基于 MapReduce 的并行化最小最大模块化支持向量机研究

赵研　李云

（南京邮电大学计算机学院　　南京　210023）

摘　要 最小最大模块化支持向量机(M3-SVM)是对大规模数据进行模式分类的有效方法。为了进一步提高M3-SVM 算法处理大规模数据的效率。本文基于 MapReduce 的编程模型实现了 M3-SVM 的并行化。并行化主要分为两个部分：1、将 M3-SVM 中的多个任务分解进行并行化；2、将 M3-SVM 中用来训练基分类器 SVM 的序列最小优化算法(SMO)进行并行化。在多个现实数据集上的实验结果表明基于 MapReduce 的并行化最小最大模块化支持向量机算法不仅具有较好的可靠性，而且比传统的最小最大模块化支持向量机算法具有更好的时间效率。

关键字：并行化；最小最大支持向量机；MapReduce；SMO

中图法分类号 TP391

Min-Max Modular SVM Parallelization Based on MapReduce

Zhao Yan　　Li Yun

(College of Computer Science , NanJing University of Posts and Telecommunications , NanJing, 210023 , China)

Abstract Min-Max Modular Support Vector Machine (M3-SVM) is an effective classifier to classify large-scale data. But with the explosion of data, we need to improve the efficiency of M3-SVM. We realize the parallelization of M3-SVM based on MapReduce. Parallelization consists of two parts: 1. The parallelization of task decomposition; 2. The parallelization of base classifier training, i.e., parallelization of SMO algorithm for SVM in M3-SVM. Experiments are conducted on several real-world data sets and the experimental results show the reliability and high computing efficiency of parallel M3-SVM based on MapReduce.

Key words：Parallelization; M3-SVM;　MapReduce; SMO

1 引言

随着信息技术和生物技术的发展,在现实生活与科学研究中涌现出大量的大规模数据，因此如何从大规模数据中挖掘出有用的知识是目前研究的热点和难点。对于大规模数据来说，一些传统的模式分类技术如 K 近邻分类（KNN）、多层感知器（MLP）和支持向量机（SVM）[1]等已经很难取得很好的分类效果，甚至不可行。但是许多研究结果和应用实例都表明集成学习是处理大规模数据的一种有效的手段[2]。而基于“部分对部分”样本分解策略的最小最大模块化支持向量机（M3-SVM）[3-5]是处理大规模数据表现比较优异的集成学习方法之一，已广泛应用到文本分类、人脸识别、网络安全检测、语音识别和蛋白质亚细胞定位等[6-11]。

为了充分利用已有的计算资源，进一步扩大最小最大模块化支持向量机处理数据的规模，提高其处理效率和减少时间开支，需要研究并行化的最小最大模块化支持向

基金项目：国家自然科学基金（61373139，61105082）江苏省自然科学基金（BK20131378，BK20140885）
收稿日期：
*通讯联系人，E-mail：liyun@njupt.edu.cn

量机。而目前最具代表性的分布式计算框架当属Google提出的MapReduce编程模型[13][14]。因此将MapReduce编程模型与最小最大模块化支持向量机算法的结合，提出了基于MapReduce的并行最小最大模块化支持向量机算法，即M3-SVM的并行化算法（P-M3-SVM）。

MapReduce是一个分布式计算框架，主要由两部分组成：编程模型和运行环境。其中，编程模型为用户提供了非常易用的编程接口，用户只需要编写Mapper函数和Reducer函数就可以实现一个分布式程序。MapReduce能够解决的问题有一个共同点：任务可以被分解为多个子任务，且这些子任务相对独立，待并行完成这些子任务后，原始任务也就相应地完成。而M3-SVM算法原本就是将大规模任务分解成多个子任务，且这些子任务之间相对独立，因此可以无缝地利用MapReduce编程模式实现M3-SVM的并行化。

本文其余章节安排如下：第二节主要介绍传统最小最大模块化支持向量机(M3-SVM)的整个学习过程。第三节中详细分析并行 M3-SVM 中多任务的并行化以及序列最小优化算法 SMO 的并行化[15-17]。第四节使用多个数据集验证 M3-SVM 并行化的可靠性及性能。第五节总结。

