文章编号：2095-2163(2019)02-0057-06 中图分类号：TP18 文献标志码：A

基于多个在线核极限学习机的并行模型训练算法

沈哲钧，凌志扬
（东华大学 计算机科学与技术学院，上海 201620）

摘 要：针对传统数据处理技术对多个极限学习机的训练问题，由于串行数据处理方式中，会造成时间复杂度的增加，根据大数据的特征以及数据处理技术，提出基于 MapReduce 的多个在线核极限学习机模型的并行算法，可使一个 MapReduce 作业完成多个模型的预测。通过算例测试验证了本文的基于 MapReduce 集成算法可以有效地提高模型的准确度和实时的训练速度。

关键词：在线极限学习机；集成学习；MapReduce；并行算法

Parallel model training algorithm based on multiple online kernel Extreme Learning Machines

SHEN Zhejun, LING Zhiyang
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

【Abstract】 Aiming at the training problem of traditional data processing technology for multiple learning machines, due to the increase of time complexity in serial data processing, according to the characteristics of big data and data processing technology, multiple online cores based on MapReduce are proposed. The parallel algorithm of the Extreme Learning Machine model allows a MapReduce job to perform predictions for multiple models. The example test proves that the MapReduce integration algorithm in this paper can effectively improve the accuracy of the model and the real-time training speed.

【Key words】 online Extreme Learning Machine; ensemble learning; MapReduce; parallel algorithm

0 引 言

随着机器学习、深度学习理论的不断发展，现代预测技术中，主要包括人工神经网络、支持向量机以及极限学习机等模型作为代表的预测方法在传统平台上都已经有着广泛的研究实践与应用。

传统神经网络方法一定程度上改善了基于时间序列预测模型的很多不足，但是容易出现局部最小、过度的信号迭代、过多的参数设定以及较慢的学习速度导致的较长模型训练时间等问题。极限学习机模型的优势在于，神经网络模型比较简单，而且可以灵活地处理非线性信息序列的问题，通过基于影响因素的非线性映射来预测数据[1]，有效降低了网络参数计算时间，在保证预测准确率的基础上提高了模型训练时间以及表现出较好的模型泛化性能的条件下，在各个领域的应用前景正日趋广阔[2]。

同时，针对极限学习机方法的各种改进研究也越来越多。文献[3]将小波分解与核极限学习机相结合克服 ELM 中存在的过拟合等缺陷；文献[4]采用 Cholesky 分解将核极限学习机（ KELM）从离线模式扩展到在线模式，提高网络的在线学习效率；文献[5]针对单个在线极限学习机输出不稳定的情况，根据数据训练误差自适应地调整在线模型集成权重，选择性能更好的模型用于集成预测，改进了模型的预测准确度和稳定性。缺点在于多个模型训练造成的时间复杂度却有所增加。

本文针对以上问题，通过集成学习的方式，提出多个在线核极限学习机的模型训练框架，得到比单个模型更好的学习性能。同时借助大数据技术提出基于 MapReduce 的多个在线核极限学习机集成学习模型的并行算法（MROS-KELM），有效地改进了多个在线核极限学习机训练造成的时间复杂度问题。

1 极限学习机相关理论

1.1 核极限学习机

根据给定的训练样本集 $\{(x_i, t_i), i = 1,2,\cdots, L\}$，采用随机方式从中挑选部分样本，将挑选出的样本组成新的训练样本 (x_i, t_i)，$i = 1,2,\cdots, N_0$，$N_0 > L$，其中 x_i 为输入变量，t_i 为输出变量。极限学习机通过以下函数来估计实际输出，其函数形式可表示为：

