DOI: 10.13382/j.jemi.B2205311

基于动态衰减网络和算法的图像识别

费春国 刘启轩
(中国民航大学电子信息与自动化学院 天津 300300)

摘 要:针对在大样本数据集下,梯度下降法长期性存在着容易收敛到局部最优和收敛速度慢等问题,通过改变网络结构和梯度下降过程,提出了一种动态衰减网络和动态衰减梯度下降算法。在现有网络的基础上,层与层的每个神经元之间增加一条衰减权重,同时在梯度下降过程中引入了衰减权重项。衰减权重值随着迭代不断衰减,最终趋于 0。由于衰减权重项的增加,可以在梯度下降的前期加快梯度下降速度和收敛速度,同时可以避免越过最优解和在最优解附近振荡,提高了网络获得最优解的概率。通过 MNIST 、CIFAR-10 和 CIFAR-100 数据集的实验结果证实,所提出的动态衰减网络和算法,相比原始网络使用 Adam 和动量随机梯度下降法,测试准确度分别提高了 0.2%~1.89%和 0.75%~2.34%,同时具有更快的收敛速度。

关键词: 深度学习;反向传播算法;局部最优;优化算法
中图分类号: TP183; TN06 **文献标识码**: A **国家标准学科分类代码**: 520.20

Image recognition based on dynamic attenuation network and algorithm

Fei Chunguo Liu Qixuan
(College of Electronic Information and Automation, Civil Aviation University of China, Tianjin 300300, China)

Abstract: To address the problems that the gradient descent method is easy to converge to the local optimum and the convergence speed is slow under large sample data sets, a dynamic attenuation network and a dynamic attenuation gradient descent algorithm are proposed by changing the network structure and gradient descent process in the paper. On the basis of the existing network, an attenuation weight is added between each neuron of each two layers, while an attenuation weight term is introduced in the gradient descent process. The attenuation weight value decreases continuously with iteration, and eventually converges to 0. Due to the addition of the attenuation weight term, the gradient descent speed and convergence speed can be accelerated in the early stage of gradient descent. At the same time, it can avoid crossing over the optimal solution and oscillating around the optimal solution. At the last, it can also improve the probability of the network to obtain the optimal solution. The experimental results on MNIST, CIFAR-10 and CIFAR-100 datasets show that the proposed dynamic attenuation network and dynamic attenuation gradient descent algorithm, compared with the original network that used Adam optimizer and stochastic gradient descent with momentum, improve the test accuracy by 0.2%~1.89% and 0.75%~2.34%, respectively, while having a faster convergence speed.

Keywords: deep learning; back propagation; local optimal; optimization algorithms

0 引 言

深度学习作为人工智能领域的一个重要分支,神经网络是其典型代表。基于深层复杂的神经网络被用于解决各种复杂问题[1],例如智能故障诊断[2]、目标跟踪[3]、大数据的分类预测[4]等。然而随着网络层数的增加,各种问题也接踵而来。

梯度下降法是目前训练深层神经网络的核心算法,通常结合反向传播(back propagation, BP)算法来训练神经网络,其通过梯度更新网络参数,使网络的输出无限逼近真实值。但是其长期性存在的问题是梯度消失,陷入局部最优,导致网络无法达到最优解状态[5]。

跳出局部最优一直是相关科学家研究的热点。为了解决这一问题,1951 年 Robbins 等[6]提出了随机逼近,后面衍生出了随机梯度下降算法(stochastic gradient descent, SGD)。SGD 算法在样本中随机挑选一个样本,通过计算一个样本的梯度来替代全部样本的梯度,再根

收稿日期: 2022-03-25 Received Date: 2022-03-25

据步长来更新网络参数。SGD 算法增加了下降的随机性,增加了网络到达最优解的概率,同时不再需要计算全部样本的梯度,减小了运行所需要的内存。SGD 算法相比于传统的梯度下降法,其拥有更高的准确率和更快的收敛速度。目前,SGD 算法也已经成为求解网络最优解问题的主流方法[7]。

然而 SGD 算法存在的问题是由于其每次参数更新依据的是一个样本的梯度,当被选择的两个样本的梯度前后差别较大时,会导致两次更新的方向出现较大的变化,使网络收敛到全局最优解的速度变慢[8]。为了避免出现参数更新震荡,一些基于 SGD 算法的改进方法被提出[9]。1999 年,Ning 等[10]在传统的 SGD 算法基础上增加了动量项,提出了带动量的随机梯度下降算法(stochastic gradient descent with momentum, SGDM)。它可以避免参数更新震荡的问题,加快了网络获得最优解的速度。但弊端是有时会使网络跳过最优解,无法保证收敛到全局最优。另一种类型的改进算法是基于梯度的二次矩估计来自适应的调整学习率,主要包括 Adagrad[11]、Adadelta[12]、Adam[13]等。这类算法在网络的训练过程中根据历史梯度信息,针对参数的不同分量自动调整学习率。目前,Adam 算法是训练神经网络中最常用的方法,具有更高的准确度,更快的收敛速度,但大量的实验证明,某些情况下,Adam 算法仅在训练集下具有较好的效果,在测试集的表现往往不如 SGDM[14]。

