Computers and Mathematics with Applications 66 (2013) 1920–1934

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A MapReduce-based distributed SVM ensemble for scalable image classification and annotation

Nasullah Khalid Alham [a], Maozhen Li [b,c,*], Yang Liu [d], Man Qi [e]

[a] Nuffield Division of Clinical Laboratory Sciences, Radcliffe Department of Medicine, University of Oxford, OX3 9DU, UK
[b] School of Engineering and Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK
[c] The Key Laboratory of Embedded Systems and Service Computing, Ministry of Education, Tongji University, China
[d] School of Electrical Engineering and Information Systems, Sichuan University, Chengdu, 610065, China
[e] Department of Computing, Canterbury Christ Church University, Canterbury, Kent, CT1 1QU, UK

ARTICLE INFO

Keywords:
Support vector machines
MapReduce framework
Ensemble classifier
Automatic image annotation

ABSTRACT

A combination of classifiers leads to a substantial reduction of classification errors in a wide range of applications. Among them, support vector machine (SVM) ensembles with bagging have shown better performance in classification than a single SVM. However, the training process of SVM ensembles is notably computationally intensive, especially when the number of replicated training datasets is large. This paper presents MRESVM, a MapReduce-based distributed SVM ensemble algorithm for scalable image annotation which re-samples the training dataset based on bootstrapping and trains an SVM on each dataset in parallel using a cluster of computers. A balanced sampling strategy for bootstrapping is introduced to increase the classification accuracy. MRESVM is evaluated in both experimental and simulation environments, and the results show that the MRESVM algorithm reduces the training time significantly while achieving a high level of accuracy in classifications.

1. Introduction

Content-based image retrieval (CBIR) was proposed to allow users to retrieve relevant images based on their low-level features such as color, texture, and shape. However, the accuracy of CBIR is not adequate, due to the existence of a *semantic gap*, a gap between the low-level visual features such as textures and colors and the high-level concepts that are normally used by the user in the search process [1]. Various methods and schemes have been investigated to reduce the semantic gap, namely, image annotation and relevance feedback.

Annotating images with labels is one of the techniques to narrow down the semantic gap [2]. Automatic image annotation is a method of automatically generating one or more labels to describe the content of an image, a process which is commonly considered as a multi-class classification. Typically, images are annotated with labels based on the extracted low-level features. Machine learning techniques have facilitated image annotation by learning the correlations between image features and annotated labels [2].

Support vector machine (SVM) techniques have been used extensively in automatic image annotation [3–11]. The qualities of SVM-based classification have been proven remarkable [12–18]. In its basic form, an SVM creates a *hyperplane* as the decision plane, which separates the positive and negative classes with the largest margin [12]. SVMs have shown a high level of accuracy in classifications due to their generalization properties. This can be evidenced from our previous work in

* Corresponding author at: School of Engineering and Design, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK. Tel.: +44 1895266748.
E-mail address: Maozhen.Li@brunel.ac.uk (M. Li).

0898-1221/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.camwa.2013.07.015

evaluating the performance of representative classifiers in image annotation [19]. The evaluation results showed that an SVM performs better than other classifiers in term of accuracy.

Due to various complexities in classification problems, it is difficult to create classifiers with enhanced performance. The combination of classifiers leads to considerable reduction of misclassification errors in a wide range of applications. An ensemble classifier is generally superior in term of classification accuracy to a single classifier when the predictions of the base classifiers have sufficient error diversity [20]. Bagging [21] is the most commonly used combination approach which combines multiple classifiers by introducing randomness in the training instances. Bagging is useful in reducing the variance component of the expected misclassification error of a classifier [22]. Therefore bagging is effective particularly for classifiers with high variance and low bias, which are described by Breiman [21] as unstable classifiers. Unstable classifiers experience significant fluctuations with a small change of the training instances or other parameters [23].

SVM ensembles based on bagging have shown improved performance in classification compared with a single SVM [22,24–28]. Although some progress has been made by these approaches, current methods of bootstrapping create training datasets from the given training dataset by randomly re-sampling with replacement. The number of training datasets required to create an effective SVM ensemble is hard to determine. Additionally, ensemble learning is extremely computationally intensive, which limits their applications in real environments. Moreover, SVM classifiers applied in ensemble learning require large computing resources due to the fact that the computation time in SVM training is quadratic in terms of the number of training instances.

This paper presents MRESVM, a distributed SVM ensemble algorithm for automatic image annotation. MRESVM builds on the Sequent Minimal Optimization (SMO) algorithm [29] for high efficiency in training, and employs MapReduce [30] for distributed computation across a cluster of computers. MapReduce has become a major enabling technology in support of data-intensive applications. MRESVM is implemented using the Hadoop implementation [31] of MapReduce. The MapReduce framework facilitates a number of important functions such as partitioning the training dataset, scheduling MapReduce jobs across a cluster of participating computer nodes, handling node failures, and managing the required network communications.

The MRESVM algorithm is designed based on the bagging architecture which trains multiple SVMs on bootstrap training datasets and combines the output in an appropriate manner. Each *map* function (called *mapper*) trains an SVM in parallel. The combination of SVMs is based on a two-layered hierarchical structure that use second layer SVM to combine the first layer SVMs. A balanced sampling strategy [32] for bootstrapping is introduced to increase the classification accuracy. The performance of the MRESVM algorithm is first evaluated in a small-scale experimental environment. Subsequently, a MapReduce simulator is implemented for further evaluation of MRESVM in large-scale simulation environments. The MRESVM algorithm reduces the training time significantly while keeping a high level of accuracy in classification compared with a single SVM.

The rest of this paper is organized as follows. Section 2 reviews some related SVM ensemble work. Section 3 briefly introduces SVM ensemble techniques. Section 4 describes in detail the design and implementation of the distributed MRESVM algorithm. Section 5 evaluates the performance of MRESVM in a small-scale experimental MapReduce environment. Section 6 presents a MapReduce simulator for evaluating the performance of MRESVM in large-scale MapReduce environments. Section 7 concludes the paper and points out some future work.

2. Related work

Various methods and schemes have been investigated for CBIR [33]. Relevance feedback (RF) is one of the techniques to improve the performance of a CBIR system. RF is based on the interactions between a user and a search engine by requiring the user to label similar or dissimilar images with the query image [34]. Zhang et al. [35] proposed a subspace learning framework called Conjunctive Patches Subspace Learning (CPSL) with side information, for learning semantic subspace by exploiting the user historical feedback log data for a collaborative image retrieval task. Quellec et al. [36] proposed adaptive wavelet-based image characterizations for CBIR applications which facilitate relevance feedback on image characterization. Biased discriminant analysis (BDA) is one of the most promising RF approaches. Zhang et al. [37] proposed a generalized BDA algorithm for CBIR: the algorithm avoids the singular problem by adopting the differential scatter discriminant criterion and handles the Gaussian distribution assumption by redesigning the between-class scatter with a nearest-neighbor approach. Zhang et al. [38] proposed a biased maximum margin analysis and a semi-supervised BMMA for integrating the distinct properties of feedbacks and utilizing the information of unlabeled samples for SVM-based RF schemes.

Currently, ensemble methods represent one of the main research issues in machine learning. Mason et al. [39] show that ensembles enlarge the margins, and consequently improve the generalization performance of learning algorithms. Schapire et al. [22] analyses ensemble learning methods based on bias–variance decomposition of the classification error, which shows that ensemble classifiers reduce the variance and bias, therefore reducing the overall classification error rate.

