计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201100032

一种新的优化机制:Rain

刘华玲　皮常鹏　刘梦瑶　汤　新
上海对外经贸大学统计与信息学院　上海 201600
(liuhl@suibe.edu.cn)

摘　要　在机器学习领域,传统模型的损失函数为凸函数,故具有全局最优解,通过传统的梯度下降算法可以求得最优解。但在深度学习领域,由于模型函数的隐式表达及同层神经元的可交换性,其损失函数为非凸函数,传统的梯度下降算法无法求得最优解,即使是较为先进的 SGDM,Adam,Adagrad,RMSprop 等优化算法也无法逃脱局部最优解的局限性,在收敛速度上虽然已经有很大的提升,但仍不能满足现实需求。现有的一系列优化算法都是针对已有优化算法的缺陷或局限性进行改进,优化效果有些许提升,但对于不同数据集的表现不一致。文中提出一种新的优化机制 Rain,该机制结合深度神经网络中的 Dropout 机制,并融入到优化算法上得以实现。该机制并不是原有优化算法的改进版,而是独立于所有优化算法的第三方机制,但可以和所有优化算法搭配使用,从而提高其对于数据集的适应性。该机制旨在对模型在训练集上的表现进行优化,测试集上的泛化问题并不作为该机制的关注点。文中利用 Deep Crossing 和 FM 两个模型搭配 5 种优化算法,分别在 Frappe 和 MovieLens 两个数据集上进行实验,结果表明,加入 Rain 机制的模型在训练集上的损失函数值明显减小,且收敛速度加快,但其在测试集上的表现与原模型相差无几,即泛化性较差。

关键词:深度学习;优化算法;Dropout 机制;Rain 机制;收敛速度
中图法分类号　TP391

New Optimization Mechanism:Rain

LIU Hua-ling,PI Chang-peng,LIU Meng-yao and TANG Xin
School of Statistics and Information,Shanghai University of International Business and Economics,Shanghai 201600,China

Abstract　The loss function of the traditional model in the field of machine learning is convex,so it has a global optimal solution. The optimal solution can be obtained through the traditional gradient descent algorithm (SGD). However,in the field of deep learning,due to the implicit expression of the model function and the interchangeability of neurons in the same layer,the loss function is a non-convex function. Traditional gradient descent algorithms cannot find the optimal solution,even the more advanced optimization algorithms such as SGDM,Adam,Adagrad,and RMSprop cannot escape the limitations of local optimal solutions. Although the convergence speed has been greatly improved,they still cannot meet the actual needs. A series of existing optimization algorithms are improved based on the defects or limitations of the previous optimization algorithms,and the optimization effect is slightly improved,but the performance of different data sets is inconsistent. This article proposes a new optimization mechanism Rain,which combines the dropout mechanism in deep neural networks and integrates it into the optimization algorithm to achieve. This mechanism is not an improved version of the original optimization algorithm. It is a third-party mechanism independent of all optimization algorithms,but it can be used in combination with all optimization algorithms to improve its adaptability to data sets. This mechanism aims to optimize the performance of the model on the training set. The generalization problem on the test set is not the focus of this mechanism. This article uses Deep Crossing and FM two models with five optimization algorithms to conduct experiments on the Frappe and MovieLens data sets respectively. The results show that the model with the Rain mechanism has a significant reduction in the loss function value on the training set,and the convergence speed is accelerated,but its performance on the test set is almost the same as the original model,that is,its generalization is poor.

Keywords　Deep learning,Optimization algorithm,Dropout mechanism,Rain mechanism,Convergence speed

1　引言

机器学习(Machine Learning,ML)是对能通过自动改进的计算机算法的研究[1],通俗来讲:让计算机从数据中进行自动学习,得到某种知识(或规律)。在最早期,机器学习被称为模式识别(Pattern Recognition,PR),直到 20 世纪 50 年末,“机器学习”一词才被提出来。随着机器学习技术的应用越来越广,模式识别已经逐渐被机器学习这一概念所替代。深度学习是机器学习发展过程中的一个分支,由于具有强大的非线性拟合能力,其开始慢慢演化为一个新的研究领域。深度学习在发展中也经过了几次比较大的发展浪潮,2015 年,AlphaGo 使用深度学习方法在围棋比赛中击败了欧洲围棋冠军[2],由此深度学习的影响更加广泛。

比较机器学习和深度学习方法哪种更好是没有意义的,

通信作者:皮常鹏(2754759189@qq.com)

因为两种方法看待问题的角度是不同的。机器学习很少进行特征变换或只能依靠上游处理来进行特征的变换[3]，也就是说它的模型训练和特征变换是独立的，并且特征变换往往需要手动解决，对先验知识依赖程度很大。深度学习通过其网络结构来执行端到端(End to End)的学习，网络自动学习特征变换，并且这种特征变换往往是深度的、非线性的，其面临的问题变为如何设计一个好的网络结构。从手动特征变换到设计网络结构是机器学习到深度学习看待问题角度的变化。

深度学习基于其非线性拟合特性，具有非常强的表达能力，但是将深度学习应用到机器学习领域仍然面临着一些难题，这些问题主要分为两大类：优化问题和泛化问题[4]。本文主要讨论优化问题。

常用的损失函数为 0-1 损失函数、平方损失函数和交叉熵损失函数。在机器学习中，平方损失函数和交叉熵损失函数是凸函数；0-1 损失函数由于不可微，因此也常常由平方损失函数和交叉熵损失函数所替代。常用 L1 和 L2 正则项是仿射函数，也是凸函数。为了充分利用凸优化中一些成熟高效的优化方法，很多机器学习方法都倾向于选择合适的模型和损失函数，并以构造一个凸函数作为优化目标[4]。在机器学习中，最简单和常用的算法是梯度下降方法，借助凸函数的特性，其优化问题能得到很好的解决。

深度学习的优化问题是非常复杂的。首先，由于模型函数的隐式表达及同层神经元的可交换性，其损失函数为非凸函数，找到全局最优解很困难。其次，由于参数规模、训练数据数量巨大，无法使用计算代价很高的二阶优化方法，而一阶优化方法训练效率又很低。最后，深度学习存在梯度消失和梯度爆炸的问题，导致基于梯度的优化方法经常失效。