2 最小最大模块化支持向量机 M3-SVM

令 $T=\{X_l,Y\}_{l=1}^{L}$ 表示一个两类分类问题的训练样本集合，$X_l\in R^n$ 表示 n 维空间中的第 l 个样本，$Y\in\{+1,-1\}$ 表示每个样本对应的类别标记，其中+1 表示正类样本，-1 表示负类样本。L 表示样本总数。M3-SVM 的整个学习过程，包括任务分解和 MIN-MAX 规则集成。

2.1 任务分解

假设给定的二类问题 T 中正类训练样本集和负类训练样本集分别为：

$$\chi^{+}=\{x_m^{+},+1\}_{m=1}^{l^{+}} \tag{1}$$

$$\chi^{-}=\{x_m^{-},-1\}_{m=1}^{l^{-}} \tag{2}$$

其中，l^{-} 和 l^{+} 分别为样本中负类和正类样本的个数，x_m^{+} 表示第 m 个正类样本，χ^{+} 表示所有的正类样本。在最小最大模块化支持向量机中，将利用“部分对部分”策略对大规模两类问题做进一步的分解。也就是对原始训练样本集中正类和负类样本分别进行分解。假设将正类样本集 χ^{+} 利用某些规则划分成 N^{+} 个样本子集：

$$\chi_i^{+}=\{x_m^{+,i},+1\}_{m=1}^{l_i^{+}}\quad,\quad i=1,\ldots,N^{+} \tag{3}$$

其中 $1\le N^{+}\le l^{+}$，l_i^{+} 为第 i 块正类样本子集中样本的个数，且 $\cup_{i=1}^{N^{+}}\chi_i^{+}=\chi^{+}$。同理也可以将负类样本集 χ^{-} 划分成 N^{-} 个子集：

$$\chi_j^{-}=\{x_m^{-j},-1\}_{m=1}^{l_j^{-}}\quad,\quad j=1,..,N^{-} \tag{4}$$

其中，$1\le N^{-}\le l^{-}$，l_j^{-} 为第 j 块负类样本子集中样本的个数，且 $\cup_{j=1}^{N^{-}}\chi_j^{-}=\chi^{-}$。

当 χ^{+} 和 χ^{-} 分别被分解成 N^{+} 和 N^{-} 个样本子集后，通过将正类样本子集与负类样本子集两两组合，原始的大规模的问题被分解成 $N^{+}\times N^{-}$ 个规模较小的子问题 $T_{(i,j)}$：

$$T_{(i,j)}=\chi_i^{+}\cup\chi_j^{-},i=1,2,..,N^{+},j=1,2,..,N^{-} \tag{5}$$

2.2 最小最大模块集成规则

任务分解完成之后，在每个子问题所对应的样本子集 $T_{(i,j)}$ 上训练相应的子模块（利用 SVM 作为基分类器），然后用 MIN-MAX 规则进行集成。

每个子模块得到的结果通过以下两个规则进行集成：MIN 规则和 MAX 规则。MIN 规则处理过程如下：对于每个子模块的训练样本集，如果它拥有相同正类训练样本和不同负类训练样本，那么它的分类结果取最小值；MAX 规则是对于每个子模块的训练样本集，如果它拥有相同的负类训练样本和不

同的正类训练样本，那么它的分类结果取最大值。于是，上述提到二类问题被分解成 $N^+ \times N^-$ 个二类平衡子问题，结果的集成过程可以描述为 N^+ 个 MIN 单元和一个 MAX 单元的组合，即

$$M_i(x) = \min_{j=1}^{N^-} M_{i,j}(x) \quad ,\mathrm{i} = 1,2,\ldots,N^+ \qquad (6)$$

$$M(\mathrm{x}) = \max_{i=1}^{N^+} M_i(\mathrm{x}) \qquad (7)$$

其中，$M_{i,j}(\mathrm{x})$ 是二类子问题 $T_{(i,j)}$ 学习所得基分类器输出结果，$M_i(\mathrm{x})$ 是采用 MIN 规则集成后的结果，$M(\mathrm{x})$ 为原始两类问题的解。

