A distributed approach for large-scale classifier training and image classification

Kuizhi Mei a,*, Peixiang Dong a, Hao Lei a, Jianping Fan b

a Institute of Artificial Intelligence and Robotics, Xi’an Jiaotong University, Xi’an 710049, PR China
b School of Information Science and Technology, Northwest University, Xi’an, 710069, PR China

A R T I C L E  I N F O

Article history:
Received 12 August 2013
Received in revised form 14 March 2014
Accepted 27 April 2014
Communicated by Deng Cai
Available online 27 May 2014

Keywords:
Parallel computing
Large-scale image classification
Huge computational cost
Huge storage/memory cost
Visual concept network
Inter-related classifier training

A B S T R A C T

In this paper, a distributed approach is developed for achieving large-scale classifier training and image classification. First, a visual concept network is constructed for determining the inter-related learning tasks automatically, e.g., the inter-related classifiers for the visually similar object classes in the same group should be trained in parallel by using multiple machines to enhance their discrimination power. Second, an MPI-based distributed computing approach is constructed by using a master–slave mode to address two critical issues of huge computational cost and huge storage/memory cost for large-scale classifier training and image classification. In addition, an indexing-based storage method is developed for reducing the sizes of intermediate SVM models and avoiding the repeated computations of SVs (support vectors) in the test stage for image classification. Our experiments have also provided very positive results on 2010 ImageNet database for Large Scale Visual Recognition Challenge.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

As digital cameras become more affordable and widespread, digital images are growing exponentially on the Internet. The availability of large-scale labeled image sets, such as ImageNet1 [1] which consists of 14,197,122 labeled images for 21,841 categories, makes it possible to leverage large-scale online images for training a large number of classifiers and evaluating their performances over large-scale benchmark image sets. For large-scale image classification applications, there may be strong visual correlations among the object classes and image concepts (i.e., strong inter-category visual correlations), thus we need to train their inter-related classifiers jointly rather than independently. Because each category (i.e., each object class or image concept) may contain large amounts of training images (i.e., there may be huge amounts of training images for multiple visually correlated object classes and image concepts which cannot be handled by a single machine), there is an urgent need to develop parallel or distributed computing frameworks which are able to use multiple machines to handle huge sizes of training images for training multiple inter-related classifiers jointly.

Recently, large-scale image classification has received significant attention in the computer vision community [2–6], and the current state of arts methods have also obtained impressive results for large-scale image classification in terms of their accuracy rates. The linear one-against-all SVM classifiers, which have achieved state-of-the-art learning tasks independently, have obtained some acceptable results for large-scale image classification tasks [4]. On the other hand, some other methods focus on developing more effective approaches for feature extraction and image representation [5,6]. In recent works, the bag-of-words (BoW) approach has become one of the most popular approaches for feature design [7–9].

In [7], Wang et al. have proposed an efficient and consistent local constrained method for visual words coding. Shabou et al. [8] have also proposed a local feature coding method and the spatial context is considered in their work. In order to consider the cluster size and shape information of each category, Zhang et al. [9] have proposed a so-called bilevel codebook generation method and a consistent coding rule is also designed.

Different from traditional classifier training and image classification problems at small or medium scales, large-scale image classification has to address two equally important issues simultaneously. The first issue for large-scale image classification is how to train a multi-class classifier for dealing with multiple inter-related learning tasks jointly. Support vector machine (SVM) has proved extremely successful for image classification. The SVM classifier is usually designed for binary classification task, and most existing...
versions of multi-class SVM classifiers originated from its binary version. One common approach for extending binary SVM classifier to multi-class SVM classifier is to decompose the issue of training the multi-class SVM classifier into training many binary SVM classifiers or solving multiple optimization problems jointly [10]. There are two well-known decomposition strategies: “one-against-one” and “one-against-all”. Both of these decomposition strategies are proved to be useful [11,12], and they are widely used in image classification tasks [13,5,6]. Unfortunately, the “one-against-one” decomposition approach may fail for large-scale image classification application due to huge computational cost and low training efficiency. On the other hand, the “one-against-all” decomposition approach may fail for large-scale image classification application because it may suffer from the serious issues of imbalance and high error rates. In addition, these two decomposition strategies (i.e., “one-against-one” and “one-against-all”) ignore the inter-category correlations completely and train all the binary classifiers independently. We argue that there are strong correlations among the categories (i.e., object classes and image concepts), and such inter-category correlations should be leveraged to train their inter-related classifiers jointly. Some pioneering researches have exploited the inter-concept semantic similarity contexts for inter-related classifier training, such as WordNet ontology [14] or Google search engine [15]. It is worth noting that the visual feature space is the common space for classifier training and image classification [16], thus it is very attractive to develop new machine learning algorithm to leverage the inter-concept visual correlations for inter-related classifier training.