作者简介：沈哲钧（1992-），男，硕士研究生，主要研究方向：数据库与信息系统、云计算；凌志扬（1993-），男，硕士研究生，主要研究方向：信息集成、云计算。

收稿日期：2018-12-08

$$f(x) = \sum_{i=1}^{L} \beta_i g_i(x_j) \quad , \tag{1}$$

其中，L 为隐层节点个数，$g(x)$ 为激活函数，是连接 i 个隐含层节点的输出权值。

式(1)可表示成矩阵形式为：

$$\boldsymbol{H\beta} = \boldsymbol{Y} \quad , \tag{2}$$

其中，$\boldsymbol{\beta} = \begin{bmatrix} \beta_1^{\mathrm{T}} \\ \cdots \\ \beta_L^{\mathrm{T}} \end{bmatrix}_{L\times m}$，$\boldsymbol{Y} = \begin{bmatrix} y_1^{\mathrm{T}} \\ \cdots \\ y_L^{\mathrm{T}} \end{bmatrix}_{N\times m}$.

极限学习机训练目标的最优化模型可以表示成：

$$L_{ELM} = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + \phi \frac{1}{2} \sum_{i=1}^{N} \varepsilon_i^2, \tag{3}$$

$$h(x_i)\boldsymbol{\beta} = t_i - \varepsilon_i \quad i = 1,2,\cdots,N \quad , \tag{4}$$

其中，β_N 为隐层输出权值；ϕ 为常参数；ε_i 为理论输出 t_i 与实际输出 $f(x)$ 的最大绝对误差；$h(x_i)$ 为隐层关于样本 x_i 的输出向量，求解上式，可得：

$$\boldsymbol{\beta}_N = \left(\frac{1}{\phi} + \boldsymbol{H}_N^{\mathrm{T}} \boldsymbol{H}_N\right)^{-1} \boldsymbol{H}_N^{\mathrm{T}} \boldsymbol{T}_N, \tag{5}$$

其中，$\boldsymbol{H}_N = [h^{\mathrm{T}}(x_1), \cdots, h^{\mathrm{T}}(x_N)]^{\mathrm{T}}$ 为神经网络关于训练样本集的隐含层输出矩阵，$\boldsymbol{T}_N = [t_1, \cdots, t_N]^{\mathrm{T}}$ 为训练样本的目标矩阵。得到 ELM 模型的实际最优输出为：

$$f(x_{op}) = h(x_{op})\boldsymbol{\beta}_N, \tag{6}$$

在 ELM 的训练过程中，$f(x_{op})$ 是通过随意赋值的方式产生，因此容易产生一系列非最优的输入权值和隐层节点的问题。为了得到更稳定的输出权值，针对电价数据的特点，将 RBF (radial basis function)作为极限学习机的核函数通过 Mercer 条件定义 ELM 核矩阵：

$$\Omega_N = \boldsymbol{H}_N \boldsymbol{H}_N^{\mathrm{T}} = K(x_i, x_j), \tag{7}$$

其中，$i,j = 1,2,\cdots,N$；$K(x_i, x_j)$ 为核函数，是该核矩阵位于第 i 列、第 j 列的元素。因此式(7)可以写成：

$$\boldsymbol{f}(x) = \begin{bmatrix} K(x_i, x_1) \\ \cdots \\ K(x_i, x_N) \end{bmatrix}^{\mathrm{T}} \alpha_{RBF}. \tag{8}$$

其中，α_{RBF} 为 KELM 模型的输出权值。从式(8)的推导可得，核极限学习机通过核函数将输出数据映射至高维空间，通过核映射避免了随机赋值导致的一定概率的隐层权值偏差，得到稳定的输出值。

1.2 在线核极限学习机

针对 ELM 的离线状态到在线状态的拓展模型训练算法分为 2 个阶段[6]，对此可阐释如下。

Step 1 离线阶段。利用随机设置的输入权值 w_i 和 $b_i(i = 1,2,\cdots,N_0)$ 计算隐含层输出矩阵 $\boldsymbol{H}_0$；并计算初始输出权值，其数学公式可表述如下：

$$\boldsymbol{\beta}^0 = \boldsymbol{P}_0 \boldsymbol{H}_0^{\mathrm{T}} \boldsymbol{T}_0, \tag{9}$$