Bagging is the most commonly used method for constructing ensemble classifiers. Bagging introduces randomness in the training instances. Recently a number of SVM ensembles based on bagging have been proposed. Kim et al. [24] proposed SVM-ensemble-based bagging to improve the classification accuracy. The experimental results show improvement of classification accuracy of the SVM ensemble. However, the experiments were performed with small training datasets. This approach of ensemble learning is extremely computational intensive for large training datasets, which limits their applications in real environments. Yan et al. [25] presented an SVM ensemble method based on bagging. The results show that the ensemble method performs better than a single SVM. However, the algorithm is evaluated using a small number of

bootstrap training datasets. Tao et al. [26] presented an SVM ensemble method based on bagging and random subspace to improve the user relevance feedback performance in content-based image retrieval. The results show an improvement in classification accuracy. However, the ensemble method cannot guarantee diversity within SVM base classifiers due to the use of purely negative user feedback in the training process of the SVMs.

Theoretical analysis of the bagging performance in terms of classification shows that the expected misclassification error of bagging has the same bias component as a single bootstrap training dataset, while the variance component is reduced significantly [22]. Valentini and Dietterich [28] present a low-bias SVM ensemble based on bagging. The aim is to reduce the bias of the base SVMs before applying bagging. They consider the bias/variance tradeoffs to improve the classification accuracy of the SVM ensemble. The experiments show an improvement in classification accuracy. However, the algorithm was evaluated with small training datasets and the efficiency of the algorithm was not analyzed. This approach of ensemble learning is also extremely computationally intensive for a large training dataset.

Silva et al. [40] proposed a distributed SVM-based ensemble system. The processing times are reported to have shown notable improvements over sequential approaches. Furthermore, the deployment of ensemble techniques improves the classification performance in terms of accuracy. The system is evaluated using Condor [41] and Alchemi middleware [42].

Re and Valentini [43] evaluated the performance of several SVM ensembles, in which each base classifier is trained on different training instances, and the outputs are aggregated based on different combination methods. Their results show that heterogeneous training instance integration through ensemble methods is highly accurate for gene function prediction. Derbeko et al. [44] proposed a new technique for aggregating SVM classifiers based on bootstrapping. In this method, a linear combination of the base classifiers using weights is optimized to reduce the variance. However, the efficiency of the ensemble algorithm is not analyzed.

Lei et al. [45] propose an SVM ensemble based on bagging and boosting for text-independent speaker recognition, and the experimental results show an improvement of classification accuracy by the SVM ensemble compared with single SVMs. However, this approach of ensemble learning is extremely computationally intensive for large training datasets. Tang et al. [46] applied bootstrapping to create training datasets from the original training dataset. An SVM is trained on each dataset. The SVM outputs are aggregated by the Bayesian sum rule for a final decision. The algorithm is efficient and scalable. However, there is a slight reduction in the accuracy level compared to a standard SVM.

Summarizing, research on SVM ensemble algorithms has been carried out from various dimensions, but mainly focuses on improving the classification accuracy [24,25,28,43]. Improving the efficiency of an SVM ensemble in training still remains an open challenge. This motivates the design of MRESVM, which is an efficient distributed SVM ensemble algorithm building on a highly scalable MapReduce implementation for image annotation.

3. SVM ensemble

A single SVM may not always provide a good classification performance on all test instances. To overcome this limitation, ensembles of SVMs have been proposed as a solution [22]. An ensemble of classifiers is a set of multiple classifiers combining a number of weak learners to create a strong learner. Training a diverse set of classifiers from a single training dataset has proven to be more accurate in classification than using a single classifier [45]. There are a number of techniques for creating a diverse set of classifiers. The most common technique is to use re-sampling to diversify the training datasets based on bootstrap aggregating (bagging). Breiman [21] showed that bagging techniques can reduce the variance component of the misclassification error, therefore increasing the reliability of the predictions. When the number of classifiers is large, the probability of error becomes small. Fig. 1 shows the general architecture of an SVM ensemble.

Each SVM is trained separately with a bootstrap training dataset created from the original training dataset based on a bootstrapping technique. The bootstrap creates m training datasets by randomly re-sampling with replacement from the original training dataset repeatedly. A particular training instance x may appear repeatedly or not at all in any particular training dataset. Once the training process is completed, the trained SVMs are aggregated using a suitable combination approach.

3.1. Aggregation method

The two-layered hierarchy approach is a combination method which uses a single SVM to aggregate the output results of a number of SVMs. Therefore, this method of combination consists of two layers of SVMs: the generated SVMs in the first layer are fed as input into a single SVM in the second layer [24]. Let $f_j(x), j = 1, 2, 3, \ldots, m$ be a decision function of the mth SVM in the SVM ensemble, and let F be a decision function of the SVM in the second layer. Then, the final decision of the SVM ensemble $f_{SVM}(x)$ for a given test instance x based on two-layered hierarchical combining is determined by $f_{SVM}(x) = F(f_1(x), f_2(x), \ldots f_m(x))$, where m is the number of SVMs in the SVM ensemble.

3.2. Balanced bootstrapping

In Monte Carlo algorithms [47], variance reduction is a technique that is used to increase the precision of the estimates that can be obtained for a fixed number of iterations in a simulation. Balanced bootstrapping is a variance reduction

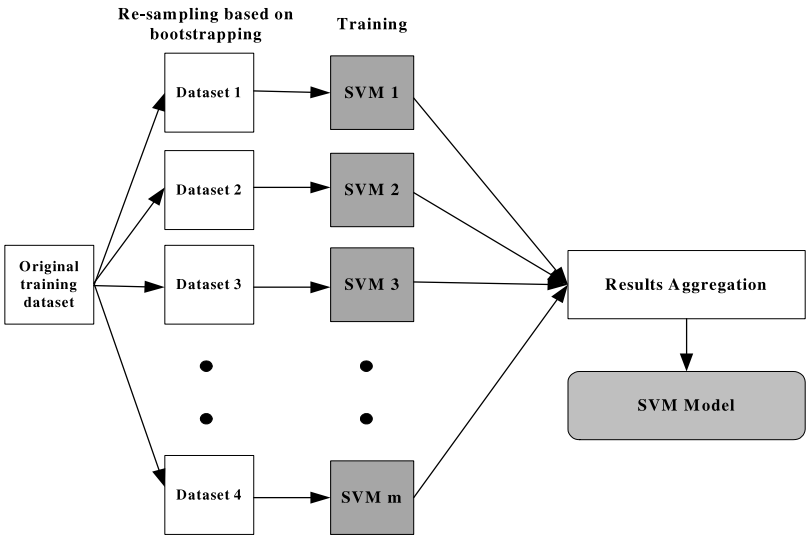

Fig. 1. Architecture of an SVM ensemble.

technique for efficient bootstrap simulation proposed by Davison et al. [32]. Esposito and Saitta [48] established the link between bagging and Monte Carlo algorithms. Despite some differences, these two algorithms have almost the same function in computation.

Balanced bootstrapping is based on the idea of controlling the number of times that training instances appear in the bootstrap training datasets, so that in the m bootstrap training datasets each training instance appears the same number of times. For the bootstrap to work, some training instances must be missing in certain bootstrap datasets, while others may appear two or more times [32]. Balanced sampling does not force each bootstrap training dataset to contain all the training instances; the first training instance may appear twice in the first bootstrap training dataset and not appear at all in the second bootstrap training dataset, while the second training instance may appear once in each training dataset.