针对上述优化算法均从梯度下降更新的角度出发,且均存在一定的问题,本文从改变网络结构来改变梯度下降的角度出发,提出了一种动态衰减网络(dynamic attenuation network, DAN)和动态衰减梯度下降(dynamic attenuation gradient descent, DAG)算法。具体做法如下:神经网络由众多神经元组成,不同层之间的神经元通过连接权重互相连接,本文在不同层每个神经元之间增加了一条权重值不断衰减的连接权重,命名为衰减权重,在反向传播计算梯度过程中,通过衰减权重在参数更新公式中引入了衰减权重项,随着网络迭代,衰减权重值会根据前一次迭代的值呈指数衰减,最终趋于 0,在这一过程中,可以起到加快参数更新的速度,同时不会由于一直加速而越过最优解和在最优解附近振荡,让网络可以更大概率的获得全局最优解。此方法结合了深层网络和 SGD 算法的优势,使网络具有更好的性能。本文分别把 DAN 嵌入到不同种类的网络模型中,并使用 DAG 算法与目前常用的几个算法进行对比实验,实验结果表明:此方法具有更好的测试准确度和更快的收敛速度。

1　网络模型和算法介绍

为了解决梯度消失的问题,一些优化算法往往直接从梯度下降公式的角度去解决,通常忽视了神经网络结构对网络性能的影响,Pasini 等[15]认为,神经网络的拓扑结构可以影响到网络的输出。SGDM 算法的梯度更新表达式为:

$$\Delta\theta_t = \eta \nabla f_{it}(\theta_t) + \rho \Delta\theta_{t-1} \tag{1}$$

$$\theta_{t+1} = \theta_t - \Delta\theta_t \tag{2}$$

式中:ρ 为动量系数。

有关动量项的实际意义,有些学者认为[16],动量项可以理解为在梯度下降的过程中施加了一个力,将 $\Delta\theta$ 看作是速度,通过施加的力,来改变速度进而改变位置不断下降。但由于不断的加速,导致速度过大错过了最优解。如果这个力可以不断的减小甚至当快接近最优解时不再加速,就可以减少错过全局最优解的概率。以此思想为基础,将恒定的动量系数改进成了一个随着迭代次数不断衰减的系数,其初始值为(0,1)之间的随机小数;为了改进网络结构,在不同层每个神经元之间引入一条值不断衰减的权重来表示这个衰减系数,当使用 BP 算法时,梯度更新公式也随之改变。

在网络结构方面,以 3 层全连接网络为例,不同层每个神经元之间在原有的连接基础上,增加了一条衰减权重,构成 3 层的动态衰减网络(DAN),如图 1 所示,图中的虚线表示衰减权重。DAN 看似是对全连接层的一个简单的改进,但其作用不单是增加了一条衰减权重,其实际意义是增加了虚拟的网络层,这使得网络模型的深度增加,使其具有更加出色的非线性表示关系。

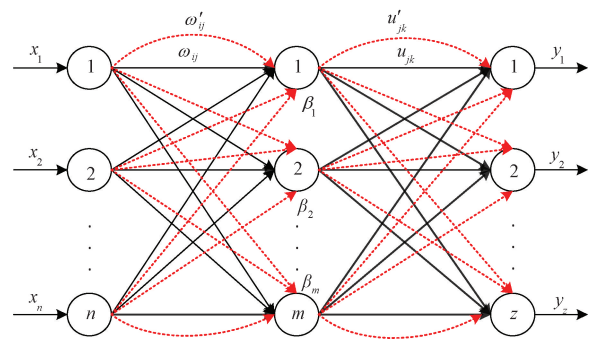

图 1　3 层 DAN 网络结构

Fig. 1　Structure diagram of DAN network with three layers

图 1 中,ω_{ij},ω'_{ij}($i=1,2,\cdots,n$;$j=1,2,\cdots,m$)分别为输入成到隐藏层的连接权重和衰减权重,u_{jk},u'_{jk}($=1,2,\cdots,z$)分别为隐藏层到输出层的连接权重和衰减权重。x_i 为网络输入,β_j 为隐藏层输出,y_k 为网络输出。

在计算梯度下降公式方面,由于网络结构发生了变化,因此网络的输出表达式也发生了变化,以 3 层全连接网络为例,假设输入层神经元个数为 n,隐藏层个数为 m,输出神经元个数为 z。输入层与隐藏层之间的连接权重为 $\omega_{11},\omega_{12},\cdots,\omega_{1m};\cdots;\omega_{n1},\omega_{n2},\cdots,\omega_{nm}$,衰减权重为

$\omega'_{11},\omega'_{12},\cdots,\omega'_{1m};\cdots;\omega'_{n1},\omega'_{n2},\cdots,\omega'_{nm}$；隐藏层与输出层之间的连接权重为 $u_{11},u_{12},\cdots,u_{1z};\cdots;u_{m1},u_{m2},\cdots,u_{mz}$，衰减权重为 $u'_{11},u'_{12},\cdots,u'_{1z};\cdots;u'_{m1},u'_{m2},\cdots,u'_{mz}$。神经元的激活函数为 f，隐藏层神经元的阈值为 b_j，输出层神经元的阈值为 θ_k，则隐藏层的输出为：

$$\beta_j = \sum_{i=1}^{n} f((\omega_{ij} + \omega'_{ij})x_i - b_j) \tag{3}$$

神经网络的输出为：

$$y_k = \sum_{j=1}^{m} f((u_{jk} + u'_{jk})\beta_j - \theta_k) \tag{4}$$

则神经网络的动态输入输出表达式为：

$$y_k = \sum_{j=1}^{m} f((u_{jk} + u'_{jk}) \sum_{i=1}^{n} f((\omega_{ij} + \omega'_{ij})x_i - b_j) - \theta_k) \tag{5}$$

