A Study on Non-Linear Machine Learning Techniques and Its Application to Content Based Image Annotation

by

Liang Sun

Supervised

by

Shinichi Yoshida

Submitted in partial fulfillment of the requirements for degree

of the Doctor of Engineering in the Graduate School of

Engineering, Kochi University of Technology

February 2012

Abstract

The number of image archives on the Internet is growing rapidly with the proliferation of user contributed images. Thus searching for the images that match a user query presents a significant challenge. In the real world image systems, many images are constantly created without direct annotations of semantic content. This creates the need for content based image retrieval (CBIR), which is conducted based on the low level features such as color, texture, and shape. However, the CBIR system still suffers from the "semantic gap problem", which implies that the low level image contents are not effective enough to encapsulate the high level semantics. One natural way to mitigate the "semantic gap problem" is to assign annotations onto images. The methods that annotate images automatically based on low level visual contents are referred to as content based image annotation (CBIA). The unceasing progress in the fields of computer vision and machine learning has provided opportunities to develop CBIA systems. However, due to the complexity of the real world image systems, effective and efficient image annotation is still a challenging problem.

This thesis is an exploratory study of non-linear machine learning techniques to address the problems in CBIA system. It is obvious that there always be a need for better optimization in the CBIA system, since the CBIA system generally possesses parameters that can be adjusted to produce more desirable outcomes. Thus one major component of this thesis is developing a cooperative particle swarm optimizer for large scale numerical optimization. Firstly, a statistical model is proposed to explore the interdependence among variables. Secondly, the algorithm partitions large scale problems into small scale sub-problems based on the prior knowledge with respect to the variable interdependencies. Finally, a CPSO framework is proposed to optimize the sub-problems cooperatively. The results of simulated experiments on the benchmark functions demonstrate the effectiveness of the proposed optimizer, as compared with the performance of other cooperative optimization algorithms.

Another important problem involved in CBIA system is data clustering, which

associates with grouping a set of data into clusters (subsets) so that the data in the same cluster are analogous in properties that are relevant to data analysis. The second major component of this thesis is developing a support vector and K-Means based hybrid data clustering algorithm. Firstly, an empirical study is conducted to guide better selection of the standard deviation of the Gaussian kernel. Following this, the outliers which increase problem complexity are identified and removed by training a global support vector clustering (SVC) model. Finally, several local SVCs are trained for the clusters and each removed data point is labeled according to the distance from it to the local SVCs. The results of simulated experiments on 2-D data sets and the UCI machine learning benchmark datasets demonstrate that the proposed algorithm compared favorably with other algorithms.

The third major component of this thesis is to develop a support vector based CBIA system. In the proposed system, clusters of images with manually tagged words are used as training instances. Images within each cluster are modeled using a kernel method, where the image vectors are mapped to a higher dimensional space and the vectors identified as support vectors are used to describe the cluster. To measure the extent of association between an image and a support vector described model, the distance from the image to the model is computed. A closer distance indicates a stronger association. Moreover, the word to word correlations are also considered in the annotation framework. For an image to be tagged, the system exploits the distance from the image to the models and the word to word correlations in a probabilistic framework to predict annotation words. The results of simulated experiments on three benchmark image sets demonstrate the effectiveness of the proposed system.

Contents

Acknowledgement

Firstly, I would like to express my great appreciation to my advisor Prof. Shinichi Yoshida, for his help, support, advice, assistance and encouragement during the period when I studied in Kochi University of Technology. The two year and a half study in will have a far reaching impact on my life.

I would like to thank Prof. Akio Sakamoto, Prof. Shuoyu Wang, Prof. Makoto Iwata, Prof. Yoshiaki Takata for their constructive suggestions and comments during the production of this thesis.

I would like to thank Prof. Xiangshi Ren in School of Information, Kochi University of Technology, Prof. Yanchun Liang in College of Computer Science and Technology, Jilin University for the promotion of collaboration between Kochi University of Technology and Jilin University. This collaboration provided me this opportunity to study in Japan.

I also would like to thank the members of Yoshida laboratory for their friendship and contributions to my work.

Meanwhile, I would like to express my appreciation to my parents for their generous love and encouragement during my stay in Japan.

List of figures

List of tables

Chapter 1: Introduction

Machine learning is an essential branch of artificial intelligence. It is a scientific discipline concerned with designing and developing algorithms that allow computers to mimic human learning activities such as classifying, repeating, recognizing, evolving, etc. Content based image annotation (CBIA) is a process by which a computer system automatically generates metadata such as caption, keywords, or filenames based on visual contents of images.

This thesis is an exploratory study of the non-linear machine learning techniques to address the problems in CBIA. It has three integral parts. The first part associates with developing a cooperative optimization algorithm. The second part associates with developing a support vector based data clustering algorithm. And the third part associates with developing a content-based image annotation system, which is proposed based on the basic formulations of support vector clustering algorithm.

1.1 Motivation

The invention of digital camera provides people the opportunities to take pictures in everyday life, and the development of Internet techniques facilitates people sharing the pictures conveniently. As a result, the number of image archives on the Internet grows at a phenomenal rate. Take a report released in 2007 as an example, the flickr.com, which is an Internet photo sharing system, has about 40 million monthly visitors and about two billion photos [1]. Due to the large amount of images, browsing, searching and retrieving images that match a user query presents a significant challenge. Many of the traditional and common image retrieval systems such as Google and Yahoo! utilize metadata such as captioning, keywords, or filenames so that the retrieval task can be performed over the textual descriptions. However, in the real word image systems, many images are constantly created without direct annotations of semantic content, which limits the ability of the text based systems. This creates the need for content based image retrieval (CBIR) [2]-[4]. CBIR is a computer system that performs retrieval task over the low level visual content of images such as texture, shape and color other

than the textural descriptions. Research on CBIR has attracted the attention of researchers in various fields including computer vision and machine learning. However, many of the CBIR systems still suffer from the “semantic gap problem”, which implies that the low-level image contents are not effective enough to encapsulate the high level semantics [2], [5].

One natural way to mitigate the “semantic gap problem” is to assign tags onto images. Appropriate tagging can help to increase the retrieval efficiency. However, the manual tagging is time consuming, labor intensive and expensive. Due to this reason, there has been a surge of research interest in content based image annotation. However, as far as CBIA concerned, there are still some problems unsolved. If they are suitably addressed, more robust and efficient systems can be designed. Thus the basic motivation of this research is developing effective and efficient methods for the CBIA system, so that the computers can understand, index, and annotate images automatically.

Optimization problems arise in a variety of fields, including engineering, science and business. Effective and effective optimization algorithms are always needed to tackle the increasingly complex real world optimization problems. It is obvious that there always be a need for better optimization algorithm in the CBIA system, since the CBIA system problem generally possesses parameters that can be adjusted to produce more desirable outcomes. Can we solve large scale optimization problem with 500 or more dimensions efficiently? This is the motivation of the first part of this research.

Another important problem involved in CBIA system is data clustering, which associates with grouping a set of data into clusters (subsets) so that the data in the same cluster are analogous in properties that are relevant to data analysis. Numerous clustering algorithms have been proposed, with varying degree of success. However, their effectiveness and advantages usually deteriorate when it is applied to solving complex real world problems, e.g., those with large proportion of noise data points and connecting clusters. Thus how to develop an even more effective data clustering algorithm for the CBIA system is the motivation of the second part of this research.

In a CBIA system, substantial machine learning techniques are required to fill the gap between the low-level image visual contents and the high-level semantics. Among the machine learning techniques, the support vector cluster (SVC) [6]-[8] is a recently developed algorithm inspired by the support vector machine (SVM) [9]. The SVC has many advantages over other data clustering algorithms for its ability to determine the system topological structure without prior knowledge with respect to the system itself, to delineate cluster boundaries of irregular shapes, and to deal with outliers by employing a soft margin constant. Whether the SVC can be used to solve problems in the CBIA system is still a question unanswered. The motivation of the third part of this research is to answer this question.

1.2 Objectives

The objectives of this thesis can be summarized as follows:

1. To develop a cooperative particle swarm optimizer for large scale numerical optimization, which includes developing a statistical model to learn prior knowledge of a problem with respect to the variable interdependencies, proposing a decomposition method based on the prior knowledge, and developing a cooperative particle swarm optimization framework.
2. To develop a support vector and K-Means based hybrid data clustering algorithm, which includes a SVC training method, an empirical study, and a support vector and K-Means based hybrid algorithm.
3. To develop a support vector based CBIA system. The objective is to consummate two major components of the system, i.e., the training process and the annotating process.

1.3 Methodology

The cooperative particle swarm optimizer developed in this thesis is achieved by adopting the divide and conquer strategy. Decomposition decision regarding variable interdependencies often plays a significant role in the algorithm's performance. Algorithms that do not consider variable interdependencies often lose their effective and

advantage when applied to solve non-separable problems. In this thesis, we propose a cooperative particle swarm optimizer with statistical variable interdependence learning (CPSO-SL). A statistical model is proposed to explore the interdependence among variables. With these interdependencies, the algorithm partitions large scale problems into overlapping small scale sub-problems. Moreover, a CPSO framework is proposed to optimize the sub-problems cooperatively. Theoretical analysis is conducted for further understanding of the proposed CPSO-SL.

In the support vector and K-Means based hybrid data clustering algorithm, an empirical study is conducted to guide better selection of the standard deviation of the Gaussian kernel. Following this, the outliers which increase problem complexity are identified and removed by training a global SVC. The refined data set is then clustered by a kernel based K-Means algorithm. Finally, several local SVCs are trained for the clusters and then each removed data point is labeled according to the distance from it to the local SVCs.

In the support vector based system for CBIA, clusters of images with manually tagged words are used as training instances. Images within each cluster are modeled using a kernel method, where the image vectors are mapped to a higher dimensional space and the vectors identified as support vectors are used to describe the cluster. To measure the extent of association between an image and a support vector described model, the distance from the image to the model is computed. A closer distance indicates a stronger association. Moreover, the word to word correlations are also considered in the annotation framework. For an image to be tagged, the system exploits the distances from the image to the models and the word to word correlations in a probabilistic framework to predict annotation words.

1.4 Thesis Outline

In Chapter 2, for a numerical optimization problem, firstly, a statistical model is proposed to learn the variable interdependencies. Based on the interdependencies, a method is then proposed to decompose large scale problem into overlapping small scale sub-problems. Following this, a cooperative particle swarm optimizer with statistical

variable interdependence learning (CPSO-SL) is proposed to optimize the sub-problems cooperatively. To give deeper insight into the proposed CPSO-SL, further theoretical analysis is carried out. The performance of the CPSO-SL is examined by means of experiments on benchmarks with different dimensions and levels of hardness and is compared with the performance of other recently reported cooperative optimization algorithms.

In Chapter 3, firstly, a review of the basic concepts of the SVC and K-Means algorithm is presented. Following this, a new SVC training method is presented. And then an empirical study is conducted to guide better selection of the standard deviation of the Gaussian kernel. The details of the SVC and K-Means based hybrid intelligent data clustering are then presented. Finally, the experimental settings and the simulated results are then presented for illustration and comparison.

In Chapter 4, firstly, some of the previous work on CBIA, in particular, the generative modeling methods are reviewed. Following this, the support vector based modeling method and the probabilistic modeling method for assignment of annotation words are presented, respectively. And then the simulated results are presented for illustration and comparison.

Chapter 5 presents a summary of the outcomes of this thesis. And some possible extensions of current research are also discussed.

Chapter 2: A cooperative particle swarm optimizer

2.1 Introduction

Optimization problems arise in a variety of fields, including engineering, science, and business [10]. Effective and efficient optimization algorithms are always needed to tackle increasingly complex real world optimization problems. Stochastic optimization algorithms, such as genetic algorithm (GA) and particle swarm optimization (PSO), have been shown to be successful in dealing with many optimization problems [11]-[21]. However, most of these algorithms still suffer from the "curse of dimensionality", i.e., their performance deteriorates rapidly as the dimensionality of the problem increases. Generally, many of the traditional stochastic algorithms can perform well on moderate scale problems, but they may have difficulty in optimizing large scale problems with 500 or more dimensions.

One natural way to address the "curse of dimensionality" is to adopt the divide-and-conquer strategy. The original divide-and-conquer algorithm is the cooperative co-evolutionary genetic algorithm (CCGA) [22]-[24]. The CCGA operates by decomposing a large scale problem into small scale sub-problems and optimizing the sub-problems by means of separate GAs; the solution to the problem is obtained by combining the sub-solutions found by each of the separate GA. Other divide-and-conquer algorithms include fast evolutionary programming with cooperative co-evolution (FEPCC) [25], cooperative particle swarm optimization (CPSO) [26]-[28], differential evolution with cooperative co-evolution (DECC) [29]-[31], and so on.

A key issue with regards to divide-and-conquer is the task of problem decomposition, the process of partitioning a large scale problem into small scale sub-problems so that the interdependencies among different sub-problems are minimized. Decomposition decision regarding variable interdependencies plays a significant role in the algorithm's performance. Generally, algorithms that do not consider variable interdependencies are

effective for separable problems, but have difficulty solving non-separable problems. Research on variable interdependencies has already attracted the attention of researchers and several methods have been proposed. For example, in [32], the authors presented a preliminary learning process for the recognition of epistemic links in problems, and in [33], a correlation based variable partitioning scheme was designed to alleviate the problems associated with selection of a number of sub-problems and variable partitioning. These methods facilitated research into variable interdependencies. However, real world optimization problems are far more complex and there is still no rigorous algorithm for giving deeper insight into the problem itself. In view of the above, this study will be conducted with the following objectives.

1. To develop a statistical model to learn prior knowledge of a problem with respect to the variable interdependencies.
2. To propose a decomposition method based on the prior knowledge.
3. To develop a cooperative particle swarm optimizer for global numerical optimization.

2.1 Problem decomposition

2.2.1 Statistical variable interdependence learning

In this thesis, the following numerical optimization problem is considered:

Find: $\vec{x}^* \in E$

Such that: $\forall \vec{x} \in E, f(\vec{x}^*) \leq f(\vec{x})$,

where $E = [b_{l1}, b_{u1}] \times [b_{l2}, b_{u2}] \times \cdots \times [b_{ln}, b_{un}] \subseteq R^n$ is the bounded solution space, $\vec{x} \in E$ is the solution vector, and $f: E \rightarrow R$ is the objective function.

The ideal decomposition method is to partition the problem into sub-problems so that the variables within each sub-problem are non-separable, and the variables among different sub-problems are separable. For example, the ideal decomposition strategy for the problem shown in Fig.2.1(a) is sub-problem 1 = $\{v_1, v_2, v_3\}$ and sub-problem 2 = $\{v_4, v_5\}$. For many problems, interdependencies occur among most variables. For example, in the problem shown in Fig.2.1(b) interdependencies occur among all the variables. As a result, an ideal decomposition strategy is difficult to obtain. An

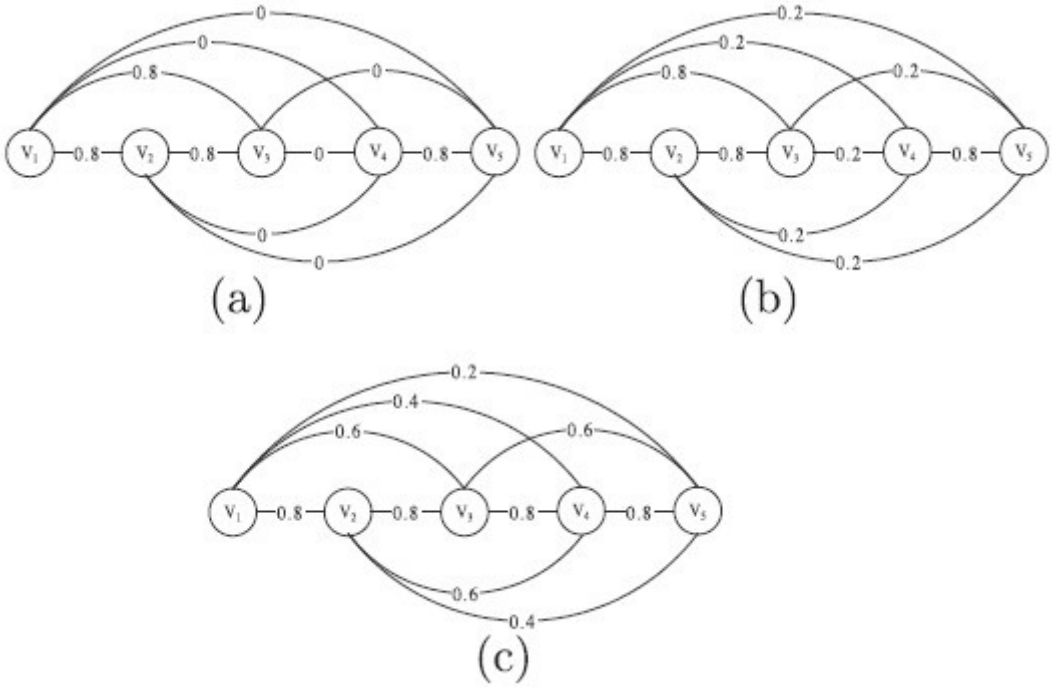


Fig. 2.1. Illustrative example for the variable interdependencies. (The digit in the middle of the connection line represents the degree of interdependence between variables)

alternative method is to partition a high dimensional problem into several low dimensional sub-problems. Within each sub-problem, the variable interdependencies are maximized, and among different sub-problems, the variable interdependencies are minimized. In this case, the optimal decomposition strategy is sub-problem 1 = $\{v_1, v_2, v_3\}$ and sub-problem 2 = $\{v_4, v_5\}$. At this point, the issue lies with quantifying the degree of interdependence between each pair of variables. To address this, a statistical variable interdependence learning model is proposed. The proposed model worked as follows.

1. Suppose we have $\vec{\alpha} = (\cdots, x_i, \cdots, x_j, \cdots)$, $\vec{\beta} = (\cdots, x_i', \cdots, x_j \cdots)$, and it is satisfied that $f(\vec{\alpha}) \leq f(\vec{\beta})$. If we change the value of x_j to x_j', resulting in $\vec{\alpha}' = (\cdots, x_i, \cdots, x_j', \cdots)$, and $\vec{\beta}' = (\cdots, x_i', \cdots, x_j' \cdots)$, and $f(\vec{\alpha}') > f(\vec{\beta}')$, then we call that variable x_i is affected by x_j under context vector $\vec{c} = (\cdots, x_{i-1}, -, x_{i+1}, \cdots, x_{j-1}, -, x_{j+1})$[1]
2. The extent to which the influence of variable x_i can be affected by x_j can be

[1] In a context vector, the variables marked with '-' have no value.

1. **Function** d = ***Interdependence*** (i, j)
   //estimate the extent to which x_i can be affected by x_j.
2. $cnt = 0$;
3. **for** k=1: N *// N is number of statistical samples.*
4. set $\vec{c}$=$(\cdots, x_{i-1}, -, x_{i+1}, \cdots, x_{j-1}, -, x_{j+1}, \cdots)$; *//create a context vector.*
5. set x_i, x_i' and x_j, satisfying $x_i \neq x_i'$;
6. let $\vec{\alpha} = (\cdots, x_i, \cdots, x_j, \cdots)$;
7. let $\vec{\beta} = (\cdots, x_i', \cdots, x_j, \cdots)$;
8. **if** $f(\vec{\alpha}) > f(\vec{\beta})$
9. exchange $\vec{\alpha}$ and $\vec{\beta}$;
10. **endif** *// to make sure $f(\vec{\alpha}) \leq f(\vec{\beta})$.*
11. set x_j', satisfying $x_j \neq x_j'$;
12. let $\vec{\alpha'} = (\cdots, x_i, \cdots, x_j', \cdots)$;
13. let $\vec{\beta'} = (\cdots, x_i', \cdots, x_j', \cdots)$;
14. **if** $f(\vec{\alpha'}) > f(\vec{\beta'})$
15. $cnt = cnt + 1$;
16. **endif**;
17. **endfor**
18. **return** $d = cnt/N$;

Fig. 2.2. Pseudo code for the statistical variable interdependence learning model.

estimated by the probability for the inequality change, i.e., $P\{f(\vec{\alpha}') > f(\vec{\beta}')\}$. We can estimate the probability $P\{f(\vec{\alpha}') > f(\vec{\beta}')\}$ by a statistical method, i.e., selecting a number of context vectors $\vec{c}$ as statistical samples, and checking the effect of variable x_j on variable x_i under each statistical sample.

Fig.2.2 shows the pseudo code for the interdependence learning model. In the proposed model, N statistical samples are created. Each sample consists of two solution vectors $\vec{\alpha}$ and $\vec{\beta}$ with inequality $P\{f(\vec{\alpha}) \leq f(\vec{\beta})\}$. The vectors $\vec{\alpha}$ and $\vec{\beta}$ are identical except x_i values. The effect of variable x_j on x_i is affected by turning x_j values; if the new solution vectors $\vec{\alpha}'$ and $\vec{\beta}'$ achieve a change in the inequality, we call x_i is affected by x_j. The extent to which the influence of variable x_i can be affected by x_j is estimated by the probability for the inequality change. And the probability is estimated

```
1. Function [S_1, S_2, ..., S_n] = Decomposition (S)
   //decompose the variable set S into n overlapping subsets.
2. set threshold value r;
3. for i=1: n   // n is number of variables in S.
4.    set S_i = {x_i};
5.    for j=1: n
6.       if d_ij ≥ r
7.          S_i = S_i ∪ {x_j};
8.       endif
9.    endfor
10. endfor
11. return [S_1, S_2, ..., S_n];
```

Fig. 2.3. Pseudo code for the decomposition method.

by the statistical approach, i.e., $d_{ij} = \frac{cnt}{N} = \hat{P}\{f(\vec{\alpha}') > f(\vec{\beta}')\}$, where cnt is the times of inequality change.

The main differences among the variable interdependence learning methods in [32], [33], and the proposed method are as follows. In [32], two variables are considered as interacted when a new candidate solution where both variables are changed is better than another candidate solution where only one variable is changed. In [33], a correlation matrix is computed based on some of the candidate solutions, and the variable interdependencies are obtained from the correlation matrix. In contrast, the proposed method captures variable interdependencies by a statistical method.

2.2.2 Decomposition

The next issue lies with decomposing large scale problems into small scale sub-problems. With a threshold value $0 \le r \le 1$, we want to obtain a partition of variable set S, which satisfies that $S_1 \cup S_2 \cup \cdots \cup S_k = S$ and $S_i \cap S_j = \emptyset$ $(i \ne j)$. For each pairs of variables x_i and x_j, if x_i and x_j are in the same subset, then $d_{ij} \ge r$ or $d_{ji} \ge r$, else, $d_{ij} < r$ or $d_{ji} < r$. It is possible, however, that the above partition is difficult to

obtain. For example, in the problem shown in Fig.2.1(c), $d_{12} = 0.8$, $d_{21} = d_{23} = 0.8$, $d_{32} = d_{34} = 0.8$, $d_{43} = d_{45} = 0.8$, and $d_{54} = 0.8$. Suppose that the threshold value is $r = 0.8$,, then the variables x_1, x_2, x_3, x_4, x_5 should be allocated to the same subset. In this case, the problem is not decomposed. This is due to the fact that the variables are interacted with different extents, and the most variables will be allocated to the same sub-problem due to their direct or indirect interdependencies.

In order to address this issue, we propose a new decomposition method. For an n dimensional problem, we first create n subsets. They are initialized as $S_1 = \{x_1\}$, $S_2 = \{x_2\}$, ⋯, $S_n = \{x_n\}$, respectively, where x_i is called the core of subset S_i. And then, for each subset S_i, we select and recruit all the variables which affect x_i with a degree no less than the predefined threshold value r. Fig.2.3 shows the pseudo code for the proposed decomposition method. Keep in mind that the subsets S_1, S_2, ⋯, S_n are overlapped. To develop cooperative optimization algorithms using subsets S_1, S_2, ⋯, S_n, the following problems need to be solved.

1. Within each subset, the variables are optimized separately. Since the optimization algorithm requires an objective function to evaluate its performance, this introduces the problem of how to calculate the objective function values.
2. To optimize the variables in subsets S_1, S_2, ⋯, S_n, n optimization algorithms work cooperatively. How to exchange information among the n optimization algorithms is another problem.
3. Since subsets S_1, S_2, ⋯, S_n are overlapped, a variable x_k that appears in one subset S_i may appear in another subset S_j, i.e., $x_k \in S_i \cap S_j$. This introduces the problem of how to construct the composite n dimensional solution with the optimized variables.

A possible solution to these problems will be presented in Section 2.3.2.

2.3 Cooperative particle swarm optimizer

2.3.1 PSO and CPSO

PSO is a swarm based computation technique which uses the metaphor of the social

behavior of flocks of birds and schools of fish [34], [35]. It has been used to solve many optimization problems such as neural network training [36] and job shop scheduling [37]. In a PSO system, particles fly around in solution space. At time t, the position and velocity of the i-th particle are represented as $\vec{x}_i(t)$ and $\vec{v}_i(t)$, respectively. The best previous position of the i-th particle is denoted as $\vec{p}_i(t)$, and the best position found by the whole swarm is denoted as $\vec{p}_g(t)$ (Hereafter, without loss of generality, the vectors $\vec{p}_i(t)$ and $\vec{p}_g(t)$ are abbreviated as $\vec{p}_i$ and $\vec{p}_g$, respectively). During a search process, the i-th particle changes its velocity and position according to the following equations:

$$v_{ij}(t+1) = w \cdot v_{ij}(t) + c_1 \cdot r1_{ij} \cdot \left(p_{ij} - x_{ij}(t)\right) + c_2 \cdot r2_{ij} \cdot (p_{gj} - x_{ij}(t)) \tag{2.1}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{2.2}$$

where w is the inertia weight, c_1 and c_2 are learning rates which are nonnegative constants, $r1_{ij}$ and $r2_{ij}$ are random numbers in the range [0, 1] [38]. The particle's velocity on each dimension is clamped to the range $[-v_{max}, v_{max}]$ to control excessive roaming of the particle. The algorithm stops when the $\vec{p}_g$ hits the global optimum with a predefined accuracy or when the algorithm runs for a maximum number of iterations. In this thesis, we use the above form of PSO due to its simplicity and efficiency compared with other PSO variants.

CPSO algorithms have already been proposed by many researchers, with varying degrees of success. For example, in [36], the authors proposed a CPSO where the input vector is partitioned into several sequential sub-vectors, with each optimized cooperatively in its own swarm. In [39], the authors applied the CCGA [22]-[24] technique to PSO and proposed two CPSO models. One model, namely CPSO-S_K, is a direct application of CCGA technique to PSO, and the other, namely CPSO-H_K, combines the standard PSO with the CPSO-S_K model. In [28], the authors applied the random grouping adaptive weighting strategies of the DECC algorithm [31] to the CCPSO and proposed a cooperatively coevolving PSO algorithm.

2.3.2 CPSO framework

We propose a CPSO framework with statistical variable interdependence learning.