其中，初始中间矩阵 $\boldsymbol{P}_0 = (\boldsymbol{H}_0^{\mathrm{T}} \boldsymbol{H}_0)^{-1}$，$\boldsymbol{T}_0 = [t_1, t_2, \cdots, t_{N_0}]^{\mathrm{T}}$；

设置 $k = 0$。

Step 2 在线阶段

(1)假设 $k + 1$ 时刻以前的历史电价训练样本 (X_t, Y_t) 已经被全部训练完成，其中 $X_t = \{x_i\}_{i=1}^{M}$，$Y_t = \{y_i\}_{i=1}^{M}$，M 表示离线状态结束后初始化训练数据个数，假设 $k + 1$ 时刻的训练数据为 (X_s, Y_s)，而且 $X_s = \{x_i\}_{i=M+1}^{M+k}$，$Y_s = \{y_i\}_{i=M+1}^{M+k}$，针对第 $k + 1$ 个获取的实时电价数据：$w_{k+1} = (x_{M+k+1}, t_{M+k+1})$，令 $\boldsymbol{T}_{k+1} = [t_{M+k+1}]^{\mathrm{T}}$，计算新到 $k + 1$ 时刻训练样本的隐含层输出矩阵 $\boldsymbol{H}_{k+1}$，计算公式如下：

$$\boldsymbol{H}_{k+1} = \begin{bmatrix} g(w_1 \cdot x_{M+k+1} + b_1) \\ \cdots \\ g(w_L \cdot x_{M+k+1} + b_L) \end{bmatrix}_{1\times L}, \tag{10}$$

(2)计算输出权值，即：

$$\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + \boldsymbol{P}_{k+1} \boldsymbol{H}_{k+1}^{\mathrm{T}} (\boldsymbol{T}_{k+1} - \boldsymbol{H}_{k+1}\boldsymbol{\beta}^k). \tag{11}$$

(3)置 $k = k + 1$，若 $k < N$，返回 Step 1 中的式(9)；否则算法结束。

2 数据处理算法比较

(1)串行数据处理算法[7]。使用串行数据处理算法的设计过程如图 1 所示。该算法对每一个核极限学习机模型的权重矩阵 $\boldsymbol{H}$ 进行串行化处理，依次选取每个节点进行训练，通过长时间的集群处理，可基于容错率来提高模型算法的稳定性。

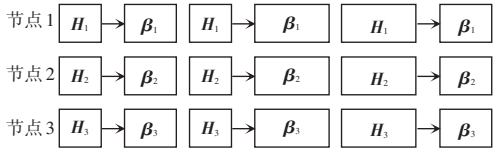

图 1 串行数据处理算法

Fig. 1 Serial data processing algorithm

(2)并行数据处理算法。本文提出的基于 MapReduce 的多个在线核极限学习机模型的并行算法(MROS-KELM)如图 2 所示，该算法分别在 Map 阶段和 Reduce 阶段对每一个核极限学习机模型的权重

矩阵 $\boldsymbol{H}$ 以及输出的权值 $\boldsymbol{\beta}$ 进行并行化处理，避免了串行数据算法中每一个核极限学习机模型只有一个节点在运算处理的缺点，明显提高了集群运算效率。

