Journal of Computer Applications
计算机应用,2020,40(10):2804 - 2810
ISSN 1001-9081
CODEN JYIIDU
2020-10-10
http://www.joca.cn

文章编号:1001-9081(2020)10-2804-07
DOI:10.11772/j.issn.1001-9081.2020020237

基于自适应学习率优化的AdaNet改进

刘　然[1,2,3,4],刘　宇[1,2,3,4*],顾进广[1,2,3,4]

(1. 武汉科技大学 计算机科学与技术学院,武汉 430065;
2. 智能信息处理与实时工业系统湖北省重点实验室(武汉科技大学),武汉 430065;
3. 武汉科技大学 大数据科学与工程研究院,武汉 430065;
4. 国家新闻出版署富媒体数字出版内容组织与知识服务重点实验室(武汉科技大学),北京 100038)
(*通信作者电子邮箱liuyu@wust.edu.cn)

摘　要:人工神经网络的自适应结构学习(AdaNet)是基于Boosting集成学习的神经结构搜索框架,可通过集成子网创建高质量的模型。现有的AdaNet所产生的子网之间的差异性不显著,因而限制了集成学习中泛化误差的降低。在AdaNet设置子网网络权重和集成子网的两个步骤中,使用Adagrad、RMSProp、Adam、RAdam等自适应学习率方法来改进现有AdaNet中的优化算法。改进后的优化算法能够为不同维度参数提供不同程度的学习率缩放,得到更分散的权重分布,以增加AdaNet产生子网的多样性,从而降低集成学习的泛化误差。实验结果表明,在MNIST(Mixed National Institute of Standards and Technology database)、Fashion-MNIST、带高斯噪声的Fashion-MNIST这三个数据集上,改进后的优化算法能提升AdaNet的搜索速度,而且该方法产生的更加多样性的子网能提升集成模型的性能。在$F1$值这一评估模型性能的指标上,改进后的方法相较于原方法,在三种数据集上的最大提升幅度分别为0.28%、1.05%和1.10%。

关键词:AdaNet;神经架构搜索;集成学习;自适应学习率方法;自动机器学习
中图分类号:TP183　　**文献标志码**:A

Improved AdaNet based on adaptive learning rate optimization

LIU Ran[1,2,3,4], LIU Yu[1,2,3,4*], GU Jinguang[1,2,3,4]

(1. *College of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan Hubei* 430065, *China*;
2. *Key Laboratory of Intelligent Information Processing and Real-time Industrial System in Hubei Province (Wuhan University of Science and Technology), Wuhan Hubei* 430065, *China*;
3. *Institute of Big Data Science and Engineering, Wuhan University of Science and Technology, Wuhan Hubei* 430065, *China*;
4. *Key Laboratory of Rich-media Knowledge Organization and Service of Digital Publishing Content, National Press and Publication Administration (Wuhan University of Science and Technology), Beijing* 100038, *China*)

Abstract: AdaNet (Adaptive structural learning of artificial neural Networks) is a neural architecture search framework based on Boosting ensemble learning, which can create high-quality models through integrated subnets. The difference between subnets generated by the existing AdaNet is not significant, which limits the reduction of generalization error in ensemble learning. In the two steps of AdaNet: setting subnet network weights and integrating subnets, Adagrad, RMSProp (Root Mean Square Prop), Adam, RAdam (Rectified Adam) and other adaptive learning rate methods were used to improve the existing optimization algorithms in AdaNet. The improved optimization algorithms were able to provide different degrees of learning rate scaling for different dimensional parameters, resulting in a more dispersed weight distribution, so as to increase the diversity of subnets generated by AdaNet, thereby reducing the generalization error of ensemble learning. The experimental results show that on the three datasets: MNIST (Mixed National Institute of Standards and Technology database), Fashion-MNIST and Fashion-MNIST with Gaussian noise, the improved optimization algorithms can improve the search speed of AdaNet, and more diverse subnets generated by the method can improve the performance of the ensemble model. For the $F1$ value, which is an index to evaluate the model performance, compared with the original method, the improved methods have the largest improvement of 0.28%, 1.05% and 1.10% on the three datasets.

Key words: AdaNet; Neural Architecture Search (NAS); ensemble learning; adaptive learning rate method; Automated Machine Learning (AutoML)

收稿日期:2020-03-05;**修回日期**:2020-05-25;**录用日期**:2020-06-08。　**基金项目**:国家自然科学基金资助项目(U1836118,61673004);教育部新一代信息技术创新项目(2018A03025);国家社会科学基金重大计划项目(11&ZD189)。

作者简介:刘然(1996—),男,湖北天门人,硕士研究生,主要研究方向:机器学习、深度学习、自动机器学习;　刘宇(1980—),男,湖北武汉人,副教授,博士,主要研究方向:知识工程、智能系统、分布式计算;　顾进广(1974—),男,湖北武汉人,教授,博士,CCF会员,主要研究方向:语义Web、新型网格计算、智能信息处理。

0　引言

人工神经网络的自适应结构学习(Adaptive structural learning of artificial neural Networks, AdaNet)是一种基于集成学习的神经结构搜索(Neural Architecture Search, NAS)框架,它不仅可以学习神经网络架构,还可以将最优的架构组合成一个高质量的集成模型。AdaNet借鉴了Boosting集成学习中提出的估计泛化误差上界的公式"泛化误差项< 经验误差项 + 学习算法容量相关项",并且在NAS领域对"学习算法容量相关项"给出了具体的定义[1],这样AdaNet在训练的过程中就能够评估所得模型的泛化误差。

AdaNet是以神经网络作为个体学习器,也就是AdaNet子网,多个子网通过混合权重进行集成,最后得到AdaNet都会评估AdaNet集成体泛化误差上界,找到使得泛化误差最大限度减小的子网,并将其添加到AdaNet集成体中,再根据所添加的这个子网来调整搜索空间。