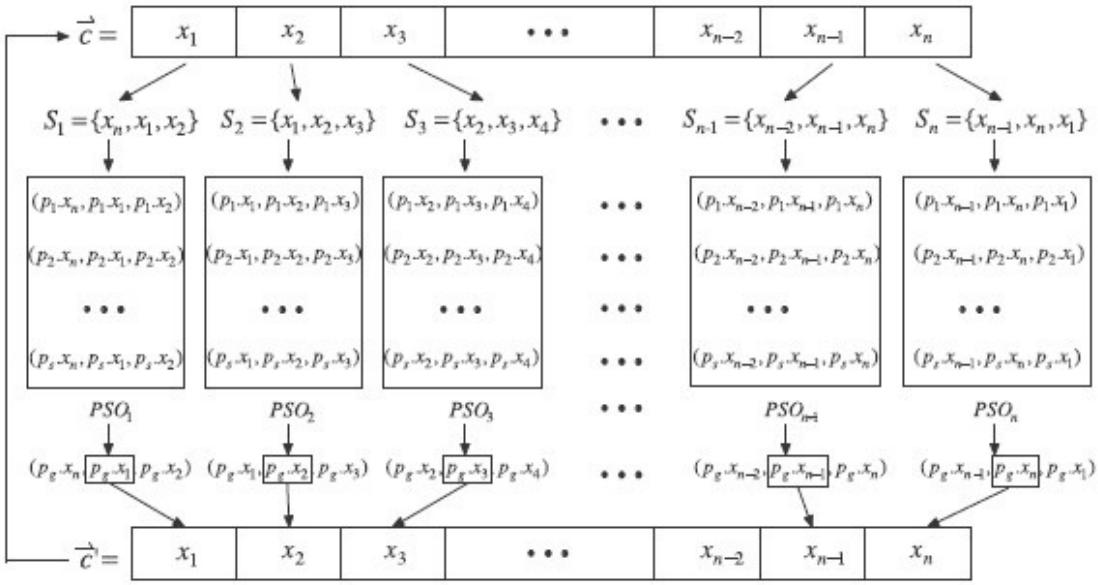


Fig. 2.4. Illustrative example for the CPSO-SL framework.

Compared with existing CPSO algorithms, the exclusive features of the proposed CPSO framework include the following.

1. The sub-problems are overlapped, with each having a sub-problem core.
2. Each sub-problem is optimized by means of a separate PSO.
3. A global solution is used as shared memory, where the separate PSOs can take variables and post optimization results.

Fig.2.4 is an illustrative example of the proposed framework. For the convenience of illustration, we assume that: $S_1 = \{x_n, x_1, x_2\}$, $S_2 = \{x_1, x_2, x_3\}$, ⋯, $S_{n-1} = \{x_{n-2}, x_{n-1}, x_n\}$, $S_n = \{x_{n-1}, x_n, x_1\}$. The key steps of the framework can be summarized as follows:

1. Create a global solution $\vec{c}$ randomly.
2. Partition the variable set into n overlapping subsets S_1, S_2, ⋯, S_n.
3. Set $itr = 1$ to start a new cycle.
4. Optimize the variables in the subsets by means of separate PSOs; in the i-the PSO, the variables that do not appear in S_i are kept constant (with their value taken from $\vec{c}$).
5. Construct a new solution $\vec{c}'$ by taking and concatenating the optimized sub-problem cores.

6. Update the global solution $\vec{c}$ by $\vec{c}'$.
7. Output $\vec{c}'$ as optimization result and stop if a maximum number of function evaluations (FEs) is reached, else, set $itr = itr + 1$, go to Step 4 for the next cycle.

2.4 CPSO-SL behavior

In this section, we conduct theoretical analysis to give deeper insight into the execution process of CPSO-SL and to guide better parameter selection.

2.4.1 Separable and non-separable problems

One important idea for the optimization problem is variable separability. The definition of separable and non-separable variables can be presented as follows.

Definition 1. Given an optimization problem $f(\vec{x})$, variable x_i is said to be separable from variable x_j, if for all the context vectors

$$\forall \vec{c}=(\cdots, x_{i-1}, -, x_{i+1}, \cdots, x_{j-1}, -, x_{j+1}, \cdots)$$

the following constraint is satisfied:

$$\forall x_i, x_i' \in [b_{li}, b_{ui}], (x_i \neq x_i') \text{ and } \forall x_j \in [b_{lj}, b_{uj}],$$

$$\text{if } f(\vec{\alpha}) \leq f(\vec{\beta})$$

$$\text{then } \forall x_j' \in [b_{lj}, b_{uj}],\ f(\vec{\alpha}') \leq f(\vec{\beta}')$$

where

$$\vec{\alpha} = (\cdots, x_i, \cdots, x_j, \cdots),\ \vec{\beta} = (\cdots, x_i', \cdots, x_j, \cdots)$$

$$\vec{\alpha}' = (\cdots, x_i, \cdots, x_j', \cdots),\ \vec{\beta}' = (\cdots, x_i', \cdots, x_j', \cdots)$$

if the above constraint is not satisfied, variable x_i is said to be non-separable from variable x_j.

Variable x_i is separable from variable x_j means that x_i is independent of x_j. Variable x_i is non-separable from variable x_j means that the influence of x_i on the objective function is affected by x_j. Based on Definition 1, we provide the definition of separable and non-separable problems.

Definition 2. An optimization problem $f(\vec{x})$ is called a separable problem if $\forall x_i, x_j \in \{x_1, x_2, \cdots, x_n\} (i \neq j)$, and variables x_i and x_j are separable from each other. Otherwise, $f(\vec{x})$ is a non-separable problem.

A separable problem is one wherein all of its variables are separable. On the other hand, a non-separable problem is one wherein some of its variables are interacted.

2.4.2 Variable interdependencies and problem decomposition

In a non-separable problem, interdependencies occur among some of its variables. The probability q_{ij} for variable x_i affected by x_j can be defined as follows.

Definition 3. Given an optimization problem $f(\vec{x})$, the probability q_{ij} for variable x_i affected by x_j is defined by:

$$q_{ij} = P\{f(\vec{\alpha}') > f(\vec{\beta}')\}$$

where (1) $\vec{\alpha}$ and $\vec{\beta}$ are identical solution vectors except for v_i values; (2) $\vec{\alpha}$ and $\vec{\beta}$ satisfy that $f(\vec{\alpha}) \leq f(\vec{\beta})$; (3) $\vec{\alpha}'$ and $\vec{\beta}'$ are obtained by turning x_j values of $\vec{\alpha}$ and $\vec{\beta}$ to x_j'.

The CPSO-SL partitions the variable set into overlapping subsets. Each of the subset is optimized by means of a separate PSO. The following theorem shows the probability for the separate PSO obtaining the global optimal variable values.

Theorem 1. Given an optimization problem $f(\vec{x})$ with global optimum solution $\vec{x}^* = (x_1^*, x_2^* \cdots, x_n^*)$, suppose:

1. $S_m = \{x_{m_1}, x_{m_2} \cdots, x_{m_k}\}$ is a subset of variable set S;
2. $\vec{c} = (\cdots, x_{m_i-1}, -, x_{m_{i+1}}, \cdots, x_{m_k-1}, -, x_{m_k+1})$ is a context vector;
3. $y_{m_1}^*, y_{m_2}^*, y_{m_k}^*$ are local optimal values of variables $x_{m_1}, x_{m_2} \cdots, x_{m_k}$ under context vector $\vec{c}$,

then the probability for $y_{m_i}^*$ to be equal to the global optimal value $x_{m_i}^*$ is:

$$P\{y_{m_i}^* = x_{m_i}^*\} = \prod_{S,(x_s \notin S_m)} (1 - q_{m_i s})$$

Proof. Because $\vec{x}^*$ is the global optimum solution, we have:

$$f(\vec{x}^*) \leq f(\vec{x}) \tag{2.3}$$

Form Eq. (2.3), it follows that:

$$P\{f(\cdots, x_{m_i}^*, \cdots, x_s^*, \cdots) \leq f(\cdots, x_{m_i}, \cdots, x_s, \cdots)\} = 1 \tag{2.4}$$

Suppose $x_s \notin S_m$ and change x_s^* to x_s whose value is equal to that in $\vec{c}$, we have:

$$P\{f(\cdots, x_{m_i}^*, \cdots, x_s, \cdots) \leq f(\cdots, x_{m_i}, \cdots, x_s, \cdots)\} = 1 - q_{m_i s} \tag{2.5}$$

This process is repeated until all the variables $x_s (x_s \notin S_m)$ are exhausted, then:

$$P\{f(\cdots,x^*_{m_1},\cdots,x^*_{m_i},\cdots,x^*_{m_k},\cdots) \le f(\cdots,x^*_{m_1},\cdots,x_{m_i},\cdots,x^*_{m_k},\cdots)\} = \prod_{S,(x_s \notin S_m)} (1 - q_{m_i s}) \quad (2.6)$$

Because $y^*_{m_1}, y^*_{m_2}, y^*_{m_k}$ are local optimal values of variables $x_{m_1}, x_{m_2} \cdots, x_{m_k}$, we have:

$$P\{f(\cdots,y^*_{m_1},\cdots,y^*_{m_i},\cdots,y^*_{m_k},\cdots) \le f(\cdots,y^*_{m_1},\cdots,y_{m_i},\cdots,y^*_{m_k},\cdots)\} = 1 \quad (2.7)$$

Combining Eqs. (2.6) and (2.7), we have:

$P\{y^*_{m_i} = x^*_{m_i}\}=\prod_{S,(x_s \notin S_m)}(1 - q_{m_i s})$,

This completes the proof.■

Theorem 1 can be explained intuitively as follows. Given a subset S_m of the variable set S, $y^*_{m_1}, y^*_{m_2}, y^*_{m_k}$ are local optimal values of variables $x_{m_1}, x_{m_2} \cdots, x_{m_k}$. For each variable x_{m_i}, the lower degree it is affected by variables outside S_m, the higher probability $y^*_{m_i}$ equals to the global optimal value $x^*_{m_i}$. An extreme case is that in a separable problem, for each pair of variables x_i and x_j, $q_{ij} = 0$. If y^*_i is a local optimal value of variable x_i, then $P\{y^*_i = x^*_i\} = 1$.

Theorem 1 implies that to optimize variable x_i, the variables that highly affected x_i should be grouped into one subset S_{m_i} so that the probability $P\{y^*_{m_i} = x^*_{m_i}\}$ is maximized. The CPSO-SL has been designed according to this principle. It firstly quantifies the degree of interdependence d_{ij} by the statistical variable interdependence learning model. The obtained degree of interdependence d_{ij} is a statistical estimator of probability q_{ij}, i.e., $d_{ij} = \hat{q}_{ij}$. Following this, it partitions an n dimensional problem into n overlapping sub-problems, where each contains variables affecting the sub-problem core with degree higher than a predefined threshold r. However, it is possible that the probability $P\{y^*_i = x^*_i\}$ decreases as the problem dimension increases. The CPSP-SL can still be an effective approach for such a problem, since it performs the optimization task by iteratively running cooperative PSOs. The following theorem shows the feasibility of the CPSO-SL framework.

Theorem 2. Given an optimization problem $f(\vec{x})$ with global optimum solution $\vec{x}^* = \{x^*_1, x^*_2, \cdots, x^*_n\}$, let $\vec{z}^*(t) = (z^*_1(t), z^*_2(t), \cdots, z^*_n(t))$ be the global solution of the CPSO-SL at the t-th iteration, let $P_i(t) = P\{z_i(t)^* = x^*_i\}$, if the local optimum solutions of the sub-problems can always be found by their corresponding PSOs, then:

1. At the 1st iteration, $t = 1$:

$$P_i(t) = \prod_{j,(x_j \notin S_i)} (1 - q_{ij})$$

2. At the t-th iteration, $t > 1$:

$$P_i(t) = \prod_{j,(x_j \notin S_i)} [1 - q_{ij} + q_{ij} P_j(t-1)]$$

Proof.

1. At the 1st iteration, in the i-th sub-problem, $y_{i_1}^*(1), y_{i_2}^*(1), \cdots, y_{i_k}^*(1)$ are local optimal values of variables $x_{i_1}, x_{i_2}, \cdots, x_{i_k}$ obtained by PSO, according to Theorem 1, we have:

$$P\{y_i^*(1) = x_i^*\} = \prod_{j,(x_j \notin S_i)} (1 - q_{ij}), \tag{2.8}$$

according to Step 5 of CPSO-SL, $y_i^*(1) = z_i^*(1)$, then:

$$P_i(1) = \prod_{j,(x_j \notin S_i)} (1 - q_{ij}), \tag{2.9}$$

2. At the t-th iteration, in the i-th sub-problem:

$$P\{f(\cdots, x_i^*, \cdots, x_j^*, \cdots) \le f(\cdots, x_i, \cdots, x_j^*, \cdots)\} = 1 \tag{2.10}$$

the global solution of CPSO-SL is $\vec{c} = (z_1^*(t-1), \cdots, z_n^*(t-1))$, suppose $x_j \notin S_i$, turning x_j^* to $z_j^*(t-1)$, then the conditional probability satisfies that:

$$P\{f(\cdots, x_i^*, \cdots, z_j^*(t-1), \cdots) \le f(\cdots, x_i, \cdots, z_j^*(t-1), \cdots) | z_j^*(t-1) = x_j^*\} = 1 \tag{2.11}$$

and

$$P\{f(\cdots, x_i^*, \cdots, z_j^*(t-1), \cdots) \le f(\cdots, x_i, \cdots, z_j^*(t-1), \cdots) | z_j^*(t-1) \neq x_j^*\} = 1 - q_{ij}. \tag{2.12}$$

Combining Eqs. (2.11) and (2.12), we have:

$$\begin{aligned} &P\{f(\cdots, x_i^*, \cdots, z_j^*(t-1), \cdots) \le f(\cdots, x_i, \cdots, z_j^*(t-1), \cdots)\} \\ &= P_j(t-1) + (1 - q_{ij})(1 - P_j(t-1)) = 1 - q_{ij} + q_{ij} P_j(t-1). \end{aligned} \tag{2.13}$$

This process is repeated until all the variables $x_j, (x_j \notin S_i)$ are exhausted, then:

$$P\{f(\cdots, x_{i_1}^*, \cdots, x_i^*, \cdots, x_{i_k}^*, \cdots) \le f(\cdots, x_{i_1}^*, \cdots, x_i, \cdots, x_{i_k}^*, \cdots)\} = \prod_{j,(x_j \notin S_i)} [1 - q_{ij} + q_{ij} P_j(t-1)] \tag{2.14}$$

3. Because $y_{i_1}^*(t), y_{i_2}^*(t), \cdots, y_{i_k}^*(t)$ are local optimal values of variables $x_{i_1}, x_{i_2}, \cdots, x_{i_k}$ obtained by PSO, we have:

$$f(\cdots, y_{i_1}^*(t), \cdots, y_i^*(t), \cdots, x_{i_k}^*(t), \cdots) \le f(\cdots, y_{i_1}^*(t), \cdots, y_i(t), \cdots, y_{i_k}^*(t), \cdots) \tag{2.15}$$

Combining Eqs. 2.14 and 2.15, we have;

$$P\{y_i^*(t) = x_i^*\} = \prod_{j,(x_j \notin S_i)} [1 - q_{ij} + q_{ij} P_j(t-1)]. \tag{2.16}$$

Because the local optimum solutions of the sub-problems can always be found by their corresponding PSOs, according to Step 5 of CPSO-SL, $y_i^*(t) = z_i^*(t)$, it follows that:

$$P_i(t) = \prod_{j,(x_j \notin S_i)} [1 - q_{ij} + q_{ij} P_j(t-1)]. \tag{2.17}$$

This completes the proof. ■

From Theorem 2, it can be seen that progressions $\{P_1(t)\}$, $\{P_2(t)\}$,…, $\{P_n(t)\}$have Markov property, and will eventually converge to stable values. The probability for the CPSO-SL locating the global optimum solution increases as the iteration of algorithm increases. The prerequisite is that the local optimum solutions of the sub-problems can always be found by their corresponding PSOs.

2.4.3 Selection of threshold value r

The statistical decomposition method partitions the problem into overlapping sub-problems. Different threshold values yield different decomposition results on the same problem, and thus yield different optimization results. As can be seen from Theorem 2, progressions $\{P_1(t)\}$, $\{P_2(t)\}$, …, $\{P_n(t)\}$ are nondecreasing, and will converge to stable values. To illustrate the effect of parameter r on these values, in this subsection, six nonseparable optimization problems are simulated. They are simulated as follows: the problem dimensions are set to 20, 50, 100, 200, 500, 1000, respectively. Within each problem, the probability q_{ij} for each pair of variables is simulated by random numbers in the range [0, 1]. Fig.2.5 plots the average stable values of progressions against different threshold values. The abscissa is the r values and the vertical axis is the average value of the stable values of progressions $\{P_1(t)\}$, $\{P_2(t)\}$, …, $\{P_n(t)\}$, whose values are computed by the iterative functions in Theorem 2. The stable

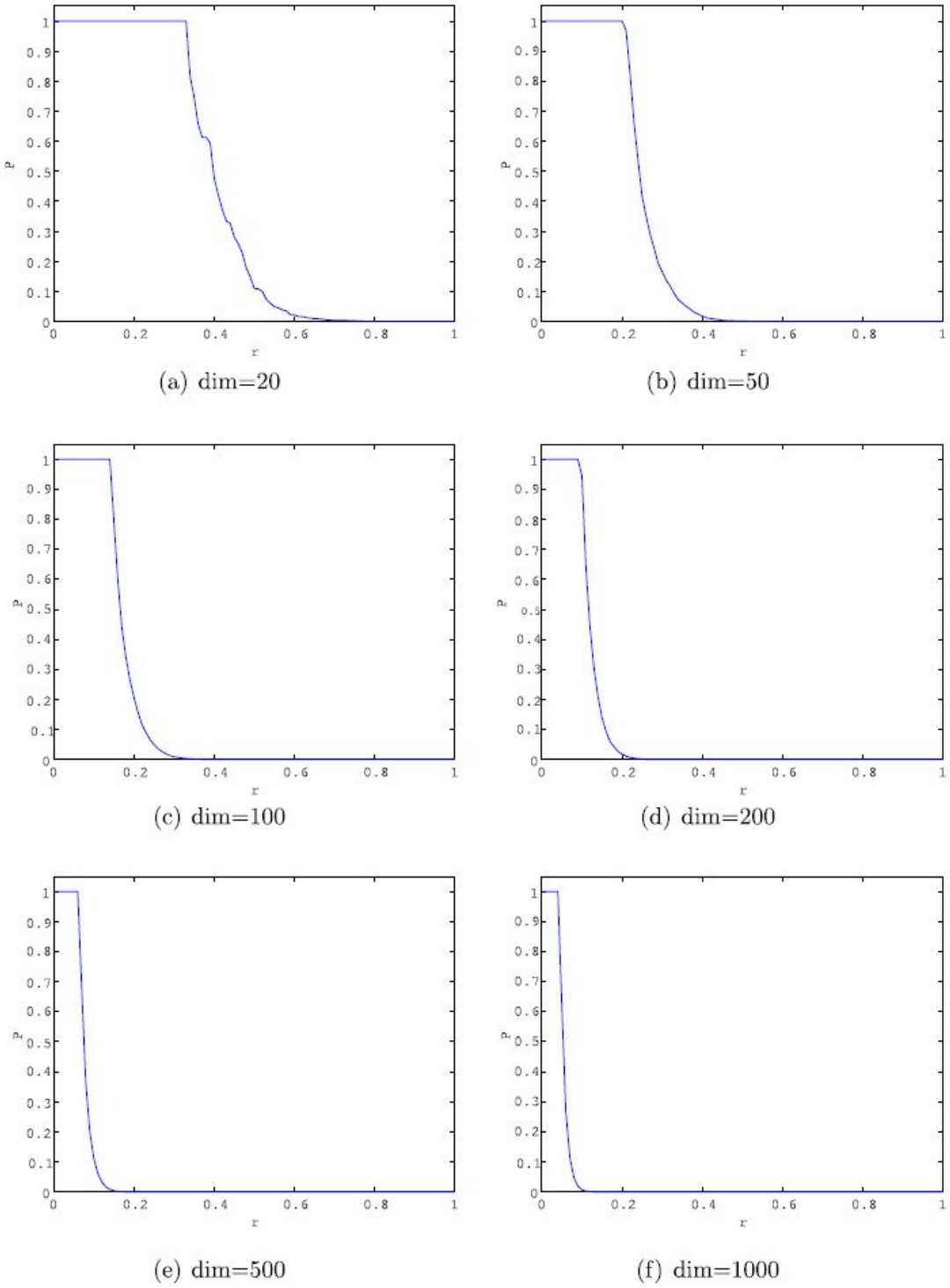


Fig. 2.5 Probabilities for CPSO-SL obtaining global optimum solutions of six simulated problems with different threshold r.

values are obtained when $|P_i(t) - P_i(t-1)| < \varepsilon$. From Fig.2.5, we can observe that r can influence the probability for CPSO-SL obtaining the global optimal solution. When r is

less than a turning point, the algorithm can converge to the global optimum solution with probability one. When r is larger than the turning point, the probability will decreases as r increases.

The parameter r makes a tradeoff between the single PSOs and probability for CPSO-SL obtaining the global optimum solution. With a lower r value, more variables are allocated to each sub-problem, making it more difficult for the single PSOs to obtain the local optimal solutions. On the other hand, with a higher r value, the probability for CPSO-SL obtaining the global optimum solution may decrease. In our experiment, we propose an iterative method to select the parameter r value, which is described as follows. Firstly, set $r = 0$, and then calculate average value P_{avg} of the stable points of progressions $\{P_1(t)\}$, $\{P_2(t)\}$, $\cdots$, $\{P_n(t)\}$ by the iterative function in Theorem 2. If $P_{avg} = 1$, then set $r = r + d$, compute P_{avg} again. The r value is set to the last value that makes P_{avg}=1. In the experiment, the parameter d is taken as 0.02.

2.4.4 Potential search space

Consider an optimization problem with solution space $E = [b_{l1}, b_{u1}] \times [b_{l2}, b_{u2}] \times \cdots \times [b_{ln}, b_{un}]$, the potential search space is $\prod_{i=1}^{n} b_{ui} - b_{li}$. The volume of the solution space increases exponentially as the number of dimension increases. In the proposed algorithm, the n-dimensional problem is partitioned into several lower dimensional sub-problems, in the i-th sub-problem, the potential search space is $\sum_{j \in S_i}(b_{uj} - b_{lj})$. Then the total solution space is $\sum_{i=1}^{n} \prod_{j \in S_i}(b_{uj} - b_{lj})$. The above analysis indicates that the proposed algorithm reduces the volume of the solution from $\prod_{i=1}^{n} b_{ui} - b_{li}$ to $\sum_{i=1}^{n} \prod_{j \in S_i}(b_{uj} - b_{lj})$.

2.5 Experimental studies

2.5.1 Test functions

In this section, we conduct numerical experiments to test the performance of the proposed CPSO-SL. In the experiment, we select 10 classical benchmark functions [19], 10 rotated classical benchmark functions, and 10 CEC2005 benchmark functions [40]. All functions are tested on 500 and 1000 dimensions. Table 2.1 lists the classical

Table 2.1

Classical benchmark functions (n = problem dimension, S = solution space, f_{min} = minimum function value).

Description	Test function	S	f_{min}
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100,100]^n$	0
Schwefel	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10,10]^n$	0
Step	$f_3(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100,100]^n$	0
Generalized Rastrigin	$f_4(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12,5.12]$	0
Generalized Penalized[a]	$f_5(x) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_{n-1})^2\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	$[-50,50]^n$	0
Generalized Penalized[a]	$f_6(x) = \frac{\pi}{10}\{10\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + 10\sin^2(3\pi x_{i+1})] + (x_n - 1)^2 (1 + \sin^2(2\pi x_n))\} + \sum_{i=1}^{n} u(x_i, 5,100,4)$	$[-50,50]^n$	0
Quardirc	$f_7(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	$[-100,100]^n$	0
Rosenbrock	$f_8(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30,30]^n$	0
Ackley	$f_9(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32,32]^n$	0
Generalized Griewank	$f_{10}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600,600]^n$	0

[a]Detailed descriptions of these functions are given in the appendix.

benchmarks functions and their key properties. Some of these functions are unimodal functions, which are relatively easy to optimize, and some are multimodal functions, which have many local optima and are relatively hard to optimize. In order to introduce variable interactions, thereby making them non-separable, the selected classical benchmark functions are further rotated. To rotate a function, firstly, an orthogonal matrix M is created. The input vector $\vec{x}$ is left multiplied by matrix M to produce rotated vector $\vec{y} = M * \vec{x}$. Then, the vector $\vec{y}$ is used as an input to calculate the

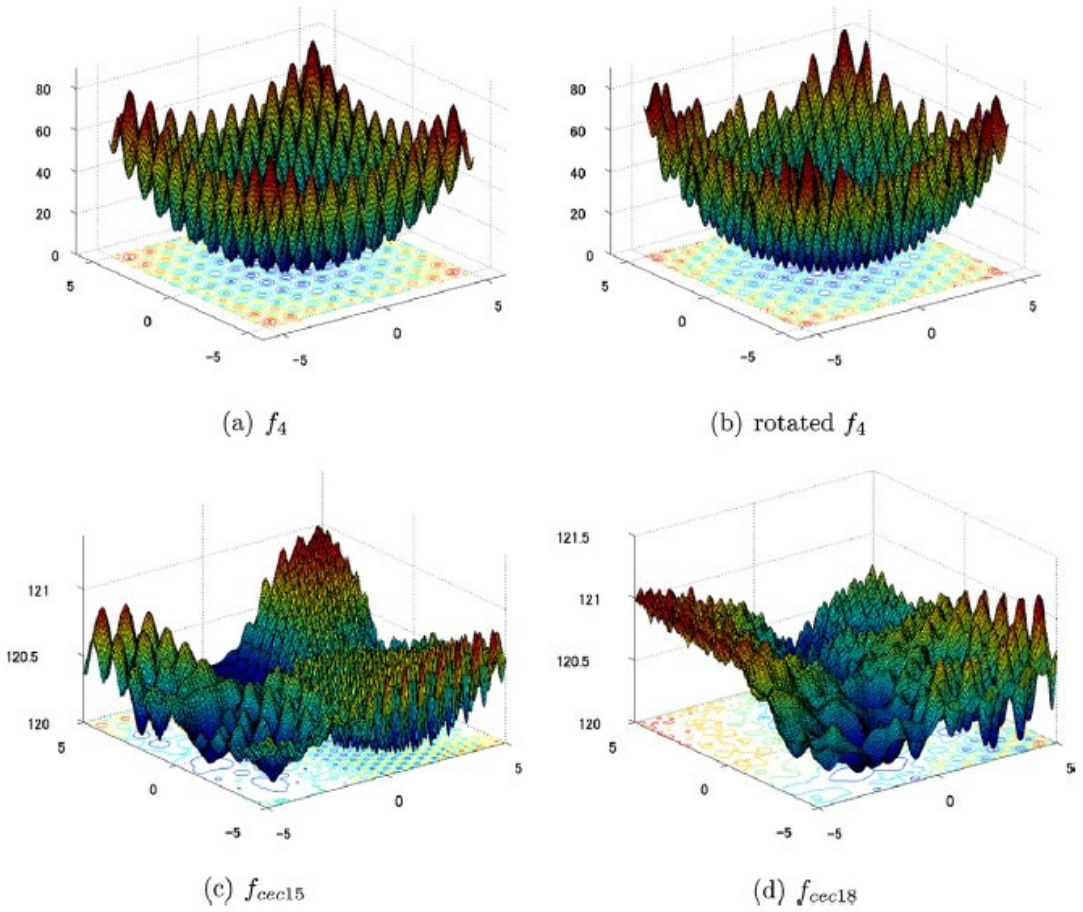


Fig. 2.6. Surface landscapes of f_4, rotated f_4, f_{cec15} and f_{cec18}.

objective function. In this thesis, the orthogonal matrix is created by Salomon's method [41]. The CEC2005 Special Session has 25 benchmark functions. Many of these functions are shifted, rotated, expanded, and combined variants of the classical benchmark functions. The properties and the formulas of these functions can be seen in [40]. We use 10 representative functions in our experiment, including two unimodal functions (f_{cec01}, f_{cec02}), four basic multimodal functions (f_{cec06}, f_{cec07}, f_{cec08}, f_{cec09}), one expanded function (f_{cec13}), and three composition functions (f_{cec15}, f_{cec18}, f_{cec21}). Among the ten selected functions, three are separable (f_{cec01}, f_{cec02}, f_{cec09}) and seven are non-separable according to Definition 2. As an example, Fig.2.6 shows the two dimensional surface landscapes of f_4, rotated f_4, f_{cec15}, and f_{cec18}.