A number of techniques have been introduced for creating balanced bootstrap training datasets. However, a simple way of creating balanced bootstrap training datasets, described in [32], is to create a string of the training instances $x_1, x_2, x_3, \ldots, x_n$ repeated m times, which generates a sequence $x^{r_1}, x^{r_2}, x^{r_3}, \ldots, x^{r_n}$. Take a random permutation p of the integers from 1 to m_n and create the first bootstrap training dataset from $x^{p_1}, x^{p_2}, x^{p_3}, \ldots, x^{p_n}$, the second bootstrap training dataset from $x^{p_{n+1}}, x^{p_{n+2}}, x^{p_{n+3}}, \ldots, x^{p_{2n}}$, and so on, until$x^{p_{(m-1)n+1}}, x^{p_{(m-1)n+2}}, x^{p_{(m-1)n+3}}, \ldots, x^{p_{mn}}$ is the mth bootstrap training dataset. The balanced bootstrapping variance reduction technique can be used in bagging to increase the classification accuracy.

3.3. Bias–variance decomposition

Bias–variance decomposition of classification error is a useful tool for analyzing supervised learning algorithms and ensemble techniques to examine the relationships between learning algorithms and ensemble methods with respect to their bias–variance characteristics [49]. Bias measures how closely a classifier's average prediction on all possible training datasets of the given training dataset size matches the true value of the class. Variance measures the variations of the classifier's prediction for different training datasets of the given size [49]. The variance will be large if different training datasets create very different classifiers, and the bias is large in cases where a learning method produces classifiers that are consistently wrong. The bias and variance decomposition is crucial in determining an optimal tradeoff between bias and variance.

Given a training instances $\{x_1, x_2, \ldots, x_n\}$, a trained classifier f can be created. Given a test instance x, the classifier predicts $y = f(x)$. Let g be the actual value of the predicted variable for the test instance x. A loss function $L(g, y)$ measures the cost of predicting y when the true value is g [50]. One of the commonly used loss functions is zero–one loss: $L(g, y) = 0$ if $y = g$; otherwise, $L(g, y) = 1$. The aim is to create a classifier with the smallest possible loss. For classification problems, bias–variance decompositions related to zero–one loss have been proposed [51–55].

In a simple bias–variance decomposition process, the training datasets are used to train SVMs and the trained SVM models are applied on a test dataset. The bias and variance for zero–one loss are calculated from the number of incorrect classifiers for the test instance [56]. Fig. 2 shows the bias–variance decomposition process.

The bias for an instance can be calculated using Eq. (1)

$$\sum_i (e_i - p_i)^2 - p_i(1 - p_i)/c - 1, \tag{1}$$

where i sums over y, c is the number of classifiers, e_i is an indicator variable that indicates whether g is equal to the ith value, and p_i is the part of the classifier that correctly predicted x_i. The variance for an instance is calculated as $1 - \sum_i p_i^2$.

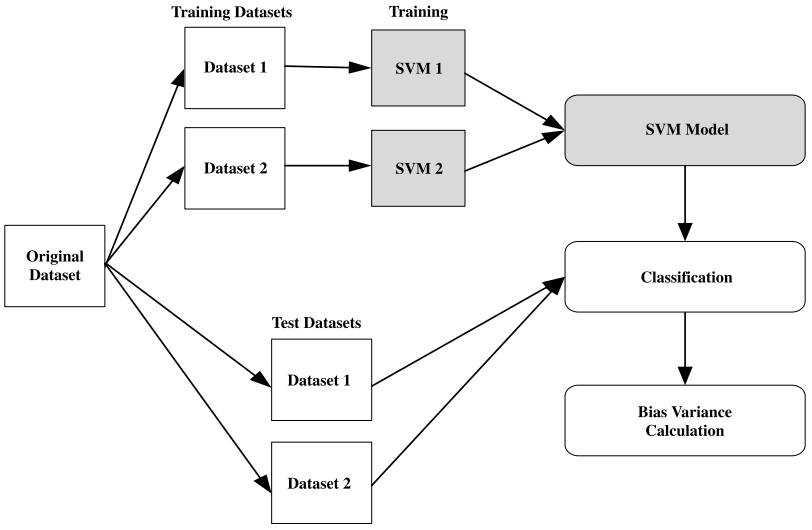

Fig. 2. Bias–variance decomposition.

4. The design of MRESVM

MRESVM builds on MapReduce for parallelization of SVM computation in training. We start this section with a brief description of the Sequential Minimal Optimization (SMO) algorithm for training an SVM, followed by MapReduce programming, and finally a detailed description of the MRESVM algorithm.

4.1. SMO algorithm

The SMO algorithm was developed by Platt [29]. Platt takes the decomposition to the extreme by selecting a set of only two points as the working set, which is the minimum due to the following condition:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2}$$

where a_i is a Lagrange multiplier and y is the class name. This allows the subproblems to have an analytical solution. Despite the need for more iterations in order to reach convergence, each iteration uses only a few operations; therefore the algorithm shows an overall speed-up of some orders of magnitude. Platt's idea provides an improvement in efficiency, and SMO is now one of the fastest SVM algorithms available. We define the following index set I, which denotes the training data pattern:

$$I_0 = \{i : y_i = 1, 0 < a_i < c\} \cup \{i : y_i = -1, 0 < a_i < c\}$$
$$I_1 = \{i : y_i = 1, a_i = 0\} \quad \text{(Positive non-support vectors)}$$
$$I_2 = \{i : y_i = -1, a_i = c\} \quad \text{(Bound negative support vectors)}$$
$$I_3 = \{i : y_i = 1, a_i = c\} \quad \text{(Bound positive support vectors)}$$
$$I_4 = \{i : y_i = -1, a_i = 0\} \quad \text{(Negative non-support vectors)},$$

where c is the correction parameter. We also define the bias b_{up} and the bias b_{low} with their associated indices:

$$b_{up} = \min\{f_i : i \in I_0 \cup I_1 \cup I_2\}$$
$$I_{up} = \arg\min_i f_i$$
$$b_{low} = \max\{f_i : i \in I_0 \cup I_3 \cup I_4\}$$
$$I_{low} = \arg\max_i f_i.$$

The optimality conditions are tracked through the vector f_i in Eq. (3)

$$f_i = \sum_{j=1}^{l} a_j y_j K(X_j, X_i) - y_i, \tag{3}$$

where K is a kernel function and X_i are the training data points. SMO optimizes two a_i values related to b_{up} and b_{low} according to Eqs. (4) and (5)

$$a_2^{new} = a_2^{old} - y_2 \left(f_1^{old} - f_2^{old}\right) / \eta \tag{4}$$

$$a_1^{new} = a_1^{old} - s \left(a_2^{old} - a_2^{new}\right), \tag{5}$$

where $\eta = 2k(X_1, X_2) - k(X_1, X1) - k(X_2, X_2)$. After optimizing a_1 and a_2, f_i which denotes the error of the ith training instance can be updated with the Eq. (6)

$$f_i^{new} = f_i^{old} + (a_1^{new} - a_1^{old}) y_1 k(X_1, X_i) + (a_2^{new} - a_2^{old}) y_2 k(X_2, X_i). \tag{6}$$

To build a linear SVM, a single weight vector needs to be stored instead of all the training instances that correspond to non-zero Lagrange multipliers. If the joint optimization is successful, the stored weight vector needs to be updated to reflect the new Lagrange multiplier values. The weight vector is updated according to Eq. (7)

$$\vec{w}^{new} = \vec{w} + y_1 (a_1^{new} - a)\vec{x}_1 + y_2 (a_2^{new,clipped} - a_2)\vec{x}. \tag{7}$$

We check the optimality of the solution by calculating the optimality gap, that is, the gap between b_{low} and b_{up}. The algorithm is terminated when $b_{low} \leq b_{up} + 2\tau$.