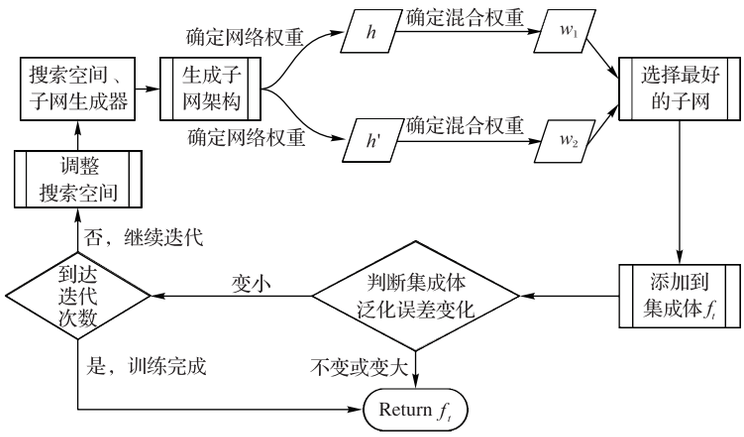

图1　AdaNet子网搜索过程

Fig. 1　AdaNet subnet search process

AdaNet为了找到每轮迭代性能最好的子网,会使用Momentum优化算法来得到子网的网络权重和子网集成时的混合权重。Momentum优化算法是在随机梯度下降(Stochastic Gradient Descent, SGD)优化算法中引入了动量相关概念,并且在整个寻优过程中学习率是固定的。这类优化算法在损失函数为凸函数时,能够保证收敛到全局最优值,缺点是收敛的时间较长。同时,受初始学习率的影响很大,每个维度的学习率一样。因此,AdaNet通过Momentum优化算法得到的子网,虽然能够保证具有较好的性能,但是子网之间差异性并不大。

由于AdaNet是基于集成学习的,而集成学习的关键是产生"好而不同"的子网,即子网之间需存在差异性,并在一定误差率限定下对子网进行集成。AdaNet集成体的泛化误差,可通过的误差分歧分解式子[2]来进行表示:

$$E = \overline{E} - \overline{A} \tag{1}$$

其中:E表示AdaNet集成体的泛化误差;$\overline{E}$表示所有子网泛化误差的均值;$\overline{A}$表示子网的加分分歧项,即所有子网的差异性度量最后取平均。该式子表明$\overline{E}$越小,并且$\overline{A}$越大,集成的效果越好。但是$\overline{A}$仅仅在集成体构造好后才能进行估计,所以不能够直接最小化$\overline{E} - \overline{A}$。

现有的AdaNet处理过程中,每次均从候选子网中选择了最好的子网,其实在训练的过程中已经对$\overline{E}$进行了限制,因此为了进一步提高AdaNet集成体的效果,应从提升子网的差异性入手,也就是从增大$\overline{A}$入手。而AdaNet使用Momentum优化算法得到的集成模型,往往为了追求较小的$\overline{E}$,而导致$\overline{A}$较小,难以减小AdaNet集成体的泛化误差$\overline{E}$。

在另一方面,神经网络中存在着大量非线性变换,它的损失函数几乎都是非凸和高维度的,不可避免存在大量的局部最小值。在实际情况中,往往存在大量梯度为0的点,导致无论用什么优化算法都很难找到唯一全局最小值。Li等[3]通过实验说明了神经网络的训练依赖于寻找高维度非凸损失函数的"较好"极小值能力,这些"较好的"局部极小值表现可能和全局最小值相差无几。同时,根据Kearns等[4]提出的集成学习中"弱学习器等价于强学习器"理论,以及Schapire[5]对这一理论给出的构造性证明,Boosting类算法的个体学习器并不要求强学习器。

针对AdaNet得到子网差异性不大,而导致$\overline{A}$过小的问题,本文改用Adagrad、RMSProp(Root Mean Square Prop)、Adam以及RAdam(Rectifier Adam)等自适应学习率的优化算法,来得到子网内部的网络权重和子网集成时的混合权重。这些算法都计算了梯度累计量(一般用r表示)或者性质类似的值,它是每次迭代梯度的历史累计值,随着迭代的进行r逐渐变大。以Adagrad优化算法流程为例,它使用学习率α除以了$\varepsilon + \sqrt{r}$(ε为极小的正数,用来防止除以0),即用r对学习率α进行缩放,这样在整个迭代过程中,前期放大α,让权重在训练前期有着很大的更新幅度,有更大机会探索到损失函数曲面中有差异的多个"较好"极小值。其他3种自适应学习率的优化算法在这一点上与Adagrad类似,也能达到相同目的。同时,由于不同维度参数的梯度不同,它们的梯度累计量也就不同,导致学习率α的缩放程度也不同,从而自适应地为不同维度参数进行不同程度的学习率缩放[6],最后优化所得的权重分布也更加分散。不同的AdaNet子网在训练时,它们参数之间学习率缩放程度差别更大,更容易产生有差异性的AdaNet子网。本文认为在损失函数曲面中,不去追求全局最小值,而是到达不同"较好"极小值点,虽然子网的性能会弱化,但增大了子网之间的差异性,即增大了$\overline{A}$,最后集成模型的性能反而会更好。

本文在多组实验中比较了不同的优化算法,实验结果表明,在AdaNet中使用RMSProp、Adam、RAdam这类自适应学习率算法来训练子网以及优化混合权重时,不仅因为自适应

学习率，不容易跳出极值点，使得收敛变快，同时由于增大了子网差异性，最后得到AdaNet集成体的性能更佳。

1 相关工作

NAS是自动机器学习（Automated Machine Learning，AutoML）领域热点之一，它可以针对特定的数据集从头开始设计性能良好的模型，在某些任务上甚至可以达到人类专家的设计水准，甚至有可能发现某些人类之前未曾提出的网络结构，所以国内外都对NAS进行了深入研究。

Zoph等[7]将NAS问题转化成了序列生成问题，并且将循环神经网络（Recurrent Neural Network，RNN）作为控制器来对序列进行求解，之后使用强化学习算法近端策略优化（Proximal Policy Optimization，PPO）学习控制器的参数。这种方法每一层的参数都是独立的，需要学习整个网络，参数搜索空间非常大，最终论文的实验动用了800个GPU。之后，Zoph等[8]又根据经典卷积神经网络（Convolutional Neural Network，CNN）每层的网络结构都具有一定相似性，提出自动搜索的网络结构也可以是基于层的，可以针对某一层来设计一种特定的结构，然后将这种结构重复多次，来得到一个完整的网络结构，实验使用了500个GPU并行处理。后续研究中，Liu等[9]根据启发式搜索，只训练有可能得到最优结果的网络结构，这样降低了需要完整训练的网络结构数量，但是搜索空间还是很大。