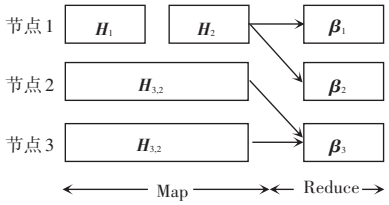

图 2　并行数据处理算法

Fig. 2　Parallel data processing algorithm

3　OS-KELM 并行化算法设计

集成学习的基本思想是通过多个单一学习器集成组合在一起，使其共同完成学习任务，结合不同模型的结果进行新的组合来求出最终结果，得到比单个模型更好的学习性能。研究得到多个在线核极限学习机训练模型框架如图 3 所示，本文在集成学习框架下根据 Bagging、随机森林以及交叉验证思想结合多个在线核极限学习机提出基于集成学习在线核极限学习机模型。其中，$(\boldsymbol{X}_{m,k}^{train}, \boldsymbol{T}_{m,k}^{train})$、$(\boldsymbol{X}_{m,k}^{valid}, \boldsymbol{T}_{m,k}^{valid})$ 分别表示子模型 m 的第 k 组训练数据、验证数据，最后结果为子模型的测试数据。

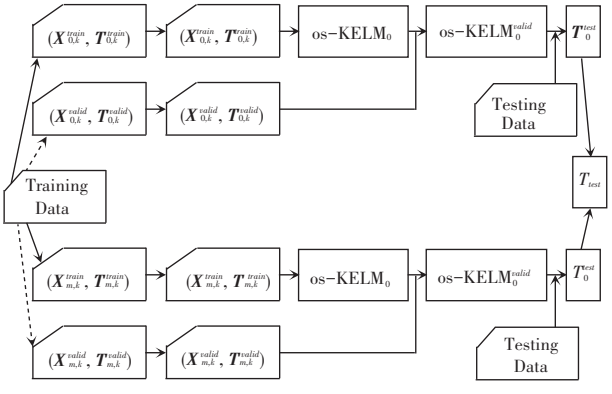

图 3　多个在线核极限学习机训练模型框架

Fig. 3　Multiple online kernel extreme learning machine training model framework

然而，在传统串行数据处理技术的环境下，基于集成学习的多个在线核极限学习机模型却会使该集成框架的并行处理优势受到限制，因此本文通过 MapReduce 的并行处理技术对 OS-KELM 模型进行并行化处理。针对 OS-KELM 模型的并行训练主要用最大期望(EM)算法，EM 算法在数学中常可用于寻找在不容易观察的隐形变量的模型中的参数最大似然估计。文中，关于 EM 算法的流程步骤可分述如下。

(1)初始化分布参数。

(2)EM 过程具体如下：

① E Step：估计未知参数期望值，给出当前的参数估计。

② M Step：重新估计分布参数，使数据训练的似然性最大。

(3)重复，直到收敛结束。

3.1　MROS-KELM 算法的 Mapper 设计

通过 MapReduce 过程实现 EM 算法基本上遵循以下的操作过程：在每一次迭代过程中，Mapper 过程训练数据实例。Reducer 整合各类期望参数，为下一轮迭代生成参数估计。

其中，Mapper 过程主要处理 2 个部分。对每一部分内容可做研究表述如下。

(1)初始化 OS-KELM 模型。根据框架相关参数，对 OS-KELM 模型初始化中间转移矩阵 $\boldsymbol{P}_{m,k}$、隐层矩阵 $\boldsymbol{H}_{m,k}$、输入权重 $\boldsymbol{\beta}_{m,K}$。

(2) *map*( ) 函数运算。训练 OS-KELM 模型的 Mapper 部分算法的伪代码详见如下。

```
For m = 0 To M - 1
    If chooseForThisModel( ) Then
      添加到 block_m 中;
      count_m ++ ;
      If count_m ≥ B Then
      Xs_m^train = GetSubSpace(block_m) ;
      H_{m,k} = calH(Xs_m^train) ;
      T_{m,k} = calH(block_m) ;
      输出 ((m,k_m,TrainTag),(H_{m,k},T_{m,k})) ;
//输出一个 key/value 对
      count_m = 0;
      k_m ++;
      EndIf
    EndIf
    If chooseForValid( ) Then
      添加到 valid_m 中;
      count_m^valid ++
      If count_m^valid ≥ B Then
      Xs_m^valid = GetSubSpace(valid_m) ;
      H_{m,k}^valid = calH(Xs_m^valid ) ;
      T_{m,k}^valid = calH(valid_m) ;
      输出 ((m,k_m^valid,ValidTag),(H_{m,k}^valid,T_{m,k}^valid )) ;
      count_m^valid = 0;
      k_m^valid ++ ;
```