定义交叉熵函数为损失函数 $F(\omega)$，学习率(步长)为 η 根据 BP 算法，DAN 的参数更新公式(以连接权重 ω 为例)：

$$\Delta\omega = -\eta \times \frac{\partial F}{\partial \omega_{t-1}} \tag{6}$$

$$\frac{\partial F}{\partial \omega_{t-1}} = \frac{\partial F}{\partial y} \times \frac{\partial y}{\partial \omega_{t-1}} = \frac{\partial(-y'_k \log y_k)}{\partial y_k} \times \frac{\partial y_k}{\partial \omega_{t-1}} \tag{7}$$

$$\frac{\partial F}{\partial \omega_{t-1}} = = -\frac{y'_{t-1}}{y_{t-1}} \times f'(u_{t-1} + u'_{t-1}) f' \sum_{i=1}^{n} x_i \tag{8}$$

$$\omega_t = \omega_{t-1} - \eta \times \frac{y'_{t-1}}{y_{t-1}} \times f'(u_{t-1} + u'_{t-1}) f' \sum_{i=1}^{n} x_i \tag{9}$$

每一次迭代，衰减权重 ω'，u' 进行一次衰减，其衰减表达式为：

$$\begin{cases} \omega'_t = (1-\alpha)\omega'_{t-1} \\ u'_t = (1-\alpha)u'_{t-1} \end{cases} \tag{10}$$

式中：α 为衰减系数。

以上就是动态衰减梯度下降算法(DAG)参数更新表达式，DAG 算法的大致流程如算法 1 所示。

算法 1 动态衰减梯度下降法(DAG)

Parameters：全局学习率 η，衰减系数 α，最大迭代次数 Γ，初始参数 ω,ω'，时间步长 t

Require：学习率 $\eta = e^{-4}, \alpha = 0.9$

Require：初始化参数 ω,ω'

$\omega \leftarrow \omega \sim N(0,1^2)$，$\omega' \leftarrow \omega' \sim N(0,1^2)$

Require：定义参数的损失函数 $F(\omega)$

While：$F(\omega)$ 不收敛 **do**

For all i in Γ **do**

$t \leftarrow t+1$

$\Delta\omega_{t+1} \leftarrow \nabla F_{t+1}(\omega_t)$：计算 t 时刻损失函数 $F(\omega)$ 的梯度

$\omega_{t+1} \leftarrow \omega_t + \Delta\omega_{t+1}$：更新参数 ω

$\omega'_{t+1} \leftarrow (1-a)\omega'_t$：更新衰减权重值 ω'

End for

Return ω,ω'

End while

2 收敛性分析

为了证明 DAG 算法收敛，需要找到一个精确值 ε，使对任意参数 ω_t 满足不等式[17]：

$$\sum_{t=1}^{\Gamma} \mathrm{E}[\nabla F(\omega_t) - \nabla F(\omega^*)] \leqslant \varepsilon \tag{11}$$

则说明 DAG 算法是收敛的，收敛率为 ε，式中 ω^* 为最优解状态下参数值，Γ 为最大迭代次数。

在分析 DAG 算法的收敛性之前，先假设损失函数 $F(\omega)$ 为连续凸函数，学习率(步长)为 η。根据这一假设，建立如下引理：

引理 1[18] 对于每个凸函数，若函数满足 Lipschitz 连续性条件，则对于所有的 $x,y \in R^d$ 都成立如下不等式：

$$\| \nabla F(\omega_{t+1}) - \nabla F(\omega_t) \| \leqslant L \| \omega_{t+1} - \omega_t \| \tag{12}$$

式中：$\| x \|$ 代表了向量的模长，L 即为 Lipschitz 常数 $L \leqslant \frac{1}{\eta}$。

引理 2 基于引理 1，任何满足 Lipschitz 连续性条件的目标函数，通过对函数 $F(\omega)$ 进行泰勒二次展开，对于任意 $x,y \in R^d$，都满足如下不等式：

$$| F(\omega_{t+1}) - F(\omega_t) - \nabla F(\omega_t)^{\mathrm{T}}(\omega_{t+1} - \omega_t) | \leqslant \frac{L}{2} \| \omega_{t+1} - \omega_t \|^2 \tag{13}$$

将 $\omega_{t+1} = \omega_t - \eta \nabla F(\omega_t)$ 代入式(13)中，得到：

$$F(\omega_{t+1}) \leqslant F(\omega_t) + \nabla F(\omega_t)^{\mathrm{T}}(\omega_{t+1} - \omega_t) + \frac{L}{2} \| \omega_{t+1} - \omega_t \|^2$$

$$F(\omega_{t+1}) = F(\omega_t) - \nabla F(\omega_t)^{\mathrm{T}} \eta \nabla F(\omega_t) + \frac{L}{2} \| \eta \nabla F(\omega_t) \|^2$$

$$F(\omega_{t+1}) = F(\omega_t) - (\eta - \frac{L\eta^2}{2}) \| \nabla F(\omega_t) \|^2$$

$$F(\omega_{t+1}) = F(\omega_t) - (1 - \frac{L\eta}{2}) \eta \| \nabla F(\omega_t) \|^2$$

$\because L \leqslant \frac{1}{\eta}$，并且 $\eta, L > 0$

$\therefore 1 - \frac{L\eta}{2} \leqslant \frac{1}{2}$

得到：

$$F(\omega_{t+1}) \leqslant F(\omega_t) - \frac{1}{2}\eta \| \nabla F(\omega_t) \|^2 \tag{14}$$