在低维参数空间中，非凸优化问题的主要难点是如何选择初始化参数和逃离局部最优解；在高维参数空间中，非凸优化的难点在于如何逃离鞍点，通过在梯度方向引入随机性可以有效地逃离鞍点[5]。针对深度学习优化问题，主要的经验性改善方法有：使用更有效的优化方法、使用好的参数初始化方法和通过数据预处理方法、修改网络结构来得到更好的优化地形等。本文主要介绍优化方法。

深度学习中，对于不同的数据与模型，最适合的优化方法是不同的[6]。常用的优化方法又可以分为两类：一阶优化方法和二阶优化方法[3]。二阶优化方法考虑了损失函数的二阶导数信息，训练效率更高，如牛顿法、共轭梯度法、BFGS 算法[9]、L-BFGS 算法[10]等，比随机梯度下降有更快的收敛速度[7]。由于深度学习中参数数量、训练数据数量巨大，无法使用计算代价很高的二阶优化方法，经常会使用计算代价较低的一阶优化方法。常用的一阶优化方法都是在梯度下降方法上进行改进，可以分为 3 类：1)调整学习率，使得优化更稳定；2)梯度估计修正，优化训练速度；3)组合学习率调整和梯度估计修正，保证优化的稳定性和快速性。调整学习率的方法主要有：学习率衰减、学习率预热[11]、周期性学习率调整[11]以及一些自适应调整学习率的方法(AdaGrad[13]，RMSprop[14]，AdaDelta[15])。梯度修正的方法主要包括 SGDM[16]、Nesterov 加速梯度[17]、梯度截断[19]等。组合学习率调整和梯度修正的方法有：Adam[20]，AMSGrad[21]，AdaBound[22]，AMSBound[22]，Lookahead[23]，RAdam[24]。

传统的梯度下降[25]算法，将每个参数的学习率都设为一样的初始值，并且参数更新时对应的学习率并不更新，这种方法不能考虑到参数的特殊性，因而造成损失函数收敛速度慢。AdaGrad[13]算法是一种自适应学习率的方法，在参数更新时它给予每个参数不同的学习率，该给予方式是基于其偏导数累积值决定的。由于参数偏导数累积值随着时间累积到很高时，该参数学习率接近于 0，故而该参数不能进行学习更新，深度学习网络损失函数也不能收敛，该问题被称为学习率的早衰减问题。为了解决 AdaGrad 算法中学习率的早衰减问题，RMSprop[14]被提出来，其将 AdaGrad 算法中偏导数累积方式变成了指数衰减移动平均方式，有效解决了学习率的早衰减问题。同样是针对 AdaGrad 算法中学习率的早衰减问题，AdaDelta[15]不仅采用了指数衰减移动平均，而且考虑参数的平方值，以完全摆脱学习率，可根据历史参数更新数据和当前计算的梯度来更新参数。SGDM[16]是在梯度下降算法上引入了一阶动量因子，对某个参数，与当前的梯度方向和最近一段时间内的梯度方向是否一致起到参数更新幅度增大、减小的作用，可使参数更新更加稳定。Adam[20]算法将 RMSprop 和动量因子结合起来，综合考虑到了学习率调整和梯度估计修正两种优化方式，快速训练的特性，使其成为了许多深度学习框架的默认优化方法。文献[21]对 Adam 算法进行了强烈抨击，指责其存在可能不收敛和可能收敛到次优解两个问题，并对其缺陷进行修正，提出了 AMSGrad 算法。上述自适应算法都是比较主流的优化算法，它们有两个相同的特点：在训练开始时收敛很快，泛化性不够好。而传统的梯度下降算法和 SGDM 虽然收敛速度比较慢，但其泛化性很好。为了解决这个问题，AdaBound[22]和 AMSBound[22]被提出，它们分别是 Adam 和 AMSGrad 的变体。这两个变体在 Adam 和 AMSGrad 算法的学习率上使用动态边界，可以看成训练开始时的自适应方法，随着时间平稳地转换为 SGDM，这样能保证训练开始时速度较快，训练结束时泛化能力较好。Lookahead[23]在不改变优化器的基础上，采用两套权重进行参数更新，实验表明，该算法具有很好的优化效果。RAdam[24]使用学习率预热[11]的方法来解决 Adam 容易收敛到局部最优解的问题，同时前期采用 SGDM 算法进行训练。

上述所有的优化算法都有一个共同的特点：针对先前的优化算法的缺陷或局限性进行改进。本文提出了一种新的优化机制 Rain，打破算法改进的传统，通过建立第三方机制，配合原有的优化算法进行优化训练。该机制避开了深度神经网络中一阶优化和二阶优化方法直接改进的优化思路，选择一种优化器，随机初始化若干组模型参数组合(子模型)，利用深度神经网络中 Dropout 机制的随机性，间隔一段时间进行召集分派搜索。本文利用 Deep Crossing 和 FM 两个模型，搭配 5 种优化算法，分别在 Frappe 和 MovieLens 两个数据集上进行实验，结果表明，加入 Rain 机制的模型在训练集上的损失函数值明显减小，且收敛速度加快，但其在测试集上的表现与原模型相差无几，即泛化性较差。

本文的主要创新和贡献有以下几点。

(1)以往的优化算法的改进都是针对先前某个优化算法的缺陷或局限性进行改进。本文提出的 Rain 机制是一个独立于所有优化算法的第三方机制，它可以与所有优化算法搭配使用，提高了对数据集的适用性。

(2)通过实验，可以发现该机制的优化效果明显提升，且对于两个模型、5 个优化算法都有很强的迁移性。

(3)通过对主流优化算法研究思路的梳理，提出建立第三方机制的想法，为广大研究者提供了研究新思路。

2 深度学习常用优化算法

自适应调整学习率就是针对每个参数设置不同的学习率，使每个参数都以合适的学习率收敛到稳定值。随机梯度下降方法中每次迭代的梯度估计方向和整个训练集上的最优梯度方向并不一致，具有一定的随机性，可通过梯度的修正来提高优化速度。实践中常用的优化算法有梯度下降[25]、SGDM[16]、AdaGrad[13]、RMSprop[14]、Adam[20]，本节对这些算法进行详细的讨论。