EndIf

EndIf

EndFor

3.2 MROS-KELM 算法的 Reducer 设计

在每次 OS-KELM 模型训练的迭代过程中，Reducer 部分伪代码详见如下。

$m = getm(\boldsymbol{key})$;

$tag = gettag(\boldsymbol{key})$;

If $tag = TrainTag$ Then

$\boldsymbol{H}_{m,k} = getH(\boldsymbol{value})$;

$\boldsymbol{T}_{m,k} = getT(\boldsymbol{value})$;

$\boldsymbol{P}_{m,k} = \boldsymbol{P}_{m,k-1} - \boldsymbol{P}_{m,k-1}\boldsymbol{H}_{m,k}^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{H}_{m,k}\boldsymbol{P}_{m,k-1}\boldsymbol{H}_{m,k}^{\mathrm{T}})^{-1}\boldsymbol{H}_{m,k}\boldsymbol{P}_{m,k-1}$;

$\boldsymbol{\beta}_{m,k} = \boldsymbol{\beta}_{m,k-1} + \boldsymbol{P}_{m,k}\boldsymbol{H}_{m,k}^{\mathrm{T}}(\boldsymbol{T}_{m,k} - \boldsymbol{H}_{m,k}\boldsymbol{\beta}_{m,k-1})$;

EndIf

If

For

$\boldsymbol{H}_{m,k}^{valid} = getH(\boldsymbol{value})$;

$\boldsymbol{T}_{m,k}^{valid} = getT(\boldsymbol{value})$;

$CrossValid(\boldsymbol{\beta}_{m,K}, \boldsymbol{H}_{m,k}^{valid}, \boldsymbol{T}_{m,k}^{valid})$;

EndFor

EndIf

整个 MapReduce 过程需要迭代多次，直到 OS-KELM 模型的训练参数达到收敛。

4 OS-KELM 算法的并行化实现

下面拟详尽给出 OS-KELM 并行化算法代码的设计实现。在这里使用 ArrayWritable 数组数据类型，用来分别存储 OS-KELM 模型的中间转移矩阵、隐层矩阵、输入权重的一行数据，代表收集到的相关值的集合。

4.1 MROS-KELM 算法的 Mapper 实现

4.1.1 初始化 OS-KELM 模型

Mapper 最先要处理的就是 KELM 模型。根据相关参数，对 KELM 模型初始化中间转移矩阵、隐层矩阵、输入权重。该过程在 Mapper 的 *setup*( ) 初始化方法中完成。部分的代码如图 4 所示。

4.1.2 Map 方法实现

OS-KELM 模型的并行化 *map*( ) 函数部分可做解析阐述如下。

研究可知，输入的 < *key*, *value* > 对中，*key* 值是训练数据；*Value* 值存储的是测试数据。对每个训练实例来说，输出多个 stripes 并且每个训练实例都输出相同的 *key* 值集合。每个 *key* 值都会在 M-step 进行优化处理。

```
public class KelmParallel {
    public static class KelmParallelMap extends Mapper<Object, Text, Text, ArrayWritable>{
        //初始化KELM模型
        public void setup(Context context)
            throws IOException, InterruptedException{
            kelm = KelmTrainer.trainSupervised(States, Observations, observedSequence, hiddenSequence, 0);
        }
```