Algorithm 1: Sequential Minimal Optimization Algorithm

Input: training instance x_i, class name y_i,
Output: sum of weight vector, α array, b,

1: Initialize: $\alpha_i = 0, f_i = -y_i$
2: Compute: $b_{high}, I_{high}, b_{low}, I_{low}$
3: Update α_{Ihigh} *and* α_{Ilow}
4: repeat
5: Update f_i
6: Compute: $b_{high}, I_{high}, b_{low}, I_{low}$
7: Update α_{Ihigh} *and* α_{Ilow}
8: until $b_{low} \leq b_{up} + 2\tau$
9: Update the threshold b
10: Store the new $\alpha 1$ and $\alpha 2$ values
11: Update weight vector w if SVM is linear

4.2. MapReduce model

MapReduce provides an efficient programming model for processing large datasets in a parallel and distributed manner. The Google File System [57] that underlies MapReduce provides efficient and reliable data management in a distributed computing environment. The basic function of the MapReduce model is to iterate over the input, compute key/value pairs from each part of input, group all intermediate values by key, then iterate over the resulting groups and finally reduce each group. The model efficiently supports parallelism. Fig. 3 presents an abstraction of a typical MapReduce framework. The map task, also called *mapper*, is an initial transformation step, in which individual input records are processed in parallel. The system shuffles and sorts the map outputs and transfers them to the reduce tasks. Reduce, also called *reducer*, is a summarization step, in which all associated records are processed together by a single entity. The number of *map* and *reduce* tasks can be defined by users depending on a particular algorithm.

4.3. The MRESVM algorithm

The MRESVM algorithm is based on the bagging architecture which trains multiple SVMs on bootstrap training datasets. Both random sampling with replacement and balanced sampling have been used.

As an initial step, training datasets to train the base classifiers are created. For random sampling with replacement, m training datasets of size n are generated according to the uniform probability distribution from a dataset. Balanced sampling is an alternative sampling method that forces each training instance to occur m times in the m bootstrap training datasets. Balanced training datasets are generated by constructing a training dataset of m copies of the original training dataset: after performing random permutation, the training dataset is partitioned into m training datasets.

Each *map* task optimizes a training dataset in parallel in the first layer. The number of *map* tasks is equal to the number of training datasets. The output of each *map* task is the a_i array (Lagrange multipliers) for a training dataset and the training

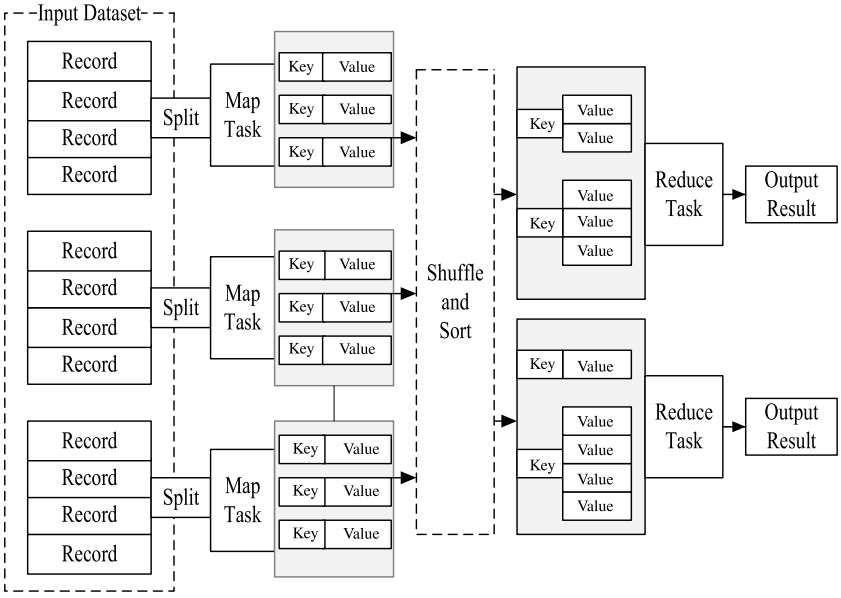

Fig. 3. The MapReduce model.

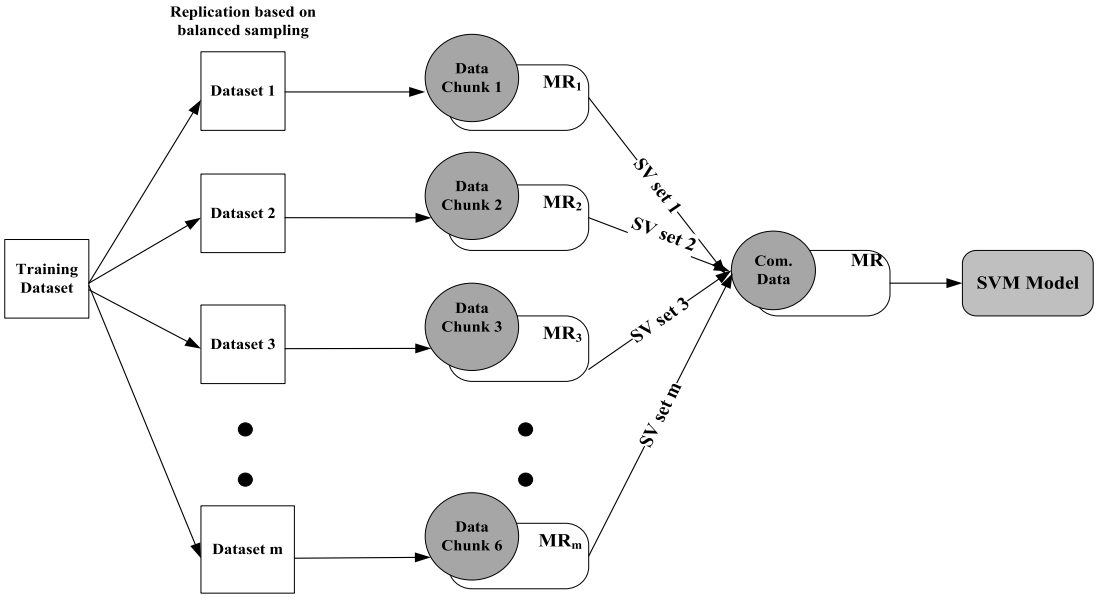

Fig. 4. The architecture of MRESVM.

instances X_i which correspond to Lagrange multipliers $a_i > 0$ in order to create input for the second layer. The output of the second layer includes the a_i array, bias threshold b, and the training instances X_i which correspond to $a_i > 0$ in order to calculate the SVM output u using Eq. (8)

$$u = \sum_{i=1}^{n} y_i a_i K(X_i, X) + b, \tag{8}$$

where X is an instance to be classified, y_i is the class label for X_i, and K is the kernel function. Each *map* task processes the associated instance datasets and generates a set of support vectors. Each set of support vectors is then combined and forwarded to the *map* task in the second layer as training instances. In the second layer, a single set of support vectors is computed, and the generated SVM model will be used in the classification. Fig. 4 shows the architecture of MRESVM, which is slightly different from that of a typical MapReduce model described in Section 4.2. MRESVM operates on map tasks only due to its particular two-layered architecture; hence there are no sort, shuffle, and reduce steps.

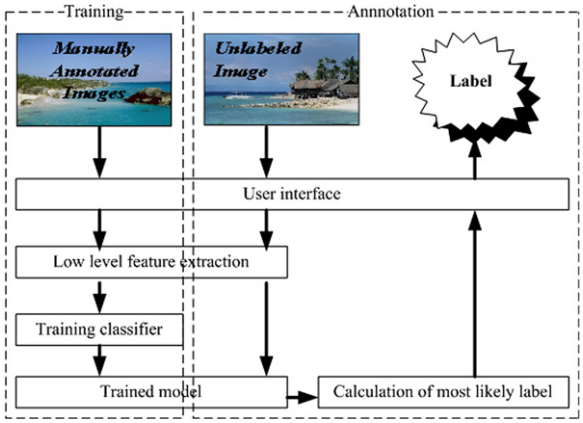

Fig. 5. The architecture of the image annotation system.

