Accelerating Relevance Vector Machine for Large-Scale Data on Spark

Fang LIU[1*], Hao ZHONG[2], Si-Han LI[2]

[1] National Engineering Laboratory for Fiber Optic Sensing Technology,Wuhan University of Technology, Wuhan 430070, China
[2] School of Computer Science and Technology,Wuhan University of Technology, Wuhan 430070, China
fangliu@whut.edu.cn, klordy@163.com, lsh751874894@163.com

Abstract: Relevance vector machine (RVM) is a machine learning algorithm based on a sparse Bayesian framework, which performs well when running classification and regression tasks on small-scale datasets. However, RVM also has certain drawbacks which restricts its practical applications such as (1) slow training process, (2) poor performance on training large-scale datasets. In order to solve these problem, we propose Discrete AdaBoost RVM (DAB-RVM) which incorporate ensemble learning in RVM at first. This method performs well with large-scale low-dimensional datasets. However, as the number of features increases, the training time of DAB-RVM increases as well. To avoid this phenomenon, we utilize the sufficient training samples of large-scale datasets and propose all features boosting RVM (AFB-RVM), which modifies the way of obtaining weak classifiers. In our experiments we study the differences between various boosting techniques with RVM, demonstrating the performance of the proposed approaches on Spark. As a result of this paper, two proposed approaches on Spark for different types of large-scale datasets are available.

1 Introduction

With the rapid development of the Internet, data based on Internet statistics and analysis have mostly become large-scale or even massive datasets. Organizing and classifying such large-scale and fast-growing data efficiently and precisely is a difficult problem. When using traditional platforms for data analysis, the majority of classification algorithms run serially. Moreover, operating platforms are largely limited by computer performance, when the amount of data increases, the performance of serial algorithms is sharply reduced.

The Relevance Vector Machine (RVM) [1] is a new supervised learning method. Unlike Support Vector Machine (SVM) [2], RVM is a Bayesian-based probabilistic sparse model. By means of Gaussian prior probabilities affected by hyperparameters, a machine learning task is performed with the Bayesian frame, and autocorrelation determination (ARD) is used to remove uncorrelated points and obtain a sparse model. Because the posterior distribution of most parameters tends toward zero during iterative learning, the training samples corresponding to non-zero parameters are not related to the decision domain samples and represent only the prototype samples in the training data. Such samples are called relevance vector, which embody the core features of the data. The biggest advantage of RVM is that they greatly reduce the computation of kernel functions, and the choice of kernel function is no longer limited by Mercer's condition. Based on the above advantages, RVM has obtained good performance in fields such as image recognition [3] and speech recognition [4].

The time complexity of RVM algorithm is $O(n^3)$ and its space complexity is $O(n^2)$. Therefore, when the number of samples that must be processed increases, the time and resources required for RVM training increase dramatically, resulting in a serious decrease in the algorithm's efficiency. In order to solve this problem, Tipping and Faul [5] proposed a fast marginal likelihood maximization method that refreshes one coefficient at a time. However, this method can get stuck in a suboptimal solution. Seeger and Ribeiro [6] applied RVM to large-scale text sets using ensemble, boosting and incremental methods, which integrates ensemble learning with RVM. This strategy performs well in accelerating the training procedure. Yang et al. [7] proposed recursive Cholesky decomposition for RVM on GPUs. This strategy was proved to be much faster, but still could not run on large-scale datasets. DIMITRIS and ARISTIDIS [8] proposed a modified RVM algorithm based on expectation maximization for image recognition. This method effectively improved the time efficiency of RVM for large-scale image recognition. However, this method only considers the time efficiency of RVM for large amounts of data, and did not solve the problem of resource consumption. Chao Dong and Lianfang Tian [9] used the idea proposed by Seeger and Ribeiro to apply RVM to a large amount of data in the Beowulf Cluster, which allows the algorithm to successfully handle large image recognition datasets in a cluster environment; however, the algorithm is difficult to implement and the efficiency of the cluster when running such tasks was poor.