图 4 模型初始化阶段

Fig. 4 Model initialization phase

而输出的 < *key*, *value* > 对包括着：在 *q* 状态开始时的初始化中间转移矩阵概率、*q* 状态产生隐层矩阵概率、*q* 状态输入权重调整概率。在讨论时，该部分研究中可参考设计代码如下。

```
// 将 map( ) 结果发射出去
        for( int i = 0;i < kelm.numStates;i ++) {
    Pi_tmp[i] = new DoubleWritable(kelm.initialProbabilities[i]);
        }
        piStripe.set(pi_tmp);
        context.write(new Text("initial"),piStripe);
        for( int i = 0;i < kelm.numStates;i ++) {
          for( int j = 0;j < kelm.sigmaSize;j ++) {
    em_tmp[j] = new DoubleWritable(kelm.adjustmentMatrix[i][j]);
          }
          adjustmentStripe.set(em_tmp);
          Context.write(new Text("adjust from" + hiddenSequence.toString( )),adjustmentStripe);
        }
```

4.2 MROS-KELM 算法的 Reducer 实现

输入的 < *key*, *value* > 对是 Mapper 部分的输出，即 < string ‘initial’, initialProbabilities > ; < string ‘emit from’, +states, Stripe adjustmentMatrix > 和 < string ‘produce from’, +states, Stripe produceMatrix > ; Reducer 部分对不同的 Mapper 部分发射来的数据进行整合迭代，即 KELM 在线状态，为下一轮迭代做准备。因此输出的 < *key*, *value* > 对依然为 < string ‘initial’, initialProbabilities > ; < string ‘emit from’, +states, Stripe adjustmentMatrix > 和 < string ‘produce from’, +states, Stripe produceMatrix > 。

MROS-KELM 算法的 Reducer 实现部分研究中可参考如下设计代码。

```
Public static class KelmParallelReduce extends Reducer<Text, ArrayWritable,Text,ArrayWritable>{
    Private ArrayWritable Cf;
    Private DoubleWritable( ) Cf_tmp;
```

```
Double[ ] doubleValue;
{
  {
  //reduce 函数部分
   Public void reduce ( Text key, Iterable < ArrayWritable> values, Context context)
      Throws IOException InterruptedException{…
```

5　算例测试

本文对在线核极限学习机模型算法(OS-KELM)、基于集成学习的在线核极限学习机模型算法(EOS-KELM)以及提出的基于 MapReduce 的多个在线核极限学习机模型的并行算法(MROS-KELM)进行实验性能对比。对此可展开论述如下。

5.1　实验设置

实验环境:在搭建异构环境的 Hadoop 集群上进行了实验。本地测试平台包含 1 个主节点(NameNode 和 JobTracker)和 2 个从节点(DataNodes 和 TaskTrackers)。通过在每个节点上搭载各种 CPU 类型、内存大小和磁盘空间来测量集群中的异构性。每台服务器配置为 Xeon E5-2620 v2(6 核双线程)CPU、16 G 内存,集群中的所有节点都与千兆以太网交换机连接,并在 Centos6.4 64 位操作系统上运行。

在实验中,集群环境部署配置的系统是市场上企业应用最多,结合最好的稳定版本 Hadoop 2.6.0,块大小为 128 MB,JDK 版本 8。并为此集群中的每个数据块维护 3 个副本,以提高数据的可用性。

在这里,真实数据使用的是 MINIST 和 ImageNet 数据集。其中,MINIST 数据集是最受欢迎的深度学习数据集之一,这是一个手写数字数据集,包含一组 60 000 个示例的训练集和一个包含 10 000 个示例的测试集。这是一个很好的数据库,用于在实际数据中尝试引入学习技术和深度识别模式,同时可以在数据预处理中花费最少的时间和成本。

主要用于对各个算法的准确率、训练速率进行评测。数据集的特征数、训练样本和测试样本数以及数据块大小见表 1。

表 1　数据集说明

Tab. 1　Dataset description

数据集	训练样本数	测试样本数	数据块大小/KB
MINIST	50 000	20 000	168 003.147
ImageNet	1 780	1 153	1 483.178

MROS-KELM、EOS-KELM、OS-KELM 采用 $g(x)=\frac{1}{1+e^{-x}}$ 作为激活核函数。在真实数据测评中,由于数据集比较小,为了比较其准确性将隐含层节点数 L 设定为 60。