但是上述几种方法都有着巨大的搜索空间，导致整个搜索过程计算量非常大，必须要有大量的GPU支撑。AdaNet是一种基于强化学习和集成学习的方法，优势在于子网基本架构是可以自定义的，后面产生的子网都是在这个基本架构上演化来的，因此需要完整训练的子网个数并不多。AdaNet通过简单的人工干预，就大幅度减小了NAS搜索空间。同时AdaNet在搜索过程中估计泛化误差上界的机制，也导致了AdaNet中验证子网泛化误差的环节并不复杂。

优化算法在深度学习领域的应用很广泛，最早出现的优化算法是随机梯度下降（Stochastic Gradient Descent，SGD）。也就是随机抽取一批样本，并且以此为根据来更新参数。SGD优化算法的优点是容易得到全局最优解，但是缺点也很明显：一是受初始学习率的影响很大，存在不收敛的可能；二是每个维度的学习率一样，很难学习到稀疏数据上的有用信息；三是学习曲线有时会产生剧烈震荡。

对于SGD优化算法中的学习曲线震荡和鞍点停滞问题，Momentum优化算法[10]可以比较好地缓解这个问题。与SGD优化算法相比，Momentum优化算法收敛更快，收敛曲线也更加稳定，但是Momentum优化算法还是存在每个维度学习率一样的问题。

在实际应用中，更新频率低的参数应该拥有较大的学习率，而更新频率高的参数应该拥有较小的学习率。Adagrad优化算法[11]采用“历史梯度累计平方和的平方根”（常用r表示）来衡量不同参数梯度的稀疏性，并且将r作为学习率的分母，某参数的r越小，则表明该参数越稀疏，该参数的学习率也就越大。

RMSProp优化算法[12]将Adagrad优化算法中的“历史梯度累计平方和的平方根”改为了“历史梯度平均平方和的平方根”，避免了分母随着时间单调递增，解决了训练提前结束的问题。RMSProp优化算法虽然解决了训练过早结束的问题，但是同样容易陷入到局部最优解。

Adam优化算法[13]集成了Momentum优化算法训练比较稳定的优点，以及RMSProp优化算法学习率随着迭代次数发生变化的优点，可以认为是带有动量项的RMSProp。Adam算法通过记录梯度一阶矩，保证了梯度之间不会相差太大，即梯度平滑、稳定地过渡，这样可以适应不稳定的目标函数。同时通过记录梯度二阶矩，保证了环境感知能力，为不同参数产生自适应的学习率。Adam优化算法可以实现快速收敛，但是它对初始学习率不够鲁棒，也会存在收敛到局部最优解的问题。并且Adam优化算法在最初几次迭代，常常需要用很小的学习率，也就是“预热”，否则可能会陷入糟糕的开始，使得训练曲线出现过度跳跃，变得更长、更困难。

近期，最优化算法领域也有许多新的研究成果，来自UIUC（University of Illinois at Urbana-Champaign）的Liu等[14]提出了一个新的优化算法RAdam，它在保证收敛较快的前提下，建立了一个“整流器（rectifier）项”来进行自动预热，在大部分预热长度和学习率的情况下都优于手动预热。RAdam提供了更好的训练稳定性，省去了手动预热的过程，在大部分人工智能（Artificial Intelligence，AI）领域都有较好的通用性。

2 AdaNet算法

2.1 AdaNet集成体

AdaNet的首要对象是集成体，每次搜索得到的子网，都会作为集成体的一部分。集成体由一个或多个子网组成，这些子网的输出通过集成器进行组合。

AdaNet中的集成模型e可以表示为：

$$e = \sum_{k=1}^{l} \boldsymbol{w}_k h_k \tag{2}$$

其中：集成模型e是所有子模型的加权和；l是集成体中的子网个数；$\boldsymbol{w}_k$代表第k个子网的混合权重；h_k代表第k个子网。

2.2 AdaNet目标函数和子网搜索过程

AdaNet在搜索与训练的过程中，需要有一个指标来评价整个集成模型的好坏，通过最小化这个指标，来寻找子网进行集成时的最优混合权重。AdaNet使用如下目标函数作为指标：

$$E(\boldsymbol{w}) = \frac{1}{m}\sum_{i=1}^{m}\phi\left(1 - y_i\sum_{j=1}^{N}\boldsymbol{w}_j h_j(x_i)\right) \tag{3}$$

$$R(\boldsymbol{w}) = \sum_{j=1}^{N}(\lambda r(h_j) + \beta)\left\|\boldsymbol{w}_j\right\|_1 \tag{4}$$

$$F(\boldsymbol{w}) = E(\boldsymbol{w}) + R(\boldsymbol{w}) \tag{5}$$

其中：$\lambda \geqslant 0$，$\beta \geqslant 0$，均为常数；m是训练集的样本个数；N是集成体中子网的个数；h_j是序号为j的子网；$\boldsymbol{w}_j$是序号为j的子网的混合权重；ϕ是子网训练时所用的损失函数（在AdaNet中常常使用指数损失函数，即$\phi(x) = \mathrm{e}^x$）；$r(h_j)$是序号为j的子网的Rademacher复杂度。

$F(\boldsymbol{w})$借鉴了Boosting算法中估计泛化误差的方式，它是$E(\boldsymbol{w})$和$R(\boldsymbol{w})$两者的和，$E(\boldsymbol{w})$为AdaNet集成体的训练误差项，$R(\boldsymbol{w})$为AdaNet集成体的复杂度惩罚项，$F(\boldsymbol{w})$则是AdaNet集成体泛化误差上界的估计。上述标准可用于判定候选子网能否添加至AdaNet集成体中：当尝试添加某个候选子网后，训练误差项的减小幅度大于复杂度惩罚项的增大幅度，即$F(\boldsymbol{w})$减小时，该候选子网会被正式添加到AdaNet集成体。

假设在$t-1$轮迭代之后得到的AdaNet模型为e_{t-1},当前混合权重向量为$\boldsymbol{w}_{t-1}$,集成模型的深度为l_{t-1}。对于第t轮迭代,子网生成器会得到两个子网h、h',其中h是深度为l_{t-1}的子网,h'是深度为$l_{t-1}+1$的子网,子网的具体形式由自定义的子网搜索空间指定。

