电子与信息工程

文章编号：1000-8608(2014)04-0461-08

一种基于 MapReduce 的动态数据流分类算法

冯 林[1,2]， 姚 远[3]， 陈 沣[1]， 金 博*[2]

（1.大连理工大学 电子信息与电气工程学部 计算机科学与技术学院，辽宁 大连 116024；
2.大连理工大学 创新实验学院，辽宁 大连 116024；
3.大连民族学院 信息与通信工程学院，辽宁 大连 116600）

摘要： 当前动态数据流下的实时分类问题存在3个难点：针对海量数据的实时处理；概念漂移的跟踪和模型的更新；模型的稳定和鲁棒性.针对上述问题，将极端支持向量机(extreme support vector machine，ESVM)与 MapReduce 框架结合，提出了带遗忘因子的鲁棒 ESVM 算法.该方法通过构造残差权重矩阵，对残差进行修正，同时加入遗忘因子，提高新样本的作用，从而实现对海量数据处理问题的求解.实验结果显示，所提出方法能够快速有效地对动态数据流进行分类，且结果不易受到噪声干扰，稳定性强.

关键词： 数据流分类；增量式学习；极端支持向量机(ESVM)；MapReduce；遗忘因子；鲁棒性

中图分类号： TP312 **文献标识码：** A **doi：** 10.7511/dllgxb201404014

0 引 言

随着大数据时代的到来，传统数据挖掘技术需要面对与解决其带来的新情况与新问题.在这些新问题中，最为核心的是数据方面的改变，即数据形式从以往的静态数据，转变为动态数据流形式[1].

有别于传统静态数据形式，所谓动态数据流形式，即数据是实时产生的，并且无法进行存储.而对分类问题来说，由于数据实时产生，需要分类模型具有快速分类的能力，而传统分类方法往往难以适应[2].此外，动态数据流还存在概念漂移现象，这要求分类模型必须能够及时更新，以适应新数据环境的要求[3].由于目前基于学习的分类方法，在面对概念漂移问题时往往无法及时自我更新，导致分类结果不能尽如人意，因此，在传统分类方法的基础上，需要提出一种新的分类方法.

针对动态数据流的分类问题，国内外学者做了大量研究，目前大部分研究是将原有方法结合 MapReduce 框架的特点，在并行环境下予以实现，取得了相当多不错的研究成果[4-5].Zhang 等[6]研究了基于 MapReduce 框架的 K 最近邻并行算法，通过将 K 最近邻并行算法与 MapReduce 框架相结合，使得分类方法在时间效率方面大大提高，并且在此基础上，使得所提出模型具有多分类能力，但分类过程由于受到并行框架限制，其距离函数确定及参数确定仍存在问题.Pitchaimalai 等[7]设计出一种并行环境下的决策树算法，将传统决策树与 MapReduce 框架相结合，以达到克服过拟合问题的目的，使用了预剪枝技术，但无法从根本上克服局部最优问题.

支持向量机(support vector machine，SVM)模型作为数据挖掘领域最为著名的方法之一，通过在特征空间中寻找最大间隔超平面进行分类，同时利用核函数将低维空间上的样本映射到高维空间解决非线性问题.其优点是克服了维度灾难的问题，泛化能力强，但对二次规划问题的求解需要多次迭代来完成.Alham 等[8]分析了二次规划问题中的最小序列求解，在并行框架下进行改造，

收稿日期： 2013-06-04； **修回日期：** 2014-03-10.
基金项目： 国家自然科学基金资助项目(61173163，51105052)；教育部新世纪优秀人才支持计划资助项目(NCET-09-0251)；辽宁省教育厅资助项目(201102037).
作者简介： 冯 林(1969-)，男，博士，教授，E-mail：fenglin@dlut.edu.cn；金 博*(1978-)，男，博士，副教授，E-mail：yaodoctor@gmail.com.

最终的并行计算思想将二次规划问题时的最小序列求解方式进行并行化处理，大大加快了整体计算过程. Caruana 等[9]将支持向量机模型的最优分割超平面计算过程中的朝向和偏置问题，通过并行化思想进行处理，有利于大数据处理. 尽管上述方法对数据流分类问题提供了一些解决的方法与思路，但仍然无法克服数据流概念漂移问题. 对此，Zhao 等[10]提出一种增量式支持向量机模型，通过分类模型进行实时更新的方式克服概念漂移问题，但由于求解过程中进行了多次迭代操作，对 MapReduce 影响较大.

以上方法均为利用传统数据流分类方法，并在此基础上进行并行化处理. 但在数据流环境下，并行环境不仅仅要提高运行速度，还要针对数据流概念漂移问题进行针对性设计，使得分类模型能同时满足数据流实时分类与动态更新量方面要求，这无疑对 MapReduce 框架下数据流分类方法提出了更高的要求. 目前，模型并行化仍然是学术界需要重点解决的问题.

针对上述问题，本文提出一种基于 MapReduce 框架的动态数据流分类算法. 首先通过增量学习方法，对数据流中的概念漂移现象进行实时跟踪与学习；然后使用权重矩阵对分类过程中的残差进行计算与修正，以达到提升分类模型抗噪声能力的目的；而遗忘因子的引入，将新旧样本对分类模型的影响进行区别对待，从整体上增强新样本对分类模型的影响力.

1 MapReduce 框架介绍

MapReduce[11]是由谷歌公司近些年提出的一种用于大规模数据集的并行运算与存储框架. 其中最主要的两个概念是“Map（映射）”和“Reduce（化简）”，其核心思想来自于函数式编程语言. MapReduce 的运行过程由 Map 和 Reduce 阶段交替进行：在 Map 阶段，对一些独立元素组成的概念上的列表的每一个元素进行指定的操作；在 Reduce 阶段，对一个列表的元素进行适当的合并. Map 和 Reduce 过程可由如下公式描述：

$$\begin{aligned} &Map\ (k_1, v_1) \rightarrow [(k_2, v_2)] \\ &Reduce\ (k_2, [v_2]) \rightarrow [(k_3, v_3)] \end{aligned} \tag{1}$$

MapReduce 框架如图 1 所示.