5.2　数据分析

真实数据集的准确率与训练时间见表 2,从测评表中可以推得以下结论:

(1)EOS-KELM 的训练准确度和测试准确度都高于 OS-KELM。说明基于集成学习的模型算法在减少噪声数据影响与增加准确率方面具有优势。

(2)MROS-KELM 的训练准确率与测试准确率都与 EOS-KELM 基本相同。在提升模型训练准确率的同时,训练时间相比较 EOS-KELM 有明显的减少。

MROS-KELM 的 Map 阶段和 Reduce 阶段算法分析与 EOS-KELM 的算法对比分析可推得,MROS-KELM 并没有改变 EOS-KELM 的执行过程,因此 MROS-KELM 方法在保证原有集成框架算法的优点的基础上同时也使用 MapReduce 框架,MROS-KELM 具备更好的模型并行处理能力,在 Map 阶段与 Reduce 阶段分别对各子模型的 $\boldsymbol{H}$ 和 $\boldsymbol{\beta}$ 矩阵进行并行计算,加速了各个模型的训练流程,因此相比 EOS-KELM 方法,MROS-KELM 的训练速度更快。

表 2　数据集结果评测

Tab. 2　Dataset result evaluation

数据集	算法	训练时间/s	训练准确率	测试准确率
MINIST	OS-KELM	3 961.352	0.765	0.811
	EOS-KELM	3 873.411	0.952	0.923
	MROS-KELM	1 016.123	0.965	0.934
ImageNet	OS-KELM	369.200	0.711	0.763
	EOS-KELM	357.100	0.808	0.871
	MROS-KELM	169.500	0.831	0.891

其中,针对 MROS-KELM 和 EOS-KELM 基于不同数据块大小情况下的准确率比较如图 5 所示。由图 5 可以看出,MROS-KELM 和 EOS-KELM 分别在 2 种不同数据集的不同数据块大小下的准确率基本一致。其中,每个数据块的前两项和后两项分别是 ImageNet 数据集和 MINIST 数据集的准确性能对比,只有 ImageNet 数据集下的不同大小数据块的准确率略有偏差,主要是因为数据的特征数量和训练样本数量的规模与 MINIST 数据集相比带来更多的数据噪声所造成的影响。综合来说,在大数据环境下基于 MapReduce 的 MROS-KELM 算法并不会影响 EOS-KELM 集成算法的准确性。

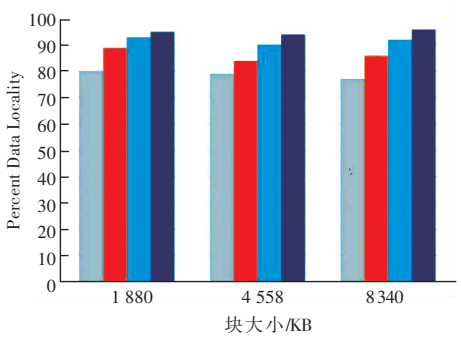

图 5 不同数据块下的准确率比较

Fig. 5 Comparison of accuracy under different data blocks

图 6 展示了基于 MapReduce 的多个在线核极限学习机模型的并行算法(MROS-KELM)、基于集成学习的在线核极限学习机模型算法(EOS-KELM)以及在线核极限学习机模型算法(OS-KELM)训练时间随训练样本数量的变化趋势。从图 6 中可以看出 MROS-KELM 的训练时间随着训练样本数量呈线性增长,也说明了 MROS-KELM 模型算法具有更好的泛化性能。同时也可以对比出 MROS-KELM 比 OS-KELM 训练速度快了一个数量级,这说明 MROS-KELM 通过并行化的算法更好地提高了模型训练效率。