此时,将第t轮迭代的子网添加到集成体的问题,可以转化为在候选子网$u\in\{h,h'\}$,并且候选子网u的混合权重$\boldsymbol{w}\in\mathbf{R}$的条件下,最小化第$t$-1轮迭代的目标函数$F_t(\boldsymbol{w},u)$的问题,$F_t(\boldsymbol{w},u)$的具体公式如下:

$$F_t(\boldsymbol{w},u)=E_t(\boldsymbol{w},u)+R_t(\boldsymbol{w},u) \tag{6}$$

$$E_t(\boldsymbol{w},u)=\frac{1}{m}\sum_{i=1}^{m}\phi(1-y_i f_{t-1}(x_i)-y_i\boldsymbol{w}u(x_i)) \tag{7}$$

$$R_t(\boldsymbol{w},u)=(\lambda r(u)+\beta)\|\boldsymbol{w}\|_1 \tag{8}$$

而在使用$F_t(\boldsymbol{w},u)$之前,需要先对子网进行局部的训练,即得到子网u内部的网络权重$\boldsymbol{w}_u$,这样的子网u才可以添加到集成体中,此时需要最小化子网u的目标函数$F_t(\boldsymbol{w},u)$,具体公式如下:

$$F_u(\boldsymbol{w}_u,u)=\frac{1}{m}\sum_{i=1}^{m}\phi(1-y_i u(x_i,\boldsymbol{w}_u)) \tag{9}$$

$F(\boldsymbol{w})$用来比较前后两次迭代的集成体性能,判断集成体是否需要添加当前子网。$F_t(\boldsymbol{w},u)$用来代入优化算法中,最小化$F_t(\boldsymbol{w},u)$的过程就是确定当前子网混合权重$\boldsymbol{w}$的过程。$F_u(\boldsymbol{w}_u,u)$用来对子网进行局部的训练,最小化$F_u(\boldsymbol{w}_u,u)$的过程就是确定当前子网内部网络权重$\boldsymbol{w}_u$的过程(最小化$F_t(\boldsymbol{w},u)$和$F_u(\boldsymbol{w}_u,u)$之前需要先确定u为某个固定的候选子网h或h')。根据式(5)、(6)、(9),AdaNet子网搜索流程可以具体表述为:

输入　训练集S;

输出　AdaNet集成体e_T。

AdaNet(S):

1) $e_0\leftarrow 0$,
   Initialize Search Space and Generator
2) for t $\leftarrow$ 1 to T do
3) 　$h,h'\leftarrow$ Generator(S,e_{t-1})
   　Get the subnet network weight $\boldsymbol{w}_h$ and $\boldsymbol{w}_{h'}$
   　by $\boldsymbol{w}_h\leftarrow$ MINIMIZE($F_u(\boldsymbol{w}_u,h)$)
   　$\boldsymbol{w}_{h'}\leftarrow$ MINIMIZE($F_u(\boldsymbol{w}_u,h')$)
4) 　$\boldsymbol{w}_1\leftarrow$ MINIMIZE($F_t(\boldsymbol{w},h)$)
5) 　$\boldsymbol{w}_2\leftarrow$ MINIMIZE($F_t(\boldsymbol{w},h')$)
6) 　if $F_t(\boldsymbol{w}_1,h)\leqslant F_t(\boldsymbol{w}_2,h')$ then
7) 　　$h_t\leftarrow h,\boldsymbol{w}^*\leftarrow\boldsymbol{w}_1$
8) 　else
9) 　　$h_t\leftarrow h',\boldsymbol{w}^*\leftarrow\boldsymbol{w}_2$
10) 　if $F(\boldsymbol{w}_{t-1}+\boldsymbol{w}^*)<F(\boldsymbol{w}_{t-1})$ then
11) 　　$e_t\leftarrow e_{t-1}+\boldsymbol{w}^*\cdot h_t$
12) 　　Use h_t to adjust Search Space
13) 　else
14) 　　return e_{t-1}
15) return e_t

第1)行:初始化AdaNet集成体子网搜索空间、子网生成器,子网搜索空间定义了生成子网的基本类型,子网生成器用来生成具体的子网,并且对子网进行训练(本文主要讨论子网为基本CNN架构的情况)。

第3)行:子网生成器生成两个候选子网h、h'的网络架构,将$u=h$、$u=h'$分别代入到$F_u(\boldsymbol{w}_u,u)$中,之后用优化算法来得到h、h'内部的网络权重,用MINIMIZE()来指代某种优化算法,每种优化算法的具体步骤将在后文介绍。

第4)~5)行:将$u=h$、$u=h'$分别代入到$F_t(\boldsymbol{w},u)$中,并且用优化算法MINIMIZE()得到h、h'的最佳混合权重$\boldsymbol{w}_1$、$\boldsymbol{w}_2$。

第6)~9)行:比较$F_t(\boldsymbol{w}_1,h)$、$F_t(\boldsymbol{w}_2,h')$,h、h'中使得$F_t(\boldsymbol{w},u)$更小的子网表示为h_t,对应的混合权重称为$\boldsymbol{w}^*$。

第10)~14)行:比较$F(\boldsymbol{w}_{t-1}+\boldsymbol{w}^*)$、$F(\boldsymbol{w}_{t-1})$,如果$F(\boldsymbol{w}_{t-1}+\boldsymbol{w}^*)<F(\boldsymbol{w}_{t-1})$,则认为引入子网$h_t$后集成模型训练误差的减小程度要优于复杂度的增大程度,也就是说引入子网h_t能够减小集成体的泛化误差上界。此时上一次迭代的集成体e_{t-1}应该包含这个子网h_t,剥离子网h_t的softmax层以暴露最后一个隐藏层,然后通过混合权重$\boldsymbol{w}^*$将h_t最后一个隐藏层连接到集成体的输出,得到新的集成体e_t,并且根据当前迭代获得的反馈信息调整子网搜索空间。如果$F(\boldsymbol{w}_{t-1}+\boldsymbol{w}^*)\geqslant F(\boldsymbol{w}_{t-1})$,则认为继续引入子网对于减小集成体的泛化误差上界没有帮助,此时直接返回上一次迭代的集成体e_{t-1}。