In recent years, distributed technology has been developing rapidly, and many distributed frameworks have emerged. The most widely used distributed frameworks include Storm, Hadoop and Spark. Using these distributed frameworks, the efficiency of algorithms can be improved effectively. For processing iterative computing, traditional MapReduce has the advantages of automatic fault tolerance, balanced load and scalability; its acyclic data flow model, however, causes a large number of disk IO operations, which greatly limits its performance. Compared with MapReduce, Spark runs iterative calculation tasks based on memory. It loads the data in memory using Resilient Distributed Dataset [10], making them easy to reuse and accelerating iterative computation dramatically. Considering the performance of Spark in memory iteration, we choose to implement the algorithm on the Spark platform.

The content of this paper revolves around the fundamental principles of RVM and ensemble learning [11]. We incorporate the Discrete AdaBoost and the modified AdaBoost in RVM and obtain two algorithms for large-scale datasets. Next section briefly reviews the basic process of RVM to lay the foundations of the proposed approaches. Section III introduces the basic concept of the AdaBoost algorithm, as well as the modification and combination of the AdaBoost algorithm with RVM algorithm on the Spark platform. Section IV mainly describes the detailed implementation of the experiments on the Spark platform and the analysis of relevant experimental results.

2 Relevance Vector Machine

RVM is a machine learning algorithm based on Bayesian theory. It uses a maximum likelihood function to obtain hyperparameters, and then assigns zero to the weights according to the hyperparameters to guarantee the sparsity of the algorithm. Given a sample set $(x_{i1},...,x_{iM},y_i), i=1,2,...,N$, where $X=(x_{i1},...,x_{iM})$ is the feature vector of the sample and $y_i \in Y=\{-1,+1\}$ is the class label of each sample, the model of RVM can be represented as follows:

$$y(x)=\sum_{n=1}^{N} w_n K(x,x_n)+w_0 \tag{1}$$

where $K(\cdot,\cdot)$ is the kernel function and w_n represents the weight of all the samples. Additionally, RVM uses a Bernoulli distribution to construct the probability density function as:

$$p(t_i \mid x)=N(t_i \mid y(x_i;w),\sigma^2) \tag{2}$$

where the expected value is $y(x_i;w)$ and variance is σ^2.

In order to avoid over-fitting, RVM defines a Gauss prior probability distribution as below:

$$p(w \mid \alpha)=\prod_{i=0}^{N} N(w_i \mid 0,\alpha_i^{-1}) \tag{3}$$

where α is a ($N+1$)-dimensional hyperparameter used to constrain the parameter. After many iterations, most of the hyperparameters tend toward infinity, leading the corresponding weights to zero which ensures the sparsity of the final model.

When executing a classification task, the posterior probability of the weights can not be calculated. However, Laplacian theory can be used to approximate the calculation: for the current fixed α, find the maximum possible weight w_{MP}. Use the second-order Newton method in (4) to get w_{MP}.

$$\log\{p(t \mid w)p(w \mid \alpha)\}= \sum_{n=1}^{N}[t_n \log y_n+(1-t_n)\log(1-y_n)]-\frac{1}{2}w^T A w \tag{4}$$

where $y_n=\sigma\{y(x_n;w)\}$ and $A=diag(\alpha_0,\alpha_1,...,\alpha_N)$.