定理 1 在引理 2 支持的条件下，假设 $\omega_{t+1} = \omega_t - \eta \nabla F(\omega_t)$，对于算法 1 中的目标函数 $F(\omega)$，将 $\omega^* = \omega$ 代入式(14)中，得到：

$$\sum_{t=1}^{\Gamma} \mathrm{E}[F(\omega_t) - F(\omega^*)] \leqslant \frac{1}{2\Gamma}\eta \left\| (u_t + u'_t) \sum_{i=1}^{n} x_i \right\|^2 \tag{15}$$

证明过程如下：

证明：设 $\omega^* = \operatorname{argmin}\sum_{t=1}^{T} F(\omega_t)$，根据式(13)，得到：

$$F(\omega_t) \leqslant F(\omega^*) + \nabla F(\omega_t)^{\mathrm{T}}(\omega_t - \omega^*) + \frac{L}{2}\|\omega_t - \omega^*\|^2$$

$$F(\omega_t) \leqslant F(\omega^*) + \nabla F(\omega_t)^{\mathrm{T}}(\omega_t - \omega^*)$$

再代入式(14)中，得到：

$$F(\omega_{t+1}) \leqslant F(\omega^*) + \nabla F(\omega_t)(\omega_t - \omega^*) - \frac{1}{2}\eta\|\nabla F(\omega_t)\|^2$$

$$F(\omega_{t+1}) - F(\omega^*) \leqslant \frac{\omega_t - \omega_{t+1}}{\eta}(\omega_t - \omega^*) - \frac{1}{2\eta}\|\omega_t - \omega_{t+1}\|^2 \leqslant$$

$$\frac{1}{2\eta}\|\omega_t - \omega^*\|^2 - \frac{1}{2\eta}\|\omega_t - (\omega^* + \eta\nabla F(\omega_t))\|^2 \leqslant$$

$$\frac{1}{2\eta}\|\omega_t - \omega^*\|^2 - \frac{1}{2\eta}\begin{pmatrix}\|\omega_t - \omega^*\|^2 - 2\eta\nabla F(\omega_t)(\omega_t - \omega^*) + \\ \|\eta\nabla F(\omega_t)\|^2\end{pmatrix}$$

$$F(\omega_{t+1}) - F(\omega^*) \leqslant \frac{1}{2\eta}(\|\omega_t - \omega^*\|^2 - \|\omega_{t+1} - \omega^*\|^2)$$

$$\sum_{t=1}^{T} F(\omega_t) - TF(\omega^*) \leqslant \frac{1}{2\eta}\begin{pmatrix}\|\omega_0 - \omega^*\|^2 - \|\omega_1 - \omega^*\|^2 + \|\omega_1 - \omega^*\|^2 + \\ \cdots + \|\omega_{T-1} - \omega^*\|^2 - \|\omega_T - \omega^*\|^2\end{pmatrix}$$

$$\sum_{t=1}^{T} F(\omega_t) - TF(\omega^*) \leqslant \frac{1}{2\eta}(\|\omega_0 - \omega^*\|^2 - \|\omega_T - \omega^*\|^2)$$

$$TF(\omega_t) - TF(\omega^*) \leqslant \frac{1}{2\eta}\|\omega_0 - \omega^*\|^2$$

$$F(\omega_t) - F(\omega^*) \leqslant \frac{1}{2\eta T}\|\omega_0 - \omega^*\|^2$$

$$\sum_{t=1}^{T}\mathrm{E}(F(\omega_t) - F(\omega^*)) \leqslant \frac{1}{2T}\eta\|\nabla F(\omega^*)\|^2 = \frac{1}{2T}\eta\|(u + u')\sum_{i=1}^{n} x_i\|^2$$

即当迭代次数达到最大迭代次数时，$\lim_{t\to T} u' = 0$，根据式(11)，能找到一个精确的约束值 $\varepsilon = O(\frac{1}{T})$，使 DAG 算法收敛，收敛率为 $O(\frac{1}{T})$。

3　数值实验分析

为了证实 DAN 的有效性，分别使用不同网络来测试 MNIST[19]、CIFAR-10[20] 和 CIFAR-100[20] 数据集，三者均属于图像识别领域。在 MNIST 数据集上使用了 3 层全连接网络和卷积神经网络(LeNet-5 模型[21])，在 CIFAR-10 数据集上使用了 ResNet18 网络[22]，在 CIFAR-100 数据集上使用了 ResNet50 网络，使用 DAN 嵌入到上述网络中，将 DAG 算法与初始网络使用 Adam 算法和 SGDM 算法在同一数据集下进行了比较，设置 SGDM 算法的动量项 $\beta = 0.9$，对于 Adam，设置 $\beta_1 = 0.9$，$\beta_2 = 0.999$，计算了训练集和测试集的准确度和损失值。为了公平起见，在保证其他共同参数一致的情况下，本文分别对每个方法进行了 3 次的数值实验，最终取得平均值，所有实验数据基于 Python 语言获得。

3.1　全连接网络测试 MNIST 数据集

在 MNIST 数据集使用了 3 层全连接网络进行测试，包含输入层、隐藏层和输出层。使用 DAN 替换了全连接层网络，两个网络参数设置如下：输入层和输出层的神经元个数分别为 784 和 10，隐藏层的神经元个数设置为 500，前两层神经元的激活函数均为 ReLU 激活函数，最后一层为 softmax 函数，选择交叉熵函数作为损失函数，设置最大迭代次数为 200 次，学习率 $\eta = 0.01$，Batch-size = 100。衰减系数 $\alpha = 5 \times 10^{-4}$。图 2 和 3 展示了 MNIST 数据集在原始网络两种算法和 DAN 上的训练和测试的损失值曲线和准确度曲线。