第15)行:完整进行T轮迭代后得到集成体e_T,最后返回它。

2.3　AdaNet的权重优化算法

AdaNet子网搜索流程中第3)行的使用优化算法最小化$F_u(\boldsymbol{w}_u,u)$得到子网内部的网络权重$\boldsymbol{w}_u$,以及第4)~5)行的使用优化算法最小化$F_t(\boldsymbol{w},u)$得到子网集成时的混合权重$\boldsymbol{w}$,在AdaNet中都是使用同一种优化算法来求解的。在AdaNet中为了得到全局最优的子网,常常都会使用Momentum优化算法来对上述两步进行求解,也就是在u设置为定值的情况下,分别将$F_u(\boldsymbol{w}_u,u)$、$F_t(\boldsymbol{w},u)$作为Momentum优化算法目标函数$f_t(\boldsymbol{w})$,分别对$\boldsymbol{w}_u$、$\boldsymbol{w}$进行寻优,Momentum优化算法的具体流程如下。

输入　学习率α,初始参数$\boldsymbol{w}_0$,初始速率$\boldsymbol{v}_0$,动量衰减参数γ,优化算法目标函数$f_t(\boldsymbol{w})$(γ常常设为0.9)。

输出　结果参数$\boldsymbol{w}_T$。

Momentum($\alpha,\boldsymbol{w}_0,\boldsymbol{v}_0,\gamma,f_t(\boldsymbol{w})$):

1) while $t=\{1,2,\cdots,T\}$ do
2) 　计算梯度:$\boldsymbol{g}_t\leftarrow\Delta_{\boldsymbol{w}}f_t(\boldsymbol{w}_{t-1})$
3) 　计算速度:$\boldsymbol{v}_t\leftarrow\gamma\cdot\boldsymbol{v}_{t-1}+\alpha\cdot\boldsymbol{g}_t$
4) 　参数更新:$\boldsymbol{w}_t\leftarrow\boldsymbol{w}_{t-1}-\boldsymbol{v}_t$
5) return $\boldsymbol{w}_T$

但是Momentum优化算法收敛较慢,导致AdaNet的整个子网搜索过程较慢;同时使用Momentum优化算法时,由于在整个训练过程中学习率都是固定的,所以每个维度的学习率都是一样的,难以学习到稀疏数据上的有用信息;并且Momentum优化算法得到的AdaNet集成体中子网的差异性较小,这样不利于AdaNet集成体泛化误差的减小。

3　自适应学习率的优化算法

本文为了增大不同子网内部的网络权重之间的差异性,改用自适应学习率的优化算法来对上述两步进行求解,具体使用了Adagrad、RMSProp、Adam、RAdam这4种优化算法,下面给出了它们的完整流程。

3.1　Adagrad优化算法

输入　学习率α,初始参数$\boldsymbol{w}_0$,数值稳定量ε,优化算法目标函数$f_t(\boldsymbol{w})$。

中间变量　梯度累计量$\boldsymbol{r}$。

输出　结果参数$\boldsymbol{w}_T$。

AdaGrad$(\alpha, \boldsymbol{w}_0, \varepsilon, f_t(\boldsymbol{w}))$:

1）梯度累计量初始化:$\boldsymbol{r} \leftarrow 0$

2）while $t = \{1, 2, \cdots, T\}$ do

3）　计算梯度:$\boldsymbol{g}_t \leftarrow \Delta_{\boldsymbol{w}} f_t(\boldsymbol{w}_{t-1})$

4）　计算梯度累计量:$\boldsymbol{r} \leftarrow \boldsymbol{r} + \boldsymbol{g}_t \odot \boldsymbol{g}_t$

5）　计算参数更新量:$\Delta \boldsymbol{w} \leftarrow -\dfrac{\alpha}{\varepsilon + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}_t$

6）　参数更新:$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} + \Delta \boldsymbol{w}$

7）return $\boldsymbol{w}_T$

3.2　RMSProp 优化算法

输入　学习率α,初始参数$\boldsymbol{w}_0$,数值稳定量ε,衰减速率ρ,优化算法目标函数$f_t(\boldsymbol{w})$。

中间变量　梯度累计量$\boldsymbol{r}$。

输出　结果参数$\boldsymbol{w}_T$。

RMSProp$(\alpha, \boldsymbol{w}_0, \varepsilon, \rho, f_t(\boldsymbol{w}))$:

1）梯度累计量初始化:$\boldsymbol{r} \leftarrow 0$

2）while $t = \{1, 2, \cdots, T\}$ do

3）　计算梯度:$\boldsymbol{g}_t \leftarrow \Delta_{\boldsymbol{w}} f_t(\boldsymbol{w}_{t-1})$

4）　计算梯度累计量:$\boldsymbol{r} \leftarrow \rho \cdot \boldsymbol{r} + (1-\rho) \cdot \boldsymbol{g}_t \odot \boldsymbol{g}_t$

5）　计算参数更新量:$\Delta \boldsymbol{w} \leftarrow -\dfrac{\alpha}{\varepsilon + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}_t$

6）　参数更新:$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} + \Delta \boldsymbol{w}$

7）return $\boldsymbol{w}_T$

3.3　Adam 优化算法

输入　学习率α,初始参数$\boldsymbol{w}_0$,数值稳定量ε,梯度一阶矩衰减系数β_1,梯度二阶矩衰减系数β_2,优化算法目标函数$f_t(\boldsymbol{w})$。

中间变量 梯度一阶矩$\boldsymbol{m}_t$,梯度二阶矩$\boldsymbol{v}_t$。

输出　结果参数$\boldsymbol{w}_T$。

Adam$(\alpha, \boldsymbol{w}_0, \varepsilon, \beta_1, \beta_2, f_t(\boldsymbol{w}))$:

1）梯度一阶矩初始化:$\boldsymbol{m}_0 \leftarrow 0$

2）梯度二阶矩初始化:$\boldsymbol{v}_0 \leftarrow 0$

3）while $t = \{1, 2, \cdots, T\}$ do

4）　计算梯度:$\boldsymbol{g}_t \leftarrow \Delta_{\boldsymbol{w}} f_t(\boldsymbol{w}_{t-1})$

5）　计算梯度一阶矩(Momentum项):
$\boldsymbol{m}_t \leftarrow \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1) \cdot \boldsymbol{g}_t$