Using the Laplacian method, the log posterior probability is quadratically approximated. Then, taking two derivatives on (4) results in the following:

$$\nabla_w \nabla_w \log p(w \mid t,\alpha)|_{w_{MP}}=-(\Phi^T / B\Phi+A) \tag{5}$$

where $\Phi=[\varphi(x_1),\varphi(x_2),..., \ \varphi(x_N)]^T$ and $B=diag(\beta_1,\beta_2,...,\beta_n)$. Use (5) to obtain the covariance matrix Σ. Next, update the hyperparameter α through Σ and w_{MP}. The updated hyperparameter can thus be represented as:

$$\alpha^{new}={\gamma_i}\big/{w_{MP}^2} \tag{6}$$

where $\gamma_i=1-\alpha_i \Sigma_{ii}$ and Σ_{ii} is the i-th diagonal element of the matrix represented in (7).

$$\Sigma=(\Phi^T B\Phi+A)^{-1} \tag{7}$$

The model function defined by the relevance vector is a high-dimensional hyperplane, which can approximate the test samples on both sides of the plane. Thus, the classification results are obtained according to the different planes..

3 Rvm Expansion Methods On Spark

3.1 Spark platform

Spark is an open-source parallel computing framework developed by the UC Berkeley AMP Lab based on MapReduce [12]. Fig. 1 shows the general Spark work-flow used in this paper. This framework proposes an abstract flexible distributed dataset called RDD (Resilient

Distributed Dataset). The researchers manipulate the data by calling RDD's Transformation and Action operations. These operations are performed on the data based on the partition established when the RDD was defined. Spark shows extraordinary effectiveness for machine learning tasks on large-scale data. Fig. 2 shows the basic running logic diagram of an RDD in Spark. The whole procedure is a directed acyclic graph. Spark divides the process into stages according to the different dependencies between RDDs. Each stage contains a series of function operations, the entire process is triggered by the RDD Action function.

3.2 Basic Principles of the AdaBoost RVM

Adaboost is an ensemble learning algorithm based on probably approximately correct(PAC) [13] learning theory.

Common AdaBoost algorithms include Discret AdaBoost, Real AdaBoost and Gentle AdaBoost. All of these algorithms have achieved great performance in fields including face detection [14] and network security [15].

The core idea of the algorithm is to train different (weak) classifiers on the training set, and then combine these weak classifiers to form a strong final classifier. The algorithm obtains the confidence of each classifier and revises the weight of each sample according to the classification of each sample and the accuracy of the overall classification. Then, the revised weights are used for the next classifier for testing. In the end, the resulting weak classifiers are merged into the final decision classifier.

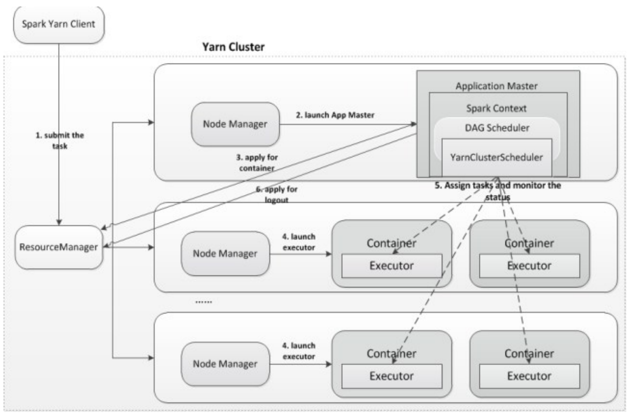

Fig. 1. Flowchart of Spark on Yarn

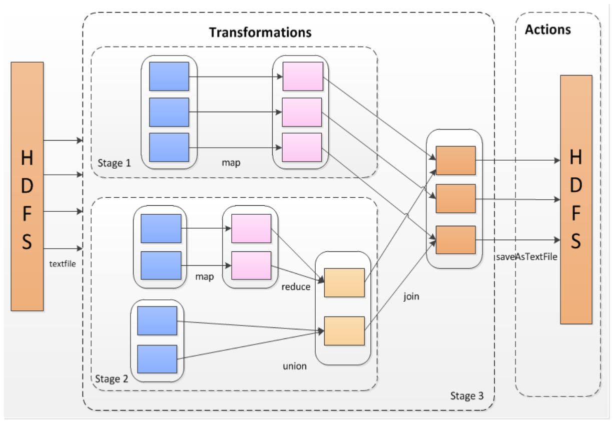

Fig. 2. Running logic of RDD

In this study, algorithms are implemented on Spark. Whereas samples are randomly distributed on Spark, which may lead to a partial data imbalance in some blocks, on this occasion, the final classifier may also be influenced. To overcome this problem, the training samples in each block are judged before training. If the threshold is exceeded, the training data of this block are regarded as an imbalance dataset. For imbalanced dataset, the training data are modified by creating synthetic examples from the minority class using the SMOTE [16] algorithm.