2.3 M3-SVM 结构

综上所述，针对大规模二类问题，M3-SVM 首先进行大规模问题的平衡化分解，得到一系列更小更平衡的子问题，并训练出子问题对应的子模块（基分类器 SVM），最后利用 MIN-MAX 规则集成得到原始大规模问题的解。整个 M3-SVM 结构如图 1 所示。

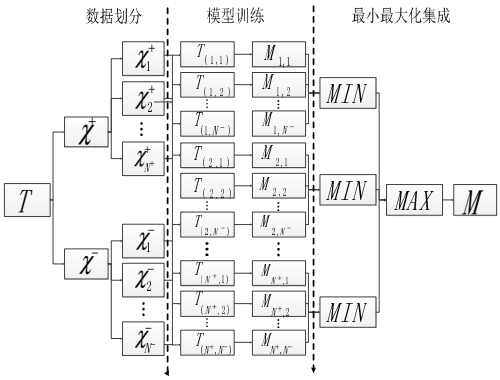

图 1. M3-SVM 的体系结构

3 最小最大模块化支持向量机的并行化

为了提高 M3-SVM 处理大规模数据的效率，提出了基于 MapReduce 并行编程框架的分布式 M3-SVM 算法--P-M3-SVM。并行主要分为两个部分：1、任务并行化：对 M3-SVM 中的任务分解进行并行处理；2、算法并行化：将 M3-SVM 中用来训练基分类器 SVM 的序列最小优化算法(SMO)进行并行化。

3.1 任务并行化

在并行的 M3-SVM 算法中，通过设定正类样本子集和负类样本子集的块数，为数据集中每个样本标注所属的块号。然后使用 MapReduce 的编程框架来实现模型的训练与测试。在模型训练的 MapReduce 作业中，Mapper 函数将所有正类样本子集与所有负类样本子集进行两两组合，得到相应训练数据子集。在 Reducer 函数中基于每个训练数据子集训练出 SVM 子模型，将所有子模型集成得到原始问题的训练模型。模型测试的 MapReduce 作业主要是用来测试模型分类的准确率。Mapper 函数主要是读取训练好的模型，然后对测试集中的样本进行逐一预测，Reducer 函数主要是统计测试的正确率。下面举例说明模型训练中任务并行化的实现过程，如图 2 所示：假定 s1, s2, …, s9 是训练样本集中的样本，+1，-1 是每个样本的类别标签。

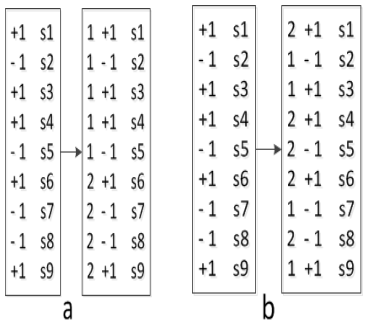

图 2. 标注样本块号的方式示例(a)顺序方式 (b)随机方式

第一步：在 P-M3-SVM 算法中，使用两种方式为样本标注块号，对应的算法实现分别为 P-M3-SVM-L 和 P-M3-SVM-R。在 P-M3-SVM-L 算法中，通过设定好的正类样本子集和负类样本子集的个数，确定每个样本子集中的样本数，然后按照顺序为数据集中的每个样本标注其所属的样本子集（块号）。如图 2 中 a 所示，首先我们假设正类样本子集和负类样本子集的个数都设为 2，那么每个正类样本子集中应该有 3 个正类样本，而每个负类样本子集应该有 2 个负类样本，这样可以依次对每个正负类样本标注其所属的块号。在 P-M3-SVM-R 算法中，首先通过设定好的正类样本子集和负类样本子集的个数，然后顺序的读取数据集中的每

个样本，并且在 1 到 χ 之间随机产生一个值作为该样本的块号。如图 2 中 b 所示，首先我们设定正类和负类样本子集的个数都为 2，然后当读取一个样本时，在 1 和 2 中随机产生一个数字作为该样本的块号。

