Contents lists available at ScienceDirect

Pattern Recognition

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

Fast support-based clustering method for large-scale problems

Kyu-Hwan Jung [a], Daewon Lee [b], Jaewook Lee [a,*]

[a] Department of Industrial and Management Engineering, Pohang University of Science and Technology, 790-784 Pohang, Kyungbuk, South Korea
[b] School of Industrial Engineering, University of Ulsan, P.O. Box 18, Ulsan 680-749, South Korea

ARTICLE INFO

Article history:
Received 26 February 2009
Received in revised form
5 December 2009
Accepted 10 December 2009

Keywords:
Large-scale problem
Kernel methods
Support vector clustering
Cluster labeling
Dynamical system

ABSTRACT

In many support vector-based clustering algorithms, a key computational bottleneck is the cluster labeling time of each data point which restricts the scalability of the method. In this paper, we review a general framework of support vector-based clustering using dynamical system and propose a novel method to speed up labeling time which is log-linear to the size of data. We also give theoretical background of the proposed method. Various large-scale benchmark results are provided to show the effectiveness and efficiency of the proposed method.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

Recently, support-based clustering methods, for example, the support vector clustering (SVC) algorithms, have attracted many researchers and successfully applied to solve difficult and diverse clustering or outlier detection problems [1–7]. They have some advantages over other clustering algorithms for their ability to generate cluster boundaries of arbitrary shape and to deal with outliers.

The conventional SVC algorithm by [3] consists in general of two main steps: SVM training step to estimate a support function [2,8] and cluster labeling step to assign each data point to its corresponding cluster. Because the time complexity of the latter cluster labeling step is $O(N^2m)$, here N is the number of data points, and $m \ll N$ is the number of sampling points on each edge, it takes most of the computation time for the entire SVC clustering process. Thus, for large-scale problems such as image pixel clustering, or even middle-scale problems, it is crucial to enhance the labeling speed. The existing labeling strategies for this, which will be briefly reviewed in Section 2, include proximity graph-based [9], divide and conquer-based [10,11], and equilibrium-based approach [5–7]. However, the former two approaches have to compensate the clustering accuracy for speed up. Although the latter equilibrium-based approach dominates most of the existing ones in labeling speed without compensating the accuracy, it is still confined to be used in moderate size problems as the former two approaches are.

To break through this bottleneck, in this paper, we propose a *fast support based clustering* method that significantly accelerates the feature space analysis by utilizing the concept of *dynamical consistency*. We show that the proposed framework significantly reduces the complexity of labeling step while maintaining advantageous characteristics of equilibrium-based approaches. We also show that the method can achieve further speed up by using intrinsic characteristic of dynamical system in the distributive computing manner and the inductivity of equilibrium-based approaches can also be enhanced by the proposed method.

The organization of the paper is as follows. The existing labeling methods for support-based clustering are briefly reviewed in Section 2. Section 3 presents the proposed method with theoretical analysis, and implementation details including algorithms and discussions about complexity are provided in Section 4. In Section 5, simulations on benchmark problems are conducted to illustrate the efficiency of the proposed method and applications to image segmentation are also presented.

2. Backgrounds on support-based clustering

2.1. Estimating a support function

A *support function* is defined as a positive scalar function $f : \Re^n \to \Re^+$ where a level set of f estimates a support of a data distribution. It can normally be decomposed into several disjoint connected sets as

$$L_f(r) = \{\mathbf{x} \in \Re^n : f(\mathbf{x}) \leq r\} = C_1 \cup \cdots \cup C_m \quad (1)$$

* Corresponding author.
E-mail address: jaewookl@postech.ac.kr (J. Lee).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.12.010