2.5.2 Experimental settings

Parameter settings are crucial for the performance of the CPSO-SL. In the variable interdependence learning stage, the total number of statistical samples N (Line 3 of Fig.2.2) is set to 50. In the decomposition stage, the threshold value r for the decomposition method is set according to the method described in Section 4.3. The next decision is on the parameters used in the main framework of CPSO-SL. Within each separate PSO, the parameters are taken as $w = 0.7298$, $c_1 = 1.4961$ and $c_2 = 1.4961$ c1 = 1.49618, which are selected through empirical study in [42]; the population size is taken as 50.

We use the number of function evaluations (FEs) to measure the computational efforts of the proposed algorithm. In the variable interdependence learning stage, the number of FEs is fixed. For an n dimensional problem, the number of FEs is $n \times n \times 50 \times 4$, where 50 is the number of statistical samples and 4 is the number of FEs within each evaluation. Thus, the computational efforts incurred by the variable interdependence learning are $5.0e+07$, $2.0e+08$ FEs for 500 and 1000 dimensional problems, respectively. In the optimization stage, the algorithm stops when a maximum number of FEs is reached. In our experiment, the maximum numbers of FEs are set to $1.0e+08$, 2.0e+08 for 500 and 1000 dimensional problems, respectively. Thus, the total numbers of FEs are $1.5e+08$, $4.0e+08$ for 500 and 1000 dimensional problems, respectively.

In order to test the stability of the CPSO-SL, the experiment on each benchmark function is repeated 30 times.

2.5.3 Existing algorithms for comparison

For the purpose of comparison, we select the following three algorithms which have been applied for all or some of the selected test functions.

1. Fast evolutionary programming with cooperative co-evolution (FEPCC) [25]: FEPCC divides the system into many modules, and then repeatedly evolves each module separately and combines them to form the whole system.
2. Self adaptive neighborhood search differential evolution with dynamic grouping

cooperative co-evolution (DECC-G) [31]: in DECC-G, a dynamic grouping and adaptive weighting strategy is proposed, and the proposed strategy is integrated to differential cooperative co-evolution algorithm for optimization.

3. Cooperatively Coevolving Particle Swarm Optimization (CCPSO) [28]: CCPSO adopts a similar grouping and weighting strategy of DECC-G, and integrates it into PSO algorithm.

All of the above three algorithms are cooperative evolutionary algorithms. They adopt the divide-and-conquer strategy, i.e., decompose the problem into several sub-problems and optimize the sub-problems cooperatively. These algorithms have been shown to be successful and can find suitable solutions, especially for large scale optimization problems. Hence, a comparison with these algorithms will demonstrate the performance of the proposed CPSO-SL, and will show whether it is better or worse than other algorithms.

To test whether the statistical decomposition technique can improve the CPSO, the CPSO that directly applies Potter's technique [24] is also included for comparison. In the CPSO, only one variable is optimized at a time. That is, each subcomponent consists of a single variable. The parameters of CPSO are taken as $w = 0.7298$, $c_1 = 1.4961$, $c_2 = 1.4968$, population size $pop = 50$, and maximum numbers of FEs are taken as $1.0e+08$, $2.0e+08$ for 500 and 1000 dimensional problems, respectively.

2.5.4 Simulated results and discussions

2.5.4.1 Results for classical benchmark functions

This subsection aims to show the results of CPSO-SL for classical benchmark functions. Table 2.2 presents the results from 30 independent runs of CPSO-SL, CPSO and the results of studies using FEPCC, DECC-G and CCPSO. "Mean" represents the average value of the results obtained using the algorithm with respect to the optimum value of the benchmark functions. "Var" represents the standard deviation of the results. "FEs-CP" represents the computational effort comparison between the CPSO-SL and the compared algorithm, whose value is obtained by dividing the number of FEs incurred by the CPSOSL with the number of FEs incurred by the compared algorithm.

Table 2.2

Results of CPSO-SL and Other Algorithms for Classical Benchmark Functions (to be continued).

Func	Dim	CPSO-SL		CPSO			FEPCC		
		Mean	Var	FEs-CP	Mean	Var	FEs-CP	Mean	Var
f_1	500	4.84e-30	4.68e-30	1.5	4.95e-30§	4.06e-30	60	4.90e-08†	1.20e-07
	1000	4.15e-29	1.29e-29	2	3.98e-29§	2.70e-29	80	5.40e-08†	2.80e-08
f_2	500	3.14e-18	9.68e-19	1.5	3.25e-18§	1.02e-18	60	1.30e-03†	3.00e-03
	1000	7.08e-18	1.82e-18	2	7.65e-18§	1.37e-18	80	2.60e-03†	3.20e-03
f_3	500	0.00e-00	0.00e-00	1.5	0.00e-00§	0.00e-00	60	0.00e-00§	0.00e-00
	1000	0.00e-00	0.00e-00	2	0.00e-00§	0.00e-00	80	0.00e-00§	0.00e-00
f_4	500	0.00e-00	0.00e-00	1.5	0.00e-00§	0.00e-00	60	1.43e-01†	2.80e-01
	1000	0.00e-00	0.00e-00	2	0.00e-00§	0.00e-00	80	3.13e-01†	4.00e-01
f_5	500	6.49e-25	2.24e-25	1.5	6.89e-25§	2.58e-25	-	-	-
	1000	6.02e-25	2.43e-25	2	7.13e-25†	2.34e-25	-	-	-
f_6	500	1.00e-22	1.36e-22	1.5	1.52e-22†	2.32e-22	-	-	-
	1000	5.07e-22	2.73e-22	2	4.62e-22‡	8.08e-22	-	-	-
f_7	500	1.67e-27	2.09e-27	1.5	1.29e-27‡	1.49e-27	-	-	-
	1000	5.08e-27	2.45e-27	2	6.06e-27†	4.90e-27	-	-	-
f_8	500	2.79e-05	2.01e-05	1.5	1.28e01†	1.60e01	-	-	-
	1000	1.47e-01	9.30e-02	2	1.03e03†	2.06e03	-	-	-
f_9	500	8.46e-16	1.45e-16	1.5	8.06e-16§	1.26e-16	60	5.70e-04†	3.90e-05
	1000	8.03e-16	1.60e-16	2	9.28e-16†	1.86e-16	80	9.50e-04†	3.40e-05
f_{10}	500	7.40e-17	9.06e-17	1.5	9.25e-17†	8.36e-17	60	2.90e-02†	8.50e-02
	1000	2.59e-16	6.40e-17	2	1.94e-16‡	1.06e-16	80	2.50e-02†	1.14e-01

To determine the statistical differences between the results obtained by the CPSO-SL and the compared algorithms, the t-test results are also provided. '†' indicates that the

Table 2.2

Results of CPSO-SL and Other Algorithms for Classical Benchmark Functions (continued).

Func	Dim	CPSO-SL		DECC-G		CCPSO		
		Mean	Var	FEs-CP	Mean	FEs-CP	Mean	Var
f_1	500	4.84e-30	4.68e-30	60	6.33e-27†	-	-	-
	1000	4.15e-29	1.29e-29	80	2.17e-25†	-	-	-
f_2	500	3.14e-18	9.68e-19	60	5.95e-15†	-	-	-
	1000	7.08e-18	1.82e-18	80	5.37e-14†	-	-	-
f_3	500	0.00e-00	0.00e-00	60	0.00e-00§	-	-	-
	1000	0.00e-00	0.00e-00	80	0.00e-00§	-	-	-
f_4	500	0.00e-00	0.00e-00	60	0.00e-00	60	0.00e-00	0.00e-00
	1000	0.00e-00	0.00e-00	80	3.55e-16†	80	3.63e-03	1.79e-02
f_5	500	6.49e-25	2.24e-25	60	4.29e-21†	-	-	-
	1000	6.02e-25	2.43e-25	80	6.89e-25†	-	-	-
f_6	500	1.00e-22	1.36e-22	60	5.34e-18†	-	-	-
	1000	5.07e-22	2.73e-22	80	2.55e-21†	-	-	-
f_7	500	1.67e-27	2.09e-27	60	6.17e-25†	60	5.35e-00	2.41e01
	1000	5.08e-27	2.45e-27	80	3.17e-23†	80	3.49e02	1.56e03
f_8	500	2.79e-05	2.01e-05	60	4.92e02†	60	4.15e02	1.85e02
	1000	1.47e-01	9.30e-02	80	9.87e02†	80	1.16e03	8.52e02
f_9	500	8.46e-16	1.45e-16	60	9.13e-14†	60	1.34e-09	6.70e-09
	1000	8.03e-16	1.60e-16	80	2.22e-13†	80	1.91e-01	2.60e-01
f_{10}	500	7.40e-17	9.06e-17	60	4.40e-16†	60	3.55e-17	5.29e-17
	1000	2.59e-16	6.40e-17	80	1.01e-15†	80	2.80e-16	4.79e-16

results of the two algorithms are statistically different with 95% certainty and that CPSO-SL is better. '‡' indicates that the results of the two algorithms are statistically different with 95% certainty and that the compared algorithm is better. '§' indicates

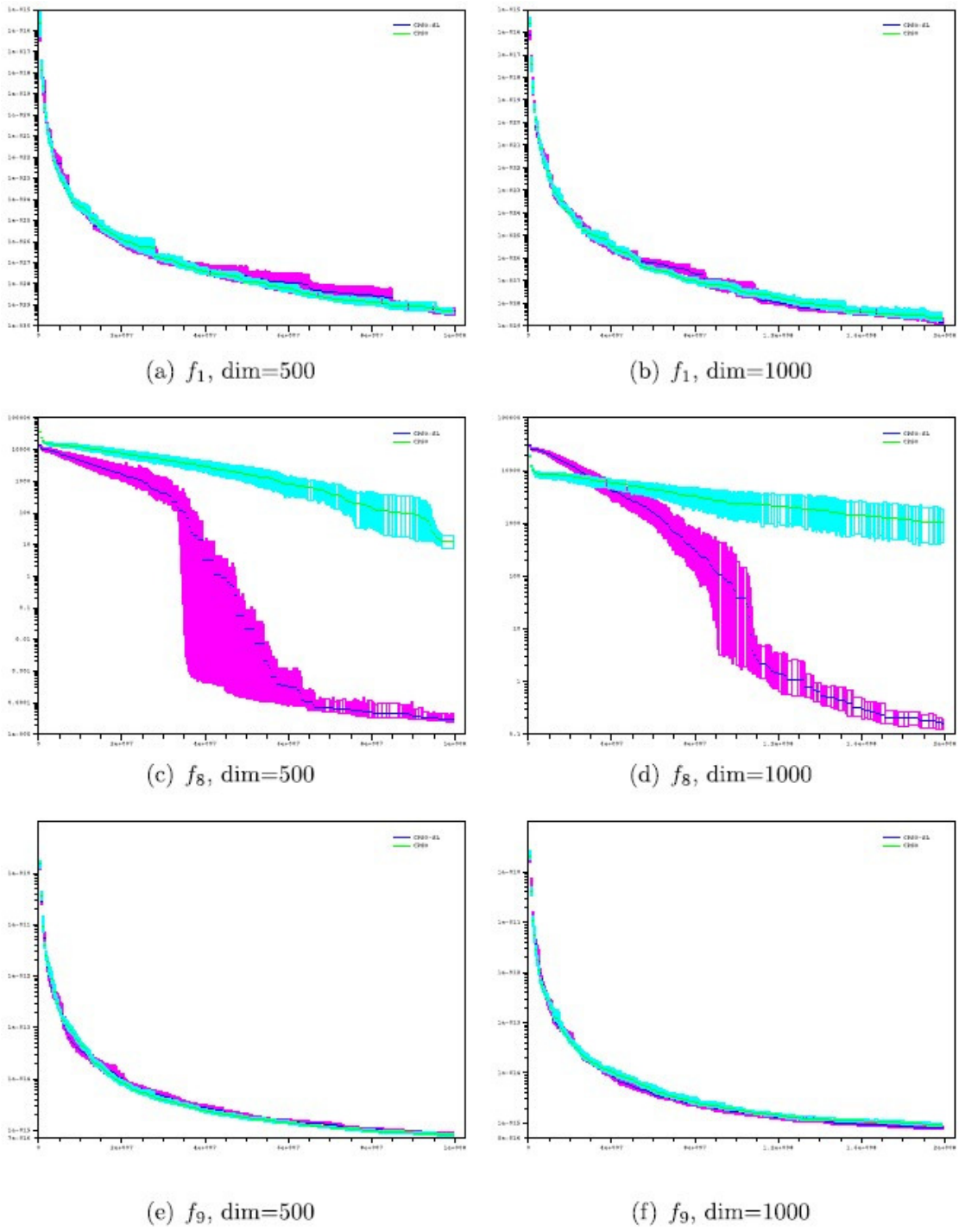


Fig. 2.7. Bands of function values of CPSO-SL and CPSO with 95% confidence interval for solutions of functions f_1, f_8 and f_9. The results were obtained from 30 independent runs of the algorithms. The abscissa is the number of FEs and the vertical axis is the objective function value.

that the results are not statistically different. For the algorithm DECC-G, because no variances have been reported in the literature, the results are compared with the mean values. '†' indicates that CPSO-SL is better. '‡' indicates that DECC-G is better. '§' indicates that the two algorithms are the same. For the algorithms compared, not all results of functions are reported, so for those functions without reported results, their corresponding comparison values are marked with '-'. Fig.2.7 is a plot of the bands of function values of CPSO-SL and CPSO with 95% confidence interval for solutions of functions f_1, f_8 and f_9. The abscissa is the number of function evaluations and the vertical axis is the objective function value. The confidence intervals of the results obtained by the two algorithms are computed using the bootstrap based method described in [43].

From Table 2.2, it can be seen that CPSO-SL can obtain accuracy of 0.1 for all functions. It also can be seen that CPSO-SL and CPSO can obtain almost the same results for functions $f_1 - f_7$, f_9 and f_{10}. For the Rosenbrock function f_8, CPSO-SL obtains accuracy of 0.1, while CPSO obtains accuracy of 10. The computational efforts of CPSO-SL are 1.5 and 2 times as intensive as those of CPSO for 500 and 1000 dimensional problems, respectively. From Fig.2.7, it can be seen that the curves of the two algorithms have the same shape for functions f_1 and f_9. For function f_8, both algorithms improve the solutions steadily; however, CPSO-SL generally obtains better results than CPSO. Compared with FEPCC, for all the compared functions, CPSO-SL obtains accuracy of 10^{-16}, while FEPCC obtains accuracy of 0.1. Among the compared algorithms, DECC-G yields the best results. CPSO-SL obtains better results than DECC-G for 17 out of the 20 functions. It is worth noting that CPSO-SL can obtain best mean values of 2.79×10^{-5} and 1.47×10^{-1} for the 500 and 1000 dimensional Rosenbrock function f_8, respectively, while DECC-G can obtain best mean values of 4.92×10^{2} and 9.87×10^{2}, respectively. Compared with CCPSO, CPSO-SL obtains accuracy of 10^{-27} and 10^{-1} for functions f_7 and f_8, respectively, while CCPSO obtains accuracy of 10^{2} and 10^{3}, respectively. The computational efforts of CPSO-SL are 60 and 80 times as intensive as those of FEPCC, DECC-G and CCPSO for 500 and 1000

Table 2.3

Results of CPSO-SL and Other Algorithms for Rotated Classical Benchmark Functions.

Func	Dim	CPSO-SL		CPSO			CCPSO		
		Mean	Var	FEs-CP	Mean	Var	FEs-CP	Mean	Var
f_1	500	1.11e-29	2.05e-29	1.5	8.52e-30‡	8.62e-30	-	-	-
	1000	7.58e-30	5.91e-30	2	3.51e-29‡	7.14e-29	-	-	-
f_2	500	1.12e-06	1.68e-06	1.5	8.90e01†	8.77e01	-	-	-
	1000	4.34e-06	8.33e-05	2	3.50e02†	2.02e02	-	-	-
f_3	500	0.00e-00	0.00e-00	1.5	0.00e-00§	0.00e-00	-	-	-
	1000	0.00e-00	0.00e-00	2	0.00e-00§	0.00e-00	-	-	-
f_4	500	4.03e-01	3.46e-00	1.5	2.18e03†	4.29e02	60	0.00e-00‡	0.00e-00
	1000	6.85e-01	1.74e-01	2	5.16e03†	1.28e03	80	5.69e-01‡	2.65e-00
f_5	500	9.41e-15	1.56e-14	1.5	1.09e-00†	3.77e-00	-	-	-
	1000	2.24e-15	1.22e-15	2	1.40e01†	1.67e01	-	-	-
f_6	500	5.87e-12	1.31e-11	1.5	5.83e-00†	6.73e-00	-	-	-
	1000	1.05e-09	1.99e-09	2	6.85e01†	1.74e01	-	-	-
f_7	500	5.73e-06	1.10e-05	1.5	2.18e03†	4.07e03	60	7.66e01†	1.57e02
	1000	4.83e-05	1.07e-04	2	1.40e04†	1.67e04	80	3.55e03†	4.80e03
f_8	500	1.78e-02	2.69e-02	1.5	1.32e03†	1.40e03	60	4.99e02†	6.11e02
	1000	5.06e-02	7.03e-02	2	6.02e03†	6.37e03	80	9.98e02†	6.14e-02
f_9	500	2.53e-09	8.47e-10	1.5	2.54e-00†	6.13e-01	60	8.69e-15‡	3.51e-15
	1000	3.16e-09	1.22e-09	2	2.83e-00†	1.01e-00	80	1.49e-10‡	5.46e-10
f_{10}	500	1.22e-15	7.65e-16	1.5	6.36e-01†	8.71e-01	60	4.44e-18	2.22e-17
	1000	1.63e-15	1.28e-15	2	5.37e-01†	1.20e-00	80	7.97e-13†	3.64e-12

dimensional problems, respectively.

2.5.4.2 Results for rotated classical benchmark functions

This subsection aims to show the results of CPSO-SL for rotated classical benchmark functions. The experiments that were conducted on functions $f_1 - f_{10}$ are

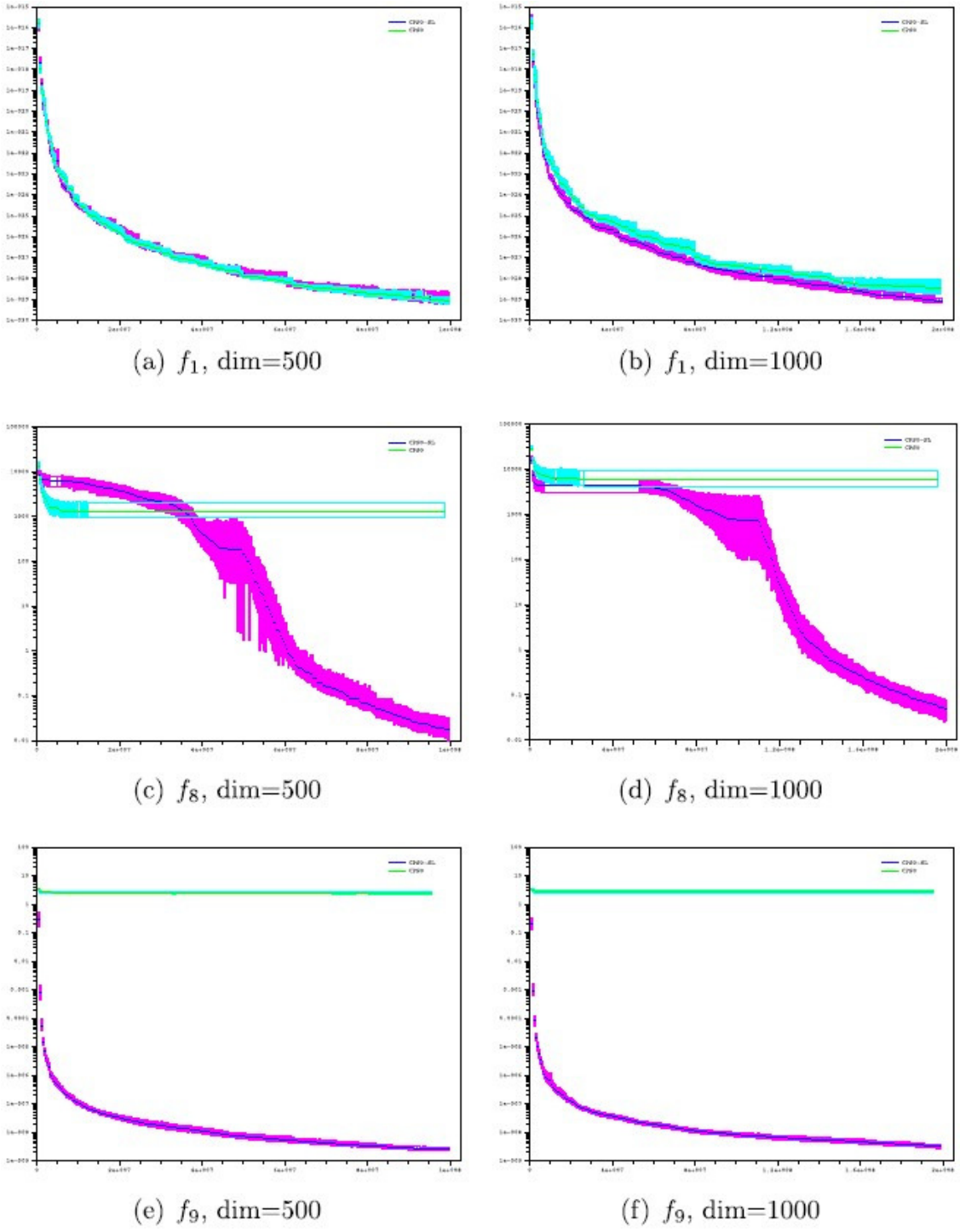


Fig. 2.8. Bands of function values of CPSO-SL and CPSO with 95% confidence interval for solutions of rotated functions f_1, f_8 and f_9. The results were obtained from 30 independent runs of the algorithms. The abscissa is the number of FEs and the vertical axis is the objective function value.

Table 2.4

Results of CPSO-SL and other algorithms for CEC2005 benchmark functions.