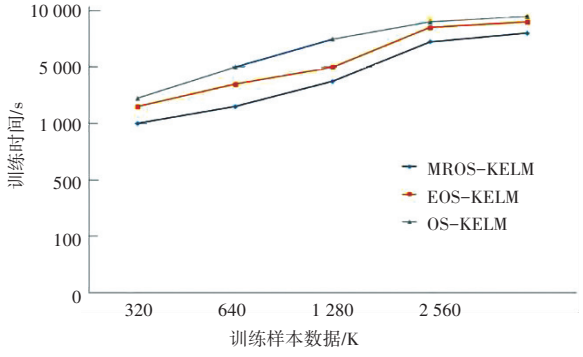

图 6 训练时间随训练样本数变化

Fig. 6 Training time varies with the number of training samples

6 结束语

本文主要对大数据并行处理技术结合多个在线核极限学习机模型算法的问题进行探讨。针对传统的串行数据处理技术难以处理多个极限学习机模型所造成的模型训练时间过长的缺点,研究中通过 MapReduce 并行处理技术提出基于 MapReduce 的在线核极限学习机集成学习的并行模型算法,通过算例测试验证了本文的基于 MapReduce 的集成学习算法可以有效提高模型的准确度和实时的训练速度,也说明了本文使用的基于大数据处理技术的预测模型在大数据环境中具备良好的适应性。

参考文献

[1] LIANG Nanying, HUANG Guangbin, SARATCHANDRAN P, et al. A fast and accurate online sequential learning algorithm for feedforward networks [J]. IEEE Transactions on Neural Networks, 2006, 17(6): 1411-1423.

[2] HUANG Guangbin, ZHU Qinyu, SIEW C K. Extreme learning machine: Theory and applications [J]. Neurocomputing, 2006, 70(1-3): 489-501.

[3] 郭瑞,樊亚敏,潘玉民. 一种结合互补集合经验模态分解和小波核极限学习机的短期电力负荷预测模型[J]. 计算机应用与软件,2016,33(12):243-247,263.

[4] 马超,张英堂. 在线核极限学习机及其在时间序列预测中的应用[J]. 信息与控制,2014,43(5):624-629.

[5] 徐勇,王东,张慧. 基于自适应在线极限学习机模型的预测方法[J]. 统计研究,2016,33(7):103-109.

[6] 杨乐, 张瑞. 在线序列 ELM 算法及其发展[J]. 西北大学学报(自然科学版), 2012, 42(6): 885-889,896.

[7] GUO Zhenhua, FOX G C. Improving MapReduce performance in heterogeneous network environments and resource utilization [C]//Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. Washington, DC, USA:IEEE, 2012: 714-716.

(上接第 56 页)

MNIST 数据集类似。最后使用自己的数据集通过 LeNet-5 模型训练卷积神经网络,对输入的手写体图片进行特征提取,取得了较好的识别效果。日后在实际应用中如何统一手写体图片的采集标准和扩展数据集数量,这将是深度卷积神经网络后期需要解决的问题。

参考文献

[1] 程国建,岳清清. 卷积神经网络在岩石薄片图像检索中的应用初探[J]. 智能计算机与应用,2018,8(2):43-46,51.

[2] 焦李成,赵进,杨淑媛,等. 深度学习、优化与识别[M]. 北京:清华大学出版社,2017.

[3] 李金洪. 深度学习之 TensorFlow 入门、原理与进阶实战[M]. 北京:机械工业出版社,2018.

[4] 李志. TesnsorFlow 深度学习[M]. 北京:人民邮电出版社,2018.

[5] HAGAN M T, DEMUTH H B, BEALE M H, et al. Neural network design[M]. 2nd ed. 章毅,等译. 北京:机械工业出版社,2018.

[6] 王文峰,李大湘,王栋. 人脸识别原理与实战:以 MATLAB 为工具[M]. 北京:电子工业出版社,2018.

[7] 林大贵. TesnsorFlow+Keras 深度学习人工智能实践应用[M]. 北京:清华大学出版社,2018.