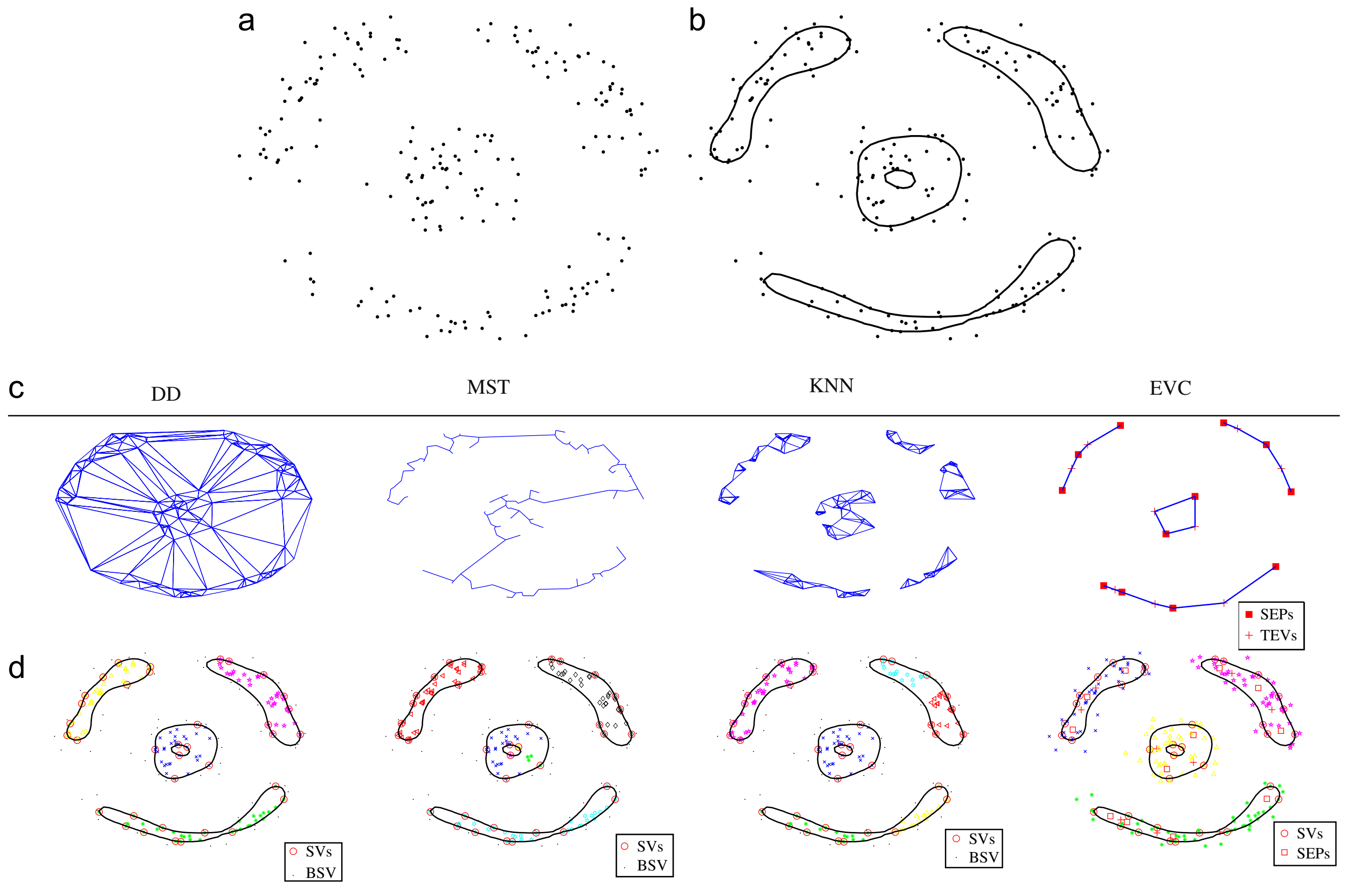

Fig. 1. Support-based clustering results by using the existing labeling methods. (a) Data points. (b) Cluster boundaries by $L_f(r)$. (c) Graphs used for constructing adjacency matrix. (d) Cluster labels.

where $C_i, i=1,\ldots,m$ are disjoint connected sets corresponding to different clusters and m is the number of clusters determined by f. See Fig. 1(b). A support function in the conventional SVC methods is generated by the support vector domain description (SVDD) method (or one-class SVM) [2,8]. The main idea of SVDD is to map data points to a high dimensional feature space and to find a sphere with minimal radius that contains most of the mapped data points in the feature space. This sphere, when mapped back to the data space, can separate into several components, each enclosing a separate cluster of points.