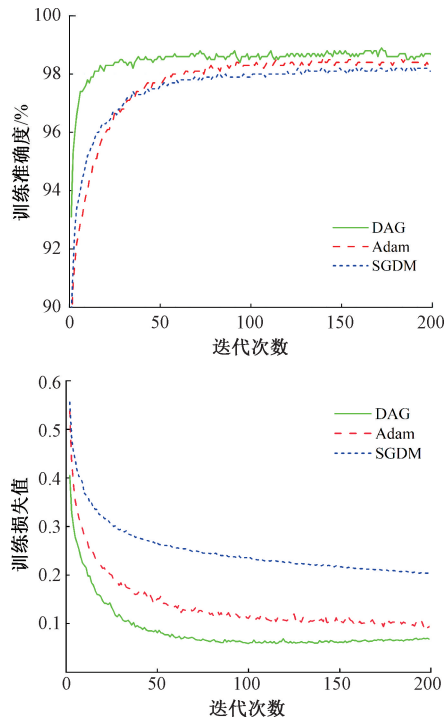

图 2　训练准确度和损失值

Fig. 2　Train accuracy and loss values

表 1 显示了 3 种方法在训练和测试时的最佳准确度和损失值。

表 1　MNIST 数据集在全连接网络上

Table 1　MNIST dataset on Full connected net

算法	训练损失值	训练准确度/%	测试损失值	测试准确度/%
DAG	0.059	98.941	0.064	98.75
Adam	0.091	98.632	0.090	98.34
SGDM	0.203	98.268	0.204	98.00

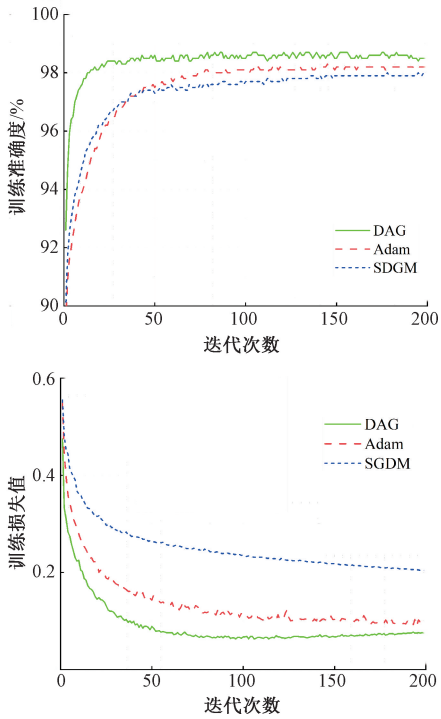

图3　测试准确度和损失值

Fig. 3　Test accuracy and loss values

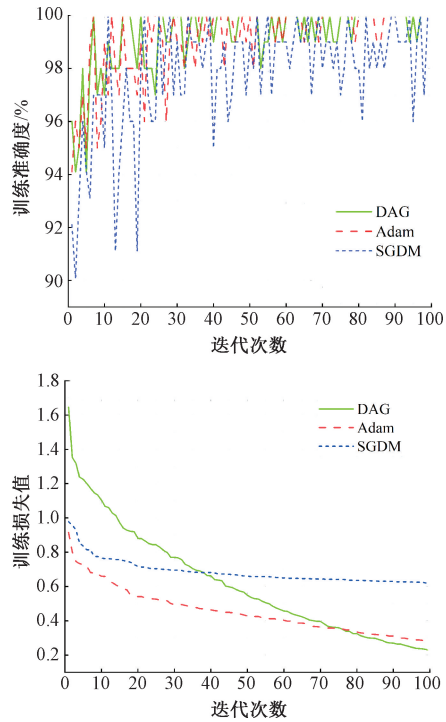

图4　训练准确度和损失值

Fig. 4　Train accuracy and loss values

由表1可知DAG算法在训练和测试时,平均准确度和误差值均优于Adam算法和SGDM算法,3次实验结果显示,DAG算法的平均测试准确度为98.72%,测试损失值为0.064,相比于Adam算法来说,平均准确度提高了0.41%,损失值减小了0.026;相比于SGDM算法来说,平均测试准确度提高了0.75%,损失值减小了0.14。由图2和3可以看出,DAG算法在收敛速度上明显快于另外两种方法。综合以上分析,在MNIST数据集上,DAG算法在准确度和收敛速度上要明显优于另外两种方法。

3.2　卷积网络测试MNIST数据集

使用卷积神经网络(convolutional neural networks, CNN)在MNIST数据集上进行测试。本文使用LeNet-5模型。使用DAN网络替换了全连接层网络,将DAG算法与初始网络使用Adam算法和SGDM算法对比。网络参数设置如下:池化层统一使用最大池化法,除了最后一层的激活函数为softmax,其余层的激活函数均为ReLU函数,损失函数选择为交叉熵函数,为了防止出现过拟合,加入了L2正则化惩罚项。设置最大迭代次数为100次,学习率$\eta = 0.001$,Batch-size=100,衰减系数$\alpha = 5 \times 10^{-4}$。图4和5分别展示了MNIST数据集在CNN上两种算法和DAN上的训练和测试的损失值曲线和准确度曲线。