在 MapReduce 框架中，将初始任务划分为 Map 任务和 Reduce 任务两部分，且两部分都使用并行处理的方式，因此 MapReduce 可以高效地处理海量数据[12].

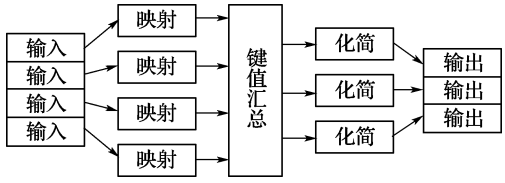


图 1 MapReduce 框架

Fig. 1 MapReduce framework

2 增量式极端支持向量机

极端支持向量机（extreme support vector machine，ESVM）是近年来发展起来的一种新的分类方法[13]，受极端学习机分类方法[14]的启发，其借鉴了极端学习机（extreme learning machine，ELM）的核函数方式，并且对初始系数及偏置使用随机给定的方式进行初始化，进而利用矩阵计算对参数进行迭代求解. 与传统 SVM 相比，ESVM 最大的优势在于训练速度的显著提高. 与 ELM 相比，由于 ESVM 计算过程中增加了结构风险最小化的约束条件，增强了其分类模型的泛化能力，从而避免在分类过程中陷入过拟合的问题. 同时，ESVM 对隐层节点数量不关注，因此相比其他分类方法参数少且稳定性强.

对二分类问题来说，假设给定训练集样本为 $\boldsymbol{S}=\{\boldsymbol{X},\boldsymbol{Y}\}$，$\boldsymbol{X}=(\boldsymbol{x}_1\ \ \boldsymbol{x}_2\ \ \cdots\ \ \boldsymbol{x}_N)^{\mathrm{T}}\in\boldsymbol{R}^{N\times d}$，表示训练集数据，$\boldsymbol{Y}=(y_1\ \ y_2\ \ \cdots\ \ y_N)^{\mathrm{T}}\in\boldsymbol{R}^{N\times 1}$，表示训练集数据对应的标签，$y_1\in[-1,1]$，$N$ 表示训练集中样本的总数量，d 表示训练集的维度. 考虑到在数据流环境下 $N\gg d$，且 d 通常小于 200，对于分类问题的 ESVM 算法，其优化条件可以公式表示为

$$\begin{aligned} &\min\ \frac{v}{2}\|\boldsymbol{\varepsilon}\|^2+\frac{1}{2}\left\|\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}\right\|^2 \\ &\text{s.t. } \boldsymbol{Y}\cdot(\boldsymbol{w}\cdot\boldsymbol{\varphi}(\boldsymbol{X})-\boldsymbol{b}\cdot\boldsymbol{e})=\boldsymbol{e}-\boldsymbol{\varepsilon} \end{aligned} \tag{2}$$

式中：$\boldsymbol{\varepsilon}=(\varepsilon_1\ \ \varepsilon_2\ \ \cdots\ \ \varepsilon_N)^{\mathrm{T}}$，为模型的误差向量；$v$ 代表误差的惩罚系数；$\boldsymbol{w}$ 和 $\boldsymbol{b}$ 代表权重向量和阈值；$\boldsymbol{e}$ 表示单位向量；$\boldsymbol{\varphi}(\boldsymbol{X})$ 代表极端学习机的核函数，其表达式如下：

$$\begin{aligned} \boldsymbol{\varphi}(\boldsymbol{X})=\boldsymbol{G}(\boldsymbol{A}\boldsymbol{x})=&\\ &\left(g\left(\sum_{j=1}^{n}A_{1j}x_j+A_{1(n+1)}\right)\ \ \cdots\right.\\ &\left.g\left(\sum_{j=1}^{n}A_{nj}x_j+A_{n(n+1)}\right)\right) \end{aligned} \tag{3}$$

式中：$\boldsymbol{A} \in \boldsymbol{R}^{d\times(d+1)}$，为输入偏置与阈值矩阵，其元素由系统随机生成；$g(x)$ 为激活函数，用于对节点进行非线性映射，使得原有线性不可分的数据变得线性可分，激活函数通常使用 Sigmoid 函数：

$$g(x) = 1/(1+\mathrm{e}^{-x}) \tag{4}$$

式(2)的目标约束条件可以表述为结构风险与经验风险的最小化，最小化结构风险可以提高分类方法的泛化性能，最小化经验风险可以使残差最小，二者通过惩罚系数对所占权重进行调整. 求解得到关于 $\boldsymbol{w}$ 和 $\boldsymbol{b}$ 的表达式：

$$\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix} = \left(\frac{\boldsymbol{I}}{\upsilon} + \boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{E}_{\varphi}\right)^{-1}\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{Y} \tag{5}$$

式中：$\boldsymbol{I}$ 为单位阵，$\boldsymbol{E}_{\varphi} = (\boldsymbol{\varphi}(\boldsymbol{Ax}) \quad -\boldsymbol{e}) \in \boldsymbol{R}^{N\times(d+1)}$. 最终极端支持向量机的判别公式如下：

$$\begin{aligned}&\boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{X})\cdot\boldsymbol{w}-\boldsymbol{b}\geqslant\boldsymbol{0};\ t=1\\ &\boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{X})\cdot\boldsymbol{w}-\boldsymbol{b}<\boldsymbol{0};\ t=-1\end{aligned} \tag{6}$$

其中 t 表示 ESVM 模型中两个支撑平面.

由式(5)可以看出，ESVM 的模型训练仅通过矩阵的运算即可得到，其结构特征便于拓展成可以进行多次学习的增量形式.