By optimizing the SVDD model with gaussian kernel $k(\mathbf{x}_i,\mathbf{x}_j)=e^{-q\|\mathbf{x}_i-\mathbf{x}_j\|^2}$, the trained support function defined by the squared radial distance of the image of $\mathbf{x}$ from the sphere center is given by

$$f(\mathbf{x}) = 1-2\sum_{j\in J}\alpha_j k(\mathbf{x},\mathbf{x}_j)+\sum_{i,j\in J}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{2}$$

where $\mathbf{x}_j$ and α_j for $j\in J$ are support vectors and their corresponding coefficients respectively.

However, we notice here that application of the proposed fast clustering framework is not restricted to conventional support function generated by SVDD but to any type of support functions that are empirically trained from data. One of them is Gaussian process support function generated by variance function of Gaussian process [7],

$$f(\mathbf{x}) = \mathbf{k}(\mathbf{x})^T\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}) \tag{3}$$

where $\mathbf{K}$ is a positive definite covariance matrix with elements $K_{ij}=K(\mathbf{x}_i,\mathbf{x}_j;\Theta)$ and $\mathbf{k}(\mathbf{x})=(K(\mathbf{x},\mathbf{x}_1),\ldots,K(\mathbf{x},\mathbf{x}_N))^T$. One commonly used covariance function is

$$K(\mathbf{x}_i,\mathbf{x}_j;\Theta) = v_0\exp\left\{-\frac{1}{2}\sum_{m=1}^{n} l_m(x_i^m-x_j^m)^2\right\}+v_1+\delta_{ij}v_2 \tag{4}$$

where $v_0,v_1,v_2,l_1,\ldots,l_n$ are hyper-parameters that are determined by model selection techniques. In addition, Parzen window or its sparse version can be also used as a support function, which are called mean-shift methods [25].

2.2. Cluster assignment

The cluster description in (1) itself does not directly assign to each data point a cluster index. Because data space contours cannot be explicitly constructed due to the nature of the feature space mapping, determining whether a pair of $\mathbf{x}_i$ and $\mathbf{x}_j$ is in the same contour is problematic. To do this, the SVC uses an indirect strategy that relies on the fact that any path connecting two data points in different contours must exit the contours in data space, which is equivalent that the image of the path in feature space exits the minimum enclosing sphere [3].

Until now, several methods for SVC cluster labeling are developed. Originally, [3] proposes Complete Graph (CG) strategy. CG strategy makes adjacency matrix A_{ij} between pairs of points $\mathbf{x}_i$ and $\mathbf{x}_j$ as follows

$$A_{ij} = \begin{cases} 1 & \text{if, for } \forall\mathbf{y} \text{ on the line segment connecting } \mathbf{x}_i \text{ and } \mathbf{x}_j, \quad f(\mathbf{y})\le r \\ 0 & \text{otherwise} \end{cases}$$

Because CG strategy of [3] has a highly intensive complexity ($O(N^2m)$) due to its full pair of checking, it is inefficient to large-scale data. To reduce the complexity of CG method, proximity graph methods are proposed by [9]. They construct appropriate proximity graphs to model a data set, in which vertices are data points and edges connect pairs of points to model their proximity and adjacency. And then they check on only the edges on the proximity graphes. Proximity graphes considered by [9] are delaunay diagram (DD), minimum spanning tree (MST), *k*-nearest neighbor (KNN). Fig. 1(c) shows the proximity graph constructed by above 3 methods. Though the proximity graph methods can significantly reduce the cluster labeling time (usually $O(N)$ or $O(N\log N)$), but fails frequently in labeling the clusters correctly and their time complexity grows exponentially as the dimension of the data points increases. Fig. 1(d) shows that MST and KNN uncorrectly spilt a cluster into two clusters compared to their own cluster boundaries (Fig. 1(b)). It is because their proximity graph missed some important edge connections.