Algorithm 2 shows the pseudo code of MRESVM with a two-layered hierarchical combination. Lines 1–3 show the bootstrapping process to create a balanced training dataset for training the SVM. Lines 4–8 show the training process of the SVM. Lines 9–12 show the assembling results of layer 1 which are used as input for layer 2, and the training process in layer 2.

Algorithm 2: MRESVM Algorithm

Input: training instances x_i

Output: support vectors sv_m, weight vectors w_i if SVM is linear

1: replicate training instances x_i based on balanced sampling;
2: perform random permutation;
3: create m datasets to train SVM;

MAP$_j$ $\forall j \in \{1 \ldots m\}$,

Input: m datasets

Output: support vectors sv_i and training instances X_i

4: train SVM on m training datasets
5: obtain sv_m set for m training datasets; $sv_m = \{\alpha_m > 0\}$
6: store all $X_m = \{x_m; a_m > 0\}$
7: store weight vectors w_m if SVM is linear
8: combine all $X_m = \{x_m; a_m > 0\}$ to create training datasets in for next layer *Map* task;
9: train SVM on X_m
10: obtain sv_i set and $X_i = \{x_i; a_i > 0\}$

We summarize the structure of the MRESVM as follows.

- Create m training datasets based on balanced bootstrapping from the original training dataset.
- Allocate each training dataset to a single *Map* task.
- Each *Map* task trains an SVM.
- The output of each *Map* task is a set of support vectors.
- The support vectors are combined to create a training dataset for the second-layer SVM.
- The output of the second-layer SVM is a trained model.

5. Experimental results

We have incorporated MRESVM into our image annotation system, which is developed using the Java programming language and the Weka package [58]. The image annotation system classifies visual features into pre-defined classes. Fig. 5 shows the architecture of the system and Fig. 6 shows a snapshot of the system.

Images are first segmented into blocks. Then, the low-level features are extracted from the segmented image blocks. Each segmented block is represented by feature vectors. We assign the low-level feature vectors to pre-defined categories. The system learns the correspondence between low-level visual features and image labels. The annotation system combines low-level MPEG-7 descriptors such as scalable color and edge histograms [59]. In the training stage, the SVM classifier is fed with a set of training images in the form of attribute vectors with the associated labels. After the SVM model is trained, it is able to classify a new image into one of the learned class labels in the training datasets.

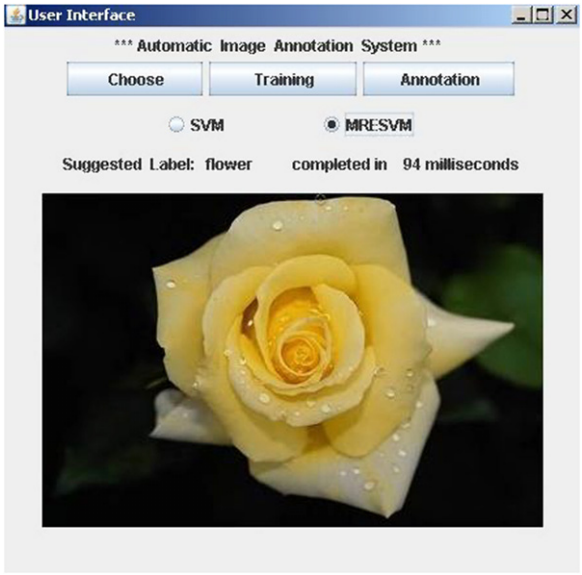

Fig. 6. A snapshot of the image annotation system.

5.1. Image corpora

The images are collected from the Corel database [60]. Images are classified into two classes, and each class of images has one label associated with it. The two pre-defined labels are *people* and *beach*. Typical images with 384×256 pixels are used in the training process. Low-level features of the images are extracted using the LIRE library [61]. After extracting the low-level features a typical image is represented in the following form:

$$0, 256, 12, 1, -56, 3, 10, 1, 18, \ldots, 2, 0, 0, 0, 0, 0, 0, 0, 0, \text{beach}.$$

Each image is represented by 483 attributes: the last attribute indicates the class name, which indicates the category to which the image belongs.

5.2. Performance evaluation

MRESVM is implemented using Weka's base machine learning libraries written in the Java programming language and tested in a Hadoop cluster. To evaluate MRESVM, the SMO algorithm provided in the Weka package is extended, configured, and packaged as a basic MapReduce job. The Hadoop cluster for this set of experiments consist of a total of 12 physical cores across three computer nodes, as shown in Table 1.

The performance of MRESVM is evaluated from the aspects of computation overhead and accuracy. Fig. 7 shows the overhead of the MRESVM in SVM training, which achieves close to 12 times speedup compared to the standalone SVM ensemble.

MRESVM outperforms the standalone SVM ensemble with an increasing number of bootstrap training datasets in terms of training time required. Fig. 8 shows the reduced computation overhead with the number of participating MapReduce *mappers* varying from 4 to 12.

Furthermore, the accuracy of MRESVM with different sampling strategies was evaluated using 2000 training instances, and the results are presented in Table 2. In total 250 test instances were used (10 test instances at a time), and the average accuracy level was considered. The results show that MRESVM using 100 bootstrap training datasets achieves up to 98% in accuracy, which is higher than with single SVM.

The number of *Map* tasks does not affect the classification accuracy because each dataset is processed by a single *Map* task; however, the classification accuracy is directly related to the number of bootstrap datasets. Fig. 9 shows the accuracy the MRESVM with increasing number of bootstrap datasets.

Figs. 10 and 11 show Receiver Operating Characteristic (ROC) curves of MRESVM with random and balanced sampling. Classification accuracy is measured by the area under the ROC curve. MRESVM balanced sampling has a slightly larger area under the ROC curve.

5.3. Measuring bias and variance

To analyze the behavior of MRESVM, the bias–variance decomposition method described in [55] was used, because it can be applied to any classifier, and decomposition does not require the training datasets to be sampled in any specific manner.

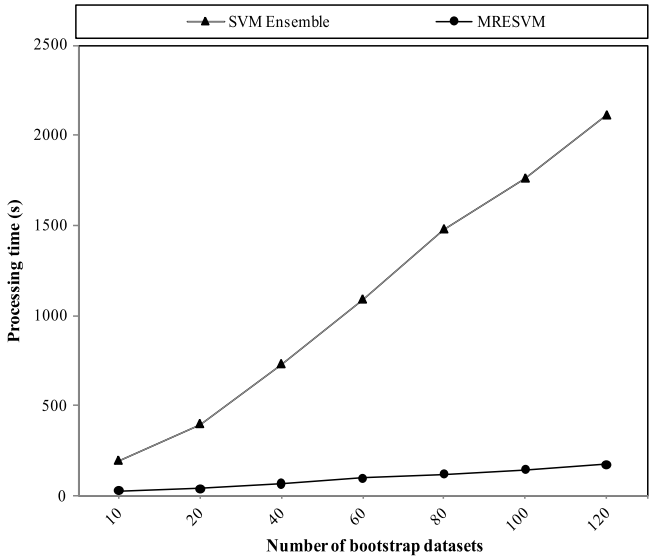

Fig. 7. The computation overhead of MRESVM using 12 *Map* tasks.