Giving the training dataset $(x_{i1},...,x_{iM},y_i), i=1,2,...,N$, where $y_i \in Y=\{-1,+1\}$ represents the class label of each sample, and the weight of each sample is initialized as $D_1(x_i)=1/N, i=1,2,...,N$. For $t=1,2,...,T$, where T represents the number of weak classifiers, a weak classifier and the corresponding weighted error ε_t are obtained. Then, calculate the confidence coefficient α_t using ε_t. Meanwhile, the weight $D_{t+1}(x_i)$ of each sample is updated for the next iteration at last.

Finally, through combining all the weak classifiers and the related confidence coefficients, the final classifier is :

$$H_{final}(x)=sign(\sum_{t=1}^{T}\alpha_t h_t(x)) \tag{8}$$

In the above process, different AdaBoost algorithms are obtained by combining different weak classifier acquisition and sample weight update methods in each iteration process. This paper integrates RVM with the AdaBoost algorithm and modified AdaBoost algorithm, and then takes full advantage of the Spark platform to implement these algorithms. Under these circumstances, the algorithms perform well on large-scale datasets. Fig. 3 shows the basic process of the algorithm.

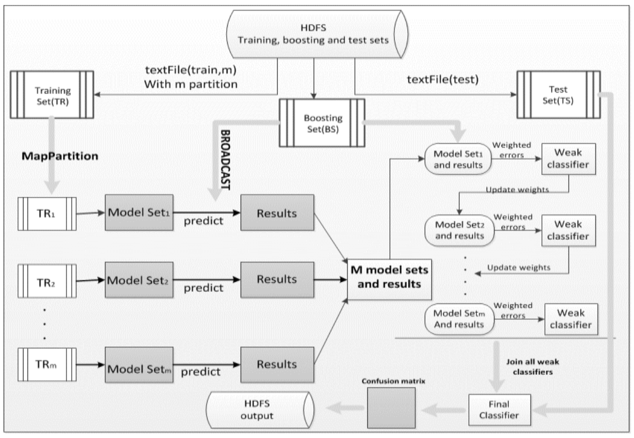

Fig. 3. Flowchart of the proposed DAB-RVM

3.3Discrete AdaBoost RVM

This section describes the combination of the Discrete AdaBoost and RVM algorithms on Spark. First, the whole dataset are divided into three parts as training, boosting and test sets. Afterward, the training set are divided into blocks and operate using RDD. RangePartitioner(#iter).

Meanwhile, the boosting set are shared with all the slaves by broadcast for boosting. In addition, modify the imbalanced dataset in each slave through the SMOTE algorithm before training weak classifiers. Then, train a weak classifier for each feature based on the data from each block and obtain the classification result of the boosting set for each weak classifier. Next, the weight of each sample is initialized and, for all weak classifiers obtained from the same block, calculate the weighted error as (9) and select the weak classifier with the smallest weighted error.

$$err_s = \sum_{k:H_c(x_k)\neq y_k} w_i(k) \tag{9}$$

Afterwards, calculate the confidence coefficient of each classifier based on the weighted error as (10) and update the weight as (11).

$$\alpha_s = \frac{1}{2}\log(\frac{1-err_s}{err_s}) \tag{10}$$

$$w_{s+1}(i) = w_s(i)\exp(-\alpha_s y_i H_s(x_i)) \tag{11}$$

Finally, the final classifier can be represented as (12).

$$G_{final}(x) = sign(\sum_{s=0}^{T-1} \alpha_s H_s(x)) \tag{12}$$

3.4All Features Boosting RVM

The proposed algorithm described above splits the data into different blocks and distributes it into different datanodes; then, for each block in a datanode, trains a weak classifier on each feature of the samples on the block. This process needs to train $m \times n$ models (m represents the number of features of each sample and n represents the number of blocks that training set has been split into). So, when dealing with a high-dimensional large-scale dataset, the training process may take too much time.