由数据集 $\boldsymbol{S}_1 = \{\boldsymbol{X}_1, \boldsymbol{Y}_1\}$ 训练得到的极端向量机形式为

$$\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix} = \left(\frac{\boldsymbol{I}}{\upsilon} + \boldsymbol{E}_{\varphi 1}^{\mathrm{T}}\boldsymbol{E}_{\varphi 1}\right)^{-1}\boldsymbol{E}_{\varphi 1}^{\mathrm{T}}\boldsymbol{Y}_1 \tag{7}$$

此时需要在 $\boldsymbol{S}_1$ 的基础上增量学习一个新的数据集 $\boldsymbol{S}_2 = \{\boldsymbol{X}_2, \boldsymbol{Y}_2\}$，可以将式(7)修改为以下的形式：

$$\begin{aligned}\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix} &= \left(\frac{\boldsymbol{I}}{\upsilon} + (\boldsymbol{E}_{\varphi 1}^{\mathrm{T}} \quad \boldsymbol{E}_{\varphi 2}^{\mathrm{T}})\begin{pmatrix}\boldsymbol{E}_{\varphi 1}\\ \boldsymbol{E}_{\varphi 2}\end{pmatrix}\right)^{-1}(\boldsymbol{E}_{\varphi 1}^{\mathrm{T}} \quad \boldsymbol{E}_{\varphi 2}^{\mathrm{T}})\begin{pmatrix}\boldsymbol{Y}_1\\ \boldsymbol{Y}_2\end{pmatrix} =\\ &\left(\frac{\boldsymbol{I}}{\upsilon} + \boldsymbol{E}_{\varphi 1}^{\mathrm{T}}\boldsymbol{E}_{\varphi 1} + \boldsymbol{E}_{\varphi 2}^{\mathrm{T}}\boldsymbol{E}_{\varphi 2}\right)^{-1}(\boldsymbol{E}_{\varphi 1}^{\mathrm{T}}\boldsymbol{Y}_1 + \boldsymbol{E}_{\varphi 2}^{\mathrm{T}}\boldsymbol{Y}_2)\end{aligned} \tag{8}$$

式(8)即为 ESVM 的增量形式(incremental ESVM，IESVM)，其中 $\boldsymbol{E}_{\varphi 1}^{\mathrm{T}}\boldsymbol{E}_{\varphi 1}$ 及 $\boldsymbol{E}_{\varphi 1}^{\mathrm{T}}\boldsymbol{Y}_1$ 与式(7)相同，因此无须重复计算，直接使用式(7)的计算结果即可.

3　极端支持向量机的并行算法

通过前面介绍，对于 MapReduce 框架以及极端学习机及其增量式学习方法已有所了解，受到这两种方法的启发，本文在原有算法的基础上，提出一种基于极端支持向量机的并行算法，其核心思想可以描述为对式(5)进行修改，将 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{E}_{\varphi}$ 和 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{Y}$ 展开得到：

$$\begin{aligned}&\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{E}_{\varphi} = (\boldsymbol{a}_1 \quad \cdots \quad \boldsymbol{a}_N)(\boldsymbol{a}_1 \quad \cdots \quad \boldsymbol{a}_N)^{\mathrm{T}} = \sum_{i=1}^{N}\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}} \in \boldsymbol{R}^{(d+1)\times(d+1)}\\ &\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{Y} = (\boldsymbol{a}_1 \quad \cdots \quad \boldsymbol{a}_N)(y_1 \quad \cdots \quad y_N)^{\mathrm{T}} = \sum_{i=1}^{N}y_i\boldsymbol{a}_i \in \boldsymbol{R}^{(d+1)\times 1}\end{aligned} \tag{9}$$

在动态数据流环境中，$N \gg d$，通常 $d < 200$，因此所提出方法的计算量主要集中在 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{E}_{\varphi}$ 和 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{Y}$. 由式(9)可知，计算 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{E}_{\varphi}$ 和 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{Y}$ 使用小矩阵叠加的方式进行，因此在 MapReduce 框架下步骤为，在 Map 阶段，对局部 $\sum_{i=1}^{\hbar}\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}$ 和 $\sum_{i=1}^{\hbar}y_i\boldsymbol{a}_i$ 进行并行计算；在 Reduce 阶段，对局部计算所得到的结果进行合并，最终得到 $\sum_{i=1}^{N}\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}$ 和 $\sum_{i=1}^{N}y_i\boldsymbol{a}_i$.

算法 1　ESVM 并行训练算法

步骤 1　随机生成权重矩阵 $\boldsymbol{A}$.

步骤 2　在 Map 过程中对文件中输入的每条数据，利用式(3)计算 $\boldsymbol{\varphi}(x_i)$，构造 $\boldsymbol{a}_i = (\boldsymbol{\varphi}(x_i) \quad -1)^{\mathrm{T}}$，计算 $local\_\boldsymbol{aa}^{\mathrm{T}} = \sum\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}$，$local\_\boldsymbol{ya} = \sum y_i\boldsymbol{a}_i$.

步骤 3　在 Reduce 过程中计算 $global\_\boldsymbol{aa}^{\mathrm{T}} = global\_\boldsymbol{aa}^{\mathrm{T}} + \sum local\_\boldsymbol{aa}^{\mathrm{T}}$，$global\_\boldsymbol{ya} = global\_\boldsymbol{ya} + \sum local\_\boldsymbol{ya}$.

步骤 4　根据式(5)或式(8)计算 $\boldsymbol{w}$、$\boldsymbol{b}$.

4　时间遗忘的鲁棒极端支持向量机

使用 IESVM 模型对数据流进行分类的基本思想是，首先利用最初得到的样本集对分类模型进行训练，构建最初的分类模型. 然后对新样本数据，根据式(8)对模型进行增量式学习，即模型重新训练，并以此克服数据流中概念漂移问题.