2.1 梯度下降

深度学习的优化问题是使目标函数式(1)达到最小，其中 N 是样本数目，i 表示样本序号，$L_i(\theta)$ 是每个样本关于参数 θ 的损失值，$L(\theta)$ 是总体样本损失值。

$$L(\theta)=\frac{1}{N}\sum_{i=1}^{N}L_i(\theta) \tag{1}$$

对于梯度下降算法，根据每次使用不同样本数据量来计算目标函数的梯度，可以分为标准梯度下降法、随机梯度下降法、批量梯度下降法。

2.1.1 标准梯度下降法

标准梯度下降是每次参数更新都要计算所有样本的损失值，如式(2)所示。因为每次都向最优的方向更新参数，所以收敛速度快。但因为每次更新需要计算整个数据集的损失，对于大数据集，这是不可接受的，计算起来非常慢。

$$\theta_t=\theta_{t-1}-\eta\nabla_\theta L(\theta_{t-1}) \tag{2}$$

其中，η 表示学习率，t 表示迭代次数序号，$\nabla_\theta L(\theta_{t-1})$ 表示 $L(\theta_{t-1})$ 关于 θ 的梯度。

2.1.2 随机梯度下降法

与标准梯度下降法不同的是，随机梯度下降法每次只输入一个样本值计算损失值与梯度，进行参数更新，如式(3)所示。虽然其计算效率得到了很大的提升，但因为每次参数更新不一定沿着最优的方向进行，所以其收敛速度会比较慢。

$$\theta_t=\theta_{t-1}-\eta\nabla_\theta L(\theta_{t-1};x^{(i)};y^{(i)}) \tag{3}$$

2.1.3 批量梯度下降法

批量梯度下降法综合了标准梯度下降法和随机梯度下降法的优点，每次只利用一小批样本，即对多个样本进行梯度下降计算，如式(4)所示。这样一方面可以降低参数更新时的方差，收敛更稳定；另一方面可以充分利用深度学习库中高度优化的矩阵操作来进行更有效的梯度计算。

$$\theta_t=\theta_{t-1}-\eta\nabla_\theta L(\theta_{t-1};x^{(i;i+n)};y^{(i+n)}) \tag{4}$$

2.2 SGDM

SGDM 是在梯度下降法的基础上引入了动量(Momentum)，如式(5)所示。每个参数的实际更新差值取决于最近一段时间内梯度的加权平均值，对某个参数，当前的梯度方向和最近一段时间内的梯度方向不一致时，其真实的参数更新幅度会变小；相反，当两者梯度方向一致时，其真实的参数更新幅度会变大，起到加速作用。一般而言，在迭代初期，两者梯度方向一致，动量起到了加速作用，加快了收敛速度。但在迭代后期，因为随机性等因素，两者梯度方向不一致，参数会在收敛值附近震荡，此时动量可以起到减速的作用，增加稳定性。

$$\begin{cases}m_t=r\,m_{t-1}-\eta\nabla_\theta L(\theta_{t-1})\\ \theta_t=\theta_t+m_t\\ m_0=0\end{cases} \tag{5}$$

其中，r 表示动量因子，通常设为 0.9。

2.3 AdaGrad

在梯度下降算法中，每个参数每次迭代时都拥有相同的学习率，但实际情况为每个参数维度上的收敛情况都不相同，因此需要根据不同参数的收敛情况分别设置学习率。

如式(6)所示，在 AdaGrad 算法中，如果某个参数的偏导数累积比较大，其学习率相对较小；相反，如果其偏导数累积较小，其学习率相对较大。但整体是随着迭代次数的增加，学习率逐渐缩小。AdaGrad 算法的缺点是若经过一定次数的迭代依然没有找到最优，由于这时的学习率已经非常小，很难再继续找到最优点。

$$\begin{cases}g_{t-1,i}=\nabla_{\theta_i}L(\theta_{t-1})\\ \theta_{t,i}=\theta_{t-1,i}-\dfrac{\eta}{\sqrt{\sum_{j=0}^{t-1}g_{j,i}+\varepsilon}}\cdot g_{t-1,i}\end{cases} \tag{6}$$

其中，$\nabla_{\theta_i}L(\theta_{t-1})$ 表示第 $t-1$ 次迭代的损失值对参数 θ_i 的梯度，$g_{t-1,i}$ 与 $\nabla_{\theta_i}L(\theta_{t-1})$ 同义；$\theta_{t,i}$ 表示第 t 次迭代时参数 θ_i 的值；η 表示初始学习率；ε 是为了保持数值稳定性而设置的非常小的常数，一般取值为 e^{-7} 到 e^{-10}。

2.4 RMSprop

针对 AdaGrad 算法中学习率在不断下降以至于出现过早衰减，难以继续找到最优点的问题，研究者们提出了 RMSprop 算法，其也是一种自适应学习率方法，如式(7)所示。

$$\begin{cases}g_{t-1,i}=\nabla_{\theta_i}L(\theta_{t-1})\\ v_{t,i}=\alpha\,v_{t-1,i}+(1-\alpha)(g_{t-1,i})^2\\ \theta_{t,i}=\theta_{t-1,i}-\dfrac{\eta}{\sqrt{v_{t,i}+\varepsilon}}\cdot g_{t-1,i}\\ v_{0,i}=0\end{cases} \tag{7}$$

其中，α 是衰减率，一般取值为 0.9。

从式(7)中可以看出，RMSprop 算法和 AdaGrad 算法的区别在于将 $v_{t,i}$ 的计算由累积方式变成了指数衰减移动平均。在迭代的过程中，每个参数的学习率并不是呈衰减趋势，既可以变小也可以变大。