To overcome the difficulties of the proximity-graph based approach, [5–7] proposed equilibrium vector-based cluster labeling (EVC) methods based on a topological property of a trained support function. EVC consists of two phases. The first phase decomposes a given data set into a small number of disjoint groups with the following dynamical system associated with support function f

$$\frac{d\mathbf{x}}{dt} = -G(\mathbf{x})^{-1}\nabla f(\mathbf{x}) \tag{5}$$

where $G(\mathbf{x})$ is a positive definite symmetric matrix for all $\mathbf{x}\in\Re^n$ (such G is called a Riemannian metric on $\Re^n$). The existence of a unique solution (or trajectory) $\Phi(\cdot,\mathbf{x}):\Re\to\Re^n$ for each initial condition $\Phi(0,\mathbf{x})$ can be guaranteed under the twice differentiability of function f with bounded norm of ∇f. Here $\Phi(t,\mathbf{x})$ is a function of time t and starting point $\mathbf{x}$ associated with the dynamical system (5). A state vector $\bar{\mathbf{x}}$ satisfying the equation $\nabla f(\bar{\mathbf{x}})=0$ is called an equilibrium vector of system (5). We say that an equilibrium vector $\bar{\mathbf{x}}$ of (5) is hyperbolic if the Jacobian matrix of ∇f at $\bar{\mathbf{x}}$ has no zero eigenvalues. A hyperbolic equilibrium vector is called (i) a (asymptotically) stable equilibrium vector (or a SEV) if all the eigenvalues of its corresponding Hessian are positive, or (ii) an unstable equilibrium vector (or an UEV) otherwise. Notice that the SEVs correspond to the local minima of the support function f.

One important concept introduced in EVC for inductive learning is a *basin cell* associated with system (5), which will be identified as the basic group consisting of similar objects. Formally, the *basin cell* of a SEV, $\mathbf{s}$, is defined as the closure of the set of all the points converging to $\mathbf{s}$ when process (5) is applied, i.e.,

$$\overline{A(\mathbf{s})} := \mathrm{cl}\{\mathbf{x}\in\Re^n : \lim_{t\to\infty}\Phi(t,\mathbf{x})=\mathbf{s}\}$$

The boundary of the basin cell defines the *basin cell boundary*, denoted by $\partial\overline{A(\mathbf{s})}$.

One nice property for the basin cell is that the whole data space can be decomposed into several separate basin cells of the SEVs under some mild conditions [12], i.e.,

$$\Re^n = \bigcup_{i}^{M}\overline{A(\mathbf{s}_i)} \tag{6}$$

where $\{\mathbf{s}_i : i=1,\dots,M\}$ is the set of the SEVs of the system (5). Therefore we can partition the whole sample space into basin cells. Because each data point converges almost surely to one of the SEVs when system (5) is applied, we can easily identify a basin cell to which a data point belongs by its corresponding SEV. And then, in the second phase, only SEVs are labeled with the adjacency matrix as in [5] or locating the transition equilibrium vectors as in [6], which results in labeling the entire data space. See the last column in Fig. 1(d). Moreover it provides us with a way to extend each cluster C_i in (1) to an enlarged cluster $\tilde{C}_i$ given by

$$\tilde{C}_i = \bigcup_{k=1}^{l_i}\overline{A(\mathbf{s}_{i_k})}\supset C_i \tag{7}$$

where $\mathbf{s}_{i_k}, k=1,\dots,l_i$ is the set of all the SEVs in a cluster C_i. As a result, the whole data sample space can be partitioned into separate enlarged clustered domains, i.e.,

$$\Re^n = \tilde{C}_1\cup\cdots\cup\tilde{C}_m \tag{8}$$

from which we can assign a cluster label to unknown data points as well as bounded support vectors that are located outside of the cluster boundaries, which enable us to conduct inductive clusterings (See Fig. 1(d)).

3. Fast support vector-based clustering using dynamical consistency

From a computational point of view, labeling the SEVs can be accomplished in a moderate computing time regardless of the training sample size because the number of SEVs are normally very small compared with that of training data. However, identifying the corresponding SEVs of all N training data points for final cluster labeling step requires intensive computing time especially when N is very large because the computational complexity for finding a SEV starting from each training point involves the minimization of support function with $O(N)$ cost, thereby leading to the quadratic $O(N^2)$ cost in total. The latter fact makes the existing algorithms for equilibrium-based clustering infeasible when applied to large-scale sample data such as image segmentation. To overcome this drawback, in this section, we propose a novel method that expedites the total labeling process time for all the data points by sequentially grouping objects of similar dynamical behavior.