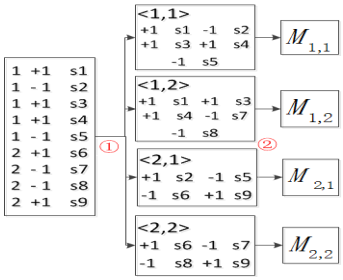

图 3. M3-SVM 模型训练中任务并行化的过程示例

第二步：将第一步的结果作为输入。在 Mapper 阶段利用每个样本所标注的块号，将正、负类样本进行分块。再将每个正类样本子块（对应着前面所介绍的正类样本子集 χ_i^+ ）与所有负类数据子块（对应着前面所介绍的负类样本子集 χ_j^- ）进行两两合并，组合得到相应的训练数据子块，即 $T_{(i,j)}$ 。并且为每个训练数据子块设置关键字〈key1, key2〉，其中 key1 为正类数据子块的块号，key2 为负类数据子块的块号。在 Reducer 阶段基于每个训练数据子块，训练相应的子模型 $M_{i,j}(x)$，并且以训练数据子块的关键字作为该子模型的关键字。如图 3 中的过程①②所示，首先将两个正类数据子块与两个负类的数据子块两两进行合并，得到四个训练数据子块；然后利用 SVM 基于每个训练数据子块训练得到相应的分类子模型。

第三步：将第二步得到所有模型作为输入，其中关键字 Key1 相同的子模型利用公式 6 所介绍的 Min 规则进行集成。然后再用公式 7 所介绍的 Max 规则进行集成，最终得到原始问题的分类模型。

3.2 算法并行化

在传统 M3-SVM 算法中为了提高支持向量机的优化效率，使用了序列最小优化算法（*Sequential Minimal Optimization*, SMO）。因此为了使 SMO 能有效处理大规模数据，将实现 SMO 算法的并行化。

3.2.1 SMO

首先，给出数据集中样本可能出现的 5 种情况：

$$\begin{aligned} I_0 &= \{\mathrm{k}: 0<\alpha_k<c\} \\ I_1 &= \{\mathrm{k}: y_k=1, \alpha_k=0\} \\ I_2 &= \{\mathrm{k}: y_k=-1, \alpha_k=c\} \\ I_3 &= \{\mathrm{k}: y_k=1, \alpha_k=c\} \\ I_4 &= \{\mathrm{k}: y_k=-1, \alpha_k=0\} \end{aligned} \tag{8}$$

其中， y_k 为第 k 个样本的类别， α_k 为第 k 个样本的 Lagrange 乘子， c 为 SVM 中的惩罚因子。首先根据文献[16][17]中求解 SMO 的处理过程，需要为样本集中每一个样本计算一个参数 f ，其中第 k 个样本 x_k 对应的 f 初始化的计算公式如下：

$$f_k = \sum_{n=1}^{l} \alpha_n y_n K(x_k, x_n) - y_k \tag{9}$$

其中， $K(x,x)$ 为核函数，本文使用的均为高斯核函数， l 为样本集中样本的个数。然后从样本集中选择两个样本 x_{up} 和 x_{low} 用作训练，其样本对应的 f 值分别为 b_{up} 和 b_{low} ，满足下列条件：

$$\begin{aligned} b_{up} &= \min\{f_k : \mathrm{k} \in I_0 \cup I_1 \cup I_2\} \\ b_{low} &= \max\{f_k : \mathrm{k} \in I_0 \cup I_3 \cup I_4\} \end{aligned} \tag{10}$$

接着为选择的两个样本计算新的 Lagrange 乘子，分别为 α_{up}^{new} 和 α_{low}^{new} ：

$$\begin{aligned} \alpha_{up}^{new} &= \alpha_{up}^{old} - \frac{y_{up}(b_{low}-b_{up})}{\eta} \\ \alpha_{low}^{new} &= \alpha_{low}^{old} - s(\alpha_{up}^{old} - \alpha_{up}^{new}) \end{aligned} \tag{11}$$

其中， $\eta = 2K(x_{low}, x_{up}) - K(x_{low}, x_{low}) - K(x_{up}, x_{up})$ ， $s = y_{low} y_{up}$ ，且 α_{low}^{new} 和 α_{up}^{new} 需要被约束在 $[0, c]$ 的范围内。根据得到的 α_{up}^{new} 和 α_{low}^{new} ，