2.5 Adam

Adam 算法综合了 RMSprop 算法和动量，除了像 RMSprop 一样储存过去梯度平方的指数衰减平均值外，也储存了像动量一样的过去梯度的指数衰减平均值，如式(8)所示。

$$\begin{cases} g_{t-1,i}=\nabla_{\theta_i}L(\theta_{t-1}) \\ v_{t,i}=\alpha v_{t-1,i}+(1-\alpha)(g_{t-1,i})^2 \\ m_{t,i}=\beta m_{t-1,i}+(1-\beta)g_{t-1,i} \\ \hat{v}_{t,i}=\dfrac{v_{t,i}}{1-\alpha^t} \\ \hat{m}_{t,i}=\dfrac{m_{t,i}}{1-\beta^t} \\ \theta_{t,i}=\theta_{t-1,i}-\dfrac{\eta}{\sqrt{\hat{v}_{t,i}+\varepsilon}}\cdot\hat{m}_{t,i} \\ v_{0,i}=0 \\ m_{0,i}=0 \end{cases} \tag{8}$$

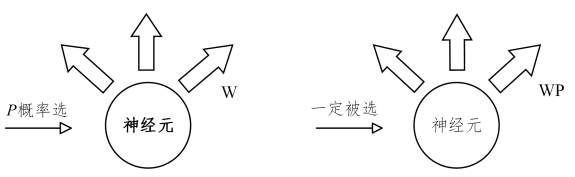
其中，α 和 β 分别为两个指数移动平均的衰减率，通常取值为 $\alpha=0.9$，$\beta=0.99$。

3 Dropout 机制

在机器学习中，常常会因为参数过多，训练样本数据过少，而出现过拟合现象。而在深度学习中，参数规模数以万计，如此大的参数规模，很容易造成过拟合现象，若只是简单地减小参数规模，模型训练的效果会差强人意。解决过拟合问题的方法，一般采用模型集成，但同时该方法会非常耗时。因此，针对深度学习中的过拟合问题，Hinton 于 2012 年提出了 Dropout[26] 机制，该机制同时也是集成模型的一个典型范例，它很好地解决了集成模型费时的问题。

该机制通过引入超参数 P 进行运作，该参数代表着 Dropout 率，但其在模型训练阶段和测试阶段有着不同的具体含义。如图 1 所示，在模型训练时，超参数 P 表示该层的每个神经元以 P 概率和下一层神经元进行连接，没有被选中进行连接的神经元失活；而在模型测试时，超参数 P 表示该层的每个神经元的权重乘 P 传输到下一层神经元中。值得注意的是，每一轮训练，每一神经元是以概率 P 被随机选择的，上一轮训练失活的神经元本轮训练仍可以被激活。

本文提出的 Rain 机制，正是借助于 Dropout 机制的随机性才得以实现。具体的实现方法见第 4 节。



(a)模型训练时 (b)模型测试时

图 1 Dropout 机制训练测试时超参数 P 所代表的不同含义

Fig. 1 Different meanings of the hyperparameter P during the training and testing of the Dropout mechanism

4 Rain 机制

梯度下降、SGDM 具有良好的泛化能力，但其损失函数值下降速度比较慢；常使用的自适应优化算法（AdaGrad，RMSprop，Adam，AMSGrad）虽然收敛速度得到了大幅提升，但泛化能力却大大减弱。针对这两个问题，似乎损失函数值下降速度和泛化能力不可兼得。本文提出的 Rain 机制，旨在增强原模型在训练集上的表现效果（更快的损失函数值下降速度，更低的损失函数值），对于训练集上的泛化问题未作更多的考虑。

常用的优化算法，通过随机初始化（不考虑预训练）一个模型参数组合来迭代搜索最优的参数配置组合，使得损失函数值达到最小。但同时存在一些问题，在参数空间内极易遇到局部最小值点或鞍点，从而导致损失函数值下降速度变慢和陷入局部最优解。为了加快损失函数值下降速度并逃离局部最优解，本文提出了一种新的机制——Rain 机制。Rain 机制的取名是因为它像雨珠一样可以自动流淌到地面的深洼处，通过阳光蒸发又可以重新回到水循环系统中并移动到下一个地点进行降落。Dropout 机制的作用是赋予它随机性，若没有 Dropout 机制，若干子模型参数在经过第一次召集后将无法进行各自不同的更新。

Rain 机制的主要思想见图 2：模型 I 表示未开始训练的模型；模型 I-i 表示训练过程中的子模型；模型 II 表示训练结束的模型。在不改变原始优化方法的基础上，随机初始化若干个模型参数组合（子模型），让它们分别按照自己的优化方法进行参数搜索，在间隔一定的迭代次数时召集所有子模型到损失函数值最低的子模型参数组合处，结合 Dropout 机制的随机性，让它们向不同方向再次搜索，循环往复，直到达到固定的迭代次数停止。算法步骤如算法 1 和算法 2 所示。

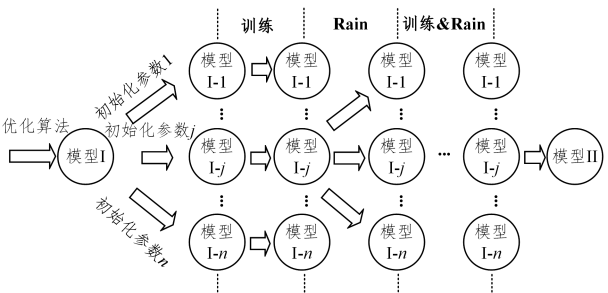


图 2 Rain 机制训练流程

Fig. 2 Rain mechanism training process

算法 1 ModelTrain

输入：X：features；Y：lables；N：subModel numbers；m：Maximum Iterations；n：Interval times

输出：θ：model parameters

1. for i=1 to N do
2. Initialization：ith subModel $Model_i$
3. Initialization：ith optimizer $Optimizer_i$
4. end for
5. for j=1 to m do
6. for i=1 to N do
7. $Loss_i=Model_i(X,Y)$
8. $\theta_i \leftarrow Optimizer_i(\theta_i, Loss_i)$
9. end for

1) http://baltrunas.info/researchmenu/frappe

2) http://grouplens.org/datasets/movielens/latest

10. if j%n=0 and j≠0 then

11. g=$\underset{i}{\text{argmin}}(Loss_i)$

12. for i=1 to N do

13. if i≠g then

14. Rain($Model_i$，$Model_g$，$Optimizer_i$，$Optimizer_g$)

15. end for

16. end for

算法 2 Rain

输入：ith subModel $Model_i$；gth subModel $Model_g$；ith optimizer $Optimizer_i$；gth optimizer $Optimizer_g$