The second issue for large-scale image classification is how to handle the critical issues of heavy computation cost and huge storage/memory cost. For large-scale image classification, the huge computational cost comes from two sources: (a) Number of training samples: The computational cost for training a nonlinear SVM classifier grows at least like \( \text{Ord}^2 \sim \text{Ord}^3 \) [17], where \( n \) is the number of training examples. (b) Number of categories: The computational cost for “one-against-all” decomposition approach grows linearly with the number of categories, on the other hand, the computational cost for “one-against-one” decomposition approach grows at least quadratically with the number of categories. For large-scale image classification application, many traditional approaches for SVM classifier training become computationally impractical due to heavy computation cost and huge storage/memory cost. In this paper, we consider the problem of large-scale image classification on ILSVRC2010 image set. In addition, 1000 dimensions SIFT BoW features are released by ILSVRC2010 for image representation. From the experimental results, we know that it takes about 3.5 s to train a single binary SVM with RBF kernel on a 3.07 GHz Intel Core i7 CPU, and it takes about 300 s for the test process over one binary classifier. For the one-against-one approach, it takes about 20 CPU days for classifier training, but it takes about 4.75 CPU years for testing due to the huge number of classifiers. Meanwhile, for the one-against-all approach, it takes about 20 CPU days for testing, but it takes about 7.6 CPU years for classifier training. For large-scale image classification application, we should consider two equally important issues simultaneously: huge computational cost and huge storage/memory cost. The high performance computation platforms, which often consider parallel or distributed processing techniques, have provided a good solution of these two critical issues. One of these successful products is the GPUs (Graphic Processing Units) based computational device. Many pioneering works have adopted GPUs as the parallel computing device for classification tasks [18,19]. Some experiments have revealed that the speed-up is limited when the training samples may be too large to be fit in GPU memory. Thus GPU is not suitable for large-scale image classification application due to the significant bottleneck of data transfer between CPU and GPU. The MapReduce [20] framework provides a high level computation model in a distributed environment, and it is employed to address many data-intensive problems [21,22]. However, this high level framework is only suitable for addressing completely irrelevant tasks on different nodes because the communication cost between different thread is very high. For this reason, MapReduce is not a good choice for training multiple inter-related classifiers jointly.

In this paper, inspired by [23], a large-scale classification algorithm is proposed to train multiple inter-related classifiers jointly and we parallelize it on a distributed platform by using the hybrid of MPI and OpenMP frameworks. The rest of this paper is organized as follows. Section 2 briefly reviews some related work. In Section 3, we propose our inter-related classifier design method for large-scale image classification. Our MPI-based distributed framework and parallel scheme are proposed in Section 4 for large-scale classifier training and image classification. Section 5 describes the experimental results and Section 6 concludes this paper.

2. Related work

In this section, we provide a brief review of the most related work on large-scale classifier training and image classification from two aspects: inter-related classifier training algorithms and parallel computing tools.

2.1. Inter-related classifier training algorithms

As we have argued that there are strong visual correlations among the object classes and image concepts, e.g., the relevant images for some object classes and image concepts may share some common or similar visual properties. Thus it is not a good idea to isolate such visually correlated object classes and image concepts and train their inter-related classifiers independently. Some pioneering works have proposed different approaches for inter-related classifiers training. Torralba et al. [24] have proposed a JointBoost algorithm to leverage the inter-task correlations for improving object detection, where the inter-task correlations are explicitly characterized by using pairwise object combinations. However, this method would not be able to guarantee that every class would be picked up into any shared sets through the boosting procedure, so the JointBoost algorithm is not suitable for large-scale scenario (i.e., large-scale classifier training and image classification). Marszalek et al. [25] have proposed a top-down approach for constructing category hierarchies which postpone the decisions when the uncertainty appears. The major problem for hierarchical learning approaches is the inter-concept error transmission, e.g., the classification errors will be propagated among the classifiers for the inter-related object classes and image concepts at different levels. Semantic hierarchies are also employed to characterize the relationship among different categories [26]. Deng et al. [27] and Zhao et al. [14] have used WordNet [28,29] to find the semantic relationships between the labels and combined discriminative classifiers through the semantic hierarchies. However, the semantic similarity measurements are not able to exactly reflect the visual correlations among the object classes and image concepts. Recently, Dong et al. [30] have proposed a subcategory-aware algorithm to boost the performance of nowadays object classification framework. In that work, both the intra-class diversities and inter-class ambiguities are
considered to guide the subcategory mining. However, the similarity modeling method incurs high computational cost. And moreover, we argue that the visually similar object categories may share some common information hence their classifiers should be learned jointly. In order to measure the visual correlations among different categories efficiently, Fan et al. [16] have constructed a visual concept network to characterize such inter-concept visual correlations explicitly and organize the inter-related learning tasks effectively. Dong et al. [23] have developed a structural learning algorithm by leveraging the visual concept network for inter-related classifier training.

2.2. Parallel computing tools

As we have described in Section 1, a single CPU or a single machine is hopeless to handle huge computation cost and huge storage/memory cost for large-scale image classification tasks, so we should turn to parallel computing by using multiple machines. The many-core GPU(s) and CPU-cluster are the mainstream hardware to run efficient parallel computing. Both CUDA (Compute Unified Device Architecture) and OpenCL (Open Computing Language) are the main programming frameworks for GPUs, whereas CUDA is specifically for NVIDIA GPUs while OpenCL is designed to work across multiple architectures including GPU. In fact, there is no essential difference between CUDA and OpenCL, the choice only depends on the type of GPU. Woodbeck et al. [31] have proposed a classification algorithm on the GPU by using a biologically motivated classifier. Herrero-Lopez et al. [19] have presented a parallel multi-class classification approach by using CUDA based on the one-against-all SVMs approach. van de Sande et al. [32] have proposed a parallel implementation to accelerate image classification by pre-computing SVM kernel values on GPU. Recently, Krizhevsky et al. [33] have used two graphics cards to train a deep network on 1.2 million images from ImageNet image set. However, the major drawback of GPU is that the speed-up is limited when the training model cannot fit in GPU memory, whereas the data transfers between CPU and GPU would cost highly. For large-scale image classification application, the model size or the number of parameters will be limited due to the memory size of GPU (e.g., 4 GB for the up to date NVIDIA GTX 690 graphics card which consists of two Kepler GPUs).