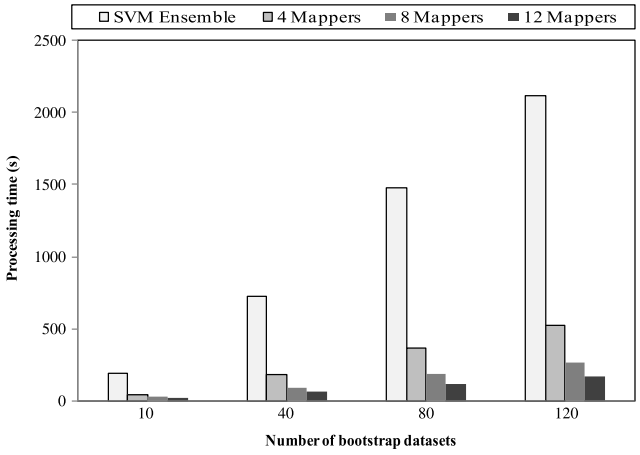

Fig. 8. The computation overhead of MRESVM with various numbers of mappers.

Table 1
Hadoop cluster configuration.

Hardware environment			
	CPU	Number of cores	RAM
Node 1	Intel Quad Core	4	4 GB
Node 2	Intel Quad Core	4	4 GB
Node 3	Intel Quad Core	4	4 GB
	Software environment		
SVM	WEKA 3.6.0 (SMO)		
OS	Fedora10		
Hadoop	Hadoop 0.20		
Java	JDK 1.6		

Table 2
Summary of accuracy results.

	Single SVM	MRESVM random	MRESVM balanced
Correctly classified	≈94%	≈96%	≈98%
Incorrectly classified	≈6%	≈4%	≈2%

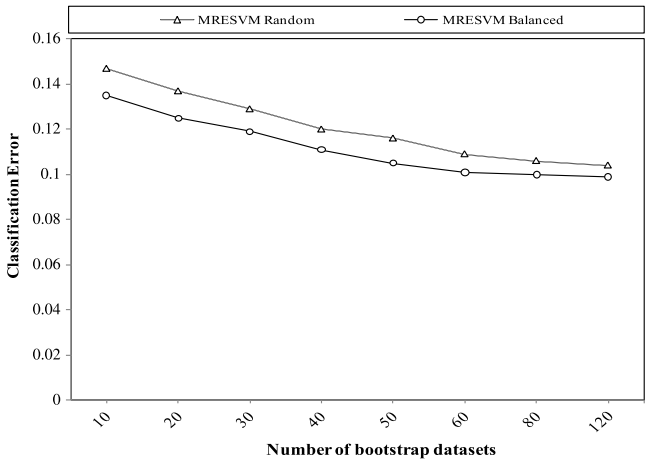

Fig. 9. The classification accuracy of MRESVM.

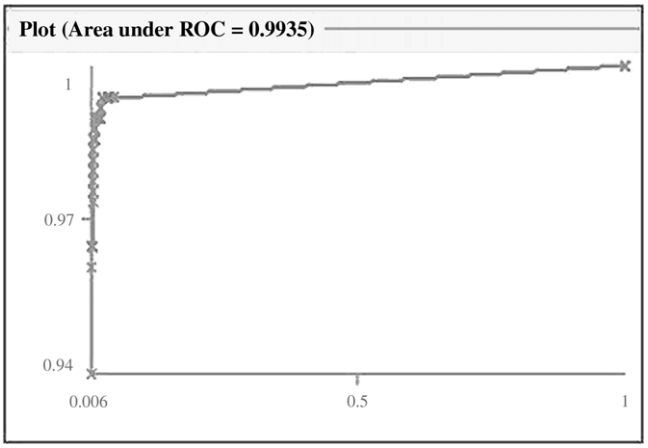

Fig. 10. ROC curve of MRESVM with balanced sampling.

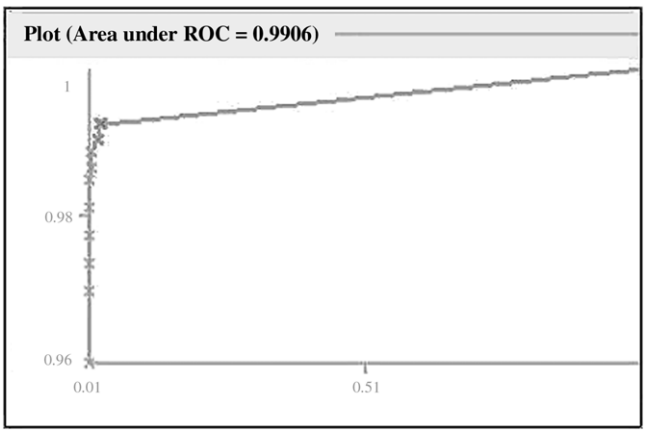

Fig. 11. ROC curve of MRESVM with random sampling.

The idea of tuning SVMs to minimize the bias presented in [28] was adopted before applying bagging to reduce the variance. As stated in [28], the bias of SVMs is controlled by two parameters. First, the parameter C, which controls the tradeoffs between fitting the instances and maximizing the margin: setting C with a large value tends to minimize bias. Second, in the polynomial kernel, the parameter is the degree d of the polynomial. In MRESVM, the base SVM algorithm was tuned using $d = 4$ and $C = 100$.

Table 3
Summary of bias, variance, and classification error.

Number of datasets	Bias (random)	Bias (balanced)	Variance (random)	Variance (balanced)	Error (random)	Error (balanced)
20	0.093	0.092	0.044	0.042	0.136	0.132
40	0.091	0.087	0.043	0.041	0.135	0.126
80	0.089	0.083	0.042	0.039	0.134	0.124
120	0.087	0.081	0.041	0.037	0.133	0.121

Table 4
Configurations for scalability evaluation.

Simulation environment	
Number of simulated nodes:	250
Data size:	100,000 MB
CPU processing speed:	0.75 MB/s
Hard drive reading speed:	80 MB/s
Hard drive writing speed:	40 MB/s
Memory reading speed:	6000 MB/s
Memory writing speed:	5000 MB/s
Network bandwidth:	1 Gbps
Total number of Map instances:	4 *Mappers* per node (1000 *Mappers*)

Bagging based on both random sampling with replacement and balanced sampling was applied to the base SVMs. The bias, variance, and error rate of MRESVM were measured. Table 3 shows that MRESVM using balanced sampling has a lower classification error rate than MRESVM with random sampling. Balanced-sampling-based bagging has lower variance than random re-sampling with replacement.

6. Simulation results

To further evaluate the effectiveness of MRESVM in large-scale MapReduce environments, we have implemented HSim [62], a MapReduce Hadoop simulator using the Java programming language. HSim was validated with established benchmark results and also with experimental environments which have shown that HSim can accurately simulate the dynamic behaviors of Hadoop clusters. In this section, we assess the performance of MRESVM in large-scale MapReduce environments using the HSim simulator.

6.1. Scalability

To further evaluate the scalability of the MRESVM algorithm, HSim was employed and a number of Hadoop environments were simulated using a varying number of nodes up to 250. Each Hadoop node was simulated with four *mappers*, and four training datasets were used in the simulation tests. Table 4 shows the configurations of the simulated Hadoop environments. HSim supports one processor per computer by default design, but the number of processors could be changed. One processor can have one or more cores. The processing speed of a processor core is defined as the volume of data units processed per second, which can be measured from real experimental tests. The CPU processing speed is measured in MB/s instead of the usual measurement of using GHz. The reason is that the CPU speed in GHz cannot accurately describe how much data can be processed by a node in the Hadoop environment since the actual processing speeds are not the same for different algorithms.

From Fig. 12, it can be observed that the processing time of MRESVM decreases as the number of nodes increases. It is also worth noting that there is no significant reduction in processing time of MRESVM beyond 700 *mappers*, due to high communication overhead with a large number of computing nodes.