输出：ith subModel $Model_i$

gth subModel $Model_g$

ith optimizer $Optimizer_i$

gth optimizer $Optimizer_g$

1. $Model_i \leftarrow Model_g$

2. $Optimizer_i \leftarrow Optimizer_g$

5 实证

5.1 数据集

在两个公开的数据集 Frappe[1)] 和 MovieLens[2)] 上进行实验。

Frappe 数据集拥有不同文本下的 96 203 个用户行为日志，除了用户 ID(user ID)和应用 ID(app ID)外，每个日志还包括 8 个场景变量，如天气(weather)、城市(city)、时辰(morning，afternoom)等。采用 one-hot 编码转化每个日志为特征向量，一共包含 5 382 个特征。标签(label)值为 1 表示用户在该场景下使用了该应用(app)。

MovieLens 数据集包含 17 045 个用户在 23 734 部电影上的 668 953 个观看评分记录。同样采用 one-hot 编码转化每个观看行为为特征向量，一共包含 90 445 个特征，标签(label)值为 1 表示用户对该电影进行了评分。

由于这两个数据集都只包含正样本，所以需要添加负样本以增强训练效果，设定负样本数量是正样本数量的两倍。在 Frappe 数据集上，我们为用户随机选取该场景下没有使用的两个应用(app)作为负样本；对于 MovieLens 数据集，随机选取用户未评分的电影作为负样本。负样本的标签(label)为 −1。数据集的具体参数见表 1。

表 1 数据集参数

Table 1 Data set parameters

Dataset	Instance#	Feature#	User#	Item#
Frappe	288 609	5 382	957	4 082
MovieLens	2 006 859	90 445	17 045	23 743

5.2 评估准则

随机将数据集分为训练集(70%)、验证集(20%)、测试集(10%)，其中训练集用于训练模型，验证集用来调整超参数并执行早停机制(Early Stopping)保存最优模型，在测试集上比较不同模型的执行效果。对于模型的评价准则，我们选用均方根误差(RMSE)，更低的 RMSE 有更好的效果。

为了综合评价加入 Rain 模型和原模型在训练集上的相对收敛效果，这里提出了 3 个指标：25ACC，100ACC，25/100ACC，具体见式(9)。

$$\begin{cases} 25ACC=\dfrac{25Min-Rain25Min}{25Min}*100\% \\ 100ACC=\dfrac{100Min-Rain100Min}{100Min}*100\% \\ 25/100ACC=\dfrac{100Min-Rain25Min}{100Min}*100\% \end{cases} \tag{9}$$

其中，$Rain25Min$，$25Min$ 分别表示加入 Rain 机制模型和原模型在训练集上前 25 个 epoch 的训练过程中损失值达到的最小值；$Rain100Min$，$100Min$ 为加入 Rain 机制模型和原模型在训练集上前 100 个 epoch 达到的最小值。

若 $25ACC$ 大于零，则说明在前 25 个 epoch 中加入 Rain 机制模型将优于原模型，小于零则相反；$25ACC$ 值越大，模型的优势越明显，最大值为 100%，当达到 100%时，则说明加入 Rain 机制模型已经收敛到全局最优解。$100ACC$ 与 $25ACC$ 解释相同。$25/100ACC$ 大于零，则说明加入 Rain 机制模型前 25 个 epoch 到达的最好效果已经优于原模型需要 100 个 epoch 所达到的最好效果，值越大，模型的优势越明显，最大值为 100%，表明加入 Rain 机制模型已经收敛到全局最优解。

5.3 Baseline

选用 FM 和 Deep Crossing 两个模型，搭配 SGD，SGDM，AdaGrad，RMSprop，Adam 优化方法，组成 10 个模型。再在该 10 个模型的基础上分别加入 Rain 机制，比较模型效果。

(1)FM。POLY2 模型大部分特征的权重缺乏有效的数据进行训练，无法收敛且训练复杂度极大增加。为了弥补上述缺陷，2010 年 Rendle 提出了 FM[27] 模型。与 POLY2 的主要区别是，FM 利用两个向量内积取代了单一的权重系数。细节上是 FM 为每个特征学习了一个相应特征隐向量，在进行特征交叉时，用两个特征隐向量的内积作为交叉特征的权重。通过引入特征隐向量的方式，使训练的复杂度显著降低，同时也更好地解决了数据稀疏性问题。

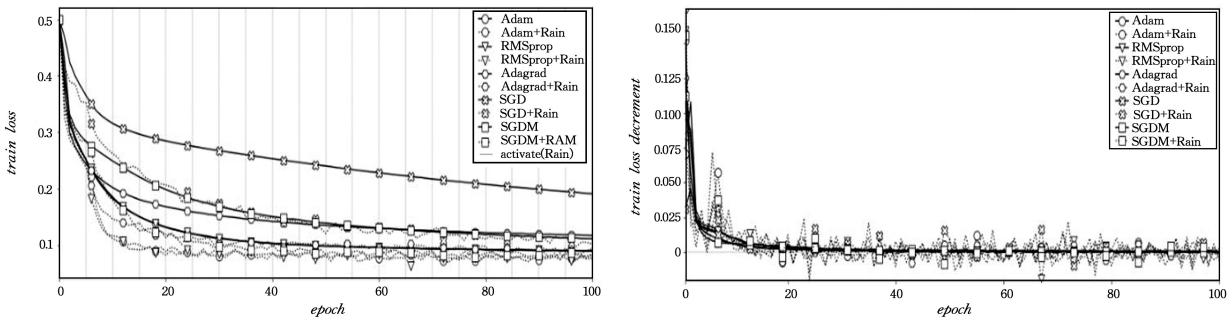
(2) Deep Crossing。2016 年微软提出的 Deep Crossing[28] 模型是深度学习框架在推荐系统上的一次完整应用，它完整地解决了从特征工程、稀疏向量稠密化、多层神经网络进行优化目标拟合等一系列深度学习在推荐系统中的应用问题。