ESVM 模型存在两个不可避免的问题：一方面由于所提出分类模型使用了核函数，其参数的初始值为随机给定的，造成 ESVM 模型训练结果的波动；另一方面，尽管使用增量式学习方式对模型进行更新，但学习过程中对样本的新旧不进行区别，以相同的权重进行学习.

针对上述两个问题，本文利用时间遗忘机制和核密度估计，提出一种鲁棒极端支持向量机算法(forgetting factor robust ESVM，FFR-ESVM). FFR-ESVM 利用核密度估计方法，对训练集误差的概率分布进行估计，进而利用估计结

果,构造权重矩阵对残差进行修正,提升 ESVM 的抗干扰能力,并通过遗忘因子提高新样本对模型的影响力,以适应当前数据环境的要求.

4.1 鲁棒性描述

利用训练集分类误差参数,所提出模型的训练结果可以表示为

$$\boldsymbol{\varphi}^{\mathrm{T}}(\boldsymbol{X})\cdot\boldsymbol{w}-\boldsymbol{b}\cdot\boldsymbol{e}=(\boldsymbol{\varphi}(\boldsymbol{X})\quad -\boldsymbol{e})\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}=\boldsymbol{E}_{\varphi}\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}=\boldsymbol{Y}+\boldsymbol{\varepsilon} \tag{10}$$

这里需要注意的是,当误差分布服从正态分布 $N(0,\sigma^2)$ 时,其经验风险的最小二乘估计值为最优情况.但在实际数据流环境下,误差估计值很难服从正态分布,因此对分类模型的泛化能力影响较大,尤其训练数据集中噪声数据较多的情况下,模型的泛化性能会大打折扣.

针对此问题,本文利用权重矩阵方法对所产生的误差进行修正.假设残差权重矩阵 $\boldsymbol{P}$,其对应的目标函数为

$$\min\ \frac{v}{2}\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{\varepsilon}+\frac{1}{2}\left\|\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}\right\|^2$$

将式(10)代入目标函数求解,可以得到 ESVM的鲁棒形式(robust ESVM,R-ESVM):

$$\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}=(\boldsymbol{I}+v\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E}_{\varphi})^{-1}\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{Y} \tag{11}$$

通过观察发现,一般噪声数据均分布于正常数据的周边,因此在已知残差概率分布的情况下,可以根据残差 $\boldsymbol{\varepsilon}$ 的分布,构造权重矩阵 $\boldsymbol{P}=\mathbf{diag}\{p(\varepsilon_1),p(\varepsilon_2),\cdots,p(\varepsilon_N)\}$,其中 $p(\boldsymbol{\varepsilon})$ 表示残差变量 $\boldsymbol{\varepsilon}$ 的概率分布函数.由于 $\boldsymbol{\varepsilon}$ 的概率分布无法提前获取,需要利用式(5)使用随机的方式先进行一次计算.假设所得到的估计值为 $\tilde{\boldsymbol{\varepsilon}}$,以它为基础代入计算,后续计算均在此基础上再对 $p(\boldsymbol{\varepsilon})$ 进行估计.其公式如下:

$$p(\boldsymbol{\varepsilon})=\frac{1}{N}\sum_{i=1}^{N}f\left(\frac{\boldsymbol{\varepsilon}-\tilde{\boldsymbol{\varepsilon}}_i}{h_N}\right) \tag{12}$$

式中:N 为样本数目;$h_N=h/\sqrt{N}$,表示滑动窗口大小;$f(x)$ 表示窗口滑动步长函数,选取正态窗口滑动函数可以获得较平滑的概率密度估计,最终计算公式如下:

$$f(x)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right) \tag{13}$$

4.2 遗忘因子

根据式(11)可知,R-ESVM 的增量形式可以表示为

$$\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}=\left(\boldsymbol{I}+v(\boldsymbol{E}_{\varphi1}^{\mathrm{T}}\quad\boldsymbol{E}_{\varphi2}^{\mathrm{T}})\begin{pmatrix}\boldsymbol{P}_1&\boldsymbol{0}\\ \boldsymbol{0}&\boldsymbol{P}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{E}_{\varphi1}\\ \boldsymbol{E}_{\varphi2}\end{pmatrix}\right)^{-1}\times(\boldsymbol{E}_{\varphi1}^{\mathrm{T}}\quad\boldsymbol{E}_{\varphi2}^{\mathrm{T}})\begin{pmatrix}\boldsymbol{P}_1&\boldsymbol{0}\\ \boldsymbol{0}&\boldsymbol{P}_2\end{pmatrix}\begin{pmatrix}\boldsymbol{Y}_1\\ \boldsymbol{Y}_2\end{pmatrix}=(\boldsymbol{I}+v\boldsymbol{E}_{\varphi1}^{\mathrm{T}}\boldsymbol{P}_1\boldsymbol{E}_{\varphi1}+v\boldsymbol{E}_{\varphi2}^{\mathrm{T}}\boldsymbol{P}_2\boldsymbol{E}_{\varphi2})^{-1}\times(\boldsymbol{E}_{\varphi1}^{\mathrm{T}}\boldsymbol{P}_1\boldsymbol{Y}_1+\boldsymbol{E}_{\varphi2}^{\mathrm{T}}\boldsymbol{P}_2\boldsymbol{Y}_2) \tag{14}$$