The performance of MRESVM can be observed by increasing the dataset size that is processed by each *mapper*. A comparison of simulating results of *mapper* overhead between dataset size 11.4 MB and 100 MB is presented in Fig. 13, which indicates higher performances with the dataset size of 100 MB due to the involvement of a smaller number of *mapper* waves. *Mapper* waves are the numbers of processings of rounds required to complete all map tasks.

The performance of MRESVM can be further observed by changing the CPU power, which is measured by the size of the data processed per second. From Fig. 14 it can be observed that the computation overhead of MRESVM using a CPU power of 0.9 MB/s is significantly less than that of using a CPU power of 0.1 MB/s.

7. Conclusions

In this paper, we have presented MRESVM, a scalable distributed SVM ensemble algorithm that capitalizes on the scalability, parallelism, and resilience of MapReduce for large-scale image annotations. MRESVM re-samples the training

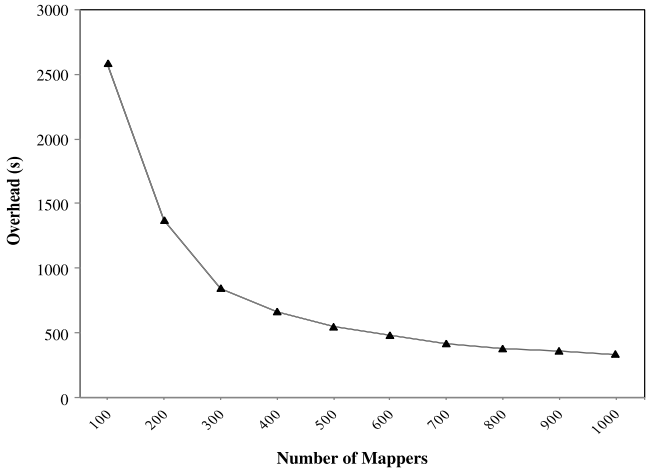

Fig. 12. The scalability of MRESVM in simulation environments.

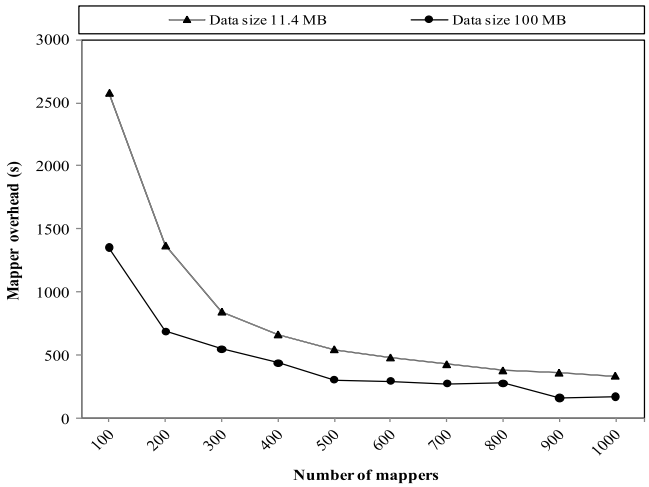

Fig. 13. The impact of dataset size on the computation overhead of mappers.

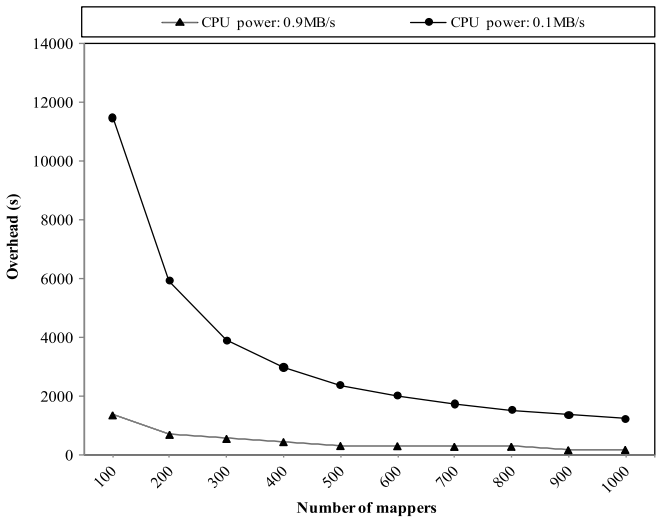

Fig. 14. The impact of CPU power on the computation overhead of MRESVM.

dataset based on bootstrapping and trains SVM in parallel using a cluster of computers. A balanced sampling strategy used for bootstrapping is introduced to increase the classification accuracy. MRESVM is evaluated in both experimental and simulation environments, showing that the distributed SVM algorithm reduces the training time significantly and achieves a high level of accuracy in classifications.

A remarkable characteristic of the MapReduce Hadoop framework is its support for heterogeneous computing environments. Therefore computing nodes with varied processing capabilities can be utilized to run MapReduce applications in parallel. However, the current implementation of Hadoop only employs first-in–first-out and fair scheduling with no support for load balancing taking into consideration the varied resources of computers. Future work will be to design load balancing schemes to optimize the performance of MRESVM in heterogeneous computing environments.