5.4 参数设置

FM 模型和 Deep Crossing 模型均采用 Dropout 机制，Dropout 率分别为 0.3 与 0.5，且模型的 embedding 维度为 256，激活函数采用 Relu 函数；FM 模型、Deep Crossing 模型中 Dropout 机制分别放在 Pair-wise Interaction[29] 层和隐藏层；Deep Crossing 模型设定 4 个隐藏层，各层的神经元为[512，256，128，64]，且每个隐藏层均加入 BathNorm 机制、Relu 激活函数、Dropout 机制；Frappe 和 MovieLens 数据集的上 batch size 分别设为 128 和 4 096；每个模型的学习率遍历[0.000 1，0.001，0.003，0.005，0.01，0.02，0.05，0.1，0.5，1，2，5，15，30]，采用最好的学习率；加入 Rain 机制的模型，随机初始化 5 个子模型，召集间隔为 5 个 epoch；所有模型均采用早停机制保存最优模型参数，且模型利用 GPU 加速。值得注意的是，原模型和加入 Rain 机制的模型中所有超参数都是一致的，唯一的区别就是是否加入了 Rain 机制。

5.5 实验结果

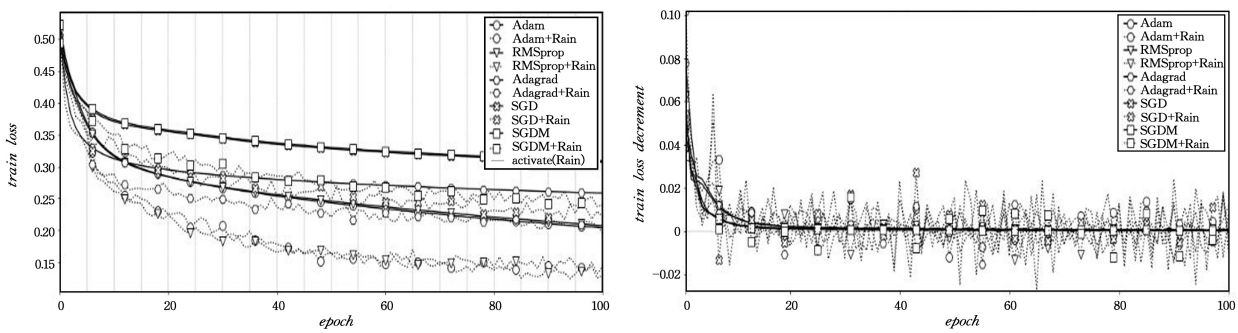
图3、图4为模型FM和Deep Crossing在数据集MovieLens上的训练结果。图中*epoch*表示所有训练集数据循环训练一次的时间，*train loss*表示训练集的损失函数值，*train loss decrement*表示该epoch较上一个epoch的train loss减小量，该值为正，则说明该epoch时刻*train loss*减小了；为负，则说明该epoch时刻*train loss*增大了；为0表示未变。



(a)ml-tag_FM_TrainLoss (b)ml-tag_FM_TrainLossDecrement

图3 FM模型在MovieLens上的训练集损失曲线及损失值减小量曲线

Fig. 3 Training set loss curve and loss value reduction curve of the FM model on MovieLens



(a)ml-tag_DeepCross_TrainLoss (b)ml-tag_DeepCross_TrainLossDecrement

图4 Deep Crossing模型在MovieLens上的训练集损失曲线及损失值减小量曲线

Fig. 4 Training set loss curve and loss value reduction curve of the Deep Crossing model on MovieLens

从训练集损失曲线可以看出，加入Rain机制的模型损失函数值下降速度明显加快，在原模型和加入Rain机制模型的损失值下降过程中，第一次分叉点就是第一次召集点(第5个epoch)处，每召集一次，损失值下降的速度就加快一次，而且加入Rain机制的模型比原模型的损失函数值更低，如表2所列，模型FM和Deep Crossing在25*ACC*平均值上分别超过了20%和35%，在100*ACC*平均值上均超过了30%，在25/100*ACC*平均值上分别超过3%和7%。这说明在前25个epoch和100个epoch，加入Rain机制模型加快了损失函数值下降速度，并且其在前25个epoch达到的效果就比原模型需要100个epoch达到的效果还要好。值得注意的是，加入Rain机制模型在所有模型和优化器上的指标值除了一个约为−0.18%以外，其它均大于0，说明Rain机制在该数据集上有良好的普适性。从损失值减小曲线可以看出，加入Rain机制的模型和原模型在前20epoch损失值下降剧烈，后期下降值在0上下震荡，因此加大训练周期并不能增强训练效果，故本文实验均采用100epoch作为训练时间，未做更长周期的训练。

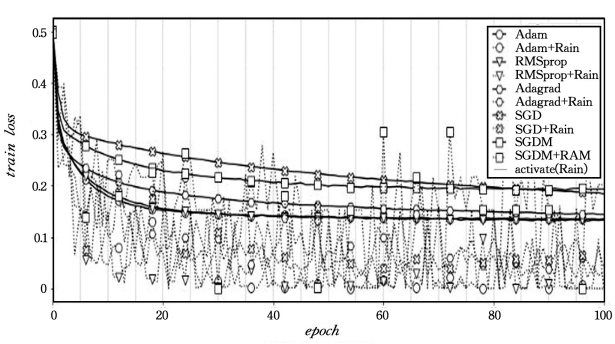
表2 MovieLens数据集上优化模型的性能对比

Table 2 Comparison of optimized model performance on MovieLens dataset

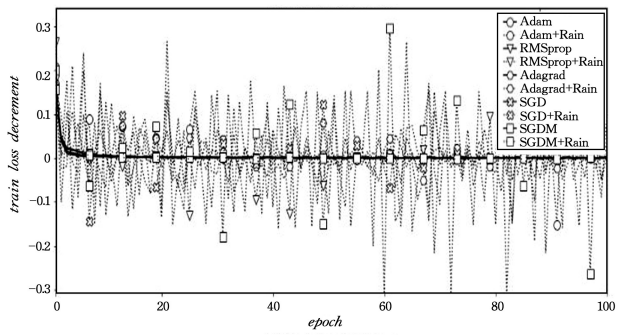
(单位：%)