As the number of the blocks which the training data is split into becomes large, and RVM belongs to a not so weak classifier, the weak classifier can be trained using all the features of the samples, which means that each block gets a single weak classifier. After training, divide all the classifiers into T groups and iterate over all groups. At each iteration, do a weighted regression based on least-squares and obtain a classifier in each group with the smallest weighted error. Finally, T classifiers are integrated into the final classifier, shown as (12).

In addition, as the output of Discrete AdaBoost can only be +1 or -1, which may be slightly rough. Thus, we adopt the principle of Gentle AdaBoost to select the weak classifiers. As for the Gentle AdaBoost, the output of the weak classifier are:

$$H(x) = P_w(y=1\,|\,x) - P_w(y=-1\,|\,x) \tag{13}$$

Meanwhile, the calculation of weighted error and weighted updating are also changed as shown in (14) and (15).

$$\varepsilon_s = \sum_{i=1}^{N} \omega_i [H_s(x_i) - y_i]^2 \tag{14}$$

$$\omega_{s+1}(i) = w_s(i)\exp(-y_i H_s(x_i)) \tag{15}$$

AFB-RVM, as shown in Algorithm 1, reduces the number of weak classifiers that need to train, especially on high-dimensional large-scale datasets. Thus, the time efficiency of this method increases significantly.

Algorithm 1. AFB-RVM

```
Input:
1:  Total $N$ samples: $<(x_{11},...,x_{1m},y_1),...,(x_{N1},...,x_{Nm},y_N)>$,
    where $y_i \in \{-1,+1\}$.
2:  Split the N training examples into $TR$, $BS$ and $TS$.
3:  Split the training data $TR$ into several blocks and
    distribute to different slaves.
4:  Modify the data of each block with K synthetic
    examples from the minority class using SMOTE
    algorithm.
5:  Train a single weak RVM classifier based on all the
    features of each block.
6:  Integer $T$ specifying the number of required weak
    RVM classifiers and $BS$ as the boosting examples.
7:  Divide all the weak classifiers into $T$ groups.
8:  Initialize $w_1^i = 1/D_{Boost}$, where $D_{Boost}$ represent the
    size of $D_2$.
9:  for c=0,2,…,T-1 do
10:     Fit the regression function $H_c(x)$ by weighted
        least-squares of $y_i$ to $x_i$ with $w_i$
11:     Update $G_{final}(x) \leftarrow G_{final}(x) + H_c(x)$
12:     Update the weight based on (15) and renormalize.
13: end for
14: Output: the final hypothesis as (12)
```

4 Experimental Results And Analysis

To verify the feasibility of the proposed algorithms, this section compares the performance of the above algorithms and the serial RVM algorithm on the UCI dataset (Image Segmentation) and artificial datasets. Image Segmentation (IS) is an image segmentation dataset with 2310 samples divided into seven categories (GRASS, PATH, WINDOW, CLEAR, FOLIAGE, SKY, and BRICKFACE). Each sample has 19 attribute values and one class label.

Because the time complexity of RVM is $O(n^3)$ and the space complexity is $O(n^2)$, when the number of samples is too large, the efficiency of RVM drops sharply. Therefore, RVM is executed on Image Segmentation and the smaller simulated datasets to compare with the three algorithms described above. For large-scale datasets, the other three algorithms in this paper are executed for comparative analysis.

4.1 Experimental Environment

All experiments are executed on a cluster composed of four nodes: the master node (namenode) and four computing nodes (datanode).All the nodes are dual-core with 4G memory.