Func	Dim	CPSO-SL		CPSO			DECC-G	
		Mean	Var	FEs-CP	Mean	Var	FEs-CP	Mean
f_{cec01}	500	3.67e-12	6.92e-13	1.5	3.22e-12 §	8.52e-13	60	3.71e-13 †
	1000	9.28e-12	2.29e-12	2	7.41e-12 ‡	2.96e-12	80	6.84e-13 ‡
f_{cec02}	500	2.95e-09	8.32e-10	1.5	6.44e-09 †	2.98e-09	-	-
	1000	5.45e-08	3.15e-08	2	5.54e-08 §	5.79e-08	-	-
f_{cec06}	500	5.28e-02	8.07e-02	1.5	2.26e02 †	1.87e02	60	1.56e03 †
	1000	3.30e-02	8.61e-02	2	2.96e02 †	4.43e03	80	2.22e03 †
f_{cec07}	500	2.96e-13	1.28e-13	1.5	1.38e-01 †	3.08e-01	-	-
	1000	1.06e-12	4.99e-13	2	1.57e-00 †	2.35e-00	-	-
f_{cec08}	500	7.31e-01	8.60e-01	1.5	1.95e01 †	5.75e-00	60	2.16e01 †
	1000	3.42e-00	1.13e-00	2	2.14e01 †	3.11e-01	80	2.16e01 †
f_{cec09}	500	3.64e-12	1.12e-12	1.5	2.94e-12 ‡	6.71e-13	60	4.50e02 †
	1000	6.88e-12	1.55e-12	2	6.94e-12 §	1.82e-12	80	6.32e02 †
f_{cec13}	500	1.42e-12	8.23e-13	1.5	4.33e02 †	1.43e02	60	2.09e02 †
	1000	1.23e-10	5.36e-10	2	9.43e02 †	2.66e02	80	3.56e02 †
f_{cec15}	500	5.97e-04	1.10e-03	1.5	6.52e-01 †	1.24e-01	-	-
	1000	1.43e-02	1.72e-02	2	6.70e-01 †	1.47e-01	-	-
f_{cec18}	500	5.97e-04	1.10e-03	1.5	7.24e-01 †	7.73e-02	-	-
	1000	4.22e-02	6.01e-01	2	7.78e-01 †	8.35e-02	-	-
f_{cec21}	500	3.06e-02	6.83e-02	1.5	1.18e-01 †	8.14e-02	-	-
	1000	5.03e-02	7.22e-02	2	1.92e-01 †	6.53e-02	-	-

repeated on rotated functions $f_1 - f_{10}$ Table 2.3 presents the results from 30 independent runs of CPSO-SL, CPSO and the results of studies using CCPSO. The item abbreviations of Table 2.3 are the same as those in Table 2.2. For the algorithms FEPCC

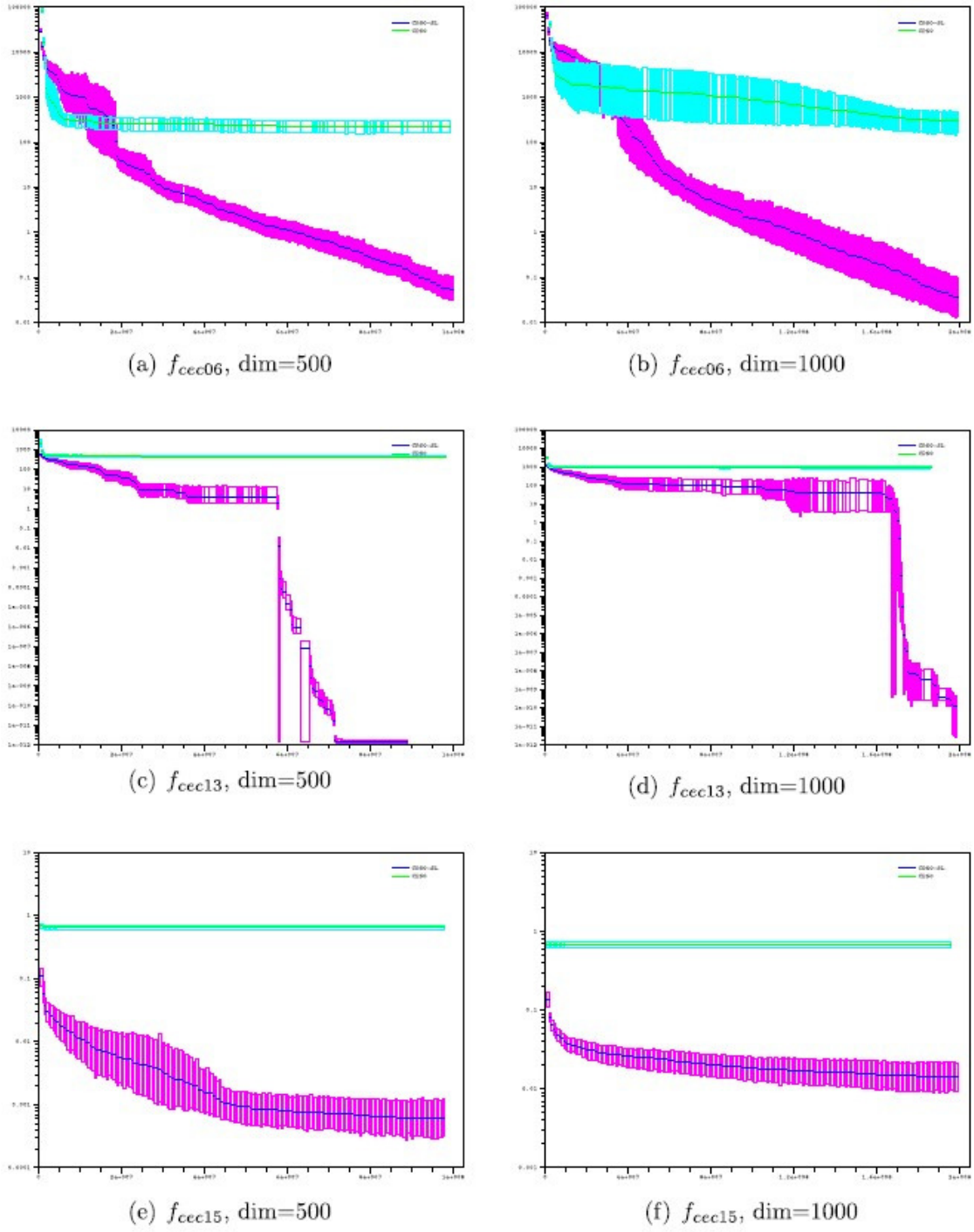


Fig. 2.9. Bands of function values of CPSO-SL and CPSO with 95% confidence interval for solutions of functions f_{cec06}, f_{cec13} and f_{cec15}. The results were obtained from 30 independent runs of the algorithms. The abscissa is the number of FEs and the vertical axis is the objective function value.

and DECC-G, the results are not reported in the literatures, so they are not included in Table 2.3. Fig.2.8 is a plot of the bands of function values of CPSO-SL and CPSO with 95% confidence interval for solutions of rotated functions f_1, f_8 and f_9.

From Table 2.3, it can be seen that CPSO-SL can obtain better results than CPSO, except for the cases of f_1 and f_3. For rotated function f_1, the results seem to be the same for both algorithms. This is because the rotated f_1 is still a separable function according to Definition 2. And the decomposition method for CPSO-SL partitions the solution into single variables, which makes the two algorithms identical. For rotated function f_3, both algorithms can obtain the optimum value. From Fig.2.8, it can be seen that for rotated function f_1, no significant difference can be found between CPSO-SL and CPSO. For rotated functions f_8 and f_9, CPSO-SL can approximate the optimum solution steadily during the search process; on the other hand, CPSO appeared to become trapped into pseudo optima. The computational efforts of CPSO-SL are 1.5 and 2 times as intensive as those of CPSO for 500 and 1000 dimensional problems, respectively. Compared with CCPSO, it can be seen that CPSO-SL can obtain better results for 5 out of the 10 compared functions. For rotated function f_4, CPSO-SL can obtain accuracy of 0.1, while CCPSO can obtain the optimum value for the 500 dimensional problem, and obtain accuracy of 0.1 for the 1000 dimensional problem. Although the results of CPSO-SL are not as good as that of CCPSO for rotated functions f_9 and f_{10}, they are not deteriorated much since they can achieve accuracy of 10^{-9}. The computational efforts of CPSO-SL are 60 and 80 times as intensive as those of CCPSO for 500 and 1000 dimensional problems, respectively.

2.5.4.3 Results for CEC2005 benchmark functions

This subsection aims to show the results of CPSO-SL for CEC2005 benchmark functions. The experiments that were conducted on classical benchmark functions are repeated on CEC2005 benchmark functions. Table 2.4 presented the results from 30 independent runs of CPSO-SL, CPSO and the results of studies using DECC-G. The item abbreviations of Table 2.4 are the same as those in Table 2.2. For the algorithms FEPCC and CCPSO, the results are not reported in literatures, so they are not included

in Table 2.4. Fig.2.9 is a plot of the bands of function values of CPSO-SL and CPSO with 95% confidence interval for solutions of f_{cec06}, f_{cec13}, and f_{cec15}.

From Table 2.4, it can be seen that CPSO-SL can obtain better results than CPSO for all functions except for functions f_{cec01}, f_{cec02} and f_{cec09}, for which the results seem to be the same. From Fig.2.9, it can be seen that CPSO-SL can improve the solutions steadily for functions f_{cec06} and f_{cec15}. For function f_{cec13}, CPSO-SL appeared to be trapped into pseudo optimal solution and escaped with the execution of the algorithm running. On the other hand, CPSO appeared to be unable to find the optimal solution. The computational efforts of CPSO-SL are 1.5 and 2 times as intensive as those of CPSO for 500 and 1000 dimensional problems, respectively. In comparison with DECC-G, it can be seen that CPSO-SL can obtain better results for 8 out of the 10 compared functions. For function f_{cec01}, CPSO-SL obtains accuracy of 10^{-12}, while DECC-G obtains accuracy of 10^{-13}. The computational efforts of CPSO-SL are 60 and 80 times as intensive as those of DECC-G for 500 and 1000 dimensional problems, respectively.

2.5.4.4 Discussions

The comparison results of CCPSO-SL on classical benchmark functions, rotated classical benchmark functions and CEC2005 benchmark functions can be summarized and explained as follows.

1. On all the classical benchmark functions except for the Rosenbrock function f_8, CPSO-SL and the compared algorithm can approximate the optimal values. On the Rosenbrock function f_8, CPSO-SL obtains accuracy of 0.1, while the compared algorithms appeared to become trapped into pseudo optimum. On some of the rotated classical benchmark functions (f_1, f_3, f_9, f_{10}), CPSO-SL and the compared algorithms can approximate the optimal values. On some of the rotated classical benchmark functions (f_2, $f_4 - f_8$), CPSO-SL can approximate the optimal solutions, while the compared algorithms appeared to become trapped into pseudo optimum. On some of the CEC2005 benchmark functions (f_{cec01}, f_{cec02}, f_{cec09}), CPSO-SL and the compared algorithms can approximate the

optimal values. On some of the CEC2005 benchmarks functions (f_{cec06}, f_{cec07}, f_{cec08}, f_{cec13}, f_{cec15}, f_{cec18}, f_{cec21}), CPSO-SL can approximate the optimal solutions, while the compared algorithms appeared to become trapped into pseudo optimum.

2. The computational efforts of CPSO-SL are 1.5 and 2 times as intensive as those of CPSO and 60 and 80 times as intensive as those of FEPCC, DECC-G, CCPSO for 500 and 1000 dimensional problems, respectively.
3. According to Definition 2, classical benchmark functions $f_1 - f_7$, f_9, f_{10}, rotated classical benchmark functions f_1, f_3, and CEC2005 benchmark functions f_{cec01}, f_{cec02}, f_{cec09} are separable, while the reminding functions are non-separable.

The simulated results indicate that CPSO-SL may perform well on separable functions, and that it maintains its stability on non-separable functions. However, the computational efforts of CPSO-SL are more intensive than those of the compared algorithms. This outcome can be referred to as the "no free lunch theorems" for optimization [44], i.e., "any elevated performance over one class of problems is offset by performance over another class". There is a cost for CPSO-SL obtaining better results on non-separable problems, and the cost is the extra computational efforts on separable problems. The reason that we get such results is that we have no prior knowledge of the problem. In CPSO-SL, the statistical variable interdependence learning model adds an extra computational burden to the algorithm. Moreover, due to the nature of the cooperative optimization framework, the CPSO-SL requires more intensive computational efforts than the compared algorithms during the optimization stage. Therefore it can be assumed that it is better to apply CPSO-SL when the problems are non-separable and the quality of the final results is of more concerned. The experiment here serves as an illustration of the usefulness of the method and provides a guideline for researchers when designing related algorithms.

2.6. Conclusion

To tackle large scale optimization problems, one of the most common ways is to adopt the divide-and-conquer strategy. In this chapter, we aimed at proposing a

divide-and-conquer algorithm. In the empirical and application aspect, we proposed a statistical model to learn the variable interdependencies among variables. With the variable interdependencies, a decomposition method was proposed to partition the problem into sub-problems such that the variable interdependencies in different sub-problems were minimized. Then the sub-problems were optimized by a CPSO framework cooperatively by putting and taking information from the global solution vector. In the theoretical aspect, firstly, we proposed and proved a theorem that explains the execution process of the proposed algorithm. From the study, we found that the probability for a local optimum solution of a subset being global optimum values is associated with degree of interdependencies of its variables with variables outside the subset. Secondly, we proposed and proved a theorem that explains why and how the proposed algorithm works. From the theorem, we could design a method that can be used to select parameter r, which determined the final results of the decomposition method. The performance of the proposed algorithm was tested on benchmarks from different data sets. Simulated results showed that CPSO-SL could find the optimal solution for most of the selected test functions.

The main contributions of this chapter include proposing the variable interdependence learning model, the problem decomposition method, and the CPSO framework. The proposed algorithm, i.e., CPSO-SL is provided to illustrate its practicality and effectiveness for numerical optimization.

Chapter 3: A support vector and K-Means based clustering algorithm

3.1 Introduction

Data clustering is a problem of grouping a set of data into clusters so that the data in the same cluster are analogous in properties. It can be found in many fields, such as engineering, computer sciences, life and medical sciences, earth sciences, social sciences, and economics [45]. The support vector clustering (SVC) algorithm, proposed by Ben-Hur et. al [6]-[8], is a recently developed unsupervised learning method inspired by the support vector machine (SVM) [9]. In SVC, data points are mapped from the original space to a higher dimensional feature space by means of a Gaussian kernel. In the feature space, the algorithm seeks the smallest sphere that encloses the images of the data points. When the sphere is mapped back to the original space, it is separated into several contours with each enclosing a separate cluster of data points.

As a kernel-based algorithm, the SVC has many advantages. It can determine the system topological structure without prior knowledge with respect to the system itself, delineate cluster boundaries of arbitrary shapes other than hyper-ellipsoid and hyper-sphere, and deal with outliers by employing a soft margin constant which does not require that the sphere enclose all the data points. Due to these advantages, the SVC has attracted a high level of interest and many variants have been derived to improve its performance. In greater detail, the method presented in [46] extended the SVC to an adaptive cell growing model, which maps data points to a higher dimension feature space through a desired kernel function. In [47], a spatial chunking algorithm is proposed to speed up the SVC algorithm for large scale data sets. In [48], a cluster labeling method for SVC is developed based on some of the invariant topological properties of a trained kernel radius function. In [49], the authors present a kernel method for clustering inspired by the classical K-Means algorithm in which each cluster is iteratively refined using a one-class support vector machine. In [50], a topological and

dynamical characterization of cluster structures described by the support vector clustering is developed. In [51], the authors developed a cluster validity measure with outlier detection for the SVC algorithm.

A review of the above literatures suggests that research on the SVC algorithms has reached a good level of maturation. However, as far as SVC design, there are still some problems requiring further consideration and investigation. We summarize the main problems as follows.

1. The most widely used method for the training of SVC is the sequential minimal optimization (SMO) [52]. Benchmarks reported in [52] show that the time complexity of SMO is $O(N^2)$. This implies that the problem of computational complexity may become intensive as the number of data points increases under real-world problems.
2. The standard deviation of the Gaussian kernel σ plays a crucial role in the clustering results. It controls the shapes of the enclosing contours in the data space. How to configure it suitably so that the SVC is capable of obtaining a desired cluster result for a given data set is an open problem for SVC designers.
3. The algorithm is sensitive to outliers although a soft margin constant C is employed. If an outlier lies close to the cluster boundary, it will distort the cluster shapes, and make the algorithm fail to obtain the desired results.
4. Since the algorithm is performed by finding connected regions of the data distribution, its performance may deteriorate when the data set has connecting clusters.

In view of the above, the study in this chapter is conducted with three integral parts: a SVC training method, an empirical study, and a support vector and K-Means based hybrid algorithm. The three integral parts provide solutions to the aforementioned problems. The SVC training method is proposed based on analysis of the Gaussian kernel radius function. The empirical study is conducted by formulating the SVC training procedure as building up weighted kernel density estimator for underlying distribution of the given data set. The proposed hybrid algorithm works by redefining

the SVC as "one cluster as one sphere" in the feature space. It is achieved in three steps. In the first step, the outliers are identified and removed by training a global SVC. In the second step, the refined data set is clustered by a kernel-based K-Means algorithm. In the final step, several local SVCs are trained for the clusters, and then the removed data points are labeled according to their distance to the local SVCs. Since the proposed algorithm can integrate the advantages of the conventional SVC and the K-Means, it may overcome the difficulties of conventional SVC for its ability to deal data set with noise and connecting clusters.

3.2 Review of support vector clustering and K-Means algorithm

3.2.1 Support vector clustering algorithm

The mathematical formulation of the SVC algorithm is summarized as follows [8]. Let $\chi \subseteq \Re^d$ be a d dimensional data space, $\{x_1, x_2, \cdots, x_N\} \subseteq \chi$ be a data set. The algorithm uses a nonlinear transformation Φ to map the data points from χ to a higher dimensional feature space, and then seeks the smallest enclosing sphere in the feature space. The optimization problem can be formulated as follows:

$$\begin{aligned} &\text{minimize } R^2 + C\sum_{j=1}^{N}\xi_j \\ &\text{subject to } \left\|\Phi(x_j) - \mathbf{a}\right\|^2 \le R^2 + \xi_j, j = 1, \cdots, N, \end{aligned} \tag{3.1}$$

where R is the radius of the enclosing sphere, ξ_1, ξ_2, $\cdots$, ξ_N are slack variables, C is the soft-margin constant, $||\cdot||$ is the Euclidean norm and **a** is the sphere center. The problem can be solved by introducing the Lagrangian function:

$$L = R^2 - \sum_{j=1}^{N}(R^2 + \xi_j - ||\Phi(x_j) - \mathbf{a}||^2)\beta_j + C\sum_{j=1}^{N}\xi_j\mu_j - \sum_{j=1}^{N}\xi_j\mu_j, \tag{3.2}$$

where $\beta_j \ge 0$ and $\mu_j \ge 0$ are Lagrange multipliers. The solution to the primal problem described in Eq. (2) can be obtained by solving the dual problem [53]:

$$\text{maximize } W = \sum_{j=1}^{N}\Phi(x_j)^2\beta_j - \sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j\Phi(x_i)\cdot\Phi(x_j) \tag{3.3}$$

$$\text{subject to } 0 \le \beta \le C_j, \sum_{j=1}^{N} \beta_j = 1, j = 1,\cdots,N$$

The inner dot products $\Phi(x_i) \cdot \Phi(x_j)$ can be replaced by a Mercer kernel $K(x_i, x_j)$. In this thesis, the Gaussian kernel is considered:

$$K(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}, \tag{3.4}$$

where σ is the standard deviation of the Gaussian kernel. After optimization, the distance from a given data point x to the sphere center can be computed by:

$$f(x) = R^2(x) = ||\Phi(x) - \mathbf{a}||^2 = K(x,x) - 2\sum_{j=1}^{N} \beta_j K(x, x_j) + \sum_{i=1}^{N}\sum_{j=1}^{N} \beta_i \beta_j K(x_i, x_j). \tag{3.5}$$

Data points can be identified based on the β values. If $\beta_j = C$, x_j is identified as a bounded support vector (BSV), else if $0 < \beta_j < C$, x_j is identified as a support vector (SV), else, x_j is identified as an inner point. When mapped back to the original data space, the BSVs lie outside of the cluster boundaries, SVs lie on the cluster boundaries, and the inner points lie inside of the cluster boundaries.

The above procedure delineates the contours of the data set. The next problem is the cluster assignment of each data point. The most widely used method for cluster labeling is the complete graph based method, which checks the connectivity for each pair of data points. Other methods include proximity graph based method [54], trained kernel radius function topology based method [48], [50], etc.

3.2.2 K-Means algorithm

The K-Means algorithm is a squared error based clustering algorithm [55]. Its key steps can be summarized as follows:

1. Initialize a K-partition randomly or based on some prior knowledge.
2. Based on current partition, calculate the cluster prototype matrix $M = [m_1, m_2, \cdots, m_K]$.
3. Assign each data point x_j $(j = 1,2,\cdots,N)$ to the cluster c_w with the closest cluster prototype, i.e., $x_j \in c_w$, if $\left|\left|x_j - m_w\right|\right| < \left|\left|x_j - m_i\right|\right|, \forall i = 1,2,\cdots,K(i \neq w)$.

4. Repeat Steps 2-3 until the cluster prototype matrix becomes stable.

The K-Means algorithm is by far the most widely used data clustering algorithm. It is simple in concept, easy in implementation, and has good performance on data sets with compact super sphere distributions.

3.3 A novel SVC training method

The optimization problem described in Eq. 3.3 is a quadratic programming (QP) problem which associates with finding the optimal values of Lagrange multipliers β_1, β_2, $\cdots, \beta_N$. The sequential minimal optimization (SMO) algorithm is by far the most widely used method to solve this problem. The benchmarks reported in [52] show that the time complexity of SMO is $O(N^2)$. This implies that its time complexity becomes intensive as the number of data points increases. In this thesis, a new SVC training method is proposed. The equality constraint $\sum_j \beta_j$ is eliminated by introducing variables a_1, a_2, $\cdots$, a_N. Let $\beta_i = a_i / \sum_k a_k$, then the QP problem described in Eq. 3.3 can be written as:

$$\begin{aligned} &\text{maximize } W = \sum_{j=1}^{N} K(x_j, x_j) \frac{a_j}{\sum_k a_k} - \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{a_i a_j}{(\sum_k a_k)^2} K(x_i, x_j) \\ &\text{subject to } 0 \le \frac{a_j}{\sum_k a_k} \le C_j, j = 1, \cdots, N. \end{aligned} \tag{3.6}$$

Eq. 3.6 describes a QP problem which associates with finding the optimal values of variables a_1, a_2, $\cdots$, a_N. A brief outline of the proposed training method is stated as follows:

1. Set a_1, a_2, $\cdots$, a_N values randomly. Initially, each of the variable a_i satisfies $0 \le a_i / \sum_k a_k \le C$.
2. Select a variable a_i randomly.
3. Calculate $u_i = \sum_{j=1, j\neq i}^{N} a_i K(x_i, x_j)$, $v_i = \sum_{j=1, j\neq i}^{N} a_j$, $a_{max} = \max a_j$, $cons_i = \sum_{j=1, k=1, j\neq i, k\neq i}^{N} a_j a_k K(x_j, x_k)$.
4. Calculate $L = \max(0, -v_i + a_{max}/C)$, $H = C v_i/(1 - C)$.
5. Calculate $a_i' = (u_i v_i - cons_i)/(u_i - v_i)$.
6. Set the optimized value of variable a_i^{new}, according to:

$$a_i^{new} = \begin{cases} L, & \text{if } a_i' \le L, \\ a_i', & \text{if } L < a_i' < H, \\ H & \text{if } a_i' \ge H. \end{cases} \tag{3.7}$$

7. Repeat Steps2-6 until a_1, a_2, ⋯, a_N turns stable.
8. For each β_i, set $\beta_i = a_i / \sum_j a_j$, return β_1, β_2, ⋯, β_N as final optimization results.

In the following, we describe the derivation of the above optimization method. Without loss of generality, let the variable to be optimized at each iteration be a_i, Since $K(x_j, x_j) = e^{-\left||x_j - x_j|\right|^2/(2\sigma^2)} = e^0$, then $K(x_j, x_j) = 1$ is satisfied. Let $u_i = \sum_{j=1,j\neq i}^{N} a_i K(x_i, x_j)$, $v_i = \sum_{j=1,j\neq i}^{N} a_j$, $a_{max} = \max a_j$, $cons_i = \sum_{j=1,k=1,j\neq i,k\neq i}^{N} a_j a_k K(x_j, x_k)$, then Eq. 3.6 can be written as:

$$\begin{aligned} &\text{maximize } W = 1 - \frac{a_i^2 + 2a_i u_i + cons_i}{(a_i + v_i)^2} \\ &\text{subject to } 0 \le \frac{a_i}{a_i + v_i} \le C, 0 \le \frac{a_{max}}{a_i + v_i} \le C. \end{aligned} \tag{3.8}$$

The extremum of the objective function W is at:

$$\frac{\partial W}{\partial a_i} = \frac{2a_i(u_i - v_i) - 2(u_i v_i - cons_i)}{(a_i + v_i)^3} = 0 \tag{3.9}$$

Following Eq. 3.9, we have:

$$a_i = \frac{u_i v_i - cons_i}{u_i - v_i}. \tag{3.10}$$

Following the inequality constraint $0 \le \beta_j \le C$, we have:

$$0 \le \frac{a_i}{a_i + v_i} \le C \quad \text{and} \quad 0 \le \frac{a_j}{a_i + v_i} \le C \quad (j \neq i). \tag{3.11}$$

Following Eq. 3.11, we have:

$$\frac{a_{max} - Cv_i}{C} \le a_i \le \frac{Cv_i}{1 - C}. \tag{3.12}$$

Considering both Eq. 3.10 and Eq. 3.12, we can calculate the new value of variable a_i according to Eq. 3.7.

Remarks.

1. Both the proposed algorithm and the SMO decompose the QP problem into sub-problems. The theorem reported in [56] indicates that the QP problem can be

broken into a series of sub-problems. Thus, both algorithms can guarantee to converge to the global optimum of the QP problem.

2. Both the proposed algorithm and the SMO can optimize the variables analytically, instead of using numerical QP optimization steps.
3. The SMO performs optimization by repeatedly executing two steps: 1) select two variables by heuristic; 2) optimize the two variables analytically. To select the two variables, the algorithm scans the entire data set, and then the data points having the highest probability for violating the KKT conditions are selected. Similar to the SMO, the proposed algorithm performs optimization by repeatedly executing two steps: 1) select one variable randomly; 2) optimize the variable analytically. Since the proposed algorithm is performed without scanning the entire data set, it has the potential to obtain a higher speed than the SMO.

3.4 Empirical Study

3.4.1 SVC Training and weighted kernel density estimator

Let $\{x_1, x_2, \cdots, x_N\} \subseteq \chi$ be a data set taken from an univariate distribution with unknown density p, the weighted kernel density estimator [57] approximates p by function $\hat{p}$:

$$\hat{p}(x) = \sum_{j=1}^{N} w_j \varphi(x, x_j), \tag{3.13}$$

where $\varphi(x, x_j)$ is the window function, $0 \leq w_j \leq 1$ is the weight of the j-th kernel, and $w_1, w_2, \cdots, w_n$ satisfy $\sum_j w_j = 1$. Without loss of generality, we consider the Gaussian kernel window function:

$$\varphi(x_i, x_j) = \frac{1}{\sqrt{2\pi\sigma^2}} K(x_i, x_j). \tag{3.14}$$

The trained kernel support function of SVC described in Eq. (3.5) can be defined by the squared radial distance from a data point x to the sphere center. Given a data point x, if x is a BSV, then:

$$f(x) = K(x,x) - 2\sum_{j=1}^{N}\beta_j K(x,x_j) + \sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j K(x_i,x_j) > R^2. \tag{3.15}$$

Since $K(x,x) = e^{-\frac{||x-x||^2}{2\sigma^2}} = e^0 = 1$ is satisfied, then:

$$\sum_{j=1}^{N}\beta_j K(x,x_j) < \frac{1}{2}\left(1 + \sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j K(x_i,x_j) - R^2\right). \tag{3.16}$$

Similarly, if x lies on the boundary of the sphere, then:

$$\sum_{j=1}^{N}\beta_j K(x,x_j) = \frac{1}{2}\left(1 + \sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j K(x_i,x_j) - R^2\right), \tag{3.17}$$

and if x lies inside of the sphere, then:

$$\sum_{j=1}^{N}\beta_j K(x,x_j) > \frac{1}{2}\left(1 + \sum_{i=1}^{N}\sum_{j=1}^{N}\beta_i\beta_j K(x_i,x_j) - R^2\right). \tag{3.18}$$

Combining Eqs. 3.13 and 3.14 with the term $\sum_j \beta_j K(x,x_j)$, we have:

$$\sum_{j=1}^{N}\beta_j K(x,x_j) = \sqrt{2\pi\sigma^2}\cdot\hat{p}(x). \tag{3.19}$$

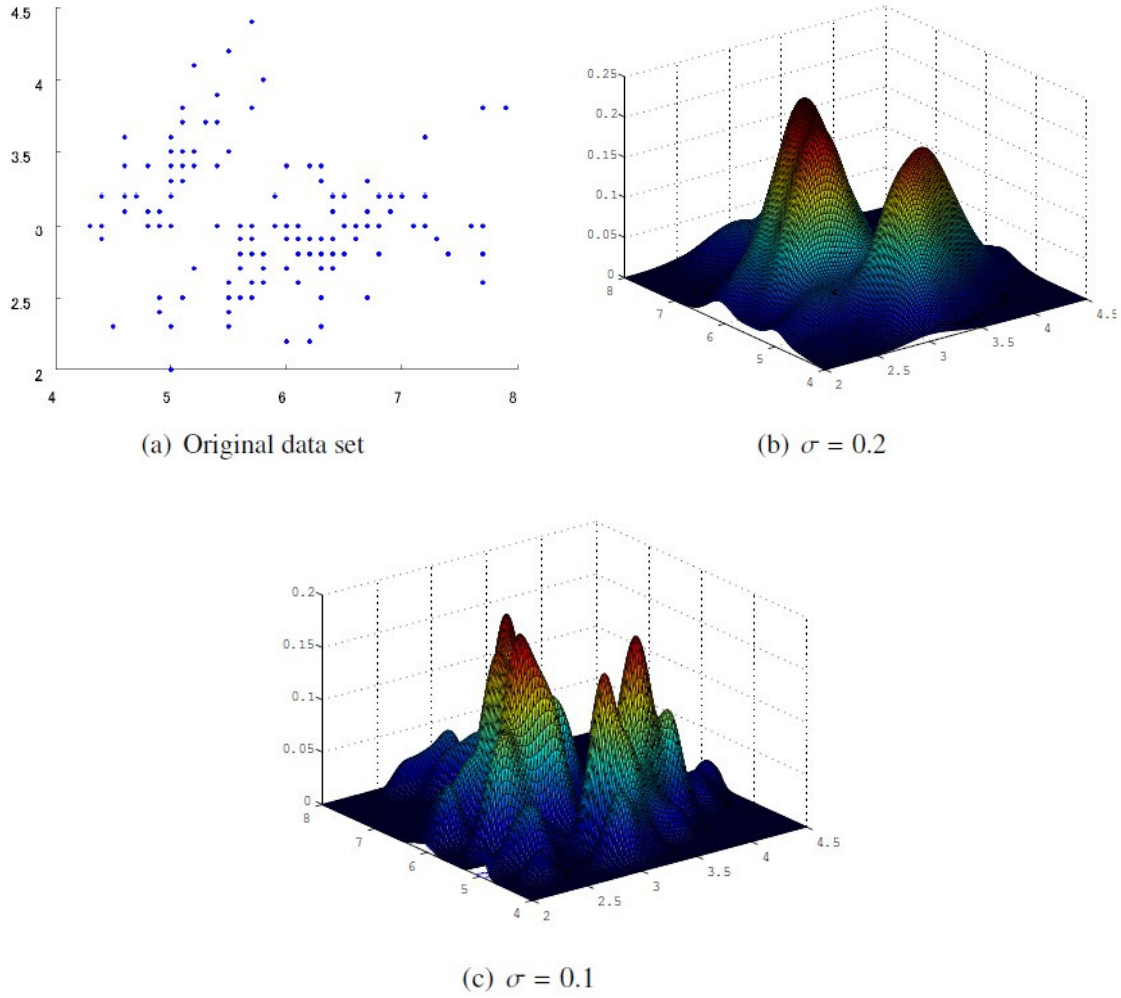
Let $\frac{1}{2}(1 + \sum_{i,j}\beta_i\beta_j K(x_i,x_j) - R^2) = \sqrt{2\pi\sigma^2}\cdot d$, we have the following relations:

1. If x is a BSV, then $\hat{p}(x) < d$.
2. If x is a SV, then $\hat{p}(x) = d$.
3. If x is a inner point, then $\hat{p}(x) > d$.