As a scalable framework that simplifies distributed computations, the MapReduce model has been widely used for the data-intensive tasks on CPUs cluster. The key advantage of the MapReduce framework is that it makes the parallelism scheme transparent to users, and hence it simplifies the parallel programming on large-scale clusters. Chu et al. [34] have adopted the map-reduce paradigm to demonstrate the parallel technique on a variety of machine learning algorithms (including SVM, LR, and naïve Bayes). However, this work is implemented on shared-memory machines, rather than the really distributed memory cluster. Tsai et al. [35] have leveraged the MapReduce model to train visual synset classifiers by using one-against-all approach, and the algorithm is implemented on a 2000-node computer cluster. Zhao et al. [14] have used map-reduce parallel computation to solve large-scale optimization problem for multi-class image classification. Wang et al. [15] have also used map-reduce programming to train multi-class classifiers by using the one-against-all approach, and they used 18-node Hadoop (an open-source implementation of the MapReduce framework) servers to evaluate the learning algorithm. However, the significant disadvantage of the MapReduce framework is that it is designed for parallel data processing, so the communication cost between different nodes is very high. Thus the MapReduce model is suitable to train independent classifiers (e.g., the “one-against-all” approach) rather than training the inter-related classifiers. In addition, we conclude that the MapReduce model is more suitable for the industry-level large-scale clusters (i.e., with hundreds or thousands of computation nodes) rather than the lab-level small-scale clusters (i.e., with few computation nodes).

The MPI\(^4\) (Message Passing Interface) model, which is another parallel programming framework for CPU clusters, is also involved as a powerful tool for classification [36,37]. The MPI framework provides fine-grained controls and massive communication functions. Compared to MapReduce, MPI is flexible enough to train our inter-related classifiers on multiple machines. Furthermore, in order to take advantage of the computing power of multi-core in modern computer, we adopt OpenMP\(^5\) as the shared memory parallel computing tool.

The flowchart of our parallel approach for large-scale classifier training and image classification is given in Fig. 1. The main contributions in this paper are the following ones:

(1) A distributed learning method is developed for large-scale classifier training and image classification application.

(2) An MPI-based distributed computation framework is developed to train the inter-related classifiers efficiently. By using this parallel computation framework, we can train our large-scale classifiers (for 1000 categories) with nonlinear SVMs (RBF kernel) on a small scale cluster (five common budget PCs) in three to four days.

(3) The storage/memory cost has been reduced significantly by only storing the indexes of SVs (support vectors) of each SVM model. Through this strategy, we can use only 750 MB to save all the SVM models in our training process.

3. Inter-related classifier training algorithm

It is very attractive to train the inter-related classifiers jointly rather than independently due to the strong inter-category visual correlations. In this section, we briefly describe our structural learning algorithm for inter-related classifier training, which consists of three steps: (1) visual concept network construction; (2) graph cut for group generation and inter-related learning task determination; and (3) group-based inter-related classifier training.

3.1. Visual concept network

A visual concept network [16] is constructed to quantify the inter-task relatedness directly in the visual feature space, and it can provide a good environment to identify the inter-related learning tasks for training multiple inter-related classifiers jointly. The visual concept network consists of two key components: object classes or image concepts and their inter-concept visual correlations.

For two given object classes or image concepts \(C_i\) and \(C_j\), their inter-concept visual similarity context \(\gamma(C_i, C_j)\) is defined as

\[
\gamma(C_i, C_j) = \frac{1}{N_i \cdot N_j} \sum_{h \in C_i} \sum_{k \in C_j} \rho(h, k)
\]

where \(N_i\) and \(N_j\) are the total numbers of image instances for the object classes or image concepts \(C_i\) and \(C_j\), respectively, \(\rho(h, k)\) is the kernel function for characterizing the visual similarity context.

\(^4\) \url{http://www.mcs.anl.gov/research/projects/mpi/}.

\(^5\) \url{http://openmp.org/}.
between the image instances $h$ and $k$ for $C_i$ and $C_j$:

$$\rho(h, k) = \exp\left(-\frac{\chi^2(h, k)}{\sigma}\right).$$

(2)

The visual concept network for ILSVRC2010 [1] image set is shown in Fig. 2. To construct a visual concept network requires the computation of the visual similarities among the categories, which has a computation complexity of $O(n^2)$, where $n$ is the number of categories. It takes about 0.7 s on a 3.16 GHz CPU to compute a similarity value between one pair of categories. However, this task can be easily parallelized because the similarities between different categories can be computed independently. It takes about 2–3 h to complete the concept network construction process on our distributed platform.

3.2. Group generation for inter-related task determination

We perform normalized cut [38] algorithm on our visual concept network and all the object categories are naturally partitioned into a set of groups. Some grouping results are shown in Fig. 3. The object classes and image concepts in the same group share some common visual properties and have strong inter-concept visual correlations; on the other hand, the object classes and image concepts in different groups have weak inter-concept visual correlations. It is worth noting that there is strong relatedness among the learning tasks for training the inter-related classifiers for multiple visually correlated object classes and image concepts in the same group. As a result, our group generation algorithm can provide a good solution for inter-related learning.

---

Fig. 1. The flowchart of our parallel approach for large-scale classifier training and image classification.

Fig. 2. The visual concept network on ILSVRC2010 image set.
task determination, which can further provide a good guidance for parallelling the process for inter-related classifier training.