The specific details of the softwares used and their configurations are below:

· Distributed platform: Hadoop 2.5.2 and Spark 1.5.1

· Development language: Scala 2.10.4 and JDK 1.7.0_51

· Operating System: Cent OS 6.5

4.2 Performance Measures

To evaluate a binary decision task, a contingency matrix representing the possible classification results, as shown in Table 1, is defined.

Metrics defined on Table 2 are generated from this contingency matrix. These metrics on Table 2 are the main methods to evaluate a classification model. Precision reflects the proportion of true positive samples in the positive cases determined by the classifier, and recall reflects the proportion of positive cases that are correctly determined. The ideal situation is that both precision and recall are close to 1. However, in real applications, precision and recall are mutually influential and contradictory. Therefore, to evaluate the model, F1 and Accuracy are chosen to evaluate the model instead of precision or recall.

Table 1. contingency matrix for classification.

	Positive class	Negative class
Assigned Positive	TP	FP
Assigned Negative	FN	TN

TABLE 2. PARAMETERS from the contingency matrix

Measure	Formula
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
Precision (P)	$\frac{TP}{TP+FP}$
Recall (R)	$\frac{TP}{TP+FN}$
F1-measure (F1)	$\frac{2\times P\times R}{P+R}$

4.3 Result on Image Segmentation

In this section, three algorithms are analyzed by comparing the performance of RVM, DAB-RVM, and AFB-RVM on IS. The experimental data is split in the ratio 6:1:3 corresponding to the training, boosting and test datasets, respectively.

For the proposed algorithms, accounting for the randomness of each partition, the average F1 and Accuracy of ten experiments are selected for evaluation. Table 3 shows the average F1 and Accuracy of these algorithms on IS. Additionally, Fig. 4 shows the quotient of training time between RVM and the proposed algorithms on Image Segmentation (r represents the proportion of training samples extracted from IS).

TABLE 3. F1 for DAB-RVM and AFB-RVM on is

	F1			Accuracy		
	DAB-RVM	**AFB-RVM**	**RVM**	**DAB-RVM**	**AFB-RVM**	**RVM**
GRASS	97.0%	92.2%	95.0%	94.5%	94.2%	96.4%
PATH	92.3%	91.4%	92.9%	94.9%	93.0%	94.1%
WINDOW	79.2%	77.9%	82.3%	81.0%	81.0%	85.9%
CEMENT	69.6%	77.1%	78.8%	81.0%	81.2%	86.9%
FOLIAGE	68.8%	64.9%	79.9%	86.5%	86.1%	90.2%
SKY	92.3%	95.8%	96.8%	87.8%	87.5%	96.5%
BRICKFACE	90.2%	96.3%	97.5%	91.9%	92.3%	98.7%
Average	84.2%	85.1%	89.0%	88.2%	87.9%	92.7%

Table 3 show that F1 value of RVM is 3-4% higher than that of DAB-RVM and AFB-RVM, and its classification accuracy is higher by 2-4%. However, Fig. 4 shows that AFB-RVM has a remarkable improvement in time efficiency.

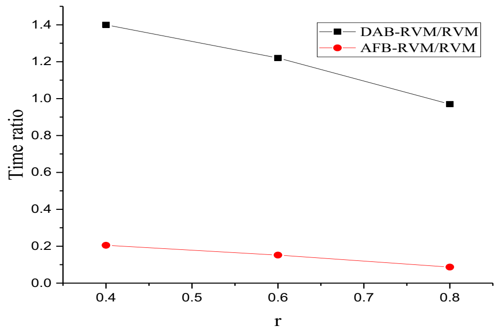

Fig.4 Quotient of training time between GAB-RVM, AFB-RVM and RVM

4.4 Result on Simulated Datasets

To test the performance of the proposed algorithms for different numbers of samples, we choose different scales of simulation data. The two selected simulation datasets contain 10,000 and 1,000,000 samples (referred to as SSD and SLD).