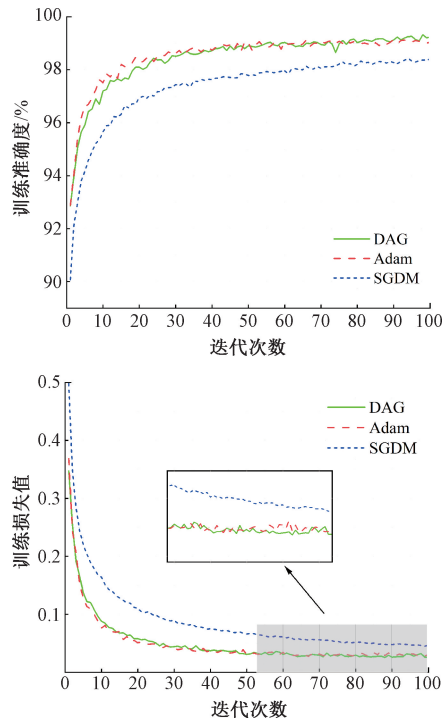

图5　测试准确度和损失值

Fig. 5　Test accuracy and loss values

表2显示了3种方法在训练和测试时的最佳准确度和损失值。

表 2　MNIST 数据集在卷积网络上

Table 2　MNIST dataset on CNN

算法	训练损失值	训练准确度/%	测试损失值	测试准确度/%
DAG	0. 226	100. 0	0. 025 4	99. 32
Adam	0. 283	100. 0	0. 027 4	99. 13
SGDM	0. 617	100. 0	0. 045 08	98. 42

由表 2 可知,DAG 算法的训练损失值最小;在测试集下,DAG 算法的平均准确度为 99. 32%,损失值 0. 025 4,相比于 Adam 算法来说,准确度提高了 0. 19%,损失值减小了 0. 002;相比于 SGDM 算法来说,平均测试准确度提高了 0. 9%,损失值减小了 0. 02。由图 4 可以看出,在训练集上,由于在损失函数中添加了 L2 正则化惩罚项,DAN 增加了若干条衰减权重,因此损失函数的初始值会变大,但最终 DAG 算法的损失值要小于另外两种方法。同时由图 5 可以看出,在测试集上,DAG 算法的收敛速度稍微慢于 Adam 算法;在准确度和损失值上,DAG 算法略优于 Adam 算法,但整体曲线趋势接近,但总体优于 SGDM 算法。

3. 3　残差网络测试 CIFAR-10

使用 18 层的残差网络(residual network,ResNet18)在 CIFAR-10 数据集上进行测试。使用 DAN 替换 ResNet18 的全连接层,两个网络参数设置如下:全连接层神经元个数设置为 512 个,激活函数选择 ReLU 函数,选择交叉熵函数为损失函数,设置最大迭代次数为 40 次,学习率 $\eta = 0.000\ 1$, Batch-size = 128, 衰减系数 $\alpha = 8.5\times10^{-4}$。图 6 和 7 分别展示了 CIFAR-10 数据集在 ResNet18 上两种算法和在 DAN 上的训练和测试的损失值曲线和准确度曲线。

表 3 显示了 3 种方法的训练和测试时的最佳准确度和损失值。

表 3　CIFAR-10 数据集在 ResNet18

Table 3　CIFAR-10 dataset on ResNet18

算法	训练损失值	训练准确度/%	测试损失值	测试准确度/%
DAG	0. 081	96. 156	0. 195 0	89. 19
Adam	0. 079	95. 896	0. 286 3	88. 95
SGDM	0. 125	94. 063	0. 309 5	86. 85

由表 3 可知,在训练集上,DAG 算法的训练准确度优于 Adam 算法与 SGDM 算法;在损失值上 DAG 算法略逊于 Adam 算法,原因是衰减权重的初始值为(0,1)的随机值,具有一定的随机性,且二者相差甚小,仅为 0. 002,对实验结论并无太大的影响,可忽略不计。但优于 SGDM 算法。在测试集上,DAG 算法的损失值优于其他两种算法,与 Adam 算法相比减小了 0. 091,相比于 SGDM 算法减小了 0. 114 5。在测试准确度上,DAG 算法

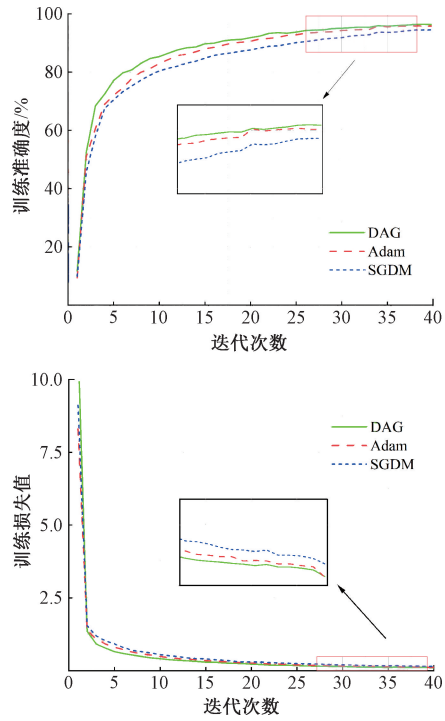

图 6　训练准确度和损失值

Fig. 6　Train accuracy and loss values

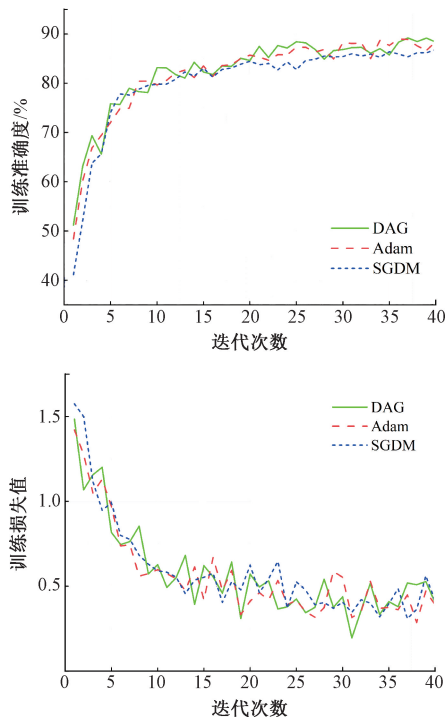

图 7　测试准确度和损失值

Fig. 7　Test accuracy and loss values

和 Adam 算法很接近,相比于 SGDM 算法准确度提高了 2. 34%。同时,根据图 6 可以看出,在训练集时,DAG 算法和 Adam 算法的整体准确度曲线和损失值曲线较为接