3.3. Group-based inter-related classifier training

For the inter-related object classes and image concepts in the same group, a structural learning algorithm is performed to learn their inter-related classifiers jointly by sharing a common prediction component. We denote the object classes and image concepts as $C_1, C_2, \ldots, C_N$, and the groups as $G_1, G_2, \ldots, G_M$. For a given group $G_i$, it consists of $N_i$ inter-related object classes and image concepts. Obviously, $\sum_{i=1}^{M} N_i = N$, where $N$ is the total number of object classes and image concepts on our visual concept network and $M$ is the total number of groups. All these groups are determined automatically by performing $N$-cut clustering algorithm over our visual concept network.

It is worth noting that the object classes and image concepts in the same group have stronger inter-concept visual correlations.
than those in different groups, thus the classifiers for the interrelated object classes and image concepts in the same group are strongly inter-related and should be trained jointly to enhance their discrimination power. It is easy for us to design multiple discrimination functions $h_{GC}(x)$ to distinguish the object classes and image concepts in different groups, because the inter-group visual correlations are much weaker. On the other hand, it could be more difficult to distinguish the inter-related object classes and image concepts in the same group because their inter-concept visual correlations are stronger. Thus multiple intra-group classifiers $h_C(x)$ should be designed and learned for distinguishing the inter-related object classes and image concepts in the same group. Besides that, in order to enhance the discrimination power of the classifiers for the object classes and image concepts in the same group, a hybrid level item $h_{BC}(x)$ is added to the classifiers. The hybrid level item $h_{BC}(x)$ is used to discriminate the object class or image concept $C$ from other groups (here the other groups mean all the groups except the one which the object class or image concept $C$ belongs to).

Based on the above observations, we train our inter-related classifiers from three levels (i.e., intra-group, inter-group, and hybrid level):

$$H_C(x) = \alpha_C \cdot h_C(x) + \beta_C \cdot h_{GC}(x) + \gamma_C \cdot h_{BC}(x)$$

(3) Designing and training the hybrid classifiers: This process is different from (1) or (2). These hybrid classifiers are used to discriminate one particular object class or image concept from those object classes and image concepts in other $(M-1)$ groups. In this situation, for any given concept $C_i$, we should train $M-1$ binary classifiers between $C_i$ and $G_j \in \{T,G_k\}$, where $M$ is the total number of groups, $T$ is a set of all groups, and $G_k$ is the group which $C_i$ belongs to. However, for any two object categories, we have found that the classification accuracy increases with the decrease of the visual similarity between them. From Fig. 4, one can observe this phenomenon clearly. For example, the visual similarity between “web site” and “mortar” is very small (about 0.1), and the classification accuracy for these two concepts is high (about 98%). The visual similarity between “chiton” and “sea cucumber” is very high (about 0.97) and the classification accuracy for these two categories is relatively low (about 70%). This phenomenon could also be obtained intuitively. So it is unnecessary to construct the classifiers between $C_i$ and all the other $M-1$ groups, because the hybrid item is just used to enhance the discrimination power of the classifiers in the same group. This work, for any concept $C$, we just perform the hybrid item on the five most relevant groups with $G_k$, where $G_k$ is the group $C_i$ belongs to:

$$h_{BC_i}(x) = \sum_{C_j \in \Omega_k} \gamma_{ij} f(x, C_i, C_j)$$

(6) That is, we first sort all the groups in descending order of the visual similarity with group $G_k$, and then pick out the top-5 to form the set $S$ in (6). For all the weight coefficients we have $\sum_{C_j \in \Omega_k} \gamma_{ij} = 1$.

According to (Eqs. (4), (5), (6) and (3)), the final discrimination function can be formulated as

$$H_C(x) = \alpha_C \cdot h_C(x) + \beta_C \cdot h_{GC}(x) + \gamma_C \cdot h_{BC}(x)$$

$$= \sum_{C_j \in \Omega_k} \alpha_{ij} f(x, C_i, C_j) + \alpha_{ij} \beta_C \cdot h_{GC}(x) + \gamma_C \sum_{C_j \in \Omega_k} \gamma_{ij} f(x, C_i, C_j)$$

(7) where $\Omega_k = G(C_i)$ is the group which the object class or image concept $C_i$ belongs to.

In order to simplify the problem, we use the average weights for classifier combination in our structural learning algorithm. Specifically, we use the average weight strategy to determine the weighted parameters in (3)–(6). We take $\beta_{ij}$ in (5) as an example, all of the weight parameters are specified an equal value and the sum of $\beta_{ij}$ is equal to 1. Hence, all the weights are subjected to $\sum_{C_j \in \Omega_k} \alpha_{ij} = 1$, $\sum_{i \neq j} (\alpha_{ij} + \beta_{ij}) = 1$, $\sum_i (\gamma_{ij} + \beta_{ij}) = 1$, and $\alpha_{ij} = \beta_{ij} = C_i = 1/3$. 

Fig. 4. The visual similarity vs. classification accuracy on some category couples.
4. Parallel training of inter-related classifiers

In this section, a visual concept network based parallel computing framework is developed to train the inter-related classifiers effectively in a distributed environment. Different from the common parallel computing task, the visual concept network can facilitate the parallelization of our structural learning algorithm effectively. In the following subsections, we first analyze the special characteristics of our visual concept network and the structural learning algorithm, and then depict the specifications of our parallel computing framework in detail.

4.1. The visual concept network based parallel scheme

For the one-against-one approach or the one-against-all approach, the final decision function is composed of a number of homogeneous tasks, so we can parallelize it directly on a distributed platform. However, for our structural learning algorithm, the final decision function is composed of a number of heterogeneous tasks. As shown in Fig. 5, our structural learning algorithm consists of three types of tasks, i.e., the intra-group classifiers, the inter-group classifiers, and the hybrid level classifiers. Obviously, the computation cost of different types of tasks is very different hence the training time is unpredictable for different types of tasks. Furthermore, as the sizes of groups are different, the computation cost for the same type of tasks is also varied. So the parallelization of our structural learning algorithm is not as direct as that of the traditional approaches, and a master–slave mode computation scheme is much more suitable to address these heterogeneous tasks.