3.1. Initial stage: constructing small balls

Most of the clustering algorithms based on the *low density separation principle* utilize similarity measures such as metric norms and group the data points close to each other with respect to this measure into the same cluster. To expedite the clustering process, in the initial stage, we decompose the whole data sample, $\mathcal{D}=\{\mathbf{x}_i : i=1,\dots,N\}$, into separate small balls, say

$$B_\rho(\mathbf{c}_j^0) := \{\mathbf{x}\in\mathcal{D} : \|\mathbf{x}-\mathbf{c}_j^0\|\le\rho\}$$

where $\rho>0$ is a pre-specified merging parameter and assign the data points inside each ball to the same label as that of initial centers $\mathbf{c}_j^0, i=1,\dots,N_c$. Notice here that because the scale of data set varies, we always normalize the scale of dataset and ρ is chosen in this normalized space. Each ball $B_\rho(\mathbf{c}_i^0)$ then falls into one of the following categories:

(i) (Case I): The ball is inside the enlarged cluster $\tilde{C}_i$; in this case, the whole data points inside the ball approach their corresponding SEVs when process (5) is applied and the obtained SEVs may be different depending on whether the ball intersects basin cell boundary or not. However, all these SEVs belong to the same cluster, say $\tilde{C}_i$ [6], and so we can get the same clustering result by locating only the SEV corresponding to $\mathbf{c}_i^0$ and assigning other data points inside the ball to the same cluster label as that of $\mathbf{c}_i^0$ without applying process (5) to these remaining points.

(ii) (Case II:) The ball intersects the enlarged cluster boundary; in this case, some of the data points inside the ball can approach to SEVs belong to difference clusters from original one. However, the constructed support function is based on the low density separation principle, the intersecting region is normally sparse, so few or no data points belong to this ball in a separable clustering problems, thereby leading to a negligible labeling error. Even for a nonseparable clustering problems, the data points near the boundary are inherently very hard to assign correct labels even in the original methods and usually the performance difference is marginal as shown in the empirical experiments.

3.2. *Main stage: merging small balls*

Now let us define $\mathcal{R}^n = \Re^n \backslash (\bigcup_i^M \partial \overline{A(\mathbf{s}_i)})$. Then $\mathcal{R}^n$ is an open set with $\overline{\mathcal{R}^n} = \Re^n$, which implies that almost all the points fall into the set $\mathcal{R}^n$. Because system (5) is completely stable, i.e. Eq. (6), we have

$$\lim_{t\to\infty} \Phi(t, \mathcal{R}^n) = \{\mathbf{s}_1, \ldots, \mathbf{s}_M\}$$

In particular, we have for an increasing positive sequence $\alpha_k \to \infty$,

$$\Phi(0,\mathcal{R}^n) \supseteq \Phi(\alpha_1,\mathcal{R}^n) \supseteq \cdots \supseteq \Phi(\alpha_k,\mathcal{R}^n) \supseteq \cdots \supseteq \Phi(\alpha_\infty,\mathcal{R}^n) = \{\mathbf{s}_1,\ldots,\mathbf{s}_M\}$$

Therefore the points in $\mathcal{R}^n$ get closer to each other as the time step α_k increases. We call the points which converges to the same SEV as dynamically consistent points. Therefore, detecting these groups of dynamically consistent points in earlier stage and labeling them according to these consistency information will lead to significant improvement of labeling efficiency.

Based on this observation, in the main stage, we iteratively merge small balls at each iteration $k = 1, \ldots$ as follows:

(i) Apply process (5) to each center point $\mathcal{C}_k = \{\mathbf{c}_i^k : i = 1, \ldots, N_c^k\}$ until some preset time step α_{k+1} to obtain $\mathcal{C}_{k+1} = \{\mathbf{c}_i^{k+1} : i = 1, \ldots, N_c^{k+1}\}$.

(ii) Decompose the data set $\Phi(\alpha_{k+1}, \mathcal{C}_{k+1}) = \{\Phi(\alpha_{k+1}, \mathbf{c}_i^{k+1}) : i = 1, \ldots, N_c^{k+1}\}$ into separate small balls, say

$$B_\rho(\mathbf{c}_i^{k+1}) := \{\mathbf{c} \in \Phi(\alpha_{k+1}, \mathcal{C}_{k+1}) : \|\mathbf{c} - \mathbf{c}_i^{k+1}\| \le \rho\}$$

and then the set $\mathcal{C}_{k+1} \subset \Phi(\alpha_{k+1}, \mathbf{c}_i^{k+1})$ is reduced to the set of centers of this new small balls.