由式(14)可知,R-ESVM 存在的问题是增量学习中不区分样本的新旧,而一视同仁加以训练.但在实际数据流环境下,相比于旧样本,新样本往往更能反映当前数据流的情况,具有更大的价值.因此,对样本的新旧进行区别对待是十分必要的.在所提出模型中,通过引入遗忘因子 θ 来提高新样本的影响力.式(14)中,对 $\boldsymbol{E}_{\varphi1}$ 乘以遗忘因子 θ:

$$\boldsymbol{E}'_{\varphi1}=\theta\boldsymbol{E}_{\varphi1} \tag{15}$$

式中:$0\leqslant\theta\leqslant1$.将式(15)代入式(14)可得到 FFR-ESVM 的增量更新公式:

$$\begin{pmatrix}\boldsymbol{w}\\ \boldsymbol{b}\end{pmatrix}=(\boldsymbol{I}+v\theta^2\boldsymbol{E}_{\varphi1}^{\mathrm{T}}\boldsymbol{P}_1\boldsymbol{E}_{\varphi1}+v\boldsymbol{E}_{\varphi2}^{\mathrm{T}}\boldsymbol{P}_2\boldsymbol{E}_{\varphi2})^{-1}\times(\theta\boldsymbol{E}_{\varphi1}^{\mathrm{T}}\boldsymbol{P}_1\boldsymbol{Y}_1+\boldsymbol{E}_{\varphi2}^{\mathrm{T}}\boldsymbol{P}_2\boldsymbol{Y}_2) \tag{16}$$

当 $\theta=1$ 时,式(16)与式(14)相同;当 $\theta=0$ 时,式(16)与式(11)相同,即训练集中全部为新样本数据.因此,可以通过遗忘因子 θ 来调整新旧样本的影响力.

根据式(16),将 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E}_{\varphi}$ 和 $\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{Y}$ 展开得到:

$$\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{E}_{\varphi}=(\boldsymbol{a}_1\quad\cdots\quad\boldsymbol{a}_N)\begin{pmatrix}p_1&\cdots&0\\ \vdots&&\vdots\\ 0&\cdots&p_N\end{pmatrix}\times(\boldsymbol{a}_1\quad\cdots\quad\boldsymbol{a}_N)^{\mathrm{T}}=\sum_{i=1}^{N}\boldsymbol{p}_i\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}\in\boldsymbol{R}^{(d+1)\times(d+1)} \tag{17}$$

$$\boldsymbol{E}_{\varphi}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{Y}=(\boldsymbol{a}_1\quad\cdots\quad\boldsymbol{a}_N)\begin{pmatrix}p_1&\cdots&0\\ \vdots&&\vdots\\ 0&\cdots&p_N\end{pmatrix}\times(y_1\quad\cdots\quad y_N)^{\mathrm{T}}=\sum_{i=1}^{N}y_i\boldsymbol{p}_i\boldsymbol{a}_i\in\boldsymbol{R}^{(d+1)\times1} \tag{18}$$

与 IESVM 类似,可以在 Map 阶段计算局部 $\sum_{i=1}^{\hbar}\boldsymbol{p}_i\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}$ 和 $\sum_{i=1}^{\hbar}y_i\boldsymbol{p}_i\boldsymbol{a}_i$ 值,在 Reduce 阶段将局部计算得到的结果进行合并,最终得到 $\sum_{i=1}^{N}\boldsymbol{p}_i\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}$ 和 $\sum_{i=1}^{N}y_i\boldsymbol{p}_i\boldsymbol{a}_i$.

算法 2 FFR-ESVM 并行训练算法

步骤 1　随机生成权重矩阵 $\boldsymbol{A}$.

步骤 2　利用算法 1 训练模型并计算残差矩阵估计值 $\tilde{\boldsymbol{\varepsilon}}$.

步骤 3　利用式(12) 估算 $p(\boldsymbol{\varepsilon})$ 并计算权重矩阵 $\boldsymbol{P}$.

步骤 4　在 Map 过程中对于文件中输入的每条数据，根据式(3) 计算 $\boldsymbol{\varphi}(x_i)$，构造 $\boldsymbol{a}_i = (\boldsymbol{\varphi}(x_i) \quad -1)^{\mathrm{T}}$，计算 $local\_\boldsymbol{paa}^{\mathrm{T}} = \sum \boldsymbol{p}_i\boldsymbol{a}_i\boldsymbol{a}_i^{\mathrm{T}}$，$local\_\boldsymbol{ypa} = \sum y_i\boldsymbol{p}_i\boldsymbol{a}_i$.

步骤 5　在 Reduce 过程中计算 $global\_\boldsymbol{paa}^{\mathrm{T}} = \theta^2 \cdot global\_\boldsymbol{paa}^{\mathrm{T}} + \sum local\_\boldsymbol{paa}^{\mathrm{T}}$，$global\_\boldsymbol{ypa} = \theta \cdot global\_\boldsymbol{ypa} + \sum local\_\boldsymbol{ypa}$.

步骤 6　根据式(16) 计算 $\boldsymbol{w}$、$\boldsymbol{b}$.

5　实验分析

本实验采用人造数据来模拟动态数据流情况，数据使用概念漂移数据流生成工具(Minku 的概念漂移数据流生成工具可从 http://www.cs.bham.ac.uk/~minkull/opensource.html 得到)自动随机生成，其生成公式如下：

$$y \leqslant -a_0 + \sum_{i=1}^{d} a_i x_i \tag{19}$$

利用上述工具，总共生成 5 个数据文件，定量为 $d=7$，$a_1=a_2=a_3=a_4=a_5=a_6=1$，每个文件代表一个独立的概念. 所有文件总计 100 000 000 条数据，总文件大小为 7.43 GB，具体参数如表 1 所示.