为所有的样本更新对应的 f_k：

$$f_k^{new}=f_k^{old}+\left(\alpha_{low}^{new}-\alpha_{low}^{old}\right)y_{low}K\left(x_{low},x_k\right)+\left(\alpha_{up}^{new}-\alpha_{up}^{old}\right)y_{up}K\left(x_{up},x_k\right) \quad (12)$$

最后，重新选择两个样本继续训练。在训练过程使用 *DualityGap* 作为结束的判断条件，*DualityGap* 代表样本原始类别与支持向量机中预测结果的误差。通过下式计算得：

$$DualityGap=\sum_{n=1}^{l}\alpha_n y_n f_n+\sum_{n=1}^{l}\varepsilon_n \quad (13)$$

其中，$\begin{cases}\varepsilon_n=C\max\left(0,b-f_n\right) & ,\text{if} \quad y_n=1\\ \varepsilon_n=C\max\left(0,-b+f_n\right) & ,\text{if} \quad y_n=-1\end{cases}$。

3.2.2 SMO 并行化

根据上述理论的分析，在串行 SMO 算法处理过程中超过 90%的时间是用来更新样本对应的 f_k，所以提高该算法效率的关键在于减少更新样本对应的 f_k 所花费的时间。并且通过公式 12 得知，由于更新每个样本对应的 f_k 是相互独立的。因此可以使用 MapReduce 的编程框架来更新样本对应的 f_k。此外，为了进一步提高该算法的时间效率，可以将寻找 x_{up} 和 x_{low} 的过程也进行并行化。SMO 算法并行化实现的伪代码如下：

算法 1：SMO 算法并行化

输入：由任务并行化中分块得到的数据块 D，结点个数 P，阈值为 δ

第一步：将数据块分成 P 份，分别为 D_i，将 D_i 分发送到不同数据结点上

第二步：初始化，使用公式 13 计算 *DualityGap* 和使用公式 9 为每个样本初始化参数 f

第三步：**WHILE** $DualityGap>\delta$

第四步：根据式 10 选出第 r 个结点中的两个训练样本 x_{up}^r 和 x_{low}^r，并计算 $DualityGap_r$

第五步：求和：$\sum_{r=1}^{i}DualityGap_r=DualityGap$，并通过比较得到 x_{up} 和 x_{low}

第六步：使用公式 11 更新 x_{up} 和 x_{low} 对应样本的 Lagrange 乘子为 α_{up}^{new} 和 α_{low}^{new}

第七步：用公式 12 更新每个结点中样本对应的 f

第八步：**END WHILE**

根据上述可知，SMO 并行化算法中 MapReduce 实现步骤如下：

1)第一步，首先将整个任务分成 P 个子任务，Mappper 函数中通过使用公式 10 找出每个子任务的 x_{up}^r 和 x_{low}^r，然后将所有子任务 x_{up}^r 和 x_{low}^r 作为 Reducer 函数的输入。在 Reducer 函数中，首先从所有 x_{up}^r 中选择一个样本作为 x_{up}，该样本对应的 f 是所有 x_{up}^r 中最小的；从所有 x_{low}^r 中选择一个样本作为 x_{low}，该样本对应的 f 是所有 x_{low}^r 中最大的。然后使用公式 11 更新两个样本对应的 α，即 α_{up}^{new} 和 α_{low}^{new}。

2)第二步，将作业 1）中两个样本对应的 α_{up}^{old}，α_{low}^{old}，α_{up}^{new} 和 α_{low}^{new} 作为该作业的输入。将任务分成多个子任务，在 Mapper 函数中使用公式 12 更新每个子任务中的样本对应的 f。

4 实验结果及分析

下面我们主要使用实验来验证算法的准确性。其中实验环境是：在一台 IBM 服务器（Intel(R) Xeon(R)E5-2609 0 @ 2.4GHz, 8 核，16GB 内存）虚拟 6 个节点集群，每个节点配置双核 2GB 内存，Apache Hadoop 0.20.2，Ubuntu 12.04，JDK1.7。将进行三组实验：第一组实验使用串行 M3-SVM 算法与 P-M3-SVM 算法分别在相同的数据集上训练模型，验证在相同数据集上的分类准确率，从而验证并行 M3-SVM 算法的可靠性。第二组实验在同等条件下比较串行 M3-SVM 算法和 P-M3-SVM 算法的时间效率。第三组实验主要验证集群规模对 P-M3-SVM 算法可扩展行的影响。实验数据集 Covertype 和 Adult 来自 UCI,

kddcup2010 和 W8a 来自[18]，数据集的具体信息如表 1 所示。

表 1 实验数据集的描述

数据集	样本数	类别	特征维数
Adult	32561	2	123
W8a	49749	2	300
Covertype	581,012	2	54
kddcup2010	19,264,097	2	29,890,095