1) *Experiment using SSD*

The overall performance of RVM, DAB-RVM and AFB-RVM on these simulated datasets is further tested with SSD (ratio 6:1:3 for training, boosting and testing). Table 4 shows the minimum, maximum and mean values of F1 and Accuracy for ten experimental results of the RVM, DAB-RVM and AFB-RVM algorithms with SSD1.

TABLE 4. F1 values for baseline RVM, DAB-RVM and AFB-RVM with SSD during ten experiments

	F1			ACCURACY		
	DAB-RVM	AFB-RVM	RVM	DAB-RVM	AFB-RVM	RVM
Min	84.86 %	89.39 %	90.80 %	86.32 %	87.42 %	90.72 %
Max	91.42 %	90.72 %	92.33 %	92.14 %	93.10 %	94.30 %
Average	88.89 %	90.08 %	91.46 %	89.07 %	90.16 %	91.69 %

It can be concluded from Table 4 that when the number of samples reaches 10,000, the disparity between DAB-RVM, AFB-RVM and baseline RVM is much smaller than on IS.

2) Experiment on SLD

The performance of these algorithms has been verified through experiments on SSD. Moreover, to determine the types of datasets to which these algorithms apply. Several copies of SLD in different dimensions are generated and used to verify the related algorithms. Additionally, these training sets revolved are all too large for RVM, so only DAB-RVM and AFB-RVM are involved in this experiment.

First, obtain a ten-dimensional SLD with class labels of +1 and -1. Table 5 shows the average F1 and Accuracy values of the DAB-RVM, and AFB-RVM results obtained by selecting different ratios of data as training data under this dataset.

TABLE 5. Average values of F1 on ten-dimensional SLD of different ratios

Percentage of training data	DAB-RVM	AFB-RVM	DAB-RVM	AFB-RVM
20%	82.21%	83.15%	84.52%	86.01%
40%	86.91%	88.34%	87.11%	89.44%
60%	87.21%	90.15%	89.12%	90.18%
80%	89.05%	91.72%	89.94%	91.05%

Because DAB-RVM and AFB-RVM obtain their weak classifiers in different ways, The data of different dimensions are used to verify the time performance of the algorithms. Fig. 5 shows the ratios of training time of DAB-RVM to AFB-RVM.

4.5 Results Analysis

From the results in Table 3, it can be concluded that the Accuracy and F1 values of RVM are 2-4% and 3-4% higher than those of DAB-RVM and AFB-RVM. However, the data in Fig. 4 show that AFB-RVM have reduced the training time greatly. Taking all this into consideration, when dealing with small datasets, it is still better to use RVM.

Moreover, from the results in Table 4, when the training data becomes larger, the Accuracy and F1 values of RVM are only 1-2% and 1.5-2.5% higher, respectively, than those of DAB-RVM and AFB-RVM. Compared with the results of Image Segmentation, the disparity with RVM gets smaller. Finally, analyzing the results in Table 5, we find that DAB-RVM and AFB-RVM perform very well with large-scale datasets, especially when the percentage of training data reaches 60%. Furthermore, from the data in Fig. 5, it is easy to find that, when the dimension of the sample is higher, the running efficiency of AFB-RVM is more obvious.

From the analyses above, it can be concluded that DAB-RVM is more suitable for large-scale datasets with few features, while AFB-RVM is more suitable for large-scale datasets with many features.