The above derivation can be explained intuitively as follows. The trained kernel support function of SVC describes a weighted kernel density estimator for the underlying data set. The objective of the SVC training procedure is to find the optimal weights $\beta_1, \beta_2, \cdots, \beta_N$ such that the lower bound of the density function values of the given data points is maximized.

3.4.2 The selection of parameter σ

The derivations in Section 3.4.1 indicate that the SVC training can be formulated as the procedure of building up weighted kernel density estimator for underlying



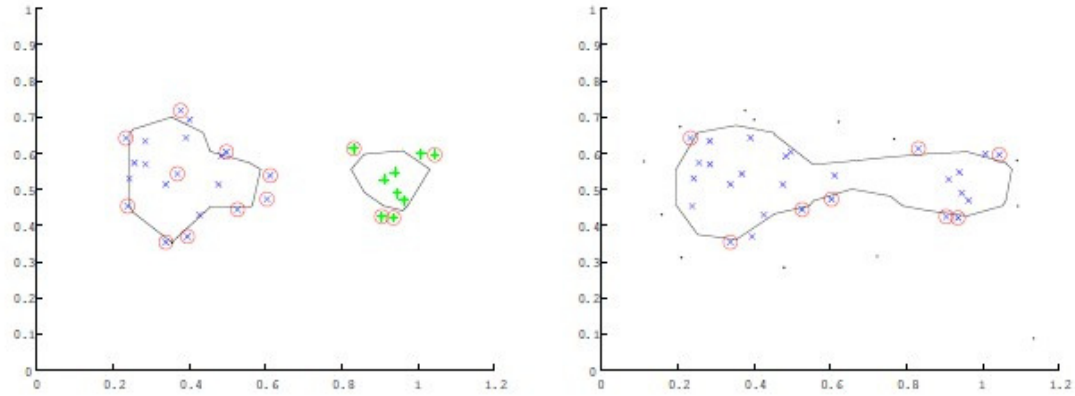
(a) Original data set

(b) $\sigma = 0.2$

(c) $\sigma = 0.1$

Fig. 3.1. Illustrative example for the effect of parameter σ on the density estimator.

distribution of the given data set. Fig.3.1 shows an illustrative example of the effect of parameter σ on the estimator, where Fig.3.1(a) is the original data set, Fig.3.1(b) is the estimator with σ = 0.2, and Fig.3.1(c) is the estimator withσ = 0.2. It can be seen from Fig.3.1 that the parameterσ has effect on the resulting estimator $\hat{p}(x)$. Moreover, the SVC also specifies the contours that enclose the peaks of the probability distribution. As σ decreases, the estimator processes more peaks, and the SVC delineates more clusters. Thus the selecting of parameter σ is equivalent to selecting a proper bandwidth parameter for the density estimator. The density estimator simulated the true density so that the mean square error (MSE) is minimized. Let the true density be $p(x)$ and the



(a) SVC clustering on data set without outliers (b) SVC clustering on data set with outliers

Fig. 3.2. Illustrative example for the effect of outliers on the cluster boundaries.

estimated density be $\hat{p}(x)$, then the MSE is defined by:

$$\mathrm{MSE}(\hat{p}(x)) = E((\hat{p}(x) - p(x))^2] = (E[\hat{p}(x) - p(x)])^2 + \mathrm{Var}(\hat{p}(x)), \tag{3.20}$$

where $E[\cdot]$is the mathematical expectation, $(E[\hat{p}(x) - p(x)])^2$ represents the squared bias of the estimator, and $\mathrm{Var}(\hat{p}(x))$ represents the variance of the estimator. Generally, a large bandwidth will reduce the variance and increase the bias. On the other hand, a small bandwidth will reduce the bias and increase the variance. In [58], the author proposed to compute the bandwidth of the Gaussian kernel according to some practical heuristic. That is:

$$\sigma = \left(\frac{4\hat{\sigma}^5}{3N}\right)^{-\frac{1}{5}} \approx 1.06\hat{\sigma}N^{-\frac{1}{5}}, \tag{3.21}$$

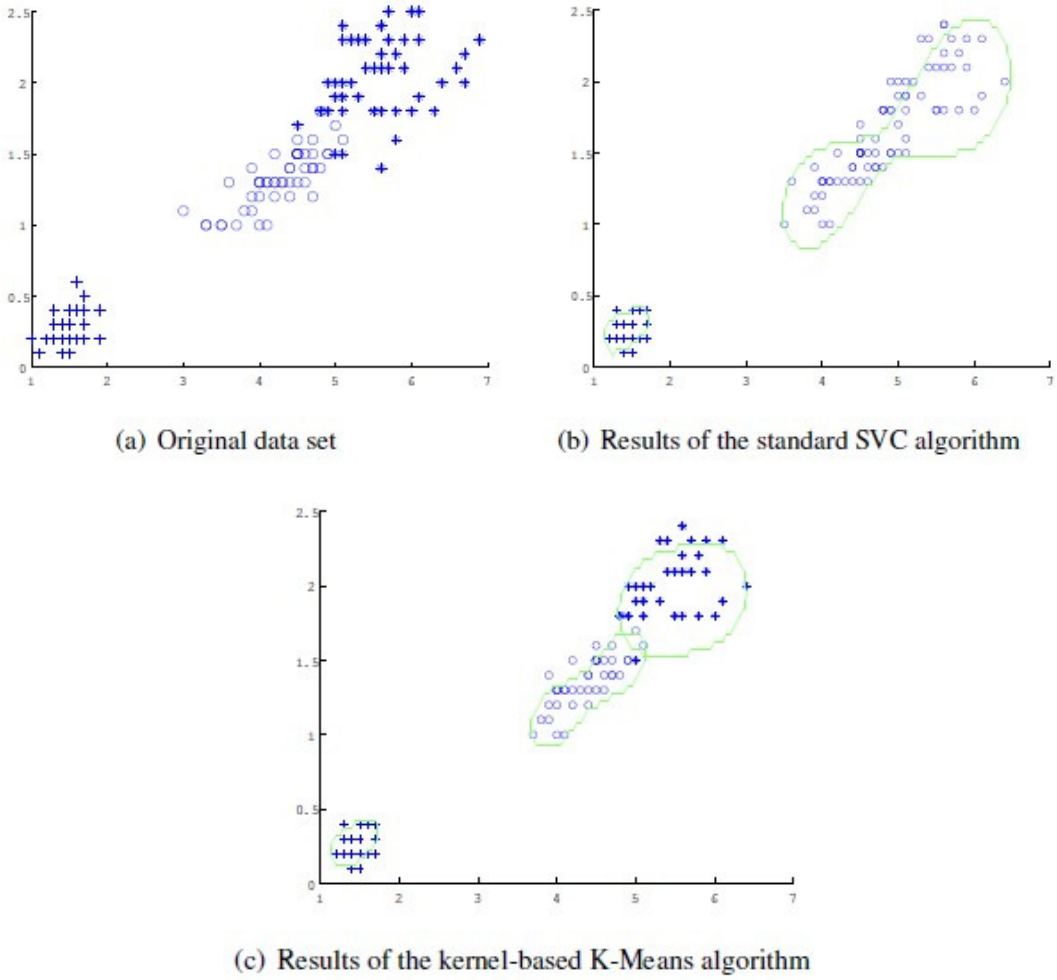
where N is the number of data points, and $\hat{\sigma}$ is the standard deviation of the samples, which is defined by:

$$\hat{\sigma} = \sqrt{\frac{1}{N}\sum_{j=1}^{N} ||x_j||^2 - ||\frac{1}{N}\sum_{j=1}^{N} x_j||^2}. \tag{3.22}$$

In view of this, we select the parameter σ according to Eq. 3.21 as a compromise between the bias and the variance.

3.5 The proposed hybrid intelligent algorithm

The SVC algorithm as reviewed in Section 3.2 has many advantages over other algorithms for its ability to delineate cluster boundaries of arbitrary shape and to deal



(a) Original data set

(b) Results of the standard SVC algorithm

(c) Results of the kernel-based K-Means algorithm

Fig. 3.3. Results of the kernel-based K-Means algorithm and the standard SVC algorithm on data set with connecting clusters.

with outliers by the soft margin constant. However, there are two major difficulties encountered when it is applied to solving real-life data. Firstly, the algorithm is sensitive to outliers although the soft margin constant is employed. The outliers will distort the cluster boundaries, and make the algorithm fail to obtain the desired clustering results. Fig.3.2 shows an illustrative example of this problem. In Fig.3.2(a), no outliers exist in the data set. It can be seen that the SVC can delineate the cluster boundaries correctly. However, in Fig.3.2(b), when the data set contains outliers, the cluster boundaries are distorted. This makes the SVC algorithm delineate undesired cluster boundaries. Secondly, the SVC algorithm is performed by identifying clusters by finding connected components among data points, thus it may have difficulty in dealing

with data sets with connecting clusters. Fig.3.3 shows an illustrative example of this problem. The simulation is conducted on the last two dimensions of IRIS data set. Fig.3.3(a) shows the original data set. As can be seen from Fig.3.3(a), cluster 1 is linearly separable from clusters 2 and 3, and clusters 2 and 3 are connected. In the simulation, we firstly remove the outliers by the data set refining method, which will be described in the rest of this section. Following this, we run the SVC algorithm on the refined data set. Fig.3.3(b) shows the clustering results. As can be seen from Fig.3.3(b), the SVC algorithm cannot distinguish clusters 2 and 3.

To overcome the above difficulties, we propose a support vector and K-Means based hybrid intelligent algorithm (HIA). The key steps of the HIA are as follows:

1. Data set refining: identify and remove the outliers by building a global SVC.
2. Clustering: cluster the refined data set by a kernel-based K-Means algorithm.
3. Local SVC modeling: build local SVCs for each of the clusters.
4. Relabeling: label each removed data point according to the distance from it to the local SVCs.

We now describe and explain the detailed procedure for the above steps.

1. Data set refining: the main objective of this step is to identify and remove the outliers. A global SVC is trained to perform this task. In the global SVC, the BSVs are considered as outliers. The constant C plays a crucial role in the identifying results. It can be seen from Eq. 3.3 that $0 \le \beta_j \le C$ and $\sum_j \beta_j = 1$. The lower bound of C is $\frac{1}{N}$, since the constraint $\sum_j \beta_j = 1$ cannot be satisfied if $C < \frac{1}{N}$. The upper bound of C is 1, since $\sum_j \beta_j = 1$, the constraint $\beta_j \le C$ has no effect if $C > 1$. We propose to select the constant C value by training SVC iteratively: starting with $C = \frac{1}{N}$, and increasing it by δ, to count the number of outliers. The final C value is selected when the number of outliers exceeds a pre-specified threshold n.

2. Clustering: the SVC algorithm may have difficulty clustering data sets with connecting clusters. One natural way to solve this problem is to use the K-Means algorithm, since the K-Means algorithm can work well for compact clusters. However, the K-Means algorithms that use geometric representation are often limited to

hyper-ellipsoids. In view of the above, we propose a novel kernel-based K-Means algorithm. The proposed algorithm works as follows. Firstly, the algorithm maps the data points from the original space to a higher dimensional feature space by a nonlinear transformation Φ. Secondly, in the feature space, the K-Means algorithm is used to cluster the mapped data points. The key steps of the algorithm are as follows:

1. Initialize a K-partition randomly.
2. For each partition, train a one class SVC, i.e., enlarge the SVC by redefining "one cluster as one sphere" in the feature space.
3. Assign each data point x_j $(j = 1,\cdots,N)$ to the sphere with the closest sphere center. The distance from the data point to the sphere center is computed by Eq. 3.5.
4. Repeat Steps 2-3 until the partition becomes stable.

In order to verify the validity of the clustering results, we use separation as the criteria for clustering evaluation. The basic idea of separation is to keep the clusters obtained by clustering algorithm as far apart as possible [51]. In statistics, the Bhattacharyya distance [59] measures the similarity of two discrete or continuous probability distributions, and is often used to measure the separability of clusters. Thus we adopt the Bhattacharyya distance as a separability function to define the degree of separability for each pair of clusters. For a pair of clusters (e.g., cluster c_i and c_j), the Bhattacharyya distance is defined as follows:

$$B(c_i, c_j) = \frac{1}{8}(\mu(c_i) - \mu(c_j))^T \left\{\frac{\sum c_i + \sum c_j}{2}\right\}^{-1} \cdot \left(\mu(c_i) - \mu(c_j)\right) + \frac{1}{2}\ln\left\{\frac{|\frac{\sum c_i + \sum c_j}{2}|}{|\sum c_i|^{\frac{1}{2}}|\sum c_j|^{\frac{1}{2}}}\right\}, \quad (3.23)$$

where $\mu(c_i)$ and $\sum c_i$ are the mean vector and the covariance matrix associated with cluster c_i. The separability function is then calculated according to:

$$B(c_1, c_2,\cdots, c_k) = \min_{i,j\ i\neq j} B(c_i, c_j). \quad (3.24)$$

Due to the random nature of the initialization step of the K-Means, the clustering results may vary in different runs. In our study, we run the kernel-based K-Means for a predefined number of times, and the cluster results that yield the biggest separability function value are adopted.

The kernel-based K-Means is performed by finding the prototypes (centers) of the clusters, while the standard SVC is performed by finding connected components among data points. Thus the kernel-based K-Means may has the advantage over the standard SVC for its ability to deal with connecting clusters. As a comparison, Fig.3.3(c) shows the clustering results of the kernel-based K-Means algorithm on the refined IRIS data set. As can be seen from Fig.3.3(c), the kernel-based algorithm can distinguish clusters 2 and 3, even though they are connected.

3. Local SVC modeling: in this step, we build local SVCs for the clusters obtained by the K-Means. Each of the cluster is described by a one class local SVC. In the local SVC, the parameter σ is determined by the method presented in Section. 3.4.2.

4. Relabeling: the main objective of this step is to label the removed data points. Let set SV_i contains the support vectors of cluster c_i. Given a removed data point x_i, we define the distance from it to cluster c_i as follows:

$$d(x_k, c_i) = \min_{x_j \in SV_i} ||x_k - x_j||^2. \tag{3.25}$$

The data point x_i is then assigned to the cluster with the minimum distance.

3.6 Experimental studies

3.6.1 Experimental setup

We have conducted three sets of experiments to test the performance of the proposed algorithm. The experiments are performed on a PC with Pentium IV 3.0GHz Processor and 4GB memory. The first set of experiments is conducted to test the time complexity of the proposed SVC training method. In the experiments, 100 data sets are simulated. They are simulated as follows: the dimension of the data space is set to 4. In the data space, the data set is generated based on mixture models with size ranging from 50 to 1000. For the experiments, the parameter σ is set by the method described in Section 3.4.2, and the soft margin constant C is set to 1. The CPU time is used to evaluate the performance of the method. The SMO algorithm is used for comparison.

The second set of experiments is conducted to illustrate the execution process of the proposed HIA. In the experiments, two 2D data sets, namely 2D-N160 and 2D-N450,

are generated. The 2D-160 is generated based on the same mixture model as used in [8], and 2D-450 is generated based on the same mixture model as used in [49]. In order to add complexity to the problem, noise data points are also included in the data set. Parameter settings are crucial for the performance of the HIA. Throughout the experiments, the parameter σ in the kernel function is selected by the method described in Section 3.4.2. In the data set refining step, the soft margin constant C is selected by training SVC repeatedly, and the parameters δ and n are taken as follows: $\delta = \frac{1}{N}$, $n = 0.1N$, where N is the total number of data points in the data set. In the clustering step, we run the kernel-based K-Means algorithm for five times, and the cluster results that yield the biggest separability function value are adopted. In the local SVC modeling step, since the outliers are removed in Step 1, the constant C is set to 1. In our experiments, we set the parameters as above and find that they are proper for all the cases.

The third set of experiments is conducted to test the performance of the HIA on benchmarks. In the experiments, three data sets taken from the UCI machine learning repository are selected. The three data sets include: the IRIS data set, the Wisconsin's breast cancer data set, and the Spam data set. Their key properties are as follows.

1. The IRIS data set: The IRIS data set contains three classes, where each class contains 50 instances and refers to one type of IRIS plant. Each instance is composed of 4 measurements of an IRIS flower. One class is linearly separable from the other two. The remaining two classes have significant overlap and are not linearly separable from each other.
2. The Wisconsin's breast cancer data set: the Wisconsin's breast cancer data set contains 699 cases of diagnostic samples. After the removal of the 16 samples with missing values, the data set consists of 683 diagnostic samples. The diagnostic samples are partitioned into two classes, benign and malignant tumors. Each sample is composed of 9 measurements of the clinical attributes. The benign tumors take about 65.5% of the data set, while the malignant tumors take about 34.5% of the data set.

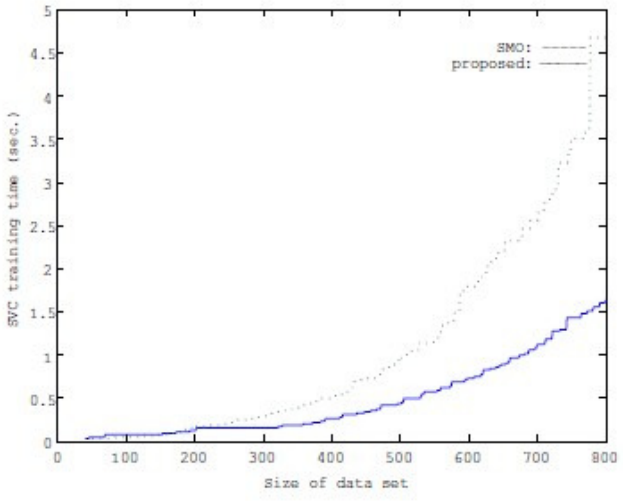


Fig. 3.4. CPU time comparison of the proposed SVC training method and the SMO algorithm.

3. The Spam data set: The Spam data set is a 57 dimension data set formed by 4601 instances. Each instance represents a spam email or a non-spam email. The spam emails came from a postmaster and individuals who had filed spam, and the non-spam ones came from filed work and personal emails.

The parameter settings are the same as those in the second set of experiments. The CPU time and the cluster labeling error rate are used to evaluate the performance of the HIA. Here, the cluster labeling error rate is the percentage of the miss-labeled data points with respect to the total data set. For the purpose of comparison, we select the following algorithms which have been applied to all or some of the selected benchmarks: 1) the support vector clustering algorithm (SVC) [8], the kernel method (Kernel) as described in [49], the SVC described topological and dynamical characterization of cluster structures (TDSVC) [50], and the SVC with cluster validity (CVSVC) [51]. All of the above algorithms are kernel methods. They are able to separate a set of complex and nonlinear data points by transforming them to a higher dimensional feature space. These algorithms have been shown to be successful and can find good enough solutions. Hence, a comparison with these algorithms will demonstrate the performance of the proposed HIA, and will show if it is better or worse than other algorithms. Since the selected benchmarks are labeled data, the conventional SVM is also included for

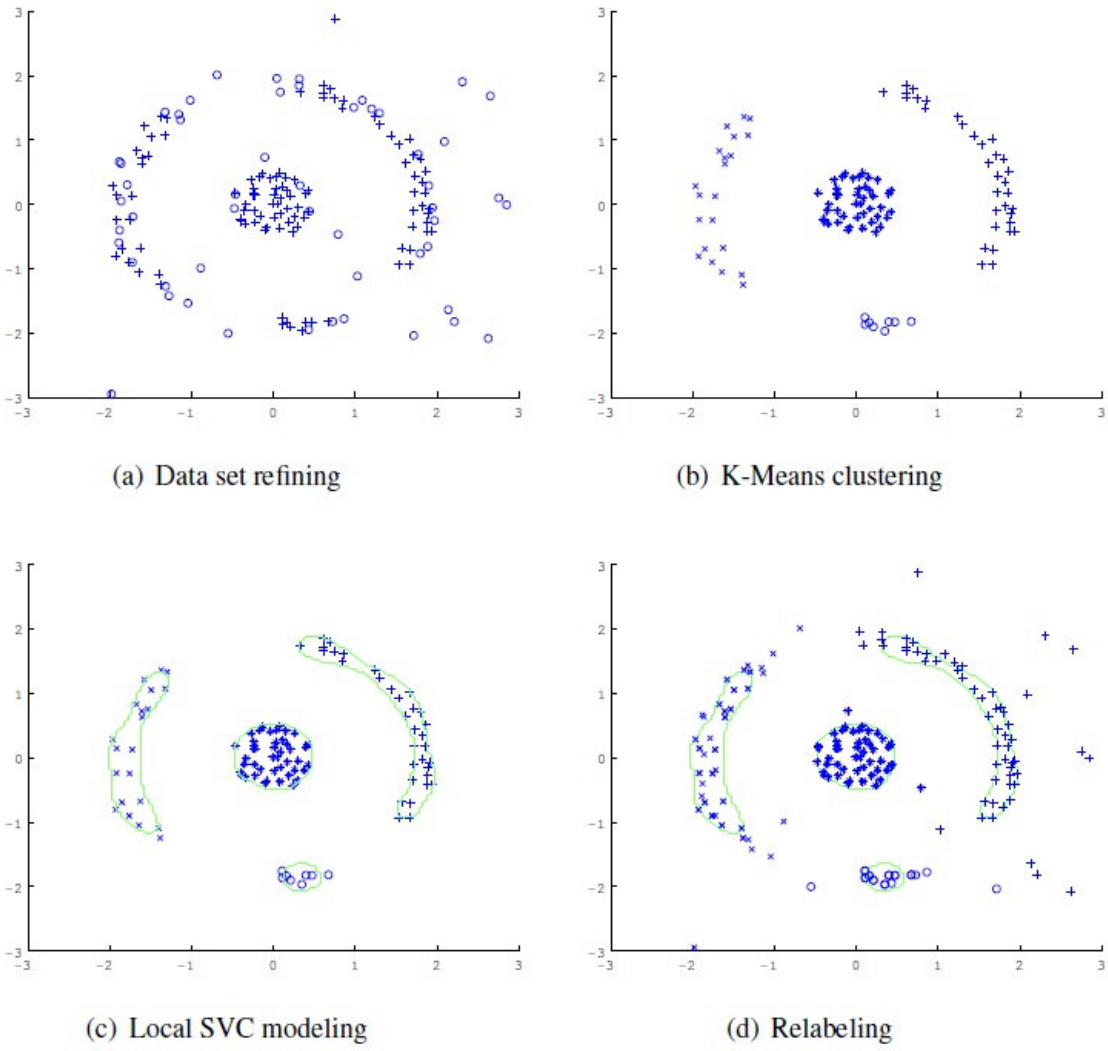


Fig. 3.5. Illustrative example for the execution process of the HIA on 2D-160 data set.

comparison. In the experiment, the SVM tool available in [60] is used. For each data set, 40% percent of the data points are used for training, 20% percent of the data points are used for validating, i.e., selecting optimal values for the parameters of SVM, and 40% percent of the data points are used for testing.