实验模型中 FF-IELM (forgetting factor IELM)与 ELM 模型的隐层节点数为 $n_{\mathrm{h}}=20$，ESVM、FF-IESVM(forgetting factor IESVM)、FFR-ESVM 残差的惩罚系数 $v=100$，FFR-ESVM中核密度估计参数 $h=20$. 为了对所提出模型的性能进行详细分析，特此给出额外 3 种度量方式，即加速比(α)、可扩展性(β) 和规模增长性(γ)：

$$\alpha(m) = \frac{1\text{ 个节点 }1\text{ 份数据的处理时间}}{m\text{ 个节点 }1\text{ 份数据的处理时间}}$$

$$\beta(m) = \frac{1\text{ 个节点 }1\text{ 份数据的处理时间}}{m\text{ 个节点 }m\text{ 份数据的处理时间}}$$

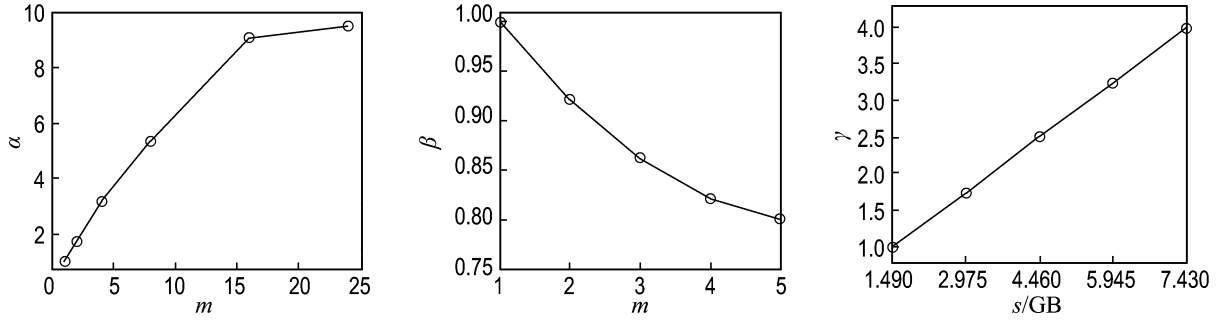
$$\gamma(m) = \frac{1\text{ 个节点 }m\text{ 份数据的处理时间}}{1\text{ 个节点 }1\text{ 份数据的处理时间}}$$

表 1　人造数据参数

Tab. 1　Synthetic dataset parameters

数据文件	变量 a_0	数据数量	正类所占比例/%
概念 1	3.0	20 000 000	57.6
概念 2	1.5	20 000 000	44.4
概念 3	0	20 000 000	50.0
概念 4	−1.5	20 000 000	50.2
概念 5	−3.0	20 000 000	46.6

加速比参数主要描述所提出算法与增加计算节点的关系，可扩展性参数用于描述所提出算法对数据集大小的处理能力，规模增长性参数用于描述所提出算法的时间复杂度，运行结果如图 2 所示. 从图 2(a)中可以看到，FFR-ESVM 算法的加速比结果分为两部分：当节点数介于 1 到 16 时，算法呈线性增长趋势，其原因为 FFR-ESVM 算法将大部分计算过程并行化，算法的加速性能与节点数成正比关系；但当节点数大于 16 时，受到 Hadoop 环境在处理数据时以 64 MB 作为单位对数据进行划分的限制，算法效率提升放缓趋向平稳. 从图 2(b)可以看到，FFR-ESVM 算法的可拓展性随着节点及数据规模的增加呈缓慢下降趋势，因此具有良好的可拓展性. 从图 2(c)可以看出，FFR-ESVM 算法具有良好的规模增长性.



(a) 加速比　　(b) 可扩展性　　(c) 规模增长性

图 2　FFR-ESVM 算法的加速比、可扩展性和规模增长性

Fig. 2　Speedup, scaleup and sizeup of FFR-ESVM algorithm

下面验证 R-ESVM 算法的有效性，使用数据概念 1 对 ELM、ESVM 以及 R-ESVM 算法进行实验，结果如表 2 所示. 由于偏置参数为随机生成，故将每种算法运行 5 次取平均值. 从表 2 中可以看出，R-ESVM 抵抗噪声的能力要优于 ELM 及 ESVM 模型，但由于 R-ESVM 算法需要对数据进行两次扫描，且 Hadoop 中无法在内存中保存数据，增加了时间消耗，可通过增加计算节点的方式进行解决. 同时，在实际实验中 R-ESVM 算法处理 20 000 000 条数据需 7 min，完全能够满足动态数据流实时分类要求.

表 2 ELM、ESVM、R-ESVM 算法性能比较

Tab. 2 Performance comparison of ELM, ESVM and R-ESVM algorithm

算法	准确率/%	时间/min
ELM	87.45±0.36	**2.24±0.05**
ESVM	88.48±0.26	2.72±0.44
R-ESVM	**90.14±0.37**	7.22±0.56

设计以下实验模拟动态数据流测试 FFR-ESVM 算法的有效性，步骤如下：

步骤 1 令 $i=1$，利用概念 i 数据生成模型 i.

步骤 2 利用模型 i 对概念 $i+1$ 数据进行预测，得到预测结果.

步骤 3 利用数据 $i+1$ 对模型 i 进行增量式学习，得到模型 $i+1$.

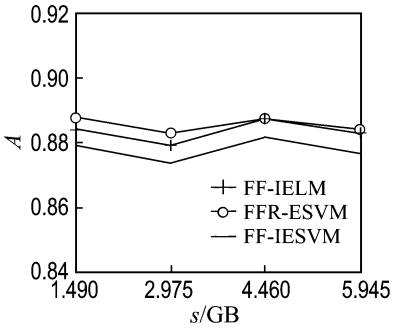
步骤 4 令 $i=i+1$，转步骤 2.

图 3 为 FF-IELM、FF-IESVM 以及 FFR-ESVM算法在遗忘因子 $\theta=1.0$ 及 $\theta=0.5$ 时运行上述实验步骤 5 次的准确率变化情况.