For the task of large-scale image classification, the data transmission between the RAM (random access memory) and the hard disk is not a trivial issue, because the training data cannot be loaded into the RAM entirely. In order to reduce the I/O transmission cost, it is better to assign the tasks to the computation nodes by groups rather than by concepts. For example, as shown in Fig. 5, if we train the basic learners \( f(C_5, C_6) \) and \( f(G_2, G_1) \) on different computation nodes, the training data for the concept \( C_5 \) would be loaded twice. Instead, if we train the two basic learners on the same node, the training data for the concept \( C_5 \) would only need to be loaded once. In other words, all the basic classifiers which are associated with a same group are trained on the same computation node in our cluster. We take Fig. 5 again as an example, the inter-group classifiers associated with group \( G_2 \), the intra-group classifiers and the hybrid classifiers with the concepts \( C_4, C_5 \) and \( C_6 \) would be trained on the same node. This strategy can reduce the data transmission cost significantly.

In our structural learning algorithm, the training data for one certain concept is used in all three types of classifiers that are associated with it, so we have reason to believe that a certain part of support vectors will be shared by different basic learners. As an example shown in Fig. 5, the decision function \( f(C_5, C_6) \) and \( f(G_2, G_1) \) may share some common SVs because they have the same training data sets for the concepts \( C_5 \) and \( C_6 \). Instead of storing the full SVM model (which contains all the support vectors) of each basic learner, we just store the indexes of the SVs corresponding to their positions in the file of the training data set.

4.2. Distributed computing platform via a master–slave mode

In order to implement our structural learning algorithm for large-scale classifier training and image classification on a small-scale cluster with few common budget PCs, an MPI/OpenMP two-layer programming framework is used in this work. Specifically, the characteristics of this two-layer structure are the following: (1) the MPI-based programming is used to allocate the tasks to the computation nodes and handle the issue of workload balance effectively; and (2) the OpenMP programming is used to execute the computation task on a single multi-core node.

The MPI is a category of parallel programming framework based on message communication. It is efficient for computation-intensive tasks. An MPI application usually consists of multiple “processes”, and the way to communicate among these processes is sending and receiving messages rather than sharing memory or state. Different from MPI, the OpenMP is a shared memory programming model. A typical OpenMP application consists of multiple “threads”, and this happens on a shared memory system where every thread can see and change the data items. In brief, the MPI model is used to partition the whole task into many “processes” and deploy all these processes across the cluster to the individual nodes, while the OpenMP model is used to parallelize each process into many “threads” on a single shared-memory node. Thus the MPI model can be considered as a parallel framework on the level of process to leverage the computation capability of different machines, and the OpenMP model can be considered as a parallel framework on the level of thread to leverage the multi-core computation power of modern computers.

For the parallel programming, one of the most important issues that we need to consider is how to avoid the workload imbalances. For this reason, we need to compare different programming patterns first. There are two well known programming patterns for the MPI framework: peer to peer model and master/slave model. In the peer to peer model, each process behaves equally, that is, all the processes run the same program on different nodes with different data sets. In the master/slave model, one process (i.e., the master process) plays the role of coordinator, and the others (i.e., the slave processes) act as the labor force. In the peer to peer model, the allocation of the tasks is done statically at the beginning of the run of program, and the allocation of tasks cannot be changed during the running time. If all the tasks are homogeneous and the run time is predictable, the peer to peer model would be very convenient to the users.

However, if the execution time of each task is unpredictable and the computational burden is heterogeneous, the peer to peer model is not appropriate. In this case some processes will stay “hungry” while others are busy, so the whole distributed system could not work at the maximum performance due to the workload imbalances. For this situation, the master/slave model would be a much better choice to handle the issue of workload imbalances. In the master/slave model, the master process is dedicated to the task allocation and the slave processes work on their own tasks (e.g., the computation tasks). That is, the allocation of the tasks is done dynamically rather than statically during the runtime of the program. For our inter-related classifier training algorithm (as shown in Section 3), all the tasks can be divided into three

---

**Fig. 5.** The design of our structural learning algorithm.
types: intra-group classifier training, inter-group classifier training, and hybrid level classifier training. Furthermore, the tasks with the same type are also heterogeneous and the execution time is also unpredictable, because our groups of categories are not uniform and the time for solving a SVM optimization problem is uncertain.

Based on the above, a two-layer distributed computation platform with the master/slave model is developed in this work, and this platform is illustrated in Fig. 6. In order to describe our platform more clearly, we state it from two inter-related aspects: programming framework and hardware architecture. From the view of programming (the left part in Fig. 6), there are one master process and a number of slave processes. The master process acts as a task scheduler and allocates the tasks in the queue to the available slave process. The slave processes act as the computation labor forces, which receive the tasks from the master process, do the main computation jobs, and return the final results to the master process. In our two-layer programming framework, the MPI programming is adopted to implement the master process and the communications between the master process and the slave processes, while the OpenMP programming dedicated for the computation jobs in the slave processes.

On the other hand, from the view of hardware architecture (the right part in Fig. 6), all the nodes in the cluster are also divided into two types: one is the master node and all the others are the slave nodes. The whole cluster acts as a distributed-memory model, and for each SMP (Symmetric Multiprocessing) node it acts as a shared-memory model. Obviously, the master node performs the master process while the slave nodes execute the slave process. Our two-layer master/slave platform can not only address the issue of workload balance but also leverage the multi-core computation power of each slave node effectively.

