	Relu

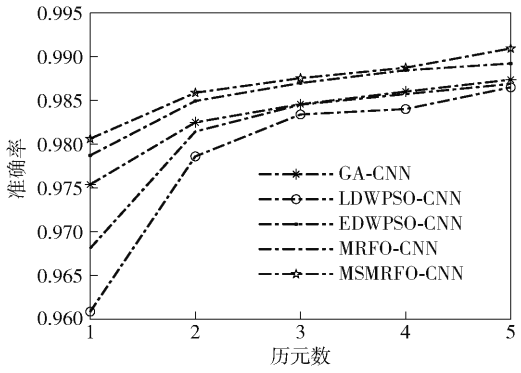
之后的参数如表 2 所示. 在卷积层中 MSMRFO-CNN 算法优化得到的卷积核大小为 3×3,用更小的卷积核可以学习到更多的特征信息. 卷积层激活函数为 Sigmoid 和 Tanh 组合的方式解决单种类激活函数引起的梯度消失问题. 全连接层神经元个数是经过优化之后用来防止发生过拟合制定的最优个数. 训练批次大小是 MSMRFO-CNN 算法自适应实验环境计算能力所优化出效率最高的参数.

图 3 所示为每个训练历元的准确率和训练损失. 可见,MSMRFO-CNN 训练算法在测试历元每个阶段的训练损失均低于其他算法,准确率高于其他算法,表明 MSMRFO-CNN 算法的性能优于其他算法. 从图 3 还可以看到,经过 MSMRFO-CNN 算法优化得到的神经网络在测试历元第 1 阶段的准确率高于其余对比算法得到的神经网络准确率. 这表明 MSMRFO-CNN 算法优化得到的神经网络在第 1 次测试时已经具备高于其余算法网络的分类能力. 此结果更能体现出 MSMRFO-CNN 算法的寻优能力强于其他算法.

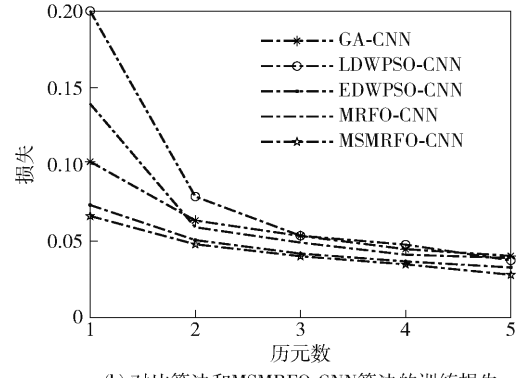
4.2.2　CIFAR-10 数据集实验结果

CIFAR-10 数据集由 10 类 3 通道彩色图片组成,其中 6 万张图片用于训练、1 万张图片用于测试. 在实验中,每 10 个历元使用优化算法进行参数优化,并且基于优化后的参数进行学习.

测试集上 MSMRFO-CNN 算法和其他对比算法的准确率和训练损失如图 4 所示. 学习 10 个历元之后 MSMRFO-CNN 算法的准确率最高为 70.5%,远高于其他对比算法. 另外,算法学习到的参数在

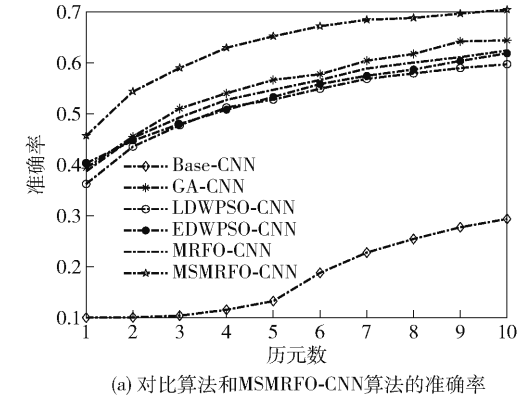


(a) 对比算法和MSMRFO-CNN算法的准确率

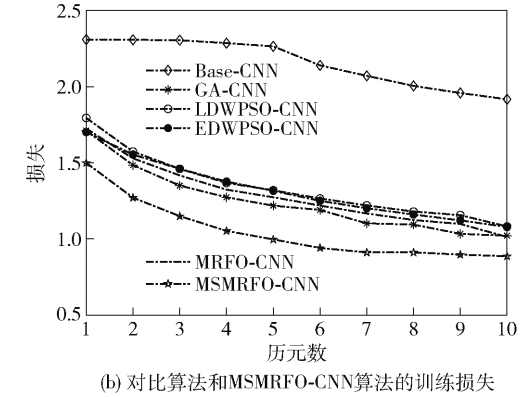


(b) 对比算法和MSMRFO-CNN算法的训练损失

图3 MNIST数据集的实验结果



(a) 对比算法和MSMRFO-CNN算法的准确率



(b) 对比算法和MSMRFO-CNN算法的训练损失

图4 CIFAR-10数据集的实验结果

第1个历元有较低的训练损失. 结果表明,MSMRFO-CNN算法面对较为复杂的数据集也可以得到一个准确率较高且损失较小的CNN模型,这进一步证明了MSMRFO-CNN算法的有效性.