第一组实验：本实验在并行 M3-SVM 和传统 M3-SVM 算法中使用的核函数均为高斯核函数，并设定每个数据块中正类/负类的样本的个数 1000。在 Adult，W8a，Covertype 和 kddcup2010 数据集上使用十折交叉验证获得算法的分类准确率，实验结果如表 2 所示。

表 2 传统 M3-SVM 和并行 M3-SVM 在不同数据集上的准确率

数据集	M3-SVM	P-M3-SVM-L	P-M3-SVM-R
Adult	0.827	0.821	0.845
Covertype	0.816	0.829	0.833
W8a	0.796	0.788	0.807
Kddcup2010	0.763	0.771	0.768

从表 2 中可以看出 P-M3-SVM 的分类准确率与传统 M3-SVM 的分类准确率基本一致，说明 M3-SVM 的并行化是可行的。在 P-M3-SVM 算法中，P-M3-SVM-R 的准确率比 P-M3-SVM-L 高，因为顺序标号会使数据集中的信息损失，以至于 P-M3-SVM-L 的准确率略低。

第二组实验：本实验从Adult数据集中随机地抽取1000，2000，4000，6000，9000，13000，17000，21000，26000，32000个样本作为训练集。分别利用传统M3-SVM与P-M3-SVM算法依据第一组实验过程训练模型。随着数据规模增加串行M3-SVM与并行M3-SVM算法的时间开销实验结果如图4所示。

从图4中可以看出当训练数据集中的样本的个数小于5000时，串行M3-SVM所需的时间与并行M3-SVM（P-M3-SVM）基本相等甚至比并行少，但是当数据集中样本的个数大于9000时，P-M3-SVM所需的时间相比M3-SVM要少了很多。这主要是由于当数据集较小时，时间开销主要部分是数据传输到HDFS上花费的时间和平台启动的延迟，所以并行的效果不是太明显，随着数据集规模的增大，时间的主要开销为训练消耗的时间，所以并行的优势更加明显。

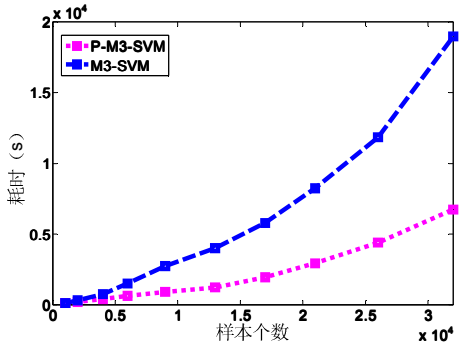

图4. 串行M3-SVM与并行M3-SVM的在不同规模的训练数据集上时间开销

第三组实验：本实验主要验证了集群规模对 P-M3-SVM 算法可扩展性的影响。实验使用的数据集为 Adult，使用 M3-SVM 算法分别在 2~6 个结点的集群上运行。实验结果如图 5 所示：

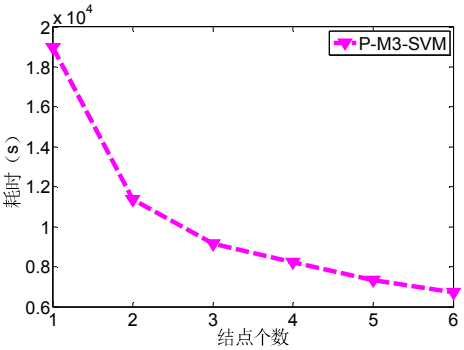

图5. 集群中结点个数与算法时间开支的关系

从图 5 可以看出，随着集群规模的不断扩大，数据处理的时间明显减少。由此可知，对于大规模的问题，随着结点数量的增加，处理数据的能力会不断增加。从而有效验证 P-M3-SVM 具有良好的可扩展性。