4.3. Parallel inter-related classifier training

In order to simplify the process of task scheduling, a task table is constructed in this work. A number of tasks are listed in our task table, and each task in this task table will be allocated to any available slave process (slave node) during the runtime. From Fig. 7, one can see the construction of our task table clearly. The right part of Fig. 7 (part (b)) illustrates the data structure of our visual concept network. After the graph cut process is performed on our visual concept network (Section 3.2), all the categories are partitioned into a number of groups which are noted as $G_0, G_1, \ldots, G_M$, where $M$ is the total number of our groups. A group $G_i$ consists of one or more concepts, and these concepts form a concepts list noted as $\text{conceptsi}$. Every item (i.e., one particular concept) in the concepts list points to a piece of disk memory where a number of training examples that belong to this concept are stored.

The structure of our task table is shown in the right part (part (a)) of Fig. 7. For large-scale image classification application, the file I/O bottleneck is a critical issue that we have to handle. Our task table is constructed according to the principle that decreases the repeated access of training files (i.e., decreasing the data transform between the hard disk and the RAM in the computer). To do this, we first sort all the groups of image concepts in descending order of the size of each group, and then merge the
groups which consists of a small number of concepts into a new group. After the merging process, all the groups are assigned into a number of cliques. More clearly, each group consists of one or more concepts which are visually similar on our visual concept network, and each clique consists of one or more groups which contains a small number of visually similar concepts. The generation of the cliques follows two principles: (1) the size of each clique is similar; and (2) the biggest two cliques can be loaded to the RAM of computer entirely. This strategy could decrease the access of hard disk significantly.

Finally, we sort the cliques in descending order of their sizes and setup a task for each cliques orderly. The tasks in our task table can be noted as \( T_0, T_1, \ldots, T_{L-1} \), where \( L \) is the total number of tasks (also the total number of cliques). The master process is in charge of allocating the unsolved tasks in our task table to the available slave nodes during the runtime of the training process. To balance the computation load among different slave nodes efficiently, the master task takes a so-called “polling” strategy to implement a task scheduler. That is, the master task detects each slave node periodically and dispatches a slave task to the one that is ready. This process is shown in Algorithm 1. Each slave process is employed to execute a task assigned by the master process. In every task, we train the intra-group, inter-group, and hybrid level classifiers for all the concepts in the current clique. The algorithm for the slave process is shown in Algorithm 2.

**Algorithm 1.** The scheduling scheme of the master task.

**Initialization**
- slave task counter: \( nSTsk \leftarrow 0 \),
- slave nodes counter: \( nSlv \leftarrow 0 \),
- A slave node queue: \( S = \{ \text{all slave nodes} \} \)

**Procedure**
1: while TRUE do
2: if \( nSlv = \text{number of slave nodes} \) then
3: break;
4: end if
5: for each \( s \in S \) do
6: if \( s \) is ready then
7: \( nSlv \leftarrow \text{the number of slave tasks} \) then
8: \( nSlv \leftarrow nSlv + 1 \); 
9: Remove \( s \) from \( S \);
10: else
11: Assign a slave task to \( s \);
12: \( nSTsk \leftarrow nSTsk + 1 \);
13: end if
14: else
15: continue;
16: end if
17: end for
18: end while

**Algorithm 2.** The algorithm of the slave task.

**Initialization**
- the total number of cliques: \( N_{clique} \),
- the index of current task (or current clique): \( F \), 
\( 0 \leq F < N_{clique} \),
\( i = 0 \),

**Procedure**
1: load all the training instances of \( F \);
2: for each group \( G_i \in F \) do
3: train intra-group classifiers for all the concepts belong to \( F \);
4: end for
5: for \( i > F; i < N_{clique} \) do
6: for each group \( G_j \in i \) do
7: train inter-group classifiers between \( G_i \) and \( G_j \); 
8: if \( G_j \) is one of the “five most related” groups of \( G_i \) (vice versa) then
9: train hybrid level for all the concepts in \( G_i \);
10: train hybrid level for all the concepts in \( G_j \);
11: end if
12: end if
13: end for
14: end for
15: end for

**4.4. Indexed storage of SVM models**

For any test sample \( x \), the decision value of a basis binary SVM classifier is formulated as

\[
 f(x) = \text{sgn}\left\{ \sum_{i \in \{SVs\}} \alpha_i k(x_i, x) - b \right\}
\]

(8)

where \( \alpha_i \) is the coefficient of \( i \)-th SV (support vector) and \( b \) is the bias. Usually, we save the SVM model which consists of the SVs and the corresponding coefficients for each basic classifier. However, the size of SVM models would be very large for large-scale classifier training and image classification application (in order of TB), due to not only the number of basic classifiers is very large but also the dimensionality of each feature vector is very high. For example, for the 10,000 categories classification problem with 500 images per class and the 1000 dimensional SIFT BOW features, it takes approximately 400 GB to save the SVM models for the one-against-all approach and 60 TB for the one-against-one approach. This number will increase when using more complex features (typically more than 10,000 dimensions). In addition, there are many duplications of SVs (support vectors) in inter-related SVM models, or rather, different basic binary SVM learners would share some common SVs. To address this issue, we only store the flag which indicates whether a training instance is an SV or not and the corresponding coefficient as shown in Fig. 8. We use the filename of the model file to indicate the source training instance of the SVs. The biggest advantage of this indexed storage method is that the size of SVM models is very small and it is independent with the dimensionality of feature vectors. By using this strategy, the storage complexity is reduced from \( O(nd) \) to \( O(n) \), where \( n \) is the number of support vectors and \( d \) is the feature dimensionality.