3.6.2 Simulated results

3.6.2.1 Results on data sets with different size

Fig.3.4 shows the CPU time comparison of the proposed SVC training method and the SMO, where the abscissa is the size of data set and the vertical axis is the SVC training time. It can be seen from Fig.3.4 that the two algorithms perform almost the

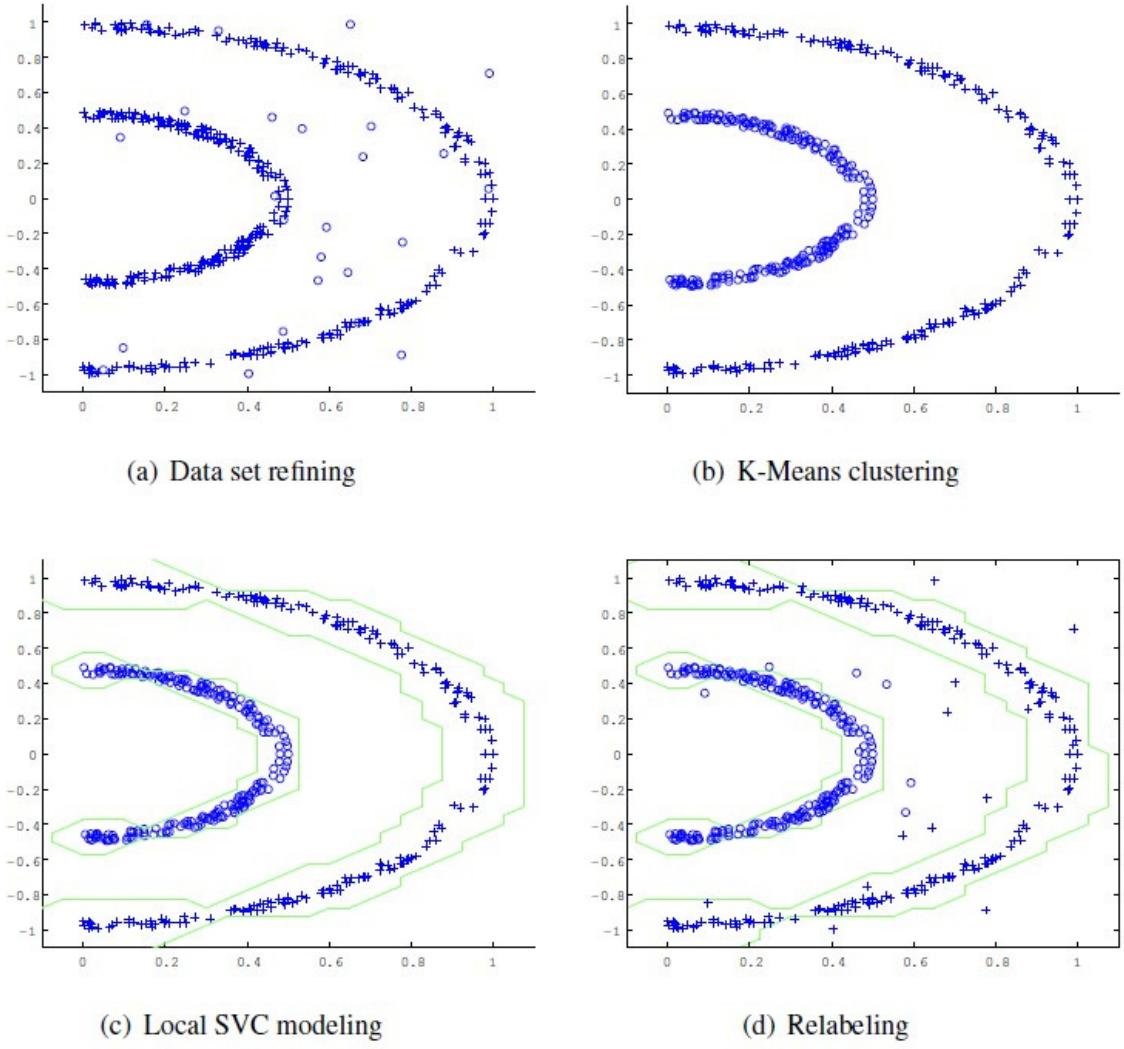


Fig. 3.6. Illustrative example for the execution process of the HIA on 2D-450 data set.

same when the size of data set is smaller than 200. However, when the data set is larger than 200, the proposed algorithm performs much better than the SMO. The results demonstrate that the proposed SVC training method is fast compared to the SMO.

3.6.2.2 Results on generated 2D data sets based on mixture models

1) Results on 2D-N160 data set: The 2D-N160 data set comprises 160 data points in a 2D space and consists of four clusters with noise. Fig.3.5 shows an illustrative example for the execution process of the HIA on the 2D-160 data set. In Fig.3.5(a), the outliers are identified by a global SVC. The data points marked with circle are considered as outliers, and the data points marked with plus are considered as support vectors or inner points. After the removal of the identified outliers, the kernel-based

Table 3.1

Comparison results of the proposed HIA with other algorithms (to be continued).

Data Set	Size	Dim	Proposed HIA		SVC	Kernel	TDSVC	
			Time	Error (%)	Error (%)	Error (%)	Time	Error (%)
IRIS	150	4	0.17	0.00	14.00	5.30	9.38	0.00
Breast	683	9	4.12	0.15	-	3.07	-	-
Spam	4601	57	301.45	2.43	-	18.71	544.96	1.70

K-Means algorithm operates on the refined data set. Fig.3.5(b) shows the kernel-based K-Means clustering results, the data points in the four clusters are marked with cross, star, circle, and plus, respectively. Following this, four local SVCs are built for each of the clusters. Fig.3.5(c) shows the cluster contours delineated by the local SVCs. In Fig.3.5(d), the removed data points are labeled according to the distance from them to the local SVCs, resulting in labeling the whole data set.

2) Results on 2D-450 data set: The 2D-450 data set comprises 450 data points in a 2D space and consists of two clusters with noise. The two clusters are not linearly separable from each other. The experiments conducted on 2D-160 are repeated on 2D-450. Fig.3.6 shows an illustrative example for the execution process of the HIA on the 2D-450.

The results on the 2D-160 data set and the 2D-450 data set demonstrate that the proposed HIA can perform well when the data set contains nonlinearly separable classes and are polluted by noise.

3.6.2.3 Results on data sets taken from the UCI machine learning repository

Table 3.1 summarizes the results of the experiments on data sets taken from the UCI machine learning repository. The contents of the table include the name of each data set (Data set), the scale of the data set (Size), the dimension of the data set (Dim), the CPU time used for the proposed algorithm (Time), the cluster labeling error rate (Error%) of the HIA, and the results reported in literatures using other kernel-based

Table 3.1

Comparison results of the proposed HIA with other algorithms (continued).

Data Set	Size	Dim	Proposed HIA		CVSVC	SVM	
			Time	Error (%)	Error (%)	Time	Error (%)
IRIS	150	4	0.17	0.00	3.33	0.02	0.00
Breast	683	9	4.12	0.15	2.93	3.98	2.64
Spam	4601	57	301.45	2.43	-	429.88	2.15

algorithms [8], [49]-[51] and the results of SVM. For the algorithm compared, not all results of selected benchmarks are reported in literature, so for those benchmarks without reported results, their corresponding comparison values are marked with '-'. For the SVM, the CPU time includes the training time and the testing time, and the cluster labeling error rate is the percentage of the miss-labeled testing data points with respect to the total testing data points.

From the table, it can be seen that the proposed algorithm yields a significant improvement in terms of cluster labeling error rate with respect to the listed results, except for the results obtained by TDSVC algorithm. Compared with TDSVC, it can be seen that both algorithms can obtain the same results for the IRIS data set, and the TDSVC algorithm has a better performance for the Spam data set. Concerning about the time complexity, the TDSVC follows the key steps of the standard SVC, i.e., solving the quadratic programming of SVC, and then labeling each data point. Generally, cluster labeling task is more computationally intensive than the SVC training task, and may become highly intensive as the scale of the data set increase. On the other hand, in the proposed algorithm, the kernel-based K-Means algorithm yields labeled data points for most data points, and each removed data point is labeled according to the distance from it to the clusters. By using this strategy, the cluster labeling step as used in the standard SVC is unneeded. Thus, compared with TDSVC, the proposed algorithm has the potential to obtain a faster computing time with a moderate labeling error rate. The results shown in Table 3.1 validate the above analysis although it cannot provide a

completely fair comparison since the CPU and memory specifications are different. In summary, the above results indicate the successful incorporation of the SVC algorithm and the K-Means.

3.7 Conclusion

The SVC algorithm has been successfully applied to solving many real-life data clustering problems. In this chapter, we aimed at proposing a new kernel-based clustering algorithm to improve the performance of SVC. A SVC training method was proposed based on theoretical analysis of the Gaussian kernel radius function. An empirical study was conducted to guide better selection of the standard deviation of the Gaussian kernel. A new data clustering algorithm, i.e., the support vector and K-Means based hybrid intelligent algorithm, was developed by integrating the merits of both SVC and K-Means algorithm. The effectiveness of the proposed algorithm was validated by three sets of experiments on generated data sets with different sizes, generated 2D data sets, and data sets taken from the UCI machine learning repository. The results demonstrated that the proposed hybrid intelligent algorithm compared favorably with existing kernel-based clustering algorithms. The current research can be further extended to study the theoretical properties of the algorithm (in particular, convergence analysis of the hybrid clustering algorithm) and to apply the algorithm to the solving of real-life problems such as image processing and computer vision.

Chapter 4: Support vector description of clusters for content based image annotation

4.1 Introduction

There has been a surge of research interest in image automatic or semi-automatic annotation based on the low-level image visual contents. These methods are referred to as content based image annotation (CBIA). In greater detail, the methods presented in [61]-[65] are based on multiple classifiers. They partition the images into different classes, and assign each class a distinct topic of interest and a set of descriptive words. For an untagged image, the system treat annotation as a classification problem and select the relevant annotation words based on the classification results. The methods presented in [1], [66]-[73] are probabilistic modeling methods, which are also referred to as generative modeling methods. They try to learn the correlations between images and annotation words by statistical tools so that the joint probability for an untagged image being labeled with each word can be computed. In the annotating process, the relevant annotation words are selected by graph based techniques [71], [72] or some other data fusion and aggregation techniques [1], [66]-[70], [73]. More recently, the development of image platform on the Internet, e.g., Flickr [74], Alipr [67], and PhotoStuff [75], has enabled users to annotate images and give feedbacks to the annotating results. This provides opportunities to develop automatic annotating methods using the user provided information and the existing search results. Methods presented in [61], [76], [77] fall into this category.

Many of the above methods require substantial machine learning techniques to fill the gap between the low-level image visual contents and the high-level semantics. Among the machine learning techniques, the support vector clustering (SVC), which was described in Section 4, has many advantages over other algorithms for its ability to determine the system topological structure without prior knowledge with respect to the system itself, to delineate cluster boundaries of irregular shapes, and to deal with

outliers by employing a soft margin constant [48], [50]. In the real world image systems, many images described by the same words often have a wide range of variety. For example, if an image is tagged with “historical building”, then it can be a picture taken in the desert, near the beach or in the city. These images are organized irregularly in the image system. Since the SVC exhibits its ability to delineate cluster boundaries of irregular shapes, it will provide opportunities to develop unified models to describe the irregularly organized images.

In view of the above, in this chapter, we present a novel algorithm for content based image annotation. In this work, images are represented by colored pattern appearance models (CPAM) [78]. The system has two major components, the training process and the tagging process. In the training process, clusters of images with manually tagged words are used as training instances. For each cluster, the image vectors are mapped to a higher dimensional space and the vectors identified as support vectors are used to describe the cluster. Since the mapping process is the same as that in SVC and its objective is to build support vector described models for image clusters, we term the system as support vector based method for image annotation (SVIA). In the tagging process, for an image to be tagged, the distances from it to the support vector described models are computed. A closer distance indicates a stronger association between the image and the model. Moreover, the word to word correlations contain rich information about the semantic meanings of the images. For example, if an image is tagged with ``France", then it will have a higher probability to be tagged with ``Europe", and if an image is tagged with ``indoor", then it will have a lower probability to be tagged with ``grass". Therefore, the word to word correlations are also considered in the annotation framework. For an image to be tagged, the system exploits the distances from the image to the models and the word to word correlations in a probabilistic framework to predict annotation words.

4.2 Related works

This work is related to the probabilistic modeling methods. In this section, we review the basic concepts and some prevailing methods of the probabilistic modeling

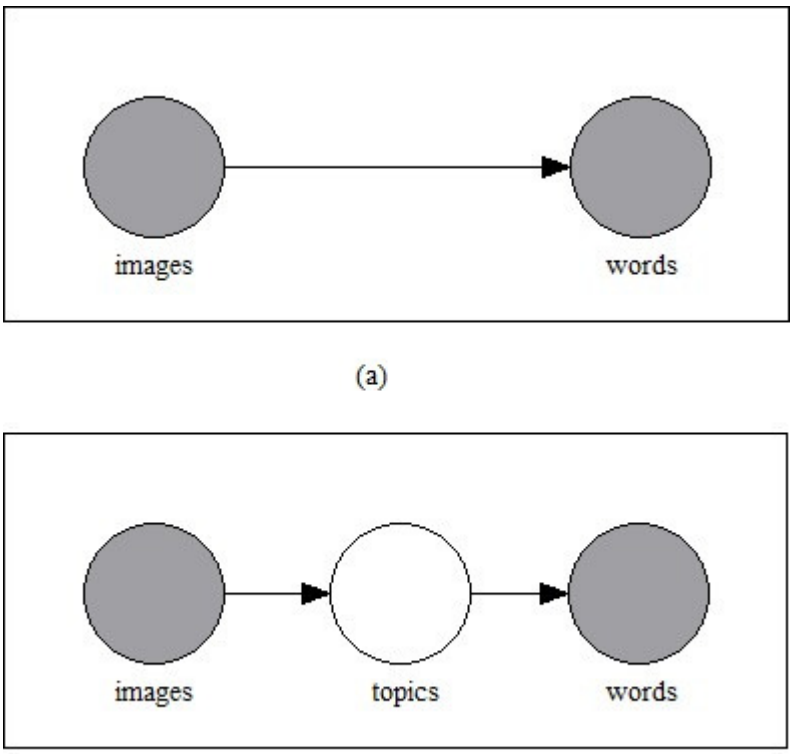


Fig. 4.1 Generative models for image auto-annotation. (a) The two-layer model. Words are directly generated from visual features. (b) The three layer model. Words are generated from a hidden layer of "topics".

approaches to CBIA.

4.2.1 Probabilistic models

The probabilistic modeling method tries to compute the joint probability for an untagged image being labeled with each annotation word. Given an untagged image I_q, the main objective is to find a group of words w^* in a given vocabulary $\mathcal{W}$, such that the conditional distributions $p(w|I_q)$ are maximized as follows:

$$w^* = \arg\max_{w \subset \mathcal{W}} p(w|I_q). \tag{4.1}$$

One type of probabilistic modeling method operates by generating words directly from the visual content of the given image. The formulation can be described as:

$$w^* = \arg\max_{w \subset \mathcal{W}} \sum_{I_i \in \mathcal{T}} p(w|I_i) p(I_q|I_i) p(I_i), \tag{4.2}$$

where $\mathcal{T}$ is the training image set, $p(w|I_i)$ is the probability that the word w can be generated by training image I_i, $p(I_q|I_i)$ is the probability that I_i is relevant (or similar)

to I_q, and $p(I_i)$ is the prior probability of I_i. This is corresponds to the generative model shown in Fig.4.1(a), in which annotations are generated directly given the images. Another type of probabilistic modeling method operates by introducing a set of "topics". The training images are distributed to the topics. Then the words are generated from the topics. The formulation can be described as:

$$w^* = \arg\max_{w \subset \mathcal{W}} \sum_{I_i \in \mathcal{T}} \left\{ \left(\sum_{t_j \in S} p(w|t_j) p(t_j|I_i) \right) \times p(I_q|I_i) P(I_i) \right\}, \tag{4.3}$$

where S is a set of topics, $p(w|t_j)$ is the probability that the word w can be generated by topic t_j, $p(t_j|I_i)$ is the probability that I_i is correlated with t_j. This is corresponds to the generative model shown in Fig.4.1(b), where there is a hidden layer of "topics" so that images are represented as a mixture of them. It is from these topics that words are generated. In addition, the word to word relation $p(w_i|w_j)$ can also be used in the probabilistic modeling formulation to maintain the semantic consistence.

4.2.2 Prevailing methods

Some of the methods are based on the formulation described in Eq. 4.2 or its variants. For instance, Duygulu et al. [66] proposed an object recognition model to translate the image regions into words. Firstly, images are segmented into regions, which are classified into region types based on the visual contents. Then a mapping between the region types and the words is learned using an expectation-maximization based algorithm. Tang et al. [72] explored a unified learning framework that combines the multiple instance and single instance of image features for annotation. An integrated graph based semi-supervised learning framework is proposed to utilize the multiple instances and single instance simultaneously. And three strategies are explored to convert from multiple instance representation into a single instance one. Lu et al. [70] proposed a discriminative stochastic method for image categorization and annotation. Images are divided into blocks. Visual keywords are generated by quantizing the features of the image blocks. A spatial Markov chain model that uses the visual keywords as input is proposed to perform categorization and annotation.

On the other hand, some of the methods are based on the formulation described in Eq. 4.3 or its variants. For instance, Li et al. [67] proposed a statistical modeling approach to CBIA. Categorized images are used to train a set of statistical models, with each representing a topic. Images of each topic are regarded as instances of stochastic process that characterizes the topic. The association between an untagged image and each topic is measured by the probability of the image generated by the stochastic process of the topic. Carneiro et al. [68] proposed a probabilistic formulation for image annotation and retrieval. Annotation and retrieval are posed as classification problems. Each class is defined as a group of training images labeled with a set of labels. Then a minimum probability of error annotation and retrieval is computed by establishing the one-to-one correspondence between labels and the image classes. Li et al. [1] proposed an automatic linguistic indexing of pictures-real time system. In the proposed system, the discrete distribution clustering is developed to group objects by bags of weighted vectors. A generalized mixture modeling technique is developed using the concept of hypothetical local mapping.

More recently, some of the methods use the user provided information and the existing search results to perform annotation tasks, i.e., the word to word relation $p(w_i|w_j)$ is used in the probabilistic modeling formulation. For instance, Wang et al. [76] proposed an attempt at model-free image annotation, which is a data-driven approach that annotates images by mining their search results. Wong et al. [77] proposed a semantic annotation technique based on the use of image parametric dimensions and meta-data. Zhou et al. [61] proposed a hybrid probabilistic model which integrates low-level image features and high-level user provided tags to automatically tag images.

4.3 Support vector description of clusters

In this section, we describe the training process of the proposed SVIA system. Firstly, the colored pattern appearance model (CPAM) representation of images [78], which is used to represent the color and texture visual contents of the images, is reviewed. Following this, the details of the support vector modeling process is presented. Finally,

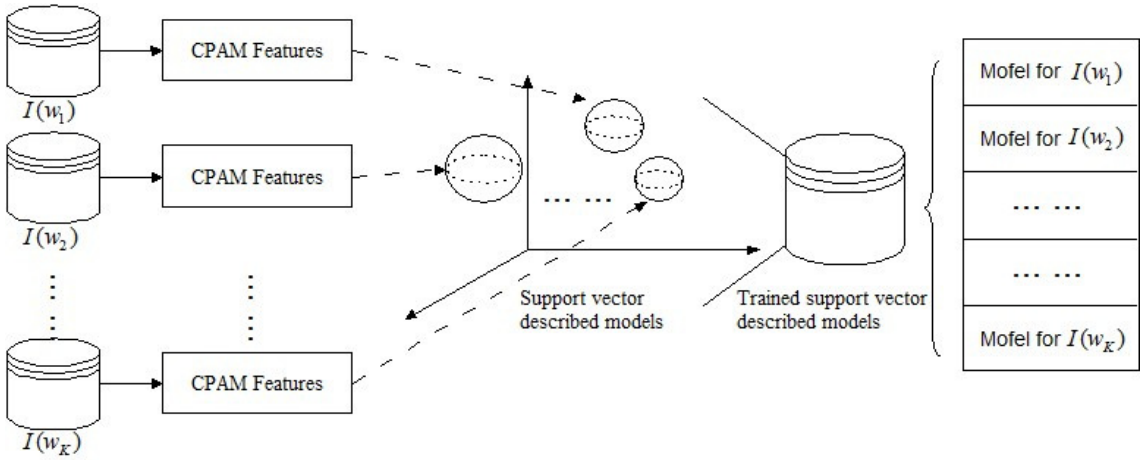


Fig. 4.2. The support vector modeling process.

the method of estimating the probability of a given image generated by the support vector described model is presented.

4.3.1 CPAM representation of images

In this work, the visual content of images are represented by CPAM[2]. The CPAM captures the statistically representative chromatic and achromatic spatial image patterns. And the distributions of these patterns are used to characterize the color and texture information of the visual content. The application of our system is to tag images taken in daily life other than special fields such as medical or geography. Since the CPAM can capture both the features around salient points and the features included in the entire image [61], it is therefore suitable for our application.

To represent an image using CPAM, firstly, the image is partitioned into a set of 4×4 blocks. Each block is then represented by CPAM appearance prototypes, which are comprised of the achromatic spatial pattern histograms (ASPH) and the chromatic spatial pattern histograms (CSPH). The CPAM appearance prototypes of the 4×4 blocks are then concatenated to construct a feature vector x of the entire image. In the experiment, a feature vector with 64 achromatic prototypes and 64 chromatic

[2] In the experiment, the software developed by Zhou et al [61] was used. The Matlab codes of the software were downloaded from:
http://www.viplab.cs.nott.ac.uk/download/CPAM.html

prototypes is selected. Thus, each image is represented by a 128-dimensional vector. Given two images, their CPAM based feature vectors are represented by x_1 and x_2, respectively. Then the distance between x_1 and x_2 can be defined as:

$$d(x_1, x_2) = \sum_{\forall i} \frac{|ASPH_1(i) - ASPH_2(i)|}{1 + ASPH_1(i) + ASPH_2(i)} + \sum_{\forall j} \frac{|CSPH_1(j) - CSPH_2(j)|}{1 + CSPH_1(j) + CSPH_2(j)}, \quad (4.4)$$

where $|\cdot|$ is the absolute value, $ASPH(i)$ is the i-th ASPH component, and $CSPH(j)$ is the j-th CSPH component.

4.3.2 The support vector described model

Let the set of distinct annotation words be $\mathcal{W}=\{w_1, w_2, \cdots, w_K\}$. Given a word w_i, let the set of images tagged with w_i be $\mathcal{I}(w_i)$. Fig.4.2 illustrates the support vector modeling process. Firstly, the CPAM based feature vectors are extracted for the images. Following this, a support vector described model is trained for each image set $\mathcal{I}(w_i)$. The training method maps the CPAM based feature vectors to a higher dimensional space by a nonlinear transformation, and then seeks the smallest enclosing sphere in the higher dimensional space. Each sphere represents a support vector described model. This process follows closely to the derivations of Eqs. 3.1, 3.2 and 3.3, which are described in Section 3.2.1. In Fig.4.2, for the convenience of illustration, we assume that the CPAM-based feature vectors are mapped to a three dimensional space. The spheres are referred to as the support vector described models, which are stored in the form of the parameters of the trained kernel radius functions.

The Gaussian kernel described in Eq. 3.4 of Section 3.2.1 cannot be directly applied to the CPAM based features. In this thesis, we define the kernel function as:

$$K(x_i, x_j) = e^{-\frac{d(x_i, x_j)}{h}}, \quad (4.5)$$

where $d(x_i, x_j)$ is the distance between the CPAM based feature vectors x_i and x_j, which is described in Eq. 4.4. The Gaussian bandwidth parameter h plays a crucial role in the support vector described model. It controls the shape of the enclosing contour in the data space, and affects the width of the kernel function. The problem of selection of the parameter h can be referred to as balancing the empirical risk and the confidence risk [79] of the support vector described model. In this thesis, the parameter h is chosen

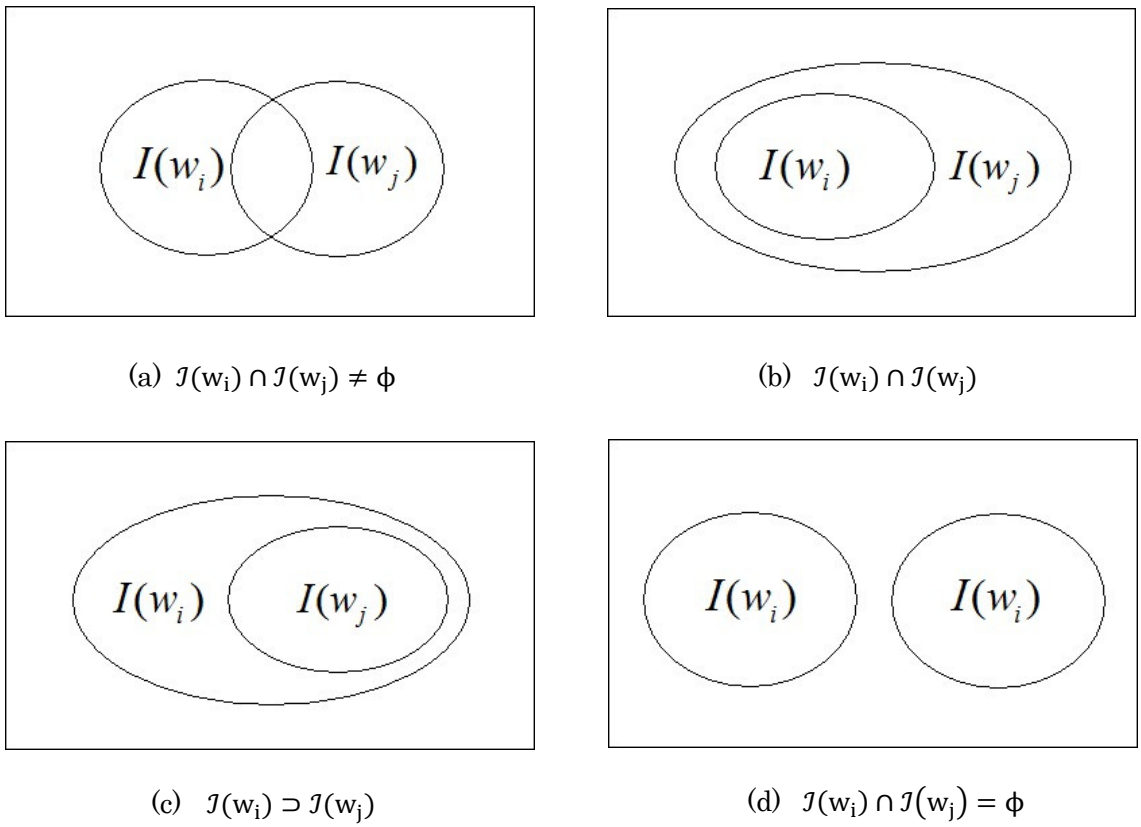
using a trial and test method. The trial and test method operates by adjusting the h value and calculating the system score iteratively. The h value that yields the highest system score is then adopted. For a set of training images and a set of validating images, the system score can be calculated as follows. Firstly, the system score is set to 1, and then a support vector described model is trained for each image set $\mathcal{I}(w_i)$ using the current h value. Following this, for each validating image, the distance from it to the support vector described models is calculated using the trained kernel radius function as described in Eq. 3.5 of Section 3.2.1. Suppose the model that yields the shortest distance be m_s, and suppose the set of models that actually generate the validating image be M_g. If $m_s \in M_g$, then the system score is increased by 1, else, it remains unchanged. The system score is obtained when all the images in the validating set are evaluated. Since the experiment on all the training images of the system is computational intensive, thus, in the experiment, for trial and test, we randomly select 10% images of the system for training and select 5% images of the system for validating.

The main computational cost in the support vector modeling comes from the solving of the quadratic programming (QP) problem described in Eq. 3.3 of Section 3.2.1. In this thesis, we use the algorithm described in Section 3.3 as a tool for the training of the support vector described models.