A naive numerical integration (or steepest descent methods) of system (5) often leads to zigzag behavior and a linear convergence rate. To expedite the process time, we notice that the clustering performance remains the same irrespective of the choice of G in (5) because $\tilde{C}_i \supset C_i$ for any G. This property implies that we have a freedom to choose a metric G that can lead to a fast convergence to a SEV when process (5) is applied, thereby enabling us to improve the convergence rate. In the following, we present a result establishing connections between numerical discretization strategies of system (5) and two widely used state-of-arts efficient numerical optimization solvers, line search methods and trust-region methods, so that the fast support vector-based clustering framework still holds (i.e. the clustering performance is preserved when these solvers are used up to discretization error.)

3.2.1. *Relations to line search methods*

Each iteration of a line search method is given by

$$\mathbf{c}_i^{k+1} = \mathbf{c}_i^k + \eta_k \mathbf{p}^k$$

where a search direction $\mathbf{p}^k$ is a descent direction (i.e. $\nabla f(\mathbf{c}_i^k)^T \mathbf{p}^k < 0$) and the step length η_k is chosen to satisfy the Wolfe conditions to guarantee global convergence (The Wolfe conditions require η_k to satisfy (i) $f(\mathbf{c}_i^k + \eta_k \mathbf{p}^k) \le f(\mathbf{c}_i^k) + z_1 \eta_k \nabla f(\mathbf{c}_i^k)^T \mathbf{p}^k$. (ii) $\nabla f(\mathbf{c}_i^k + \eta_k \mathbf{p}^k)^T \mathbf{p}^k \ge z_2 \nabla f(\mathbf{c}_i^k)^T \mathbf{p}^k$ with $0 < z_1 < z_2 < 1$) [13]. Now a simple numerical discretization strategy of system (5) is given by

$$\mathbf{c}_i^{k+1} = \mathbf{c}_i^k + \eta_k F(\mathbf{c}_i^k)$$

To show its relation to a line search method, we should show that the search direction $\mathbf{p}^k$ has the form

$$\mathbf{p}^k = F(\mathbf{c}_i^k) = -G_k^{-1} \nabla f(\mathbf{c}_i^k) \tag{9}$$

where G_k is a symmetric positive definite matrix. This can be easily proved by utilizing the well-known theorem saying that if F is a descent direction, i.e., $F(\mathbf{c}_i)^T \nabla f(\mathbf{c}_i) < 0$ for all x, then there exists a smooth Riemannian metric G such that $F(\mathbf{c}_i) = -\nabla_G f(\mathbf{c}_i)$ [14].

3.2.2. *Relations to trust-region methods*

We build the following numerical discretization strategy applied to a first-order linearized form of system (5):

$$\mathbf{c}_i^{k+1} = \mathbf{c}_i^k + \eta_k [F(\mathbf{c}_i^k) + J_F(\mathbf{c}_i^k)(\mathbf{c}_i^{k+1} - \mathbf{c}_i^k)]$$

If we use $G(\mathbf{c}_i) = I$, then we get the following iteration

$$\left(\nabla^2 f(\mathbf{c}_i^k) + \frac{1}{\eta_k} I\right)(\mathbf{c}_i^{k+1} - \mathbf{c}_i^k) = -\nabla f(\mathbf{c}_i^k)$$

which takes the form of solutions to the trust-region subproblem

$$\min_{\mathbf{p}} \quad m_k(\mathbf{p}) = f(\mathbf{c}_i^k) + \nabla f(\mathbf{c}_i^k)^T \mathbf{p} + \tfrac{1}{2} p^T \nabla^2 f(\mathbf{c}_i^k) \mathbf{p} \text{s.t.} \|\mathbf{p}\| \le \Delta_k$$

By using the trust-region strategy for the stepsize η_k, we can obtain a trust-region algorithm [15].

With respect to the issue of the choice of G, we've adopted the following strategy: For the first few iterations (say, three to five iterations) we've used $G(\mathbf{c}_i) = I$ (i.e. steepest descent method) because the first-order gradient information is enough to merge small balls and it reduces the cost involved in computing the inverse G_k^{-1} where the sample data set $\mathcal{D}_k$ to be applied is relatively large. Then we've used a line search with BFGS update for medium case and a trust-region method with Newton-CG update. Line-search methods and trust-region methods are well-known that they are globally convergent to locate local minima under some mild conditions [15].