Consistent with the model storage method, in the test stage, we first compute the kernel values between the test instances and all the SVs recorded in the model files, and then we look up the specific kernel values and the coefficients to make the final decision. This strategy could remove the repeated computation and accelerate the test process significantly.
By using a visual concept network based classifier design scheme and an MPI based distributed platform, our parallel computing framework could

(a) balance the computation load among different nodes in a computing cluster efficiently via a MPI-based task scheduler;
(b) accelerate the kernel computation on each cluster node by performing shared memory OpenMP programming method;
(c) reduce the storage space for the intermediate SVM models tremendously by performing efficient indexing;
(d) accelerate the test stage by avoiding repeated kernel computation.

Because the partition of our visual concept network is not balanced in terms of group size, it is difficult to estimate the computational complexity exactly. However, we can give some analysis for the situation of balance partition. Assuming that there are totally $N$ categories which are partitioned into $M$ groups, each group consists of $N/M$ categories. By using the LIBSVM with RBF kernel, training SVM has complexity $O(n^2)$, where $n$ is the number of training samples. First we make some analysis of the computation complexity for the training phase for different approaches. For the one-against-all approach, we need to train $N$ binary classifiers each with $Nn$ training samples, so the computation complexity is $O(Nn^2)=O(N^3)$. For the pairwise approach, we need to train $N(N−1)/2$ basic SVM classifiers each with $Nn$ training samples, so the computation complexity is $O(N^2)$. For our inter-related classifier training algorithm, the basic classifiers can be divided into three types: intra-group classifiers, inter-group classifiers, and hybrid classifiers. For the intra-group classifiers, we need to train $M-1(N/M)(N/M−1)/2$ classifiers with $Nn$ training examples and $M-1(N/M)$ classifiers with $Nn$ training examples, so the computation complexity for training intra-group classifiers is $O_{intra-group}(O=n^2/M^2)$. For the inter-group classifier training, we need to train $M(M−1)/2$ classifiers with $2Nn/M$ training samples, so the computation complexity is $O_{inter-group}(O=n^2)$. For training hybrid classifiers, we need to train $NT$ classifiers with $(N/M+1)m$ training samples, where $T$ is the number of most similar groups with each group, so the computation complexity is $O_{hybrid}(O=n^2/m^2)$. Hence the overall computation of our inter-related classifier training algorithm is $O_{intra-group}+O_{inter-group}+O_{hybrid}=O(n^2+N^2/M^2)$. Usually, we have $M < N$ and $M^2 > N$, the training computation complexity of our inter-related classifiers is with the same order of that of the pairwise approach.

In the testing phase, the computational cost is mainly determined by the number of support vectors, so we will give some experimental results to compare the testing time of different approaches in Section 5.

5. Experimental results for algorithm evaluation

To evaluate the performance of our proposed parallel learning method, the experiments are carried out on the following aspects:

(a) we compare the average classification accuracy rates of our parallel classifier training method with the traditional algorithms (i.e., the one-against-all approach and the pairwise approach);
(b) we evaluate the comprehensive performance of parallel classification system on large scale image set.

5.1. Experiment 1: our parallel learning method vs. the traditional approaches

In this experiment, we aim at comparing our method with the one-against-all approach and the pairwise approach. We use two image sets for this experiment: one is a subset of ImageNet image set with 300 classes (noted as ImageNet300 in this section) and the other one is the Caltech256 image set [39]. For the ImageNet300 image set, we use the 1000 dimensional SIFT BOW features which are released by ImageNet. For the Caltech 256 image set, we evaluate our learning algorithm over 255 classes of the Caltech256 image set excluding its clutter category, and the features are also the 1000 dimensional SIFT BOW features. For each image set, we use a universal couple of parameters (i.e., the kernel parameters $\gamma$ and the cost parameters $C$) for all basic SVM learners. The parameters $C$ and $\gamma$ are obtained by the cross-validation process using the one-against-one approach.

The experimental results are shown in Table 1. From these experimental results, one can see that our parallel learning algorithm can provide comparable classification accuracy with the traditional approaches but the computation cost is much lower.

5.2. Experiment 2: performance our distributed classification system on ILSVRC2010

In this section we evaluate our parallel framework for large-scale classifier training and image classification on ILSVRC2010 [1] image set. The ILSVRC2010 image set contains 1000 classes and 1.4M images, and it is used in the ImageNet Large Scale Visual Recognition Challenge in 2010. In order to focus on the parallel large-scale learning, we use the 1000 dimensions BoW representations provided by ILSVRC2010. The nonlinear SVM with RBF kernel (Gaussian kernel) is used as our basic learner. The kernel parameters $\gamma$ and the cost parameters $C$ are obtained by the same manner used in Experiment 1. In the training stage, we randomly sample 500 instances per class from the standard training set. And we use the standard testing set (150K samples) in the test process. The specifications of our distributed platform are shown in Table 2.

As a result, our classification system achieves 13.7% average classification accuracy on the ILSVRC2010 test set, and the classification accuracy rates for all the 1000 categories are shown in Fig. 9. The histogram of classification accuracy on the 1000 classes is shown in Fig. 12. Some hard categories and some easy categories are illustrated in Fig. 13, and one can see that the diversity of the hard categories is much higher than that in the easy categories.