近,但 DAG 算法的收敛速度略快于 Adam 算法,且优于 SGDM 算法;由图 7 可以看出,在测试集上,DAG 算法和 Adam 算法较为接近。相比于 SGDM 算法,DAG 算法具有明显的优势。

3.4　残差网络测试 CIFAR-100

使用 ResNet50 网络在 CIFAR-100 数据集上进行测试,将初始网络搭建成 DAN,两个网络参数设置如下:设置最大迭代次数为 50 次,学习率 $\eta=0.000\ 1$,Batch-size=256,衰减系数 $\alpha=8.5\times10^{-4}$。图 8 和 9 分别展示了 CIFAR-100 数据集在 ResNet50 上两种算法和在 DAN 上的训练和测试的损失值曲线和准确度曲线。

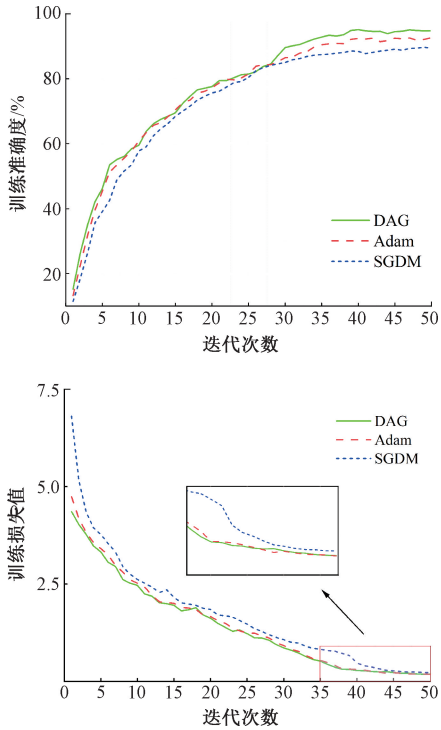

图 8　训练准确度和损失值

Fig. 8　Train accuracy and loss values

表 4 显示了 3 种方法在训练和测试时的最佳准确度和损失值。

表 4　CIFAR-100 数据集在 ResNet50

Table 4　CIFAR-100 dataset on ResNet50

算法	训练损失值	训练准确度/%	测试损失值	测试准确度/%
DAG	0.182	95.06	0.761	71.42
Adam	0.186	92.65	1.089	69.53
SGDM	0.230	89.40	0.854	70.84

由表 4 可知,在训练集上,DAG 算法和 Adam 要优于 SGDM 算法。在测试集上,DAG 算法的损失值要优于其他两种算法,与 Adam 算法相比减小了 0.328,相比于 SGDM 算法减小了 0.093;在测试准确度上,DAG 算法相比于 Adam 算法和 SGDM 算法,分别提高了 2.89%和 1.58%。根据图 8 可以看出,在训练集,DAG 算法和 Adam 算法的收敛速度较相近,整体准确度曲线和损失值曲线较为接近,优于 SGDM 算法;由图 9 可以看出,在测试集上,DAG 算法的曲线较平稳,且在准确度和损失值上,相对于两种算法,DAG 算法具有一定的优势。

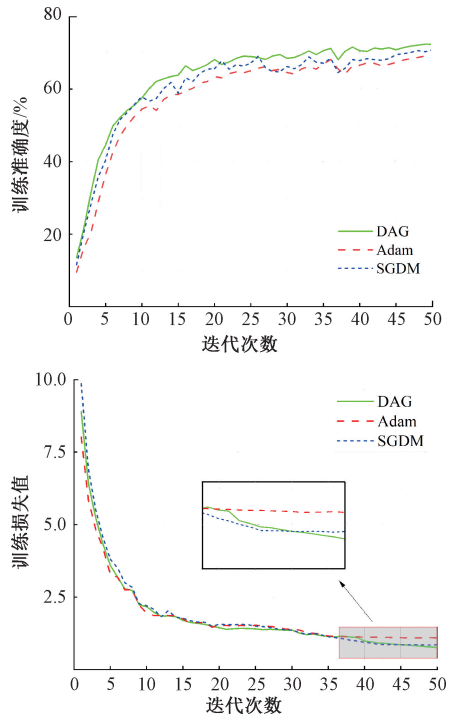

图 9　测试准确度和损失值

Fig. 9　Test accuracy and loss values

4　结　论

本文针对梯度下降法在训练过程中容易陷入局部最优和收敛速度慢的问题,通过在层与层每个神经元之间增加一条衰减的连接权重,构造了一种动态衰减网络模型(DAN),并提出了动态衰减梯度下降法(DAG),并使用了一系列数学定理对 DAG 算法做了收敛性分析,证明了其稳定性。此方法结合了深层网络和随机梯度下降的优势,能更好的表示输入和输出的非线性关系,提高了网络获得全局最优解的概率,加快了收敛速度。基于 MINST、CIFAR-10 和 CIFAR100 数据集的实验数据表明,将 DAN 替换了不同种类的初始网络之后,相较 Adam 算法和带动量随机梯度下降法(SGDM)而言,DAG 算法具有更快的收敛速度和更高的测试准确度;与 Adam 算法相比,测试准确度提高了 0.2%~1.89%;与 SGDM 算法

相比,测试准确度提高了 0.75%~2.34%,由准确度曲线和损失曲线可以看出,DAG 算法收敛到全局最优的速度更快。综上所述,本文提出的方法对于神经网络跳出局部最优具有重要的意义。

参考文献

[1] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553):436.