优化器	DeepCross			FM		
	25*ACC*	100*ACC*	25/100*ACC*	25*ACC*	100*ACC*	25/100*ACC*
Adam+Rain	26.1061	38.8760	−0.1821	31.4292	20.2564	6.8072
RMSProp+Rain	29.2944	36.0936	5.6738	32.7015	29.7548	8.5661
Adagrad+Rain	13.7603	24.2779	2.9293	30.2264	27.6267	4.2557
SGD+Rain	17.5636	29.3977	5.8583	36.6707	49.1274	8.2549
SGDM+Rain	14.4057	23.2642	2.3282	44.9302	33.0934	8.8588
平均	20.2260	30.3819	3.3215	35.1916	31.9717	7.3485

图5、图6为模型FM、Deep Crossing在数据集Frappe上的训练结果。与数据集MovieLens上效果不同的，加入Rain机制模型的损失曲线震荡更加激烈，在损失值减小量曲线上也能看出相同的效果。这种震荡存在于该数据集下的所有模型和优化器上，这可能是由于该数据集存在较多的奇异值，使得优化地形变得崎岖，造成模型参数收敛出现问题。但总体上看，损失函数值下降速度十分快，最低损失值接近于0。从表3可以看出，比起数据集MovieLens，各个指标值十分高，甚至部分模型的指标值达到了近100%，意味着损失函数值下降到了全局最小值，模型优化效果十分明显。



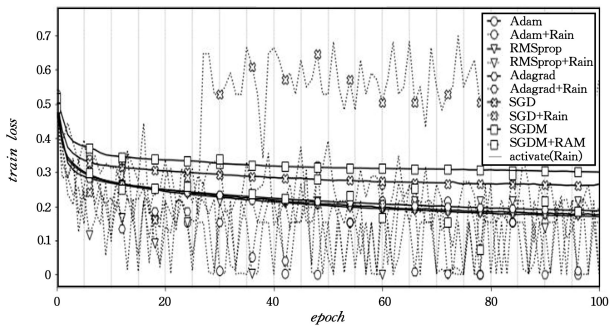
(a)frappe_FM_TrainLoss



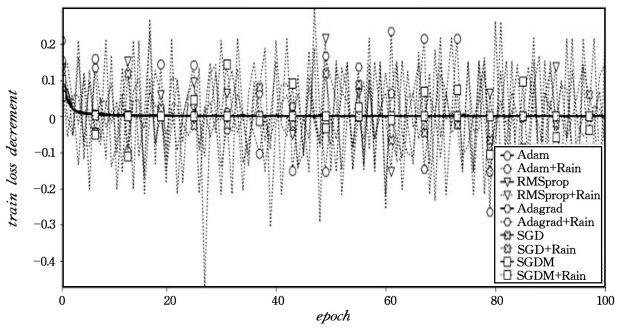
(b)frappe_FM_TrainLossDecrement

图 5 FM 模型在 Frappe 上的训练集损失曲线及损失值减小量曲线

Fig. 5 Training set loss curve and loss value reduction curve of the FM model on Frappe



(a)frappe_DeepCross_TrainLoss



(b)frappe_DeepCross_TrainLossDecrement

图 6 Deep Crossing 模型在 Frappe 上的训练集损失曲线及损失值减小量曲线

Fig. 6 Training set loss curve and loss value reduction curve of the Deep Crossing model on Frappe

表 3 Frappe 数据集上优化模型的性能对比

Table 3 Comparison of optimized model performance on Frappe dataset

（单位：%）

优化器	DeepCross			FM		
	25*ACC*	100*ACC*	25/100*ACC*	25*ACC*	100*ACC*	25/100*ACC*
Adam+Rain	91.1034	99.9996	87.6567	95.3728	99.9994	94.8615
RMSProp+Rain	99.1104	99.9999	98.7616	91.9519	99.9995	90.7627
Adagrad+Rain	77.3881	99.7775	70.6015	77.9065	85.9956	72.1090
SGD+Rain	54.4723	47.5761	47.5761	82.6938	92.1589	75.8781
SGDM+Rain	61.5904	75.9191	57.3703	99.9339	100.0000	99.9227
平均	76.7329	84.6544	72.3932	89.5718	95.6307	86.7068

值得注意的是，在两个数据集上，原模型的训练损失曲线和损失值减小量曲线都是光滑的，但加入 Rain 机制模型的两种曲线都是震荡的，只不过在 MovieLens 数据集上震荡较小，在 Frappe 数据集上震荡较大。对于这种现象，本文未做更深入的研究。表 4 列出了优化模型在测试集上的 RMSE，可以看出加入 Rain 机制的模型和原模型的测试集效果相差不大，说明加入 Rain 机制并不能很好地解决泛化性的问题。如何解决 Rain 机制的泛化性问题是后续研究的重点。

表 4 优化模型测试集的 RMSE

Table 4 RMSE of the optimized model test set

optim	Frappe		MovieLens	
	DeepCross	FM	DeepCross	FM
Adam	0.2222	**0.1814**	**0.2841**	**0.2411**
Adam+Rain	**0.2135**	0.1875	0.2844	0.2449
RMSProp	0.2246	**0.1844**	0.2837	**0.2492**
RMSProp+Rain	**0.2219**	0.1908	**0.2835**	0.2528
Adagrad	0.2335	0.1926	0.2852	0.2463
Adagrad+Rain	**0.2232**	**0.1828**	**0.2843**	**0.2455**
SGD	**0.2677**	0.2114	0.3270	0.2677
SGD+Rain	0.2936	**0.1916**	**0.2972**	**0.2605**
SGDM	0.3082	**0.2054**	0.3324	0.2568
SGDM+Rain	**0.2626**	0.2327	**0.2938**	**0.2527**