6）　计算梯度二阶矩(RMSProp项):
$\boldsymbol{v}_t \leftarrow \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \cdot \boldsymbol{g}_t^2$

7）　修正一阶矩的偏差:$\hat{\boldsymbol{m}}_t \leftarrow \dfrac{\boldsymbol{m}_t}{1-\beta_2^t}$

8）　修正二阶矩的偏差:$\hat{\boldsymbol{v}}_t \leftarrow \dfrac{\boldsymbol{v}_t}{1-\beta_2^t}$

9）　计算参数更新量:$\Delta \boldsymbol{w} \leftarrow -\dfrac{\alpha \cdot \hat{\boldsymbol{m}}_t}{\varepsilon + \sqrt{\hat{\boldsymbol{v}}_t}} \odot \boldsymbol{g}_t$

10）　参数更新:$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} + \Delta \boldsymbol{w}$

11）return $\boldsymbol{w}_T$

3.4　RAdam 优化算法

输入　学习率α,初始参数$\boldsymbol{w}_0$,数值稳定量ε,梯度一阶矩衰减系数β_1,梯度二阶矩衰减系数β_2,优化算法目标函数$f_t(\boldsymbol{w})$。

中间变量　梯度一阶矩$\boldsymbol{m}_t$,梯度二阶矩$\boldsymbol{v}_t$。

输出　结果参数$\boldsymbol{w}_T$。

RAdam$(\alpha, \boldsymbol{w}_0, \varepsilon, \beta_1, \beta_2, f_t(\boldsymbol{w}))$:

1）梯度一阶矩初始化:$\boldsymbol{m}_0 \leftarrow 0$

2）梯度二阶矩初始化:$\boldsymbol{v}_0 \leftarrow 0$

3）SMA(Simple Moving Average)最大值初始化:

$\rho_\infty \leftarrow \dfrac{2}{1-\beta_2} - 1$

4）while $t = \{1, 2, \cdots, T\}$ do

5）　计算梯度:$\boldsymbol{g}_t \leftarrow \Delta_{\boldsymbol{w}} f_t(\boldsymbol{w}_{t-1})$

6）　计算梯度二阶矩:
$\boldsymbol{v}_t \leftarrow \beta_2 \cdot \boldsymbol{v}_{t-1} + (1-\beta_2) \cdot \boldsymbol{g}_t^2$

7）　计算梯度一阶矩(Momentum项):
$\boldsymbol{m}_t \leftarrow \beta_1 \cdot \boldsymbol{m}_{t-1} + (1-\beta_1) \cdot \boldsymbol{g}_t$

8）　修正一阶矩的偏差:$\hat{\boldsymbol{m}}_t \leftarrow \dfrac{\boldsymbol{m}_t}{1-\beta_2^t}$

9）　计算SMA最大值:$\rho_t \leftarrow \rho_\infty - \dfrac{2t\beta_2^t}{(1-\beta_2^t)}$

10）　if $\rho_t > 4$ then

11）　　修正二阶矩的偏差:$\hat{\boldsymbol{v}}_t \leftarrow \dfrac{\boldsymbol{v}_t}{1-\beta_2^t}$

12）　　计算整流器项:
$r_t \leftarrow \sqrt{\dfrac{(\rho_t - 4)(\rho_t - 2)\rho_\infty}{(\rho_\infty - 4)(\rho_\infty - 2)\rho_t}}$

13）　　使用自适应动量进行参数更新:
$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} - \dfrac{\alpha_t r_t \hat{\boldsymbol{m}}_t}{\hat{\boldsymbol{v}}_t}$

14）　else

15）　　使用非自适应动量进行参数更新:
$\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} - \alpha_t \hat{\boldsymbol{m}}_t$

16）return $\boldsymbol{w}_T$

4　实验与评估

本文使用了上述几种方法,在AdaNet的子网搜索空间设置为基本CNN的前提下,优化子网内部的网络权重和子网集成时的混合权重,并且在标准测评数据集上测试了上述方法建立AdaNet集成模型的性能,以说明使用RMSProp、Adam、RAdam这类优化算法时,相较于AdaNet常用的Momentum优化算法,AdaNet的整个训练过程不仅能够快速收敛,并且得到的AdaNet集成模型效果也更好。

4.1　数据集

MNIST是美国国家标准与技术研究院收集整理的大型手写数字数据库,包含60 000个训练集样本以及10 000个测试集样本,样本是28×28的图片,共有10个类别。这个数据集较为简单,AdaNet原始CNN模型就能获得较好效果,对于这个数据集的分类,目的是为了检验本文观点,即能否让AdaNet集成体学习到一些比较难学到的内容。

Fashion-MNIST是一个替代MNIST手写数字集的图像数据集。Fashion-MNIST的格式、大小和训练集/测试集划分与MNIST完全一致,主要是增加了分类的难度。对这个数据集的分类更加具有挑战,目的是为了检验本文观点在更复杂的分类问题上是否同样有效。

为了检验通过本文方法所得到的AdaNet集成体,能否保持集成学习类算法能避免噪声的特性。本文又在添加了高斯噪声后的Fashion-MNIST上进行了对比实验,具体添加方法为:源数据在分批训练时,随机选取20批次的数据,为这些数据中的所有图像都引入高斯噪声。

4.2　输入参数设置

对于不同的优化算法,需要给定对应的输入参数。首先

是全部优化算法共有的两个输入参数,即初始参数$\boldsymbol{w}_0$和学习率α。初始参数$\boldsymbol{w}_0$决定了在梯度下降过程中出现梯度消失或者梯度爆炸的可能性大小,本文使用He initialization[15]这种初始化方法来给定,这种方法不仅尽可能保持输入和输出服从相同分布,避免出现激活函数输出值趋向于0的情况,而且它还能很好适应ReLU(Rectified Linear Unit)这种非线性激活函数。学习率α决定了权重更新比率,它需要根据训练集的大小来选择,一般训练集数据量越大,α就应该设在更小的值上[16]。如果α设得太小,训练速度会很慢,但可能将目标函数值降到最低,如果α设得太大,训练速度会很快,但可能会导致学习震荡,甚至可能无法收敛[17]。本文尝试了α为1、0.5、0.1、0.05、0.01、0.005、0.001、0.000 5这些情况,经过前期测试,当α为0.005时,既能够保证训练速度,又能够得到一个很低的目标函数值。同时,由于Momentum优化算法在整个寻优过程中都使用唯一的α,如果α选择不当可能出现非常差的情况。而Adagrad等自适应学习率优化算法每次迭代都对α进行不同程度的缩放,前期距离寻优目标远,对α进行放大,能够更有效率地进行探索,后期距离寻优目标近,对α进行缩小,保证能够收敛到一个较好的结果,所以Adagrad等自适应学习率优化算法对α的要求比Momentum优化算法更低。由于本文3组对比实验的训练集数据量大致相同,所以本文所有实验都基于相同的$\boldsymbol{w}_0$和α来进行。