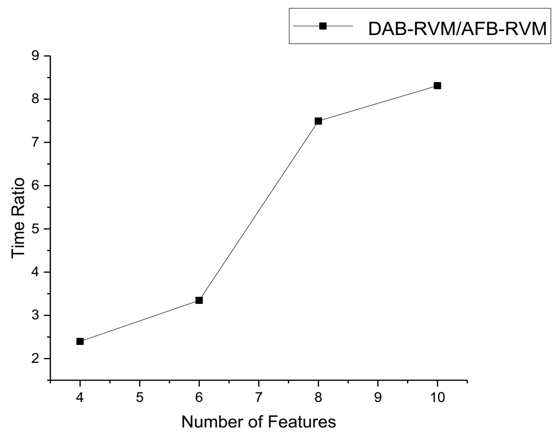

Fig. 5. Training time ratios of DAB-RVM to AFB-RVM

5 Conclusion And Future Work

In this paper, we combine Discrete AdaBoost and Gentle AdaBoost with RVM to classify large-scale datasets on a distributed platform. Then, to reduce the training time for high-dimensional large-scale datasets, we propose the AFB-RVM, which alters the method of Gentle AdaBoost for obtaining weak classifiers and incorporates it into RVM. Experimental results on Spark for image segmentation and simulated datasets demonstrated that the proposed algorithms perform well when dealing with large-scale dataset and have achieved great improvements in time efficiency. Future improvements could be carried on by extending the algorithms to multiclass classification and optimizing the output of the weak classifiers.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61402345) and the Fundamental Research Funds for the Central Universities (WUT: 2017 IVB072).

References

[1]. Tipping, Michael E. "Sparse bayesian learning and the relevance vector machine." Journal of Machine Learning Research 1.3(2001):211-244.
[2]. Vapnik, Vladimir N. The Nature of Statistical Learning Theory. The nature of statistical learning theory /. Springer, 2000:1564-1564.

[3]. Gupta, Rishabh, K. U. R. Laghari, and T. H. Falk. "Relevance vector classifier decision fusion and EEG graph-theoretic features for automatic affective state characterization." Neurocomputing 174(2015):875-884.

[4]. Zhou, Yan, et al. "Deception detecting from speech signal using relevance vector machine and non-linear dynamics features."Neurocomputing 151(2015):1042-1052.

[5]. Tipping, Michael E, and A. C. Faul. "Fast Marginal Likelihood Maximisation for Sparse Bayesian Models." International Workshop on Artificial Intelligence and Statistics 2003.

[6]. Silva, C, and B. Ribeiro. "Towards expanding relevance vector machines to large scale datasets. " International Journal of Neural Systems18.1(2008):45-58.

[7]. Yang, Depeng, et al. "High Performance Relevance Vector Machine on GPUs." Symposium on Application Accelerators in High PERFORMANCE Computing 2010.

[8]. DIMITRIS TZIKAS, ARISTIDIS LIKAS, and NIKOLAS GALATSANOS. "LARGE SCALE MULTIKERNEL RELEVANCE VECTOR MACHINE FOR OBJECT DETECTION." International Journal of Artificial Intelligence Tools 16.6(2011):967-979.

[9]. Dong, et al. "Accelerating Relevance-Vector-Machine-Based Classification of Hyperspectral Image with Parallel Computing."Mathematical Problems in Engineering 2012.1024-123X(2012):865-883.

[10]. Zaharia, Matei, et al. "Spark: Cluster Computing with Working Sets." (2010):10-10.

[11]. Friedman, Jerome, T. Hastie, and R. Tibshirani. "Additive logistic regression: a statistical view of boosting." Annals of Statistics 28.2(2000):págs. 374-376.

[12]. Dean, Jeffrey, and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters. " Conference on Symposium on Opearting Systems Design & Implementation 2004:107-113.

[13]. Valiant, L. G. "A theory of the learnable." Communications of the Acm27.11(1984):1134-1142.

[14]. P. Ithaya Rani, and K. Muneeswaran. "Facial Emotion Recognition Based on Eye and Mouth Regions." International Journal of Pattern Recognition & Artificial Intelligence 30.07(2016).

[15]. Zhuang, Weiwei, Q. Jiang, and T. Xiong. "An Intelligent Anti-phishing Strategy Model for Phishing Website Detection." International Conference on Distributed Computing Systems Workshops IEEE Computer Society, 2012:51-56.

[16]. Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of Artificial Intelligence Research 16.1(2002):321-357.