结束语 加入 Rain 机制的模型，在训练集上损失函数的收敛速度大大加快，并且收敛值更小，但是该结果是通过增加子模型的数量来获得的，因此其参数数量成比例的增长，优化的时间复杂度也成比例的增长。如何平衡时空复杂度和损失函数值下降速度成为了非常重要的问题，本文未对该问题做进一步探讨。但从实验结果上看，本实验在 Rain 机制的基础上初始化了 5 个子模型，在 25 个 epoch 就达到了原模型 100 个 epoch 的效果，虽然看似这种牺牲时空复杂度换取的损失函数值下降速度的措施是并不值得的，但是加入 Rain 机制的模型在最低损失函数值上远低于原模型，从损失值减小量曲线中可以看出损失值的进一步降低并不是靠训练时间就可以获得的，因此 Rain 机制在最低损失值上的表现是不可比拟的。从多个模型、数据集、优化算法上的实验可以看出，Rain 机制具有强大的可移植性。

通过实验可以发现一些问题，这也为未来研究指明了方向。

(1)如何平衡时空复杂度和损失函数值下降速度的问题。

(2)Rain 机制的优化效果显著，但泛化能力不强，如何提

升其泛化能力成为了亟需解决的问题。

(3)在 Frappe 数据集上,加入 Rain 机制后,不同的模型、优化器组合都出现损失值下降振幅过大的问题,初步猜想是由于 Frappe 数据集异常值过多的原因,具体原因需要进一步研究。若确实是由于数据异常值过多,但在原模型中却未出现该现象,是否可将加入 Rain 机制的模型作为判断数据集优良的一个标准可做进一步思考。

(4)加入 Rain 机制的模型在两个数据集上都出现了损失曲线震荡现象,但原模型的损失曲线却是光滑的,该现象的具体原因需要进一步探讨。

(5)目前 Rain 机制采用的是固定时间间隔召集分派的方法,在召集的过程中直接略过了很大的参数群,考虑是否可以采用逐步靠近的方法,从而尽可能降低直接略过较优参数组合的可能性。

(6)目前 Rain 机制完全依赖模型中 Dropout 机制的随机性,为了提高 Rain 机制的泛化性,是否可以弥补这一缺陷也值得思考。

参 考 文 献

[1] MITCHELL, TOM M. Machine learning [M]. McGraw-Hill, 1997.

[2] SILVER D, HUANG A, MADDISON C. Mastering the game of go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.

[3] HUANG Y, LIU D Y, HUANG K, et al. Overview on deep learning[J]. CAAI Transactions on Intelligent Systems, 2019, 1(14): 1-19.

[4] QIU Y P. Neural networks and deep learning[M]. Beijing: China Machine Press, 2020.

[5] DAUPHIN Y, PASCANU R, GULCEHRE C, et al. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization[J]. Advances in Neural Information Processing Systems, 2014, 27.

[6] LE Q V, NGIAM J, COATES A, et al. On optimization methods for deep learning[C]// International Conference on Machine Learning. 2011.

[7] RUDER S. An overview of gradient descent optimization algorithms[Z]. 2016.

[8] YOUSOFF S N M, BAHARIN A, ABDULLAH A. A review on optimization algorithm for deep learning method in bioinformatics field[C]// 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES). 2017.

[9] FLETCHER R. Practical methods of optimization[J]. Journal of the Operational Research Society, 2013, 33(7): 675-676.

[10] NOCEDAL J. Updating quasi-newton matrices with limited storage[J]. Mathematics of Computation, 1980, 35(151): 773-782.

[11] GOYAL P, DOLLAR P, GIRSHICK R, et al. Accurate, large minibatch sgd: Training imagenet in 1 hour[J]. arXiv: 1706.02677, 2017.

[12] LOSHCHILOV I, HUTTER F. Sgdr: Stochastic gradient descent with warm restarts[C]// International Joint Conference on Artificial Intelligence. 2016.

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

[14] TIELEMAN T, HINTON G. Lecture 6. 5-rmsprop: Divide the gradient by a running average of its recent magnitude[C]// COURSERA: Neural Networks for Machine Learning. 2012.

[15] ZEILER D M. Adadelta: An adaptive learning rate method[C]// International Joint Conference on Artificial Intelligence. 2012.

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

[17] NESTEROV Y. Gradient methods for minimizing composite functions[J]. Mathematical Programming, 2013, 140(1): 125-161.

[18] SUTSKEVER I, MARTENS J, DAHL G, et al. On the importance of initialization and momentum in deep learning[C]// International Conference on Machine Learning. 2013.

[19] PASCANU R, MIKOLOV T, BENGIO Y. On the difficulty of training recurrent neural networks[C]// Proceedings of the International Conference on Machine Learning. 2013.

[20] KINGMA D, BA J. Adam: A method for stochastic optimization [J]. arXiv: 1412. 6980, 2014.

[21] REDDI S J, KALE S, KUMAR S. On the convergence of adam and beyond[C]// Proceedings of the International Conference on Learning Representations. 2019.

[22] LUO L, XIONG Y, LIU Y, et al. Adaptive gradient methods with dynamic bound of learning rate[C]// Proceedings of the International Conference on Learning Representations. 2019.

[23] ZHANG M R, LUCAS J, HINTON G, et al. Lookahead optimizer: k steps forward, 1 step back[C]// Proceedings of the Neural Information Processing Systems. 2019.

[24] LIU L, JIANG H, HE P, et al. On the variance of the adaptive learning rate and beyond[C]// Proceedings of the International Conference on Learning Representations. 2019.

[25] RUMELHART D E, HINTON G E, WILLIAMS R J. Learning representations by back-propagating errors[J]. Nature, 1986, 323(6088): 533-536.

[26] HINTON G E, SRIVASTAVA N, KRIZHEVSKY A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. Computerence, 2012, 3(4): 212-223.

[27] RENDLE S. Factorization machines [C]// IEEE International Conference on Data Mining. 2011.

[28] SHAN Y, HOENS T R, JIAO J, et al. Deep crossing: Web-scale modeling without manually crafted combinatorial features[C]// 22nd ACMSIGKDD International Conference. 2016.

[29] XIAO J, YE H, HE X, et al. Attentional factorization machines: Learning the weight of feature interactions via attention networks[C]// 26th International Joint Conference on Artificial Intelligence. 2017.

LIU Hua-ling, born in 1964, Ph.D, professor. Her main research interests include privacy protection data mining and Internet financial intelligent monitoring.

PI Chang-peng, born in 1996, postgraduate. His main research interests include machine learning, deep learning and semantic recognition.