References

[1] A. Smeulders, M. Worring, S. Santini, A. Gupta, A. Jain, Content based image retrieval at the end of the early years, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12) (2000) 1349–1380.
[2] C. Tsai, C. Hung, Automatically annotating images with keywords: a review of image annotation, Recent Patents on Computer Science 1 (1) (2008) 55–68.
[3] W. Wong, S. Hsu, Application of SVM and ANN for image retrieval, European Journal of Operational Research 173 (3) (2006) 938–950.
[4] Y. Gao, J. Fan, Semantic image classification with hierarchical feature subset selection, in: Proceedings of the ACM Multimedia Workshop on Multimedia Information Retrieval, 2005, pp. 135–142.
[5] M. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognition 37 (9) (2004) 1757–1771.
[6] Y. Chen, J.Z. Wang, Image categorization by learning and reasoning with regions, Journal of Machine Learning Research 5 (2004) 913–939.
[7] C. Cusano, G. Ciocca, R. Schettini, Image annotation using SVM, in: Proceedings of SPIE Conference on Internet Imaging, 2004, pp. 330–338.
[8] J. Fan, Y. Gao, H. Luo, G. Xu, Automatic image annotation by using concept-sensitive salient objects for image content representation, in: Proceedings of the 27th Annual International Conference on Research and Development in Information Retrieval (SIGIR), 2004, pp. 361–368.
[9] B. Le Saux, G. Amato, Image recognition for digital libraries, in: Proceedings of the ACM Multimedia Workshop on Multimedia Information Retrieval, MIR, 2004, pp. 91–98.
[10] L.P. Wang, X.J. Fu, Data Mining with Computational Intelligence, Springer, Berlin, 2005.
[11] A. Khandoker, M. Palaniswami, C. Karmakar, Support vector machines for automated recognition of obstructive sleep apnea syndrome from ECG recordings, IEEE Transactions on Information Technology in Biomedicine 13 (1) (2009) 37–48.
[12] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000.
[13] C. Waring, X. Liu, Face detection using spectral histograms and SVMs, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 35 (3) (2005) 467–476.
[14] F. Colas, P. Brazdil, Comparison of SVM and some older classification algorithms in text classification tasks, in: Proceedings of IFIP-AI World Computer Congress, 2006, pp. 169–178.
[15] T. Do, V. Nguyen, F. Poulet, Speed up SVM algorithm for massive classification tasks, in: Proceedings of the 4th International Conference on Advanced Data Mining and Applications, ADMA, 2008, pp. 147–157.
[16] V. Kecman, Learning and soft computing, support vector machines, in: Neural Networks and Fuzzy Logic Models, The MIT Press, Cambridge, MA, 2001.
[17] L.P. Wang (Ed.), Support Vector Machines: Theory and Application, Springer, Berlin, 2005.
[18] A. Sloin, D. Burshtein, Support vector machine training for improved hidden Markov modeling, IEEE Transactions on Signal Processing 56 (1) (2008) 172–188.
[19] N. Khalid Alham, M. Li, S. Hammoud, H. Qi, Evaluating machine learning techniques for automatic image annotations, in: Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD, 2009, pp. 245–249.
[20] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorization, Information Fusion 6 (1) (2005) 5–20.
[21] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[22] R. Schapire, Y. Freund, P. Bartlett, W. Lee, Boosting the margin: a new explanation for the effectiveness of voting methods, The Annals of Statistics 26 (1) (1998) 1651–1686.
[23] G. Fumera, F. Roli, A. Serrau, Dynamics of variance reduction in bagging and other techniques based on randomisation, Multiple Classifier Systems (2005) 316–325.
[24] H. Kim, S. Pang, H. Je, D. Kim, S. Bang, Support vector machine ensemble with bagging, in: SVM, 2002, pp. 397–407.
[25] G. Yan, G. Ma, L. Zhu, Support vector machines ensemble based on fuzzy integral for classification, in: ISNN, 2006, pp. 974–980.
[26] D. Tao, X. Tang, X. Li, X. Wu, Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (3) (2006) 1088–1099.
[27] N. Stepenosky, D. Green, J. Kounios, C. Clark, R. Polikar, Majority vote and decision template based ensemble classifiers trained on event related potentials for early diagnosis of Alzheimer's disease, in: IIEEE International Conference. Acoustic, Speech and Signal Proceedings, 2006, pp. 901–904.
[28] G. Valentini, T. Dietterich, Low bias bagged support vector machines, in: Proceedings of the International Conference of Machine Learning, ICML, 2003, pp. 752–759.
[29] J.C. Platt, Sequential minimal optimization: a fast algorithm for training support vector machines, in: Technical Report, Microsoft Research, MSR-TR-98-14, 1998.
[30] R. Lämmel, Google's MapReduce programming model — revisited, Science of Computer Programming 70 (1) (2008) 1–30.
[31] Apache Hadoop, [Online]: http://hadoop.apache.org/ (Last accessed: 3.04.2011).
[32] A.C. Davison, D. Hinkley, E. Schechtman, Efficient bootstrap simulation, Biometrika 73 (3) (1986) 555–566.
[33] J. Chen, C. Su, W. Grimson, J. Liu, D. Shiue, Object segmentation of database images by dual multiscale morphological reconstructions and retrieval applications, IEEE Transactions on Image Processing 21 (2) (2012) 828–843.
[34] Y. Rui, T. Huang, M. Ortega, S. Mehrotra, Relevance feedback: a power tool for interactive content-based image retrieval, IEEE Transactions on Circuits and Systems for Video Technology 8 (1) (1998) 644–655.
[35] L. Zhang, L. Wang, W. Lin, Conjunctive patches subspace learning with side information for collaborative image retrieval, IEEE Transactions on Image Processing 21 (4) (2012) 3707–3720.
[36] G. Quellec, M. Lamard, G. Cazuguel, B. Cochener, C. Roux, Fast wavelet-based image characterization for highly adaptive image retrieval, IEEE Transactions on Image Processing 21 (4) (2012) 1613–1623.
[37] L. Zhang, L. Wang, W. Lin, Generalized biased discriminant analysis for content-based image retrieval, IEEE Transactions on System, Man, Cybernetics, Part B: Cybernetics 42 (1) (2012) 282–290.
[38] L. Zhang, L. Wang, W. Lin, Semi-supervised biased maximum margin analysis for interactive image retrieval, IEEE Transactions on Image Processing 21 (4) (2012) 2294–2308.
[39] L. Mason, P. Bartlett, J. Baxter, Improved generalization through explicit optimization of margins, Machine Learning 38 (3) (2000) 243–255.
[40] C. Silva, U. Lotric, B. Ribeiro, A. Dobnikar, Distributed text classification with an ensemble kernel-based learning approach, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 40 (3) (2010) 287–297.

[41] Condor, high throughput computing, [online]: http://www.cs.wisc.edu/condor/ (Last accessed: 30.05.2011).
[42] Alchemi, [online]: http://www.cloudbus.org/~alchemi/ (Last accessed: 30.05.2011).
[43] M. Re, G. Valentini, Prediction of gene function using ensembles of SVMs and heterogeneous data sources, in: Applications of Supervised and Unsupervised Ensemble Methods, 2009, pp. 79–91.
[44] P. Derbeko, R. El-Yaniv, R. Meir, Variance optimized bagging, in: Proceedings of the 13th European Conference on Machine Learning, ECML, 2002, pp. 60–71.
[45] Z. Lei, Y. Yang, Z. Wu, Ensemble of support vector machine for text-independent speaker recognition, International Journal Computer Science and Network Security 6 (1) (2006) 163–167.
[46] Y. Tang, Y. He, S. Krasser, Highly scalable SVM modeling with random granulation for spam sender detection, in: Proceedings of International Conference on Machine Learning and Applications, ICMLA, 2008, pp. 659–664.
[47] G. Brassard, P. Bratley, Algorithmic: Theory and Practice, Prentice-Hall, 1988.
[48] R. Esposito, L. Saitta, Monte Carlo theory as an explanation of bagging and boosting, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, 2003, pp. 499–504.
[49] G. Valentini, T. Dietterich, Bias–variance analysis of support vector machines for the development of SVM-based ensemble methods, Journal of Machine Learning Research 5 (2004) 725–775.
[50] P. Domingos, A unified bias–variance decomposition for zero–one and squared loss, in: Proceedings of the Seventeenth National Conference on Artificial Intelligence, 2000, pp. 564–569.
[51] E. Kong, T. Dietterich, Error-correcting output coding correct bias and variance, in: The XII International Conference on Machine Learning, San Francisco, 1995, pp 313–321.
[52] L. Breiman, Bias, variance and arcing classifiers, in: Technical Report TR 460, Statistics Department, University of California, Berkeley, CA, 1996.
[53] T. Heskes, Bias/variance decomposition for likelihood-based estimators, Neural Computation 10 (2) (1998) 1425–1433.
[54] H. Friedman, On bias, variance, 0/1 loss and the curse of dimensionality, Data Mining and Knowledge Discovery 1 (1) (1997) 55–77.
[55] R. Kohavi, D.H. Wolpert, Bias plus variance decomposition for zero–one loss functions, in: Proceedings of the Thirteenth International Conference on Machine Learning, The Seventeenth International Conference on Machine Learning, 1996, pp. 275–283.
[56] R. Bouckaert, Practical bias variance decomposition, in: Australasian Conference on Artificial Intelligence, 2008, pp. 247–257.
[57] S. Ghemawat, H. Gobioff, S. Leung, The Google file system, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles, SOSP, 2003, pp. 29–43.
[58] Weka 3, [Online]: http://www.cs.waikato.ac.nz/ml/weka/ (Last accessed: 3.04.2011).
[59] T. Sikora, The MPEG-7 visual standard for content description—an overview, IEEE Transactions on Circuits and Systems for Video Technology 11 (2) (2001) 696–702.
[60] Corel image databases, [Online]: http://www.corel.com (Last accessed: 3.04.2011).
[61] Lire, an Open Source Java content based image retrieval library, [Online]: http://www.semanticmetadata.net/lire/ (Last accessed: 3.04.2011).
[62] Y. Liu, M. Li, N.K. Alham, S. Hammoud, HSim, a MapReduce simulator in enabling cloud computing, Future Generation Computer Systems 29 (1) (2013) 300–308.