然后是每个优化算法特有的输入参数,本文根据Ruder[18]以及仝卫国等[19]的研究进行相关设置。对于Momentum优化算法,动量衰减参数γ取0.9,它类似物理中的摩擦系数,用来抑制动量,在接近寻优目标后能快速收敛。对于Adagrad优化算法,数值稳定量ε取10^{-8},ε用一个极小的正数,防止计算过程出现除以0,其他3种自适应学习率优化算法RMSProp、Adam、RAdam中的ε也取相同值。对于RMSProp优化算法,衰减速率ρ取0.9,它用来实现只累积近期梯度信息,用平均平方梯度替换累计平方梯度,避免梯度累计量过大。对于Adam优化算法和RAdam优化算法,梯度一阶矩衰减系数β_1取0.9,梯度二阶矩衰减系数β_2取0.999,梯度一阶矩是历史梯度与当前梯度的平均,它借鉴于Momentum用来保持惯性,实现梯度平滑、稳定的过渡,梯度二阶矩是历史梯度平方与当前梯度平方的平均,它借鉴于RMSProp用来环境感知,保留能为不同维度参数自适应地缩放学习率的优点。

4.3　评价标准

在对比实验中,本文采用在测试集上的正确率(*precision*)、召回率(*recall*)、*F*1值(*F*1_*score*)作为集成模型性能的评价指标,定义如下:

$$precision = \frac{TP}{TP + FP} \tag{10}$$

$$recall = \frac{TP}{TP + FN} \tag{11}$$

$$F1\_score = \frac{2*precision*recall}{precision + recall} \tag{12}$$

而对于子网之间的差异性,由于AdaNet中各个子网架构都是相同的,主要差别在于子网内部的网络权重。本文先计算了每个子网网络权重均值,并将两子网网络权重均值之差取平方作为它们之间的“差异性距离”,之后让AdaNet集成体中所有子网两两组合,对所有组合结果的“差异性距离”求和,最后对这个和取均值作为子网分歧项。具体定义如下:

$$dist(\bar{\boldsymbol{w}}_a, \bar{\boldsymbol{w}}_b) = (\bar{\boldsymbol{w}}_a - \bar{\boldsymbol{w}}_b)^2 \tag{13}$$

$$\overline{A} = \frac{\sum_{a=1}^{n-1}\sum_{b=a+1}^{n} dist(\bar{\boldsymbol{w}}_a, \bar{\boldsymbol{w}}_b)}{C_n^2} \tag{14}$$

其中:$\bar{\boldsymbol{w}}_a$是序号为a的子网网络权重均值;$\bar{\boldsymbol{w}}_b$是序号为b的子网网络权重均值;$dist(\bar{\boldsymbol{w}}_a, \bar{\boldsymbol{w}}_b)$是两子网的“差异性距离”;$n$是AdaNet集成体中子网的个数。

同时为了比较AdaNet在使用不同优化算法时的网络架构搜索速度,本文对所有实验的训练过程所用时间进行了统计(本文在8块NVIDIA Tesla V100-SXM2-16GB的GPU分布式计算环境中进行实验,并且用TensorBoard对整个训练过程进行了监控)。

4.4　实验分析

表1展示了5种优化算法在MNIST数据集上的结果,可以看到*F*1值最高的是RAdam,同时它有着最大的子网分歧项。在这个较简单数据集上RMSProp、Adam、RAdam这3种优化算法效果普遍更好,即它们能让AdaNet集成体学习到一部分难以学到的内容。

表1　在MNIST数据集上不同优化算法的结果

Tab. 1　Results of different optimization algorithms on MNIST dataset

优化算法	正确率/%	召回率/%	*F*1值/%	子网分歧项	时间/s
Momentum	98.35	98.33	98.34	0.102 1	1 417.341 6
Adagrad	96.50	96.49	96.49	0.121 8	**1 388.558 1**
RMSProp	98.53	98.51	98.52	0.125 0	1 428.087 2
Adam	98.51	98.51	98.51	0.125 3	1 439.572 8
RAdam	**98.63**	**98.62**	**98.62**	**0.125 5**	1 393.153 1

表2则展示了在Fashion-MNIST数据集上的结果,可以看到*F*1值最高的是RAdam,它同样有着最大的子网分歧项。在Fashion-MNIST数据集上RMSProp、Adam、RAdam这3种优化算法也得到了更好的效果,即它们在使用AdaNet处理更复杂的分类任务时也是有帮助的。

表2　在Fashion-MNIST数据集上不同优化算法的结果

Tab. 2　Results of different optimization algorithms on Fashion-MNIST dataset

优化算法	正确率/%	召回率/%	*F*1值/%	子网分歧项	时间/s
Momentum	89.13	89.18	89.14	0.032 1	1 483.285 7
Adagrad	87.46	87.54	87.48	0.113 0	**1 395.914 0**
RMSProp	90.01	90.01	90.00	0.116 2	1 398.880 2
Adam	90.11	90.13	90.11	0.166 4	1 426.782 6
RAdam	**90.18**	**90.22**	**90.19**	**0.167 1**	1 439.837 0

表3则展示了在添加高斯噪声的Fashion-MNIST数据集上的结果,*F*1值最高的都是Adam,它同样有着最大的子网分歧项。可以看到RMSProp、Adam、RAdam在数据集中添加了噪声后,损失函数均值和*F*1值的影响比较小,即它们能够帮助AdaNet集成体在一定程度上避免噪声。