4.3.3 The probabilities of a given image generated by the support vector described models

The derivation described in Section 3.4.1 indicates that the trained kernel radius function of the support vector described model delineates a weighted density estimator for underlying distribution of the data set. Thus, in this thesis, we use this density estimator to compute the probability of a given image being generated by the support vector described models. Given a CPAM based feature vector x of image I, and a trained kernel radius function of model m, the probability of x being generated by m is computed by:

$$p(x|m) = \frac{1}{h}\sum_{j=1}^{n} \beta_j K(x, x_j). \tag{4.6}$$



(a) $\mathcal{I}(w_i) \cap \mathcal{I}(w_j) \neq \phi$ (b) $\mathcal{I}(w_i) \cap \mathcal{I}(w_j)$

(c) $\mathcal{I}(w_i) \supset \mathcal{I}(w_j)$ (d) $\mathcal{I}(w_i) \cap \mathcal{I}(w_j) = \phi$

Fig. 4.3 The relationship between $\mathcal{I}(w_i)$ and $\mathcal{I}(w_j)$.

4.4 The annotation method

In this section, we describe the tagging process of the proposed SVIA system. A unified probabilistic framework which generates words from the support vector described models and the word to word correlations is proposed.

4.4.1 The word to word correlations $p(w_i|w_j)$

The word to word correlations contain rich information about the semantic meanings of the images. For instance, if an image is tagged with “France”, then it will have a higher probability to be tagged with “Europe”. If an image is tagged with “indoor”, then it will have a lower probability to be tagged with “grass”. If the system uses only the visual contents of images to generate tagging words, the information contained in the word to word correlations will be lost. Therefore, the word to word correlations are considered in the probabilistic framework. Let $\mathcal{T} = \{I_1, I_2, \cdots, I_N\}$ be a set of images in the

training set and $\mathcal{W}=\{w_1, w_2, \cdots, w_M\}$ be a given vocabulary. Each image $I_i \in \mathcal{T}$ is manually tagged with a set of M_i words $\{w_{i1}, w_{i2}, \cdots, w_{iM_i}\}$. Denote the set of images that contain word w_i in their annotations by $\mathcal{I}(w_i)$. Given a pair of words w_i and w_j, we use the following formulations to estimate the word to word correlations:

$$p(w_i|w_j) = \frac{|\mathcal{I}(w_i) \cap \mathcal{I}(w_j)|}{|\mathcal{I}(w_j)|}, \tag{4.7}$$

$$p(w_i|w_j) = \frac{|\mathcal{I}(w_i) \cap \mathcal{I}(w_j)|}{|\mathcal{I}(w_i)|}, \tag{4.8}$$

where $|\cdot|$ is the number of images in the set. Generally, the relationship between $\mathcal{I}(w_i)$ and $\mathcal{I}(w_j)$ falls into four categories, which correspond to the illustrative figures shown in Fig.4.3 and can be summarized as follows:

1. If $\mathcal{I}(w_i) \cap \mathcal{I}(w_j) \neq \phi$, then $p(w_i|w_j) > 0$ and $p(w_j|w_i) > 0$;
2. If $\mathcal{I}(w_i) \subseteq \mathcal{I}(w_j)$, then $0<p(w_i|w_j) \leq 1$ and $p(w_j|w_i) = 1$;
3. If $\mathcal{I}(w_i) \supset \mathcal{I}(w_j)$, then $p(w_i|w_j) = 1$ and $0 < p(w_j|w_i) < 1$;
4. If $\mathcal{I}(w_i) \cap \mathcal{I}(w_j) = \phi$, then $p(w_i|w_j) = 0$ and $p(w_j|w_i) = 0$;

4.4.2 The probabilistic framework

Given an untagged image I_q, if the system has not selected any word as annotation, then the conditional probability that I_q being tagged with w_i can be represented by $p(w_i|I_q)$, which can be computed by applying the Bayesian rule:

$$p(w_i|I_q) = \frac{p(w_i)p(I_q|w_i)}{p(I_q)} \tag{4.9}$$

where $p(I_q|w_i)$ is the probability of I_q being generated by w_i, $p(I_q)$ is the probability of the image I_q, and $p(w_i)$ is the probability of the word w_i. If the system has selected w_{q1}, $w_{q2}, \cdots$, w_{qM_q} as annotations, taking into account the word to word correlations among the new word w_i and the selected words w_{q1}, $w_{q2}, \cdots$, w_{qM_q}, then the conditional probability that I_q being tagged with w_i can be represented by $p(w_i|I_q, w_{q1}, w_{q2}, \cdots, w_{qM_q})$, which can be computed by applying the Bayesian rule:

$$p\left(w_i \middle| I_q, w_{q1}, w_{q2}, \cdots, w_{qM_q}\right) = \frac{p(w_i)p(I_q, w_{q1}, w_{q2}, \cdots, w_{qM_q}|w_i)}{p\left(w_{q1}, w_{q2}, \cdots, w_{qM_q} \middle| I_q\right) p(I_q)}. \tag{4.10}$$

For a new word w_i, we can assume that I_q, w_{q1}, w_{q2}, $\cdots$, w_{qM_q} are independent of

each other, thus:

$$p\left(I_q, w_{q1}, w_{q2}, \cdots, w_{qM_q} \middle| w_i\right) = p\left(I_q \middle| w_i\right) \prod_{t=1}^{M_q} p\left(w_{qt} \middle| w_i\right). \tag{4.11}$$

Then Eq. 4.10 can be written as:

$$p\left(w_i \middle| I_q, w_{q1}, w_{q2}, \cdots, w_{qM_q}\right) = \frac{p(w_i) p\left(I_q \middle| w_i\right) \prod_{t=1}^{M_q} p\left(w_{qt} \middle| w_i\right)}{p\left(w_{q1}, w_{q2}, \cdots, w_{qM_q} \middle| I_q\right) p(I_q)}. \tag{4.12}$$

$p(w_i)$ can be estimated using the training set. Suppose the total number of annotations on the training images is n, and the number of annotations using word w_i is n_i, then $p(w_i)$ can be estimated by:

$$p(w_i) = \frac{n_i}{n}. \tag{4.13}$$

Suppose the CPAM based feature vector of I_q is x_q, and the support vector described model that corresponds to w_i is m_i, then the probability $p(I_q|w_i)$ can be computed by the formulation described in Eq. 4.6, i.e., $p\left(I_q \middle| w_i\right) = p(x_q|m_i)$. For w_{q1}, w_{q2}, $\cdots$, w_{qM_q}, the word to word correlations can be computed by Eqs. 4.7 and 4.8. In our implementation, we assume that $p\left(w_{q1}, w_{q2}, \cdots, w_{qM_q} \middle| I_q\right)$ and $p(I_q)$ are kept constant across different new words.

4.4.3 Annotation of untagged images

Let the set of distinct annotation words be $\mathcal{W} = \{w_1, w_2, \cdots, w_K\}$. Given an untagged image I_q, let the set of words that have been selected as annotations by the system be $\mathcal{V} = \{w_{q1}, w_{q2}, \cdots, w_{qM_q}\}$, where M_q is the number of words in $\mathcal{V}$. To annotate I_q, its CPAM based feature vector is extracted first. For each $w_i \in \mathcal{W} - \mathcal{V}$, the conditional probability that I_q being tagged with w_i is then computed. If $\mathcal{V} = \phi$, i.e., $M_q = 0$, then the conditional probability is computed by Eq. 4.9, else if $\mathcal{V} \neq \phi$, i.e., the system has selected a set of words as annotations, then the conditional probability is computed by Eq. 4.12. By computing the conditional probability for the words in the set $\mathcal{W} - \mathcal{V}$, the words can be sorted in descending order, and the top ranked word is selected as next annotation. Suppose the top ranked word be w^*, then we set $\mathcal{V} = \mathcal{V} \cup \{w^*\}$ and

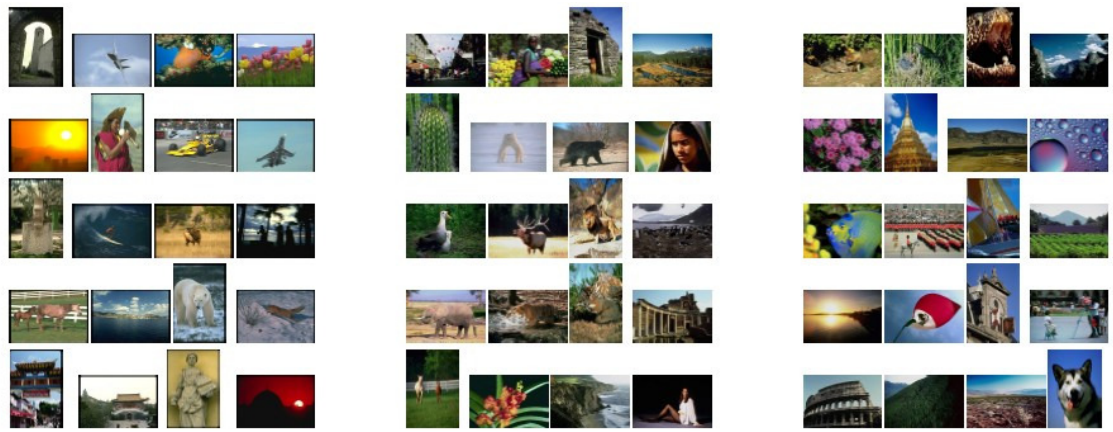
$M_q = M_q + 1$. This process is repeated until the number of selected words reaches a predefined number k.

4.5 Experimental studies

In this section, we conduct experiments to evaluate the performance of the proposed SVIA. The experiments were performed on a PC with pentium IV 3.0GHz processor and 4GB memory.

4.5.1 Image datasets

In the experiment, we select the Corel5k dataset [66], the Corel30k dataset [68], and the Corel60k dataset [1] as benchmarks. These three datasets are originated from the Corel stock photograph collection and are widely used in evaluating the image annotation methods. The Corel5k dataset contains 5000 images that are stored with size 192×128 or 128×192. There are 371 distinct words in the vocabulary. Each image is tagged with 1-5 words, and the average number of words per image is 3.5. According to Ref. [66], the data set is divided into two parts. Out of the 5000 images, 4500 images are used for training and 500 images are used for testing. The testing set vocabulary contains 260 distinct words out of the entire vocabulary. The Corel30k dataset is of similar property as the Corel5k dataset except that it is substantially larger. It contains 31695 images that are stored with size 384×256 or 256×384. There are 5587 distinct words in the vocabulary. Each image is tagged with 1-5 words, and the average number of words per image is 3.6. In the experiments of Refs. [66] and [61], the dataset is divided into training and testing set with a ratio of 9:1. In the training set, only the words that are used as annotations by at least 10 images are considered. Therefore, the total number of words in the training set vocabulary is 1035. In the testing set, the vocabulary contains 950 words out of the entire vocabulary. The Corel60k dataset contains about 60000 images that are stored with size 384×256 or 256×384. There are 417 distinct words in the vocabulary. The images are assigned to 600 categories, where each category has about 100 images and represents a distinct topic of interest. Each category is tagged with 1-7 words, which describe the category as



(a) Corel5k images (b) Corel30k images (c) Corel60k images

Fig. 4.4. Thumbnails of some randomly selected images from the Corel5k, Corel30k, and Corel 60k datasets.

a whole, but not accurately describe each individual image. In our experiment, for each image, the ground truth tagging words are taken as the words that describe its category. As an example, Fig.4.4 shows the thumbnails of some randomly selected images from the Corel5k, Corel30k, and Corel60k dataset, respectively. As can be seen from Fig.4.4, the datasets contain a wide variety of images, ranging from nature sceneries to historical buildings or people activities, which reflect the diversity of the datasets. Since the application of the proposed SVIA is to tag images taken in daily life other than special fields such as medical or geography, thus the selected datasets are suitable for our performance evaluation.

4.5.2 Experimental settings

Experimental settings are crucial for the performance evaluation. The first decision is on the selection of the training images and the testing images. For the Corel5k and Corel30k datasets, to make a fair comparison, we use the same training and testing sets as that used in [66] and [68], i.e., 90% of the images are used for training and 10% of the images are used for testing. For the Corel60k datasets, we randomly select 90% of the images for training and 10% of the images for testing. In the experiment, to fix the Gaussian bandwidth parameter h described in Eq. 4.5, 15% images of the entire dataset

are further randomly selected from the training dataset. Among the selected images, 10% images of the entire dataset are used for training and 5% images of the entire dataset are used for validating. After the h value is fixed, the entire training image set is then used as a whole to train the support vector described models.

Basically, we use the precision and recall rate on the testing image set to assess the performance of the proposed SVIA. For each word w, denote the number of images tagged by the system by n_s, denote the number of ground truth related images in the testing set by n_t, and denote the number of images correctly annotated by the system by n_c, then the precision and recall rate can be computed by:

$$\text{precision}(w) = \frac{n_c}{n_s}, \quad \text{and} \quad \text{recall}(w) = \frac{n_c}{n_t}. \tag{4.14}$$

We then compute the average precision and recall rates over all the words in the testing set vocabulary to evaluate the performance of the system. Generally, there is a trade-off between precision and recall rate. For the testing images, when the number of words provided by the system increase, the recall rate will usually increase, whereas the precision rate will usually decrease. In the experiment, to compare the precision rates at different levels of recall rates, we change the parameter k (The number of word provided by the system, which was described in Section 4.4.3 from 1 to 15. Moreover, we use the coverage rate to show the generalization ability of the system. The coverage rate is calculated as follows. Denote the number of words with positive recall rate by n^+, denote the number words in the testing set by n, then the coverage rate can be computed by:

$$\text{coverge} = \frac{n^+}{n}. \tag{4.15}$$

4.5.3 Existing systems for comparison

For the purpose of comparison, we select the following systems which have been evaluated for all or some of the selected benchmark datasets.

1. Supervised multiclass labeling (SML) [68]: the SML poses annotation and retrieval as classification problems where each class is defined as the group of database images labeled with a common semantic label. A minimum probability

of error annotation and retrieval is computed by establishing a one-to-one correspondence between semantic labels and semantic classes. Then the images are labeled based on the probabilities.

2. Automatic Linguistic Indexing of Pictures-Real Time (ALIPR) [1]: in ALIPR, the Discrete Distribution (D2-) Clustering method, which is in the same spirit as K-Means for vectors, is developed to group objects represented as bags of weighted vectors. And a generalized mixture modeling technique for non-vector data is developed using the concept of Hypothetical Local Mapping (HLM).
3. Graph Learning Model (GLM) [71]: in GLM, firstly, the image-based graph learning is performed to obtain the candidate annotations for each image. To capture the complex distribution of image data, a nearest spanning chain method is proposed to construct the image based graph. Secondly, the word based graph learning is developed to refine the relationships between images and words to get final annotations for each image. Moreover, two types of word correlations based on web search results are designed.
4. Hybrid Probabilistic Model (HPM) [61]: the HPM integrates low level image features and high level user provided tags to automatically tag images. For images without any tags, HPM predicts new tags based solely on the low level image features. For images with user provided tags, HPM jointly exploits both the image features and the tags in a unified probabilistic framework to recommend additional tags to label the images. Moreover, a collaborative filtering method based on the nonnegative matrix factorization is developed for tackling the data sparsity issue.

The SML, ALIPR, and HPM are probabilistic modeling methods. They perform labeling tasks by computing the joint probabilities for an untagged image being labeled with the annotation words. The GLM is performed by propagating the keywords from the tagged images to the untagged images by visual similarities. These systems have been shown to be successful and can obtain suitable annotation results. Hence, a comparison with them will demonstrate the performance of the proposed SVIA, and will

			
SVIA Tagging	jet, plane, clouds, runway, sky	beach, sand, tree, valley, sky	tree, cat, tiger, forest, bengal
Human Tagging	jet, plane, sky	beach, palm, tree, people	bengal, cat, forest, tiger
			
SVIA Tagging	sky, water, street, buildings, mountain	sky, tree, water, buildings, street	reefs, coral, ocean, fish, water
Human Tagging	buildings, harbor, water, shore	tree, scalpture, street, buildings	coral, fan, ocean, reefs, sea
			
SVIA Tagging	plants, lawn, garden, house, petals	foals, mare, horses, fence, field	
Human Tagging	garden, house, lawn, tree	horse, mare, meadow	
			
SVIA Tagging	plants, leaf, stems, petals, flowers	turn, formula, tracks, straight, prototype	
Human Tagging	flowers, grass, petals	cars, formula, tracks, wall	

Fig. 4.5. Examples of some annotations generated by the SVIA and human tagging on the Corel5k dataset.

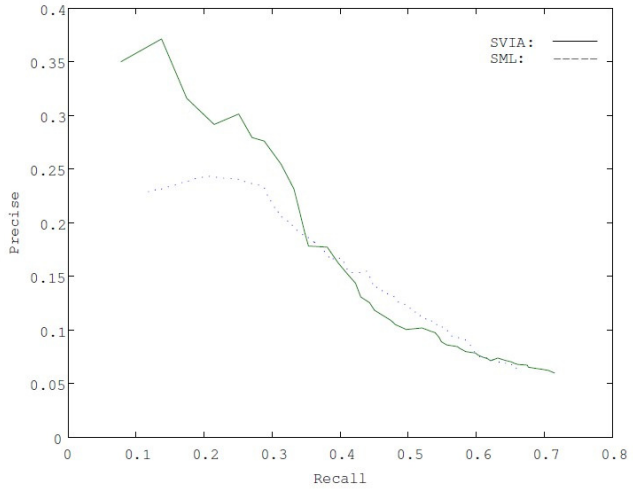


Fig. 4.6. Precision-recall curves for annotation on Corel5k testing dataset using SVIA and SML.

show whether it is better or worse than other systems.

4.5.4 Comparison results

4.5.4.1 Results for the Corel5k data set

This subsection aims to show the results of SVIA for Corel5k dataset. In the training process, on average, the computing time involved in selecting the Gaussian width parameter h was 55.1 seconds, the computing time involved in training the support vector described models was 214.6 seconds. In the tagging process, on average, the computing time was as high as 0.16 seconds for tagging one image. We note that these times are not definitive since the computing time varies when the computer configurations are different. The reason why we show them here is to roughly illustrate the time complexity of the SVIA. Since the images in the Corel5k dataset is tagged with 1-5 words, we basically show the results of the SVIA when the system provides 5 words for each image. Fig.4.5 presents the examples of some annotations generated by the SVIA. The images were randomly selected from the testing image set. Fig.4.6 is a plot of precision-recall curves for annotation on Corel5k testing dataset using SVIA and SML. The curve generated by SML is taken from Ref. [68]. In the experiment, the Gaussian Mixture Model mixed with Discrete Cosine Transform representation, i.e., the

Table 4.1

Results of SVIA and the compared systems for Corel5k testing dataset.

	n_{image}	n_{word}	precise (%)	recall (%)	coverage (%)
SVIA	500	260	25.5	31.3	51.2
SML			23.0	29.0	52.7
GLM			25.3	29.1	50.4
HPM			25.0	28.0	52.3

GMM-DCT representation is adopted. The curve is generated when the dimension of the DCT feature space is 63. Table 4.1 presents the results of SVIA when the system provides 5 words for each images, and the results of studies using SML, GLM, and HPM, respectively. The contents of the table include the number images in the testing set n_{image}, the number of words in the testing set vocabulary n_{word}, the average precision and recall rates over the entire testing set vocabulary (precision %, recall %), and the coverage rate of the system (coverage %). As far as HPM concerned, the system uses user provided tags to enhance the tagging accuracy. In order to present a fair comparison, we list the results of HPM under the Given 0 protocol, i.e., the result of HPM that didn't use any user provided tags as hints. For the ALIPR, the results on Corel5k dataset are not reported in the literature, so they are not included in this subsection.

As can be seen from Fig.4.5, though the SVIA tagging doesn't match the human tagging exactly, it is usually more plausible. Take the second image in the top row as an example, the SVIA provided words are “beach, sand, tree, valley, sky”, and the human provided words are “beach, palm, tree, people”. From the contents of the image, it can be seen that the words “sand, sky” are more plausible than the human provided words. From Fig.4.6, it can be seen that the SVIA curve has the best precision at 0.371, and its precision is superior to that of SML when the recall rate is below 0.352. There are also some levels of recall rates where the SML obtain a better precision, especially when the recall rate is between 0.4 and 0.6. This is due to the fact that the system has to provide

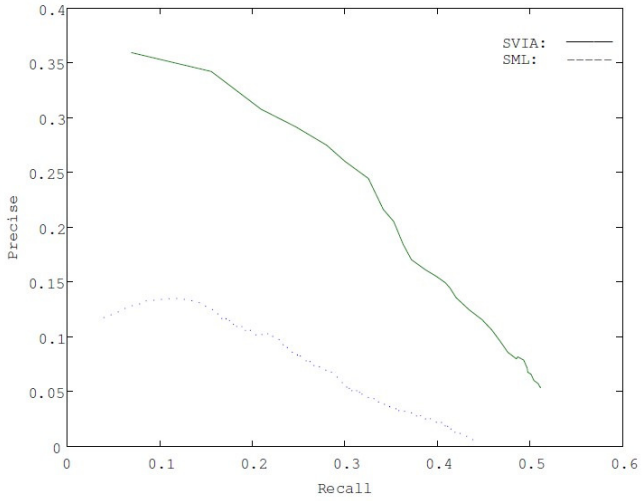


Fig. 4.7. Precision-recall curves for annotation on Corel30k testing data set using SVIA and SML.

more words to obtain a higher recall rate, and the number of correct annotations doesn't increase as the number of system pamrovided word increase. From Table 4.1, it can be seen that the SVIA obtains the best results in terms of recall, precision rates. More specifically, when compared with the previous best results (GLM), the SVIA obtains recall rate 31.3% and precision rate 25.5%, while the GLM obtains recall rate 29.1% and precision rate 25.3%. The coverage rate of SVIA is lower than those of SML and HPM, and higher than that of GLM. Although the result is worse than those of SML and HPM, it is not deteriorated much since it can obtain coverage rate 51.2%.

4.5.4.2 Results for the Corel30k data set

This subsection aims to show the results of SVIA for Corel30k dataset. The experiments that were conducted on Corel5k dataset are repeated on Corel30k dataset. In the training process, on average, the computing time involved in selecting the Gaussian width parameter h was 443.1 seconds, the computing time involved in training the support vector described models was 2167.6 seconds. In the tagging process, on average, the computing time was as high as 0.18 seconds for tagging one image. Fig.4.7 is a plot of precision-recall curves for annotation on Corel30k testing dataset using SVIA and SML. The curve generated by SML is taken from Ref. [68]. In the experiment, similar to the experiments on Corel5k dataset, the GMM-DCT

Table 4.2

Results of SVIA and the compared systems for Corel30k testing dataset.

	n_{image}	n_{word}	precise (%)	recall (%)	coverage (%)
SVIA	500	260	27.5	28.1	41.1
SML			12.0	21.0	44.6
HPM			10.0	19.0	46.2

representation is adopted. The curve is generated when the dimension of the DCT feature space is 128 and the Gaussian mixture model is learned with 3-level hierarchy. Table 4.2 presents the results of SVIA when the system provides 5 words for each images, and the results of studies using SML and HPM, respectively. The item meanings of Table 4.2 are the same as those in Table 4.1. As far as HPM concerned, the same as the experiments on the Corel5k dataset, we list the results of HPM under the Given 0 protocol. Similar to the experiments on Corel5k dataset, the results of ALIPR and GLM are not reported in the literature, so they are not included in this subsection.

It can be seen from Fig.4.7 that the SVIA curve has the best precision at 0.359, and its precision and recall rates are superior to that of SML significantly. From Table 4.2, it can be seen that the SVIA obtains the best results in terms of recall and precision rate. When compared with SML and HPM, the SVIA obtains recall rate 28.1%, and precision rate 27.5%, while the SML obtains recall rate 12.0%, precision rate 21.0% and HPM obtains recall rate 19.0%, precision rate 10.0%. As far as coverage rate concerned, the SVIA obtains coverage rate 41.1%, while the SML and HPM obtain coverage rate 44.6% and 46.2%, respectively. The results demonstrate that the SVIA maintains its scalability as the size of the dataset increases.

4.5.4.3 Results for the Corel60k data set

This subsection aims to show the results of SVIA for Corel60k dataset. The experiments that were conducted on Corel5k and Corel30k datasets were repeated on Corel60k dataset. In the training process, on average, the computing time involved in selecting the Gaussian width parameter h was 1439.2 seconds, the computing time

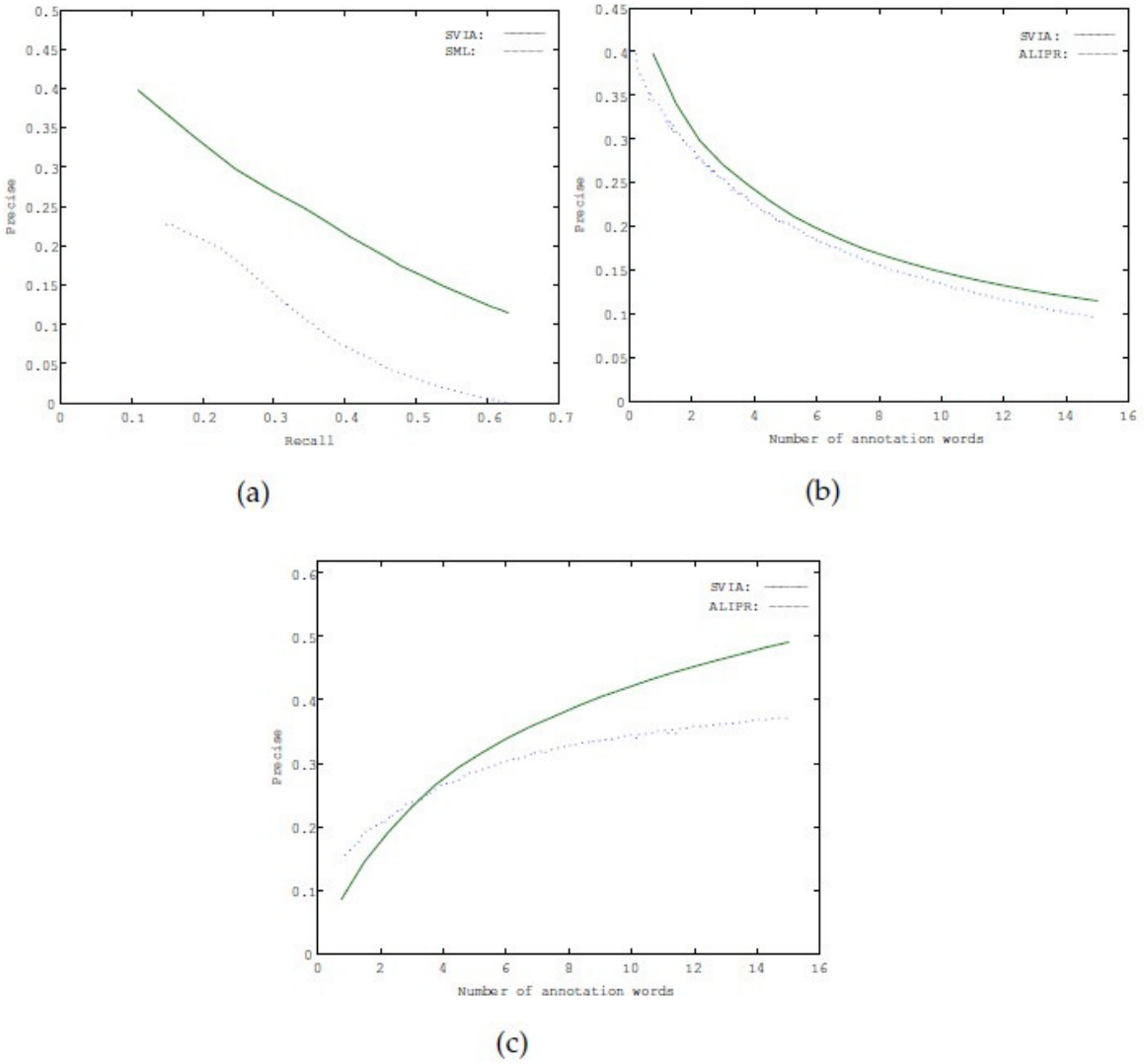


Fig. 4.8. Comparing annotation results of SVIA, SML and ALIPR. (a) Precision-recall curves for annotation using SVIA and SML. (b) Comparing precision rates obtained by SVIA and ALIPR. (c) Comparing recall rates obtained by SVIA and ALIPR.

involved in training the support vector described models was 6972.7 seconds. In the tagging process, on average, the computing time was as high as 0.22 seconds for tagging one image. Fig.4.8 presents the comparing annotation results of SVIA, SML and ALIPR for annotation on Corel testing dataset. Fig.4.8(a) is a plot of precision-recall curves using SVIA and SML. In the experiment of SML, the GMM-DCT representation is adopted. The SML classifiers were learned using 3-level hierarchies. Fig.4.8(b) and (c) compare the results of SVIA and ALIPR in terms of precision and recall rates,

Table 4.3

Results of SVIA and the compared systems for Corel60k testing dataset

	n_{image}	n_{word}	precise (%)	recall (%)
SVIA	60000	417	24.9	26.6
SML			23.6	15.3
ALIPR			22.4	28.7

respectively, where precision and recall rates are shown with the number of words provided by the system ranging from 1 to 15. Table 4.3 presents the results of SVIA when the system provides 5 words for each images, and the results of studies using SML and ALIPR, respectively. The item meanings of Table 4.3 are the same as those in Tables 4.1 and 4.2. The coverage rate of SVIA is 42.6% when the system provides 5 words for each image. Since the coverage rates of SML and ALIPR are not reported in the cited literatures Refs. [68]} and [1], they are not included in Table 4.3. For the GLM and HPM, the results on Corel60k dataset are not reported in the literature, so they are not included in this subsection.