The computation cost (measured in training time and test time) of our parallel classification system and the traditional ones (i.e., one-against-one and one-against-all) on the same distributed

<table>
<thead>
<tr>
<th>Table 1</th>
<th>Average classification accuracy of our parallel learning method and the traditional approaches.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning methods</td>
<td>Average accuracy rate</td>
</tr>
<tr>
<td></td>
<td>ImageNet300</td>
</tr>
<tr>
<td>One-against-all</td>
<td>0.1338</td>
</tr>
<tr>
<td>Pairwise</td>
<td>0.2054</td>
</tr>
<tr>
<td>Our parallel learning method</td>
<td>0.2092</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2</th>
<th>The specifications of our distributed platform.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
<td>CPU frequency (GHz)</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>3.47</td>
</tr>
<tr>
<td>3</td>
<td>2.50</td>
</tr>
<tr>
<td>4</td>
<td>3.20</td>
</tr>
<tr>
<td>5</td>
<td>3.40</td>
</tr>
<tr>
<td>6</td>
<td>3.07</td>
</tr>
</tbody>
</table>
Computation environment are shown in Table 3. In order to describe the speedup efficiency of our proposed distributed algorithm, we first give the variety of the grouping results, as shown in Fig. 9.

<table>
<thead>
<tr>
<th>Learning methods</th>
<th>Training time (days)</th>
<th>Test time (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-against-all</td>
<td>89.4</td>
<td>0.7</td>
</tr>
<tr>
<td>Pairwise</td>
<td>0.8</td>
<td>60.8</td>
</tr>
<tr>
<td>Our parallel learning method</td>
<td>3.5</td>
<td>0.35</td>
</tr>
</tbody>
</table>

* The results of one-against-one and one-against-all approaches are obtained by estimation rather than experiments.

Fig. 10. The size of each visual similar groups for the ILSVRC2010 data set.

Fig. 11. The illustration of speedup by using the OpenMP programming. We take three representative tasks according to the three task types. Specifically, we chose task 1 for intra-group task, task 177 for inter-group task, and task 294 for hybrid task.

Fig. 12. The histogram of the classification accuracy of the ILSVRC2010 image set.
From Fig. 10, the group size is very diverse from each other, so the computational cost of the training is also very diverse. Hence the hybrid of MPI and OpenMP framework is suitable for our inter-related heterogeneous classifier training tasks.

The inter-related classifier training process can be efficiently accelerated by our distributed approach from two aspects: the task level MPI framework and the computation level OpenMP programming. Benefitting from the MPI based task level parallelism, our proposed algorithm can obtain a $n \times$ speedup due to the efficient balancing strategy, where $n$ is the number of computational nodes. The acceleration induced by the OpenMP programming is not as clear as that caused by the MPI based method because of some complex on-chip communication mechanism. It is infeasible to train all the classifiers sequentially to obtain

---

**Fig. 13.** Some hard categories and some easy categories of ILSVRC2010: the category names, the classification accuracy rates, and some sample images. (a) The easiest categories and (b) the hardest categories.
the acceleration effect of the parallel approach due to the high computational cost. However, we can give some representative results with and without OpenMP acceleration, as shown in Fig. 11. In Fig. 11, we evaluate the OpenMP parallel mechanism by using three tasks corresponding to the three types of training tasks (i.e., intra-group task, inter-group task, and hybrid task). From the experimental results, we can find that about 5–6× speedup can be obtained by using the OpenMP parallel mechanism. Based on the above analysis, about 25–30× speedup can be obtained by our hybrid MPI and OpenMP parallel approach.

Besides that, the total storage size in our experiments is listed in Table 4. From the experimental results, one can see that our proposed distributed computing method could accelerate both the training and testing stages and reduce the storage size effectively.

6. Conclusions

In this paper, we mainly focus on addressing two critical issues of huge computation cost and huge storage/memory cost for the task of large-scale classifier training and image classification. We first explore the parallelism of our structural learning method proposed in this paper, and then develop an efficient storage method to save SVM models and avoid the repeated computation of support vectors (SVs) in the testing process for image classification. A high performance MPI-based parallel framework is developed to perform large-scale classifier training and image classification on multiple machines. The positive results are obtained from ILSVRC2010 image set. Obviously, more comprehensive features (such as color-SIFT and GIST) and more accurate feature coding methods (e.g., LLC or image set) can be applied to improve classification accuracy of our parallel computing approach in our future work.

Acknowledgments

The authors would like to thank the reviewers for their insightful comments and suggestions to make this paper more readable. This research is supported by National Natural Science Foundation of China (Grant nos. 61272285 and 61375023) and National High-Technology Research and Development Program of China (863 Program, Grant nos. 2014AAA012301 and 2012AAA010904).

References


Kuizhi Mei received the B.E and M.S. degrees in electronics engineering and the Ph.D. degree in pattern recognition and intelligence systems in 1999, 2002, and 2006, respectively, Xi’an Jiaotong University, Xi’an, China. He is currently an Associate Professor of the Institute of Artificial Intelligence and Robotics, Xi’an Jiaotong University. His research interests include image analysis and embedded vision computing.

Peixiang Dong received the B.S. degree in electronic engineering from Xi’an Jiaotong University, Xi’an, China, in 2008. He is currently working toward the Ph.D degree in Institute of Artificial Intelligence and Robotics at Xi’an Jiaotong University. His research interests include computer vision and large scale image classification.

Hao Lei received the B.S. degree in electronic engineering from Xi’an Jiaotong University, Xi’an, China, in 2008. He is currently pursuing the Ph.D degree in Institute of Artificial Intelligence and Robotics at Xi’an Jiaotong University. His research interests include computer vision, image/video analysis and representation.