通过对上述2个标准数据集的实验结果进行分析可知,MSMRFO-CNN算法在优化卷积神经网络超参数方面效果明显,面对不同复杂程度的数据集均能得到优于其他对比算法的神经网络. 这表明MSMRFO-CNN算法具有普遍适用的特点,在损耗资源相同的情况下,具有更好的实用性和工程应用前景.

5 结束语

针对CNN的超参数优化问题,提出了一种用多策略蝠鲼优化算法优化CNN超参数的方法. 通过提出的初始化策略、权重因子更新策略和个体分裂策略可提高MRFO算法的收敛速度和拟合精度;然后通过实数编码策略将算法应用于CNN超参数优化中,并通过不同数据集验证了MSMRFO-CNN算法的有效性. 然而只针对固定深度的网络进行的优化研究具有一定的局限性,进一步研究优化深度不定情况下网络中的超参数是后续工作的重点.

参考文献:

[1] Wang Yulong, Zhang Haoxin, Zhang Guangwei. CPSO-CNN: an efficient PSO-based algorithm for fine-tuning hyper-parameters of convolutional neural networks[J]. Swarm and Evolutionary Computation, 2019, 49: 114-123.

[2] Zhao Weiguo, Zhang Zhenxing, Wang Liying. Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications[J]. Engineering Applications of Artificial Intelligence, 2020, 87: 103300.

[3] Wang Bin, Sun Yanan, Xue Bing, et al. Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification[C]// 2018 IEEE Congress on Evolutionary Computation (CEC). Rio de Janeiro: IEEE Press, 2018: 1-8.

[4] Sun Yanan, Xue Bing, Zhang Mengjie, et al. Completely automated CNN architecture design based on blocks[J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 31(4): 1242-1254.

(下转第95页)

a software defined decentralized mobile network architecture toward 5G[J]. IEEE Network, 2015, 29(2): 16-22.

[9] 3GPP. 3GPP TS 23. 501—2020, System architecturefor the 5G system (release 16) [S]. Sophia Antipolis: 3rd Generation Partnership Project, 2020: 29-404.

[10] 3GPP. 3GPP TS 29. 244—2020, Interface between the control plane and the user plane nodes (release 16) [S]. Sophia Antipolis: 3rd Generation Partnership Project, 2020: 20-294.

[11] 王立文, 王友祥, 唐雄燕, 等. 5G 核心网 UPF 硬件加速技术[J]. 移动通信, 2020, 44(1): 19-23, 32.

Wang Liwen, Wang Youxiang, Tang Xiongyan, et al. Research on UPF hardware acceleration technology in 5G core networks[J]. Mobile Communications, 2020, 44(1): 19-23, 32.

[12] Hong Cheol-Ho, Lee Kyungwoon, Hwang Jaehyun, et al. Kafe: can OS kernels forward packets fast enough for software routers? [J]. IEEE/ACM Transactions on Networking, 2018, 26(6): 2734-2747.

[13] Cerović D, Del Piccolo V, Amamou A, et al. Fast packet processing: a survey[J]. IEEE Communications Surveys & Tutorials, 2018, 20(4): 3645-3676.

(上接第 88 页)

[5] Mirjalili S, Lewis A. The whale optimization algorithm [J]. Advances in Engineering Software, 2016, 95: 51-67.

[6] Elhosseini M A, Haikal A Y, Badawy M, et al. Biped robot stability based on an *A-C* parametric whale optimization algorithm [J]. Journal of Computational Science, 2019, 31: 17-32.

[7] Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization[J]. Soft Computing, 2019, 23(3): 715-734.

[8] Heidari A A, Mirjalili S, Faris H, et al. Harris hawks optimization: algorithm and applications[J]. Future Generation Computer Systems, 2019, 97: 849-872.

[9] Liu Peng, El Basha M D, Li Yangjunyi, et al. Deep evolutionary networks with expedited genetic algorithms for medical image denoising[J]. Medical Image Analysis, 2019, 54: 306-315.

[10] Serizawa T, Fujita H. Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization [EB/OL]. (2020-01-16) [2021-01-21]. https://arxiv. org/abs/2001. 05670.