[2] 宫文峰, 陈辉, 张美玲. 基于深度学习的电机轴承微小故障智能诊断方法[J]. 仪器仪表学报, 2020, 41(1):195-205.
GONG W F. CHEN H, ZHANG M L. Intelligent diagnosis method for incipient fault of motor bearing based on deep learning [J]. Chinese Journal of Scientific Instrument, 2020,41(1):195-205.

[3] 李鑫,刘帅男,杨桢,等. 基于改进 Cascade R-CNN 的输电线路多目标检测[J]. 电子测量与仪器学报, 2021,35(10):24-32.
LI X, LIU SH N, YANG ZH, et al. Multi-target detection of transmission lines based on improved cascade R-CNN [J]. Journal of Electronic Measurement and Instrumentation,2021,35(10):24-32.

[4] WIJESINGHE S. Time series forecasting: Analysis of LSTM neural networks to predict exchange rates of currencies[J]. Instrumentation, 2020, 7(4):25.

[5] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks [C]. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, 2010: 249-256.

[6] ROBBINS H, MONRO S. A stochastic approximation method[J]. Annals of Mathematical Statistics, 1951, 22(3):400-407.

[7] 史加荣, 王丹, 尚凡华, 等. 随机梯度下降算法研究进展[J]. 自动化学报,2021,47(9):2103-2119.
SHI J R, WANG D, SHANG F H, et al. Research advances on stochastic gradient descent algorithms[J]. Acta Automatica Sinica, 2021,47(9):2103-2119.

[8] CHOROMANSKA A, HENAFF M, MATHIEU M, et al. The loss surfaces of multilayer networks[C]. Artificial Intelligence and Statistics, PMLR, 2015: 192-204.

[9] 刘建伟, 赵会丹, 罗雄麟, 等. 深度学习批归一化及其相关算法研究进展[J]. 自动化学报, 2020, 46(6): 1090-1120.
LIU J W, ZHAO H D, LUO X L, et al. Research progress on batch normalization deep learning and its related algorithms[J]. Acta Automatica Sinica, 2020, 46(6): 1090-1120.

[10] NING Q. On the momentum term in gradient descent learning algorithms[J]. Neural Netw, 1999, 12(1): 145-151.

[11] DUCHI J, HAZAN E, SINGER Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research, 2011, 12(7): 2121-2159.

[12] ZEILER M D. Adadelta: An adaptive learning rate method[J]. ArXiv Preprint,2012, arXiv:1212.5701.

[13] KINGMA D P, BA J. Adam: A method for stochastic optimization[J]. ArXiv Preprint,2014,arXiv:1412.6980.

[14] YUAN W, HU F, LU L . A new non-adaptive optimization method: Stochastic gradient descent with momentum and difference [J]. Applied Intelligence, 2021:1-15.

[15] PASINI M L, YIN J, LI Y W, et al. A scalable algorithm for the optimization of neural network architectures[J]. Parallel Computing, 2021, 104: 102788.

[16] RUDER S. An overview of gradient descent optimization algorithms [J]. ArXiv Preprint, 2016, arXiv:1609.04747.

[17] YANG Z, CHEN Z, WANG C. An accelerated stochastic variance-reduced method for machine learning problems[J]. Knowledge-Based Systems, 2020, 198: 105941.

[18] BUBECK S. Convex optimization: Algorithms and complexity[J]. Foundations and Trends in Machine Learning, 2015, 8(3-4): 231-357.

[19] LECUN Y, CORTES C, BURGES C. MNIST handwritten digit database[J]. 2010.

[20] KRIZHEVSKY A, HINTON G. Convolutional deep belief networks on cifar-10 [J]. Unpublished Manuscript, 2010, 40(7): 1-9.

[21] SZEGEDY C, LIU W, JIA Y, et al. Going deeper with

convolutions[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1-9.

[22] HE K, ZHANG X, REN S, et al. Deep residual learning for image recognition [C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016: 770-778.

作者简介

费春国,1996 年于天津理工学院获得学士学位,2003 年于天津科技大学获得硕士学位,2006 年于上海交通大学获得博士学位,现为中国民航大学副教授,主要研究方向为神经网络、故障诊断、非线性控制等。

E-mail: fchunguo@163.com

Fei Chunguo received his B. Sc. degree from Tianjin Polytechnic in 1996, his M. Sc. degree from Tianjin University of Science and Technology in 2003 and his Ph. D. from Shanghai Jiao Tong University in 2006, respectively. He is now an associate professor in the Civil Aviation University of China. His main research interests include neural networks, fault diagnosis and nonlinear control.

刘启轩(通信作者),2020 年于杭州电子科技大学获得学士学位,现为中国民航大学硕士研究生,主要研究方向为神经网络优化研究、图形处理。

E-mail: lqxdictator@163.com

Liu Qixuan (Corresponding author) received his B. Sc. degree from Hangzhou Dianzi University in 2016, and now he is a M. Sc. candidate at Civil Aviation University of China. His main research interests include neural network optimization and image processing.