It can be seen from Fig.4.8(a) that the SVIA curve has the best precision at 0.398, and its precision and recall rates are superior to that of SML significantly. From Fig.4.8(b), it can be seen that the curves of SVIA and ALIPR have the same shape, and the SVIA has a higher precision rate. From Fig.4.8(c), it can be seen that the curves of SVIA and ALIPR have the same shape. When the systems provide less than 3 words, the ALIPR has a higher recall rate, and when the system provides more than 3 words, the SVIA has a higher recall rate. From Table 4.3, it can be seen that the SVIA obtains the best results in terms precision rate. The SVIA obtains precision rate 24.9%, while the SML obtains precision rate 23.6%, and ALIPR obtains precision rate 22.4%. As far as recall rate concerned, the SVIA obtains recall rate 26.6%, the SML obtains recall rate 15.3%, and ALIPR obtains recall rate 28.7%. The results demonstrate that SVIA mains it scalability on the Corel60k datasets.

4.5.5 Discussions

The comparison results of SVIA on Corel5k, Corel30k and Corel60k datasets can be summarized and explained as follows.

1. When compared with SML, SVIA obtains a better precision and recall rates on all the Corel5k, Corel30k and Corel60k datasets. The precision-recall curves of Fig.4.6, Fig.4.7 and Fig.4.8(a) show that SVIA has a higher precision rate at most levels of recall rates. The coverage rates of SVIA are lower than those of SML on the Corel5k and Corel30k datasets.
2. When compared with ALIPR on the Corel60k dataset, SVIA obtains a better precision rate as the number of words provided by the system increasing from 1 to 15. ALIPR has a higher recall rate when the system provides less than 3 words, and SVIA has a higher recall rate when the system provides more than 3 words.
3. When compared with GLM on the Corel5k dataset, SVIA obtains a better result in terms of precision, recall and coverage rates.
4. When compared with HPM on the Corel5k and Corel30k datasets, SVIA obtains a better result in terms of precision and recall rates. And the coverage rates of SVIA are lower than those of HPM.

The simulated results indicate that the SVIA may obtain a better result in terms of precision and recall rates, and maintains its stability as the size of the dataset increases. However, the coverage rates are not as high as the compared systems in most cases. This is due to the fact that in the SVIA, the support vector description of cluster is adopted, and its most important advantages is to describe clusters of arbitrary shape. When the cluster in the training dataset contains a large amount of images, the advantage of the SVIA will be exhibited. On the other hand, when the cluster in the training dataset contains less images, the SVIA will lose its advantage, thus obtain a lower coverage rate.

4.6 Conclusion

In this chapter, we aims at proposing an image annotation algorithm based on support vector descriptions of clusters. The main novelty of this chapter is in the

proposed SVC-based approach, which aims at describing the clusters of training images that manually annotated by semantic words. The fact that it can exploit the advantage of SVC for its ability to delineate cluster boundaries of arbitrary shape makes it particularly useful when the training images are not regularly organized. The performance of the proposed algorithm is tested on Corel5k, Corel30k and Corel60k data set. The simulated results validate the effectiveness of the proposed algorithm.

Chapter 5: Conclusions and future perspectives

5.1 Conclusions

Images are major media on the Internet. To mitigate the "semantic gap problem" between the low level images and the high level semantic of documents, image automatic annotation is an imperative but highly challenging task. In this thesis, we aimed at developing non-linear machine learning techniques to address the problems in the CBIA system.

Referring to the parameters in the CBIA system, we proposed a cooperative optimization algorithm. A statistical model is proposed to learn the variable interdependencies among variables. With the variable interdependencies, a decomposition method was proposed to partition the problem into sub-problems so that the variable interdependencies in different sub-problems are minimized. Then the sub-problems were optimized by a cooperative particle swarm optimization framework. We further proposed and proved a theorem that explains the execution process of the proposed algorithm. From the study, we found that the probability for a local optimum solution of a subset being global optimum values is associated with degree of interdependencies of its variables with variables outside the subset. Then another theorem that explains why and how the proposed algorithm works is proposed and proved. The performance of the proposed algorithm was tested on benchmarks from different data sets. Simulated results showed that the proposed algorithm could find the optimal solution for most of the selected test functions. This work has shown that large scale optimization problem can be partitioned into small scale sub-problems and can be optimized cooperatively.

Referring to the data clustering problems in the CBIA system, we proposed a support vector and K-Means hybrid clustering algorithm. A SVC training method was proposed based on theoretical analysis of the Gaussian kernel radius function. An empirical study was conducted to guide better selection of the standard deviation of the Gaussian kernel. A new data clustering algorithm was developed by integrating the merits of both SVC

and K-Means algorithm. The performance of the proposed algorithm was tested on generated data sets with different sizes, generated 2D data set, and data sets taken from the UCI machine learning repository. The results shown that the proposed algorithm compared favorably with existing kernel based clustering algorithms.

We further proposed a support vector based CBIA system. The system contains two major components, the training process and the annotating process. In the training process, clusters of images that manually annotated by semantic words are used as training instances. Images within each cluster are modeled using a support vector based method. The fact that it can exploit the advantage of SVC for its ability to delineate cluster boundaries of arbitrary shape makes it particularly useful when the training images are not regularly organized. In the annotating process, the system exploits the distance from the image to the support vector described models and the word to word correlations in a probabilistic framework to predict annotation words. The performance of the proposed algorithm is tested on Corel5k, Corel30k, and Corel60k data sets. The simulated results show the performance of the proposed system.

5.2 Future perspectives

Future work of this thesis can take many directions. Firstly, the parameters used in the CBIA system described in Chapter 4 are selected empirically. The incorporation of the CPSO-SL described in Chapter 2 to the CBIA system may produce more desirable results. Thus one direction of future is to integrate the CPSL-SL with CBIA system so that better parameter settings can be obtained. This includes the problems of how to formulate the optimization problems, how to design the PSO operators, and how to define the objective functions, etc. Secondly, the support vector and K-Means data clustering algorithm described in Chapter 3 is effective for data sets with less than 10000 data points. However, whether it can deal with large scale data sets is still a question to be answered. On the other hand, in the CBIA system described in Chapter 4, the training images are annotated manually, which is known to be tedious and labor intensive. Thus, one direction of future work is to develop data clustering algorithm for large scale data sets so that the training images of the CBIA system can be generated

automatically by computer. Finally, a review of literatures suggests that the accuracies of CBIA systems are less than 50% in most case. Obviously, it is not enough for the user demand. Thus, the third direction of current research is to improve the accuracy of the system. The orientations are multi-fold. Shape information can be utilized to improve the training and annotating process. Better and larger amounts of training images may produce more robust models. Contextual information may also help in the modeling process. And the system can be integrated with other retrieval methods to improve usability.

References

[1] J. Li and J. Wang, "Real-Time Computerized Annotation of Pictures," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 6, pp. 985-1002, 2008.

[2] A. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, "Content-Based Image Retrieval at the End of the Early Years," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 12, pp. 1349-1380, 2000.

[3] N. Vasconcelos and M. Kunt, "Content-Based Retrieval from Image Databases: Current Solutions and Future Directions," Procedings 2001 International Conference on Image Processing, vol. 3, pp. 6-9, 2001.

[4] R. Datta, D. Joshi, J. Li and J.Z. Wang, "Image Retrieval: Ideas, Influences, and Trends of the New Age," ACM Transactions on Computing Surveys, vol. 40, no. 2, pp. 1-60, 2008.

[5] C. Dorai and S. Venkatesh, "Bridging the Semantic Gap with Computational Media Aesthetics," IEEE Multimedia, vol. 10, no. 2, pp. 15-17, 2003.

[6] A. Ben-Hur, D. Horn, H. Siegelmann and V. Vapnik, "A Support Vector Clustering Method," Proceedings of 15th International Conference on Pattern Recognition, Barcelona, Spain, vol. 2, pp. 724-727, 2000.

[7] A. Ben-Hur, D. Horn, H. Siegelmann and V. Vapnik, "A support Vector Method for Clustering," Advances in Neural Information Processing Systems, vol. 13, pp. 367-373, 2001.

[8] A. Ben-Hur, D. Horn, H. Siegelmann and V. Vapnik, "Support Vector Clustering," Journal of Machine Learning Research, vol. 2, pp. 125-137, 2001.

[9] V. Vapnik, "An Overview of Statistical Learning Theory," IEEE Transactions on Neural Networks, vol. 10, no. 5, pp. 1191-1199, 1999.

[10] A. Torn and A. Zilinskas, Global Optimization, Springer-Verlag, New York, 1989.

[11] W. Chu, X. Gao and S. Sorooshian, "A new evolutionary search strategy for global optimization of high-dimensional problems," Information Sciences, vol. 181, no. 22,

pp. 4909-4927, 2011.

[12] A. Ghosha, S. Das, A. Chowdhurya, and R. Giria, "A improved diferential evolution algorithm with fitness-based adaption of the control parameters," Information Sciences, vol. 181, no. 18, pp. 3749-3765, 2011.

[13] D. Jia, G. Zheng, and M. Khan, "An effective memetic differential evolution algorithm based on chaotic local search," Information Sciences, vol. 181, no. 16, pp. 3175-3187, 2011.

[14] C. Lee, X. Yao, "Evolutionary programming using mutations based on the Lêvy probability distribution," IEEE Transactions on Evolutionary computation, vol. 8, no. 1, pp. 1-13, 2004.

[15] J. Liang, A. Qin, P. Suganthan, S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, pp. 281-295, 2006.

[16] E. Mezura-Montes, M Miranda-Verela, R. Gomez-ramon, "Differential evolution in constrained numerical optimization: an empirical study," Information Sciences, vol. 180, no. 22, pp. 4223-4262, 2010.

[17] Y. Ong, A. Keane, "Meta-Lamarckian learning in memetic algorithms," IEEE Transactions on Evolutionary Computation, vol. 8, no. 2, pp. 99-110, 2008.

[18] Z. Tu, Y. Lu, "A robust stochastic genetic algorithm (StGA) for global numerical optimization," IEEE Transactions on Evolutionary Computation, vol. 2, no. 3, pp. 456-470, 2004.

[19] X. Yao, Y. Liu, G.M. Lin, "Evolutionary programming made faster," IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 82-102, 1999.

[20] Q. Yuan, F. Qian, W. Du, "A hybrid genetic algorithm with the Baldwin effect," Information Sciences, vol. 180, no. 5, pp. 640-652, 2010.

[21] C. Zhang, Y. Zhang, "Scale-free fully informed particle swarm optimization algorithm," Information Sciences, vol. 180, no. 20, pp. 4550-4568, 2011.

[22] M. Potter, The design and analysis of a computational model of cooperative coevolution, Ph. D Thesis, George Mason University, 1997.

[23] M. Potter and K. De Jong, “A cooperative coevolutionary approach to function optimization,” in: Proceedings of the Third Conference on Parallel Problem Solving from Nature, Jerusalem, Israel, vol. 2, 1994, pp. 249-257.

[24] M. Potter and K. De Jong, “Cooperative coevolution: an architecture for evolving coadapted subcomponents”, Evolutionary Computation, vol. 8, no. 1, pp. 1-29, 2000.

[25] Y. Liu, X. Yao, Q. Zhao, T. Higuchi, “Scaling up fast evolutionary programming with cooperative coevolution,” in: Proceedings of the 2001 Congress on Evolutionary Computation, Piscataway, NJ, USA, 2001, pp. 1101-1108.

[26] F.V.D. Bergh, A.P. Engelbrecht, “Cooperative learning in neural networks using particle swarm optimizers”, South African Computer Journal, vol. 26, pp. 84-90, 2000.

[27] F.V.D. Bergh, A.P. Engelbrecht, “A cooperative approach to particle swarm optimization,” IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, pp. 225-239, 2004.

[28] X. Li, X. Yao, “Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms,” in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, 2009, pp. 1546-1553.

[29] Z. Yang, K. Tang, X. Yao, “Differential evolution for high dimensional function optimization,” in: Proceedings of the 2007 Congress on Evolutionary Computation, Singapore, 2007, pp. 3523-3530.

[30] Z. Yang, K. Tang, X. Yao, “Multilevel cooperative coevolution for large scale optimization,” in: Proceedings of the 2008 Congress on Evolutionary Computation, Hong Kong, 2008, pp. 1663-1670.

[31] Z. Yang, K. Tang, X. Yao, “Large scale evolutionary optimization using cooperative coevolution,” Information Sciences, vol. 178, no. 15, pp. 2985-2999, 2008.

[32] K. Weicker, N. Weicker, “On the improvement of coevolutionary optimizers by learning variable interdependencies,” in: Proceedings of the 1999 Congress on Evolutionary Computation, Washington, D.C. USA, 1999, pp. 1627-1632.

[33] T. Ray, X. Yao, “A cooperative coevolutionary algorithm with correlation based

adaptive variable partitioning," in: Proceedings of the 2009 Congress on Evolutionary Computation, Trondheim, Norway, 2009, pp. 983-989.

[34] R. Eberhart, Y. Shi, "Particle swarm optimization: developments, applications and resources," in: Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, 2001, pp. 81-86.

[35] J. Kennedy, R. Eberhart, "Particle swarm optimization," in: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, vol. 4, 1995, pp. 1942-1948.

[36] F.V.D. Bergh, A.P. Engelbrecht, "Cooperative learning in neural networks using particle swarm optimizers," South African Computer Journal, vo. 26, pp. 84-90, 2000.

[37] H. Ge, L. Sun, Y. Liang, F. Qian, "An effective PSO and AIS-Based hybrid intelligent algorithm for job shop scheduling," IEEE Transactions on Systems, Man, and Cybernetics. Part A: Systems and Humans, vol. 38, no. 2, pp. 358-368, 2008.

[38] Y. Shi, R. Eberhart, "A modified particle swarm optimizer," in: Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, USA, 1998, pp. 69-73.

[39] F.V.D. Bergh, A.P. Engelbrecht, "A cooperative approach to particle swarm optimization," IEEE Transactions on Evolutionary Computation, vol. 10, no. 3, pp. 225-239, 2004.

[40] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization," Technical Report, Nanyang Technological University, Singapore, 2005. <http://www.ntu.edu.sg/home/EPNSugan>.

[41] S. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions," Biosystems, vol. 39, pp. 263–278, 1996.

[42] R. Eberhart, Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in: Proceedings of the IEEE Congress on Evolutionary Computation, San Diego, USA, 2000, pp. 84-88.

[43] E.D. Taillard, "Few guidelines for analyzing methods," in: The Sixth Metaheuristics

International Conference, Vienna, Austria, 2005.

[44] D.H. Wolpert, W.G. Macready, “No free lunch theorems for optimization,” IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82, 1997.

[45] R. Xu and D. Wunsch I, “Survey of clustering algorithms,” IEEE Transactions on Neural Networks, vol. 16, no. 3, pp. 645-678, 2005.

[46] J.H. Chiang and P.Y. Hao, “A new kernel-based fuzzy clustering approach: support vector clustering with cell growing,” IEEE Transactions on Fuzzy System, vol. 11, no. 4, pp. 518-527, 1999.

[47] T. Ban and S. Abe, “Spatially chunking support vector clustering algorithm,” in: Proceedings of 2004 International Joint Conference on Neural Networks, vol. 1, pp. 413-418, 2004.

[48] J. Lee and D. Lee, “An improved cluster labeling method for support vector clustering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 3, pp. 461-464, 2005.

[49] F. Camastra and A. Verri, “A novel kernel method for clustering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 5, pp. 801-805, 2005.

[50] J. Lee, and D. Lee, “Dynamic characterization of cluster structures for robust and inductive support vector clustering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no.11, pp. 1869-1874, 2006.

[51] J.S. Wang and J.C. Chiang, “A cluster validity measure with outlier detection for support vector clustering,” IEEE Transactions on System, Man and Cybernetics, Part B: Cybernetics, vol. 38, no.1 ,pp. 78-89.

[52] J. Platt, “Fast training of support vector machines using sequential minimal optimization,” In Advances in Kernel Methods-Support Vector Learning, B. Schölkopf, C.J.C. Burges, and J.J. Smola, editors, 1999.

[53] R. Fletcher, Practical methods of optimization, Wiley-Interscience, Chichester, 1987.

[54] J. Yang, V. Estivill-Castro, and S.K. Chalup, “Support vector clustering through

proximity graph modeling," in: Proceedings of the 9th International conference on Neural Information Processing, pp. 898-903, 2002.

[55] J.A. Hartigan and M.A. Wong, "A K-Means clustering algorithm," Applied Statistics, vol. 28, pp. 100-108, 1979.

[56] E. Osunna, R. Freund and F. Girosi, "An improved training algorithm for support vector machines," Neural Networks for Signal Processing VII. Proceedings of the 1997 Workshop, pp. 276-285, 1997.

[57] P. Hall and B.A. Turlach, "Reducing bias in curve estimation by use of weights," Computational Statistics & Data Analysis, vol. 30, no. 1, pp. 67-86, 1999.

[58] B.W. Silverman, Density estimation for statistics and data analysis, Monographs on statistics and applied probability, London: Champman and Hall, 1986.

[59] A. Bhattacharyya, "On the measure of divergence between two statistical populations defined by probability distributions," Bulletin of the Calcutta Mathematical Society, vol. 35, pp. 99-109, 1943.

[60] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," 2001, Software Available at: http://www.csie.ntu.edu.tw/cjlin/libsvm.

[61] N. Zhou, W.K. Cheung, G. Qiu, and X. Xue, "A hybrid probabilistic model for unified collaborative and content based image tagging," IEEE Transactions on pattern analysis and machine intelligence, vol. 33, no. 7, pp. 1281-1294, 2011.

[62] J. Luo, A. Savakis, and A. Signhal, "A bayesian network based framework for semantic image understanding," Pattern Recognition, vol. 38, no. 6, pp. 1331-1338, 2005.

[63] J. Fan , Y. Gao, and H. Luo, "Integrating concept ontology and multitask learning to achieve more effective classifier training for multilevel image annotation," IEEE Transactions on Image Processing, vol. 17, no. 3, pp. 407-426, 2008.

[64] J. Fan, Y. Shen, C. Yang, and N. Zhou, "Structured max-margin learning for inter-related classifier training and multi-label image annotation," IEEE Transactions on Image Processing, vol. 20, no. 3,pp. 837-854, 2011.

[65] J. Su, C. Chou, C. Lin, and V. Tseng, "Effective semantic annotation by

image-to-concept distribution model," IEEE Transactions on Multimedia, vol. 13, no. 3, pp. 530-538, 2011.

[66] P. Duygulu, K. Barnard, J.F.G. de Freitas, and D.A. Forsyth, "Object recognition as machine translation: learning a lexicon for a fixed image vocabulary," Proceedings of Seventh European Conference on Computer Vision, pp. 349-354, 2002.

[67] J. Li and J. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, no. 9, pp. 1075-1088, 2003.

[68] G. Carneiro, A. Chan, P. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 3, pp. 394-410, 2007.

[69] S. Zhu and Y. Liu, "Semi-supervised learning model based efficient image annotation," IEEE Signal Processing Letters, vol. 16, no. 11, pp. 989-992, 2009.

[70] Z. Lu and H. Ip, "Spatial Markov kernels for image categorization and annotation," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 41, no. 4, pp. 976-989, 2011.

[71] J. Liu, M. Li, Q. Liu, H. Lu, and S. Ma, "Image annotation via graph learning," Pattern Recognition, vol. 42, no. 2, pp. 218-228, 2009.

[72] J. Tang, H. Li, G. Qi, and T. Chu, "Image annotation by graph based inference with integrated multiple/single instance representations," IEEE Transactions on Multimedia, vol. 12, no. 2, pp. 131-141, 2010.

[73] F. Yu and H. Ip, "Semantic content analysis and annotation of histological images," Computers in Biology and Medicine, vol. 38, no. 6, pp. 635-649.

[74] Flickr, http://www.flickr.com

[75] C. Halaschek-Wiener, J. Golbeck, A. Schain, M. Grove, B. Parsia, and J.-Hendler, "Photostuff- an image annotation tool for the semantic web," Proceedings of the Fourth International Semantic Web Conference, 2005.

[76] X.J. Wang, L. Zhang, X. Li, and W.Y. Ma, "Annotating images by mining image search results," IEEE Transactions on Pattern Analysis and Machine Intelligence,

vol. 30, no. 11, pp. 1919-1932, 2008.

[77] R. Wong and C. Leung, “Automatic semantic annotation of real-world web images,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 11. pp. 1933-1944, 2008.

[78] G. Qiu, “Indexing chromatic and achromatic patterns for content-based color image retrieval,” Pattern Recognition, vol. 35, pp. 1675-1685, 2002.

[79] V. Vapnik, Statistical learning theory, Wiley, New York, 1998.

Appendix

Detailed descriptions of Generalized penalized functions

$$f_5(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2 \cdot [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(y_i, 10, 100, 4)$$

$$f_6(x) = \frac{1}{10}\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2 \cdot [1 + 10\sin^2(3\pi x_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

List of publications

B.1 Papers in referred journals

[1] L. Sun, S. Yoshida, X. Cheng, and Y. Liang, "A Cooperative Particle Swarm Optimizer with Statistical Variable Interdependence Learning", Information Sciences, vol.186, no.1, pp.20-39.

[2] L. Sun, S. Yoshida, and Y. Liang, "A Support Vector and K-Means Based Hybrid Intelligent Data Clustering Algorithm", IEICE Transactions on Information and Systems, vol.E94-D, no.11, pp.2234-2243, 2011.

[3] L. Sun, S. Yoshida, H. Ge, and Y. Liang, "Support Vector Description of Clusters for Content-Based Image Annotation", IEEE Transactions on Pattern Analysis and Machine Intelligence. (will be submitted)

B.2 Paper in referred conferences

[4] L. Sun, S. Yoshida, and Y. Liang, "A Novel Support Vector and K-Means Based Hybrid Clustering Algorithm", Proceedings of the 2010 IEEE International Conference on Information and Automation, Haerbin, China, pp.126-130, 2010.

[5] L. Sun, S. Yoshida, and Y. Liang, "Cooperative Particle Swarm Optimizer for Large Scale Numerical Optimization", Joint 5th International Conference on Soft Computing and Intelligent Systems and 11th International Symposium on Advanced Intelligent Systems, Okayama, Japan, 2010.

[6] L. Sun, S. Yoshida, and Y. Liang, "Support Vector Description of Clusters for Image Annotation", International Workshop on Advanced Computational Intelligence and Intelligent Informatics, Suzhou, China, 2011. (Session Best Presentation Award)

B.3 Paper in other journals

[7] H. Ge, L. Sun, Y. Liang, and F. Qian, "An Effective PSO and AIS-Based Hybrid Intelligent Algorithm for Job-Shop Scheduling", IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, vol.38, no.2, pp.358-368, 2008.

[8] J. Yang, L. Sun, H. P. Lee, Y. Qian, and Y. Liang, "Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems", Journal of Bionic Engineering,

vol.5, no.2, pp.111-119, 2008.

[9] J. Tang, S. Le, L. Sun, X. Yan, M. Zhang, J. Macleod, B. Leroy, N. Northrup, A. Ellis, T. Yeatman, Y. Liang, M. Zwick, and S. Zhao, “Copy number abnormalities in sporadic canine colorectal cancers”, Genome Research, vol.20, no.3, pp.341-350, 2010.

[10] L. Sun, X. Cheng, and Y. Liang, “Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function”, International Journal of Intelligent Information Processing, vol.1, no.2, pp.65-77, 2010

[11] L. Sun, X. Cheng, and Y. Liang, “A Cooperative Lamarckian Evolutionary Algorithm for Numerical Optimization”, International Journal of Computational Intelligence Systems. (Under Review)

B.4 Paper in other conferences

[12] H. Ge, L. Sun, and Y. Liang, “Solving Job-Shop Scheduling Problems by a Novel Artificial Immune System”, AI2005: Advances in Artificial Intelligence, Lecture Notes in Computer Sciences, Sydney, Australia, vol.3809, pp.839-842, 2005.

[13] L. Sun, F. Melgani, S. Yoshida, and Y. Liang, “Application of GA and SVM Based Hybrid Algorithm to the Classification of Power-Quality Disturbances”, 3rd International Conference on Power Electronics and Intelligent Transportation System, Shenzhen, China, pp.207-212, 2010.

[14] L. Sun, S. Yoshida, Y. Liang, “A Particle Swarm Optimizer for the Training of Support Vector Clustering,”, International Workshop on Information Technology, Kochi, Japan, 2010.