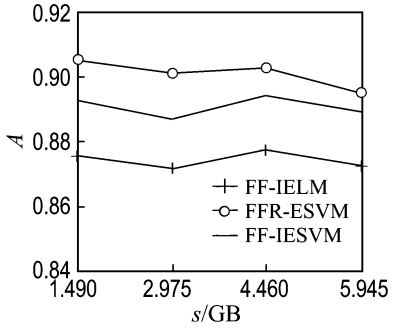
从图 3 中可以看出，FF-IESVM 与 FFR-ESVM算法在 $\theta=0.5$ 时准确率明显要优于 $\theta=1.0$ 时. 这是因为当 $\theta=1.0$ 时，新样本和旧样本对模型的权重相同，此时遗忘因子不产生作用. 对 FF-IELM 来说，其误差计算使用最小二乘法，因此极易受到噪声数据的干扰，其泛化能力与另外两种算法相比差一些.

图 4 为 FF-IELM、FF-IESVM 以及 FFR-ESVM算法在遗忘因子 $\theta=1.0$ 及 $\theta=0.5$ 时运行上述实验步骤 5 次的平均时间消耗情况.

从图 4 中可以看出，由于增量学习公式形式不变，遗忘因子的引入对 3 种算法的运行时间消耗影响不大. 此外，因为 FFR-ESVM 算法需要对文件进行两次读取，故时间消耗较大，约为其他算



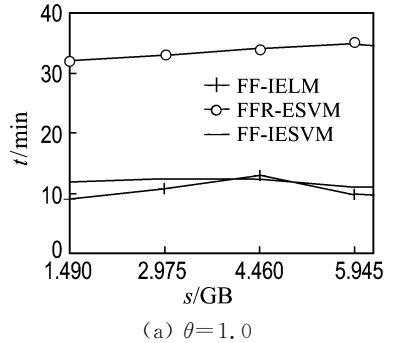
(a) $\theta=1.0$



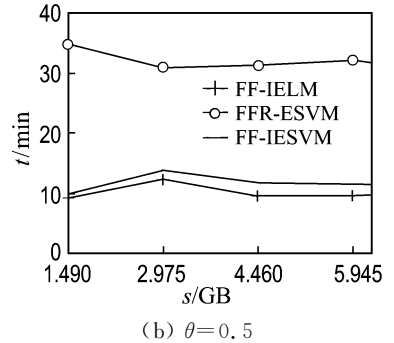
(b) $\theta=0.5$

图 3 3 种算法在遗忘因子 $\theta=1.0$ 及 $\theta=0.5$ 的准确率变化情况

Fig. 3 Accuracy of three algorithms with forgetting factor $\theta=1.0$ and $\theta=0.5$



(a) $\theta=1.0$



(b) $\theta=0.5$

图 4 3 种算法在遗忘因子 $\theta=1.0$ 及 $\theta=0.5$ 的时间变化情况

Fig. 4 Time cost of three algorithms with forgetting factor $\theta=1.0$ and $\theta=0.5$

法所消耗时间的 3 倍左右. 尽管如此，FFR-ESVM 模型仍然可以在 30 min 内处理完毕 100 000 000 条数据，基本满足实时分类的要求.

当 $\theta=0.5$ 时，FF-IELM、FF-IESVM 以及 FFR-ESVM 算法 5 次运行的平均分类准确率、灵敏度、特指度的结果如表 3 所示. 其中，灵敏度和特指度的定义如下：

$$\text{灵敏度}=\frac{\text{分类为正的正样本数}}{\text{分类为正的正样本数}+\text{分类为负的正样本数}}$$

$$\text{特指度}=\frac{\text{分类为负的负样本数}}{\text{分类为负的负样本数}+\text{分类为正的负样本数}}$$

由表 3 可以看出，FFR-ESVM 算法的整体准确率和灵敏度要优于 FF-IELM 和 FF-IESVM 两种方法，但特指度与 FF-IESVM 算法差距不明显. 综合分析，在数据流环境中，所提出算法使用残差权重矩阵对残差最小二乘误差值进行修正，并通过遗忘因子对样本影响力进行调整，可以提高分类模型的适应能力. 此外，从计算量角度分析，由于所提出的 FFR-ESVM 算法，是在传统的 ESVM 模型基础上增加了核函数选择和遗忘因子，因此，相比传统 ESVM 模型，计算量势必增加. 但是由于核函数选择计算过程使用了 MapReduce 技术，将选择过程进行并行化，可以将 $O(n)$ 的计算量压缩为 $O(1)$，因此计算过程几乎可以忽略不计. 此外，所提出模型使用遗忘因子对样本权重进行分配时，计算量随着样本空间发生变化，因此也为 $O(n)$. 与其他算法进行比较，本文算法由于使用 MapReduce 技术，将所涉及的计算过程并行化，从而在时间复杂度上并没有明显提升，反而在分类准确率方面有所提高，与其他分类方法（FF-IELM 和 FF-IESVM）相比，计算量增加程度有限.

表 3　FF-IELM、FF-IESVM 与 FFR-ESVM 算法整体性能比较

Tab. 3　Performance comparison of FF-IELM, FF-IESVM and FFR-ESVM algorithm

阶段	准确率/%			灵敏度/%			特指度/%		
	FF-IELM	FF-IESVM	FFR-ESVM	FF-IELM	FF-IESVM	FFR-ESVM	FF-IELM	FF-IESVM	FFR-ESVM
1	87.58±1.23	89.30±1.33	**90.56±1.64**	82.38±1.63	84.86±2.00	**87.60±4.01**	91.98±0.96	**93.14±1.22**	93.12±1.27
2	87.18±1.22	88.70±1.23	**90.14±1.77**	82.20±1.39	83.30±1.80	**86.32±3.38**	92.14±1.13	**94.06±1.13**	94.00±0.98
3	87.74±1.20	89.44±1.30	**90.30±1.19**	88.76±1.39	88.76±1.70	**91.94±2.28**	86.90±1.23	**90.02±1.25**	88.92±0.90
4	87.30±1.15	88.92±1.23	**89.54±0.89**	90.62±1.45	91.58±1.72	**91.96±4.77**	84.84±1.11	86.94±1.19	**87.02±1.88**

6　结　语

本文分析了国内外动态环境下数据流分类技术的相关研究进展，针对动态数据流的数据海量性、概念漂移等特点，利用构造残差权重矩阵对ESVM的残差矩阵进行修正，并结合遗忘因子，提出了一种FFR-ESVM算法. FFR-ESVM可以对不同概念漂移类型进行判断，使得分类模型能够及时更新，提高分类模型的分类准确率和稳定性. 实验中，所提出算法与 FF-IELM 及 FF-IESVM 算法的比较证实了模型的性能，后续将针对其他分类模型在 MapReduce 框架下的应用展开研究，进一步提高分类准确率.