3组对比实验所用数据集的复杂程度是依次递增的,它们对应的损失函数曲面复杂程度也是依次递增的。可以看到损失函数曲面越复杂,“较好的”局部极小值点越多,不同优化算法得到的子网分歧项差别越大,即对于复杂问题,本文的改进方法可以获得更好的效果。

在收敛方面，本文的3组实验都是Adagrad优化算法收敛最快，但是Adagrad优化算法所得到结果的$F1$值普遍不高，这是因为Adagrad流程中式子的分母随着时间单调递增。当分母积累值过大后，收敛变得很小，导致训练后期变化幅度很小。对于SGD和Momentum，随机的特性导致了其收敛本身就不快，它们所得$F1$值也不高。对于RMSProp、Adam、RAdam则可以在保证较快收敛的前提下，同时得到较高的$F1$值。

表3 在添加了高斯噪声的Fashion-MNIST数据集上不同优化算法的结果

Tab. 3 Results of different optimization algorithms on Fashion-MNIST dataset with Gaussian noise

优化算法	正确率/%	召回率/%	$F1$值/%	子网分歧项	时间/s
Momentum	88.73	88.79	88.75	0.392 1	1 477.516 2
Adagrad	86.99	87.07	87.00	0.414 8	**1 368.305 1**
RMSProp	89.53	89.59	89.54	0.597 8	1 418.823 6
Adam	**89.84**	**89.87**	**89.85**	**0.646 4**	1 412.049 9
RAdam	89.69	89.76	89.72	0.636 5	1 378.447 7

5 结语

本文基于AdaNet这种NAS方法，引入RMSProp、Adam、RAdam等自适应学习率优化算法来得到子网内部的网络权重和子网集成时的混合权重。实验结果表明，相对于现有的Momentum优化算法，本文方法在保证收敛较快，也就是网络搜索效率较高的前提下，还能够通过提高子网之间的差异性，使得AdaNet集成模型拥有更高的准确率，在越复杂的分类任务中优化效果越明显。

本文的方法在AdaNet子网为简单CNN架构时效果较好，在未来的工作中，将探讨AdaNet子网为各种复杂CNN架构以及RNN架构时，如何将它们组合为更好的AdaNet集成模型。

参考文献（References）

[1] FREUND Y, SCHAPIRE R E. A decision-theoretic generalization of on-line learning and an application to boosting[J]. Journal of Computer and System Sciences, 1997, 55(1): 119-139.

[2] KROGH A, VEDELSBY J. Neural network ensembles, cross validation, and active learning [C]// Proceedings of the 7th International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 1994: 231-238.

[3] LI H, XU Z, TAYLOR G, et al. Visualizing the loss landscape of neural nets [C]// Proceedings of the 2018 Advances in Neural Information Processing Systems, 2018: 6389-6399.

[4] KEARNS M, VALIANT L. Cryptographic limitations on learning Boolean formulae and finite automata[J]. Journal of the ACM, 1994, 41(1): 67-95.

[5] SCHAPIRE R E. The strength of weak learnability[J]. Machine Learning, 1990, 5(2): 197-227.

[6] 金海东，刘全，陈冬火. 一种带自适应学习率的综合随机梯度下降Q-学习方法[J]. 计算机学报, 2019, 42(10): 2203-2215. (JIN H D, LIU Q, CHEN D H. Adaptive learning-rate on integrated stochastic gradient decreasing Q-learning[J]. Chinese Journal of Computers, 2019, 42(10): 2203-2215.)

[7] ZOPH B, LE Q V. Neural architecture search with reinforcement learning[EB/OL]. [2020-02-25]. https://arxiv.org/pdf/1611.01578.pdf.

[8] ZOPH B, VASUDEVAN V, SHLENS J, et al. Learning transferable architectures for scalable image recognition [C]// Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Piscataway: IEEE, 2018: 8697-8710.

[9] LIU C, ZOPH B, NEUMANN M, et al. Progressive neural architecture search [C]// Proceedings of the 2018 European Conference on Computer Vision, LNCS 11205. Cham: Springer, 2018: 19-35.

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

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

[12] MCMAHAN B, STREETER M. Delay-tolerant algorithms for asynchronous distributed online learning [C]// Proceedings of the 7th International Conference on Neural Information Processing Systems. Cambridge: MIT Press, 2014: 2915-2923.

[13] KINGMA D P, BA J. Adam: a method for stochastic optimization[EB/OL]. [2020-02-25]. https://arxiv.org/pdf/1412.6980.pdf.

[14] LIU L, JIANG H, HE P, et al. On the variance of the adaptive learning rate and beyond[EB/OL]. [2020-02-25]. https://arxiv.org/pdf/1908.03265.pdf.

[15] HE K, ZHANG X, REN S, et al. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification [C]// Proceedings of the 2015 IEEE International Conference on Computer Vision. Piscataway: IEEE, 2015: 1026-1034.

[16] WILSON D R, MARTINEZ T R. The general inefficiency of batch training for gradient descent learning[J]. Neural Networks, 2003, 16(10): 1429-1451.

[17] 朱振国，田松禄. 基于权值变化的BP神经网络自适应学习率改进研究[J]. 计算机系统应用, 2018, 27(7): 205-210. (ZHU Z G, TIAN S L. Improvement of learning rate of feed forward neural network based on weight gradient [J]. Computer Systems and Applications, 2018, 27(7): 205-210.)

[18] RUDER S. An overview of gradient descent optimization algorithms [EB/OL]. [2020-02-25]. https://arxiv.org/pdf/1609.04747.pdf.

[19] 仝卫国，李敏霞，张一可. 深度学习优化算法研究[J]. 计算机科学, 2018, 45(11A): 155-159. (TONG W G, LI M X, ZHANG Y K. Research on optimization algorithm of deep learning [J]. Computer Science, 2018, 45(11A): 155-159.)

This work is partially supported by the National Natural Science Foundation of China (U1836118, 61673004), the New Generation Information Technology Innovation Project of the Ministry of Education (2018A03025), the Major Plan of the National Social Science Foundation of China (11&ZD189).

LIU Ran, born in 1996, M. S. candidate. His research interests include machine learning, deep learning, automated machine learning.

YU Liu, born in 1980, Ph. D, associate professor. His research interests include knowledge engineering, intelligent systems, distributed computing.

GU Jinguang, born in 1974, Ph. D, professor. His research interests include semantic Web, new grid computing, intelligent information processing.