5 总结

为了快速地训练出处理大规模数据流的分类模型，本文利用 Hadoop 平台上的 MapReduce 编程模型，实现了 M3-SVM 算法的并行化。其中，主要包括 SMO 算法的并行优化和任务分解的并行化两部分。同时在数据集上验证并行算法的有效性和可靠

性，以及可保证的算法运行效率。在未来的工作中，将更深入分析并行化本身对算法的性能的影响。

参 考 文 献

[1] J. Han, M. Kamber. Data mining: concepts and techniques [M], Morgan Kaufman, 2001

[2] T. G. Dietterich. Ensemble methods in machine learning[C].Proc. of the 1st Int'l Workshop on Multiple Classifier Systems, Cagliari, Italy, 2000: 1-15

[3] B. L. Lu, M. Ito. Task decomposition and module combination based on class relations: a modular neural network for pattern classification [J].IEEE Trans. on Neural Networks, 1999, vol.10: 1244-1256

[4] B. L. Lu, K. A. Wang, M. Utiyama and H. Isahara.A part-versus-part method for massively parallel training of support vector machines[C]. Proc. of Int'l Joint Conf. on Neural Networks (IJCNN), 2004: 735-740

[5] B. L. Lu, X. L. Wang.A parallel and modular pattern classification framework for large-scale problems [M]. Handbook of Pattern Recognition and Computer Vision (4th Edition), 2009: 725-746

[6] B. L. Lu, Q. Ma, M. Ichikawa, H. Isahara. Efficient part-of-speech tagging with a min-max modular neural network[J]. Applied Intelligence,2003, vol.19: 65-81

[7] B. L. Lu, J. Shin, M. Ichikawa. Massively parallel classification of single-trial EEG signals using a min-max modular neural network[J]. IEEE Trans. on Biomedical Engineering ,2004, vol. 51(3): 551-558

[8] X. Chu, C. Ma, J. Li, B. L. Lu, M. Utiyama, H. Isahara. Large-scale patent classification with min-max modular support vector machines[C].Proc. of Int'l Joint Conf. on Neural Networks (IJCNN), 2008, vol. 1: 3972-3979

[9] K. Wu, B. L. Lu, M. Uchiyama, H. Isahara . An empirical comparison of min-max-modular k-NN with different voting methods to large-scale text categorization [J]. Soft Computing - A Fusion of Foundations, Methodologies and Applications, 2008, vol.12 (7): 647-655

[10] Y. Y. Chen, B. L. Lu, H. Zhao. Parallel learning of large-scale multi-label classification problems with min-max modular LIBLINEAR[C].Proc. of Int'l Joint Conf. on Neural Networks (IJCNN), 2012, vol.1:1-7

[11]Y. Yang, B. L. Lu. Protein sub cellular multi-localization prediction using a min-max modular support vector machine [J].International Journal of Neural Systems, 2010,vol.20(1): 13-28

[12] C. Cortes, V. Vapnik .Support Vector Networks. Machine Learning,1995, 20: 273-297

[13]R. Vernica, M. J. Carey, and C. Li. Efficient parallel set-similarity joins using MapReduce. In Proceedings of the International Conference on Management of data,2010.

[14]J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. Operating Systems Design and Implementation, 2004, pp:137–149.

[15]J. C. Platt, B. Scholkopf, C. Burges, and A. J. Smola, Eds., Fast training of support vector machines using sequential minimal optimization,in Advances in Kernel Methods—Support Vector Learning. Cambridge, MA: MIT Press, 1999, pp. 185–208

[16]S. S. Keerthi, S. K. Shevade, C. Bhattaacharyya, and K. R. K. Murthy ,Improvements to Platt's SMO algorithm for SVM classifier design,Neural Computing, vol. 13, pp. 637–649, 2001.

[17]L. J. Cao, S. S. Keerthi, C.J. Ong, J. Q. Zhang, Uvaraj Periyathamby, X. J.Fu, and H. P. Lee. Parallel Sequential Minimal Optimization for the training of Support Vector Machines. IEEE Trans. on Neural Networks,2006, vol. 17(4): 1039-1049

[18] C. J. Lin, http://www.csie.ntu.edu.tw/~cjlin/