参考文献：

[1] 黄树成，曲亚辉. 数据流分类技术研究综述[J]. 计算机应用研究，2009，**26**(10)：3604-3609.
HUANG Shu-cheng, QU Ya-hui. Survey on data stream classification technologies [J]. **Application Research of Computers**, 2009, **26**(10): 3604-3609. (in Chinese)

[2] 尹志武，黄上腾. 一种自适应局部概念漂移的数据流分类算法[J]. 计算机科学，2008，**35**(2)：138-139，143.
YIN Zhi-wu, HUANG Shang-teng. Adaptive method for handling local concept drift of data streams classification [J]. **Computer Science**, 2008, **35**(2): 138-139, 143. (in Chinese)

[3] 陈照阳，黄上腾. 流数据分类中的概念漂移问题研究[J]. 计算机应用与软件，2009，**26**(2)：254-256，279.
CHEN Zhao-yang, HUANG Shang-teng. On concept drift in stream data classification [J]. **Computer Applications and Software**, 2009, **26**(2): 254-256, 279. (in Chinese)

[4] 吴　枫，仲　妍，吴泉源. 基于增量核主成分分析的数据流在线分类框架[J]. 自动化学报，2010，**36**(4)：534-542.
WU Feng, ZHONG Yan, WU Quan-yuan. Online classification framework for data stream based on incremental kernel principal component analysis [J]. **Acta Automatica Sinica**, 2010, **36**(4): 534-542. (in Chinese)

[5] 欧阳震诤,罗建书,胡东敏,等. 一种不平衡数据流集成分类模型[J]. 电子学报, 2010, **38**(1):184-189.
OUYANG Zhen-zheng, LUO Jian-shu, HU Dong-min, *et al*. An ensemble classifier framework for mining imbalanced data streams [J]. **Acta Electronica Sinica**, 2010, **38**(1): 184-189. (in Chinese)

[6] ZHANG Chi, LI Fei-fei, Jestes J. Efficient parallel kNN joins for large data in MapReduce [C] // **15th International Conference on Extending Database Technology, EDBT 2012**. New York: Association for Computing Machinery, 2012:38-49.

[7] Pitchaimalai S K, Ordonez C, Garcia-Alvarado C. Comparing SQL and MapReduce to compute naive Bayes in a single table scan [C] // **The 2nd International Workshop on Cloud Data Management, CloudDB'10**. New York: Association for Computing Machinery, 2010:9-16.

[8] Alham N K, LI Mao-zhen, Hammoud S, *et al*. A distributed SVM for image annotation [C] // **2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010**. Piscataway: IEEE Computer Society, 2010:2983-2987.

[9] Caruana G, LI Mao-zhen, QI Man. A MapReduce based parallel SVM for large scale spam filtering [C] // **2011 8th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2011**. Piscataway: IEEE Computer Society, 2011:2659-2662.

[10] ZHAO Jun, LIANG Zhu, YANG Yong. Parallelized incremental support vector machines based on MapReduce and Bagging technique [C] // **2012 IEEE International Conference on Information Science and Technology, ICIST 2012**. Piscataway: IEEE Computer Society, 2012:297-301.

[11] Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters [J]. **Communications of the ACM**, 2008, **51**(1):107-113.

[12] 黄 詠,易晓东,李姗姗,等. 面向高性能计算机的海量数据处理平台实现与评测[J]. 计算机研究与发展, 2012, **49**(S1):357-361.
HUANG He, YI Xiao-dong, LI Shan-shan, *et al*. Implementation and evaluation of massive data processing paradigm on high performance computers [J]. **Journal of Computer Research and Development**, 2012, **49**(S1):357-361. (in Chinese)

[13] LIU Qiu-ge, HE Qing, SHI Zhong-zhi. Extreme support vector machine classifier [C] // **12th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2008**. Heidelberg: Springer Verlag, 2008:222-233.

[14] HUANG Guang-bin, ZHU Qin-yu, Siew Chee-kheong. Extreme learning machine: Theory and applications [J]. **Neurocomputing**, 2006, **70**(1-3): 489-501.

A dynamic data stream classification algorithm based on MapReduce

FENG Lin[1,2], YAO Yuan[3], CHEN Feng[1], JIN Bo[*2]

(1. School of Computer Science and Technology, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian 116024, China;
2. School of Innovation Experiment, Dalian University of Technology, Dalian 116024, China;
3. College of Information and Communication Engineering, Dalian Nationalities University, Dalian 116600, China)

Abstract: There are three difficulties in real-time dynamic data stream classification: real-time processing of massive data, tracking of concept drift and model updates, model's stability and robustness. To solve these problems, extreme support vector machine (ESVM) is combined with MapReduce framework, and a forgetting factor robust ESVM algorithm (FFR-ESVM) is proposed. The proposed algorithm amends the residuals by constructing a residual matrix, while improves the effect of new samples by forgetting factor. Experimental results show that the proposed algorithm can rapidly and effectively classify dynamic data stream, and the results are stable and less affected by noise interference.

Key words: data stream classification; incremental learning; extreme support vector machine (ESVM); MapReduce; forgetting factor; robustness