3.3. *Stopping*

It is well-known [16] that if $\mathbf{s}$ is a SEV of system (5) and $\Re(\lambda_j) < -\alpha < 0$ for all of the eigenvalues λ_j of the Jacobian matrix $J_F(\mathbf{s})$, then given $\varepsilon > 0$ there exists a $\delta > 0$ such that for all $\mathbf{x} \in N_\delta(\mathbf{s})$, the flow $\Phi(t, \mathbf{x})$ of (5) satisfies

$$\|\Phi(t,\mathbf{x}) - \mathbf{s}\| \le \varepsilon e^{-\alpha t} \quad \forall t \ge 0$$

which implies an exponential convergence to a SEV when process (5) is applied to its nearby points. Therefore, in regards to a stopping criterion, we check whether the center point $\mathbf{c}_i^{k+1}$ obtained at iteration k satisfies $\|\nabla f(\mathbf{c}_i^{k+1})\| \le \varepsilon$.

4. Implementation

In this section we present detailed description of the proposed method discussed in the previous sections with algorithms of major steps and time complexity analysis. We also give remarks which distinguish the proposed method from others.

4.1. *Algorithm*

We start the fast labeling framework by partitioning whole data samples into subsets of separate small balls. The algorithm 1 for partitioning initial balls is similar with that of [17] except that we do not need to update the center of each ball.

Algorithm 1. Partitioning data points into small balls

```
1:  Given a data Set $\mathcal{D}=\{\mathbf{x}_k\}_{k=1}^N$
2:  $N_c=0, \mathcal{C}_0=\emptyset, i=1$;
    //the number of balls, a set of center points and center index
3:  while $\mathcal{D}\neq\emptyset$ do
4:    choose a random point $\mathbf{x}$ from $\mathcal{D}$.
5:    $\mathbf{c}_i^0 \leftarrow \mathbf{x}$;
6:    if $\exists \mathbf{x}_j \in \mathcal{D}$ that is $\|\mathbf{c}_i^0-\mathbf{x}_j\|<\rho$ given merging parameter $\rho$ then
7:      $\mathbf{x}_j \in \langle \mathbf{c}_i^0 \rangle$;
8:      $\mathbf{x}_j \backslash \mathcal{D}$;
9:    end if
10:   $\mathbf{c}_i^0 \in \mathcal{C}_0$;
11:   $N_c=N_c+1$;
12:   $i=i+1$;
13: end while
```

Notice that although in Algorithm 1 we start from calculating the distance of all samples from initial centers, the computation is significantly reduced as the iteration is repeated because we exclude the pre-computed centers and samples from dataset. And also we do not iterate for N times but on the average N_c. Therefore this strategy of Algorithm 1 is fast enough to construct initial balls and their centers especially when the resulting number of center is small. We provide experimental result on empirical cost of this procedure.

We already trained a kernel support function f from which we construct a associated dynamical system. And then we find SEVs for the centers points obtained above using f as follows:

Algorithm 2. Finding SEVs for center points

```
1:  $N_s=0, \mathcal{S}=\emptyset, k=0$; // number of SEVs, a set of SEVs and iteration
2:  while $N_c^k>0$ do
3:    $N_c^{k+1}=0$ and $\mathcal{C}_{k+1}=\emptyset$;
4:    for $i=1$ to $N_c^k$ do
5:      Numerically intergrate (5) one step to move from $\mathbf{c}_i^k$ to $\mathbf{c}_i^{k+1}$
6:      if $\|\nabla f(\mathbf{c}_i^{k+1})\|\leq\varepsilon$ //converged to local minimum then
7:        $\mathbf{c}_i^*=\mathbf{c}_i^{k+1}$;
8:        if $\mathbf{c}_i^*\notin\mathcal{S}$ //create a new SEV then
9:          $N_s\leftarrow N_s+1$ and $\mathbf{s}_{N_s}\leftarrow\mathbf{c}_i^*$;
10:       else
11:         find $\mathbf{s}_k\in\mathcal{S}$ such that $\mathbf{c}_i^*=\mathbf{s}_k$ and $\mathbf{c}_i^*\in\langle\mathbf{s}_k\rangle$;
12:       end if
13:     else
14:         $N_c^{k+1}=N_c^{k+1}+1$ and $\mathbf{c}_i^{k+1}\in\mathcal{C}_{k+1}$;
15:       end if
16:     end for
17:     subroutine : $IntermediateMerging(\mathcal{C}_{k+1}, N_c^{k+1}, \mathcal{S}, N_s)$
18:     for $i=1$ to $N_c^{k+1}$ do
19:       if $\exists\mathbf{s}_j\in\mathcal{S}$ that is $\|\mathbf{c}_i^{k+1}-\mathbf{s}_j\|<\rho$ then
20:         $\mathbf{c}_i^{k+1}\in\langle\mathbf{s}_k\rangle$ and $N_c^{k+1}\leftarrow N_c^{k+1}-1$;
21:       else if $\exists\mathbf{c}_j^{k+1}\in C$ that is $\|\mathbf{c}_i^{k+1}-\mathbf{c}_j^{k+1}\|<\rho$ then
22:         $\mathbf{c}_j^{k+1}\in\langle\mathbf{c}_i^{k+1}\rangle$ and $N_c^{k+1}\leftarrow N_c^{k+1}-|\mathbf{c}_j^{k+1}|$;
23:       end if
24:     end for
25:     end subroutine
26:     $k=k+1$;
27: end while
```

There are two types of intermediate merging procedures. When there exist a SEV that is closer than pre-specified merging parameter ρ, then that centers are regarded as already converged to that SEV. We call this procedure as ‘attraction’. Otherwise, check if there exist other centers that are within the range of merging parameter and then merge these centers into one if this is the case. We call this procedure as ‘joint’. By doing this intermediate merging step, we can further speed up convergence of dynamically consistent points. In this intermediate step, we can also use different parameter ρ' rather than initial merging parameter ρ or anneal it as ρ_k which is a decreasing value of iteration to prevent centers converging to different SEVs from merging. However, because SEVs are relatively distant by nature and centers close to each other after several iterations are almost surely converges to same SEV, this is not always advantageous because small merging parameter leads to increased computational cost.

After obtaining all SEVs from centers, we can assign cluster label to each SEVs by investigating the adjacency matrix as in [5] or locating the so-called transition equilibrium vectors as in [6]. Because the number of the SEVs are very small compared with that of the training points and often irrespective of its size, this process can be accomplished in a moderate computing time. When this is done, we finally label the whole data samples to the corresponding clusters. Because we keep the membership information of each sample to its centers obtained in merging step (Algorithm 1), this step is performed straightforwardly.

4.2. Complexity analysis of the fast support-based clustering

The time cost of partitioning step (Algorithm 1) and intermediate merging step (Algorithm 2 line 16–24) is dependent on the radius parameter, hence the number of centers. In partition step, it is at worst $O(N^2)$ complexity when ρ is close to 0 because every sample becomes center and the pairwise distance from every sample should be checked. When ρ becomes larger to cover whole data space, it converges to $O(N)$ because only few samples become centers (see Fig. 2(b)). However, as we observed from experimental result (Table 1), because the elementary computation is the calculation of simple Euclidean distance, this is ignorable compared to other steps. The intermediate merging step also takes ignorable time because there remains few centers or SEVs to be computed in this step. The complexity of most time consuming labeling step involves the numerical integration for each centers. Because we use gradient information for each iteration, the complexity is therefore $O(N_c d)$ where N_c is the number of center points and d is the dimension of input. Here, even though the number of centers tends to be increased when the number of data points is increased because the data samples are highly likely to be spread over all input space, it is less than log-linear to the input size and the upper bounded to some constant k regardless the size of input (see Fig. 2).

Cluster assignment of SEVs from adjacency matrix by DFS algorithm and cluster matching of each data sample takes $O(N_s)$ and $O(N)$ times of elementary calculation respectively, which are ignorable in practice. Therefore, total time complexity of proposed labeling method is analytically bounded on $O(kd)$ and log-linear to the input data size which results in remarkable efficiency for large-scale applications. Here we note that we can further reduce the complexity of the proposed method by adopting more sparse and fast training algorithm of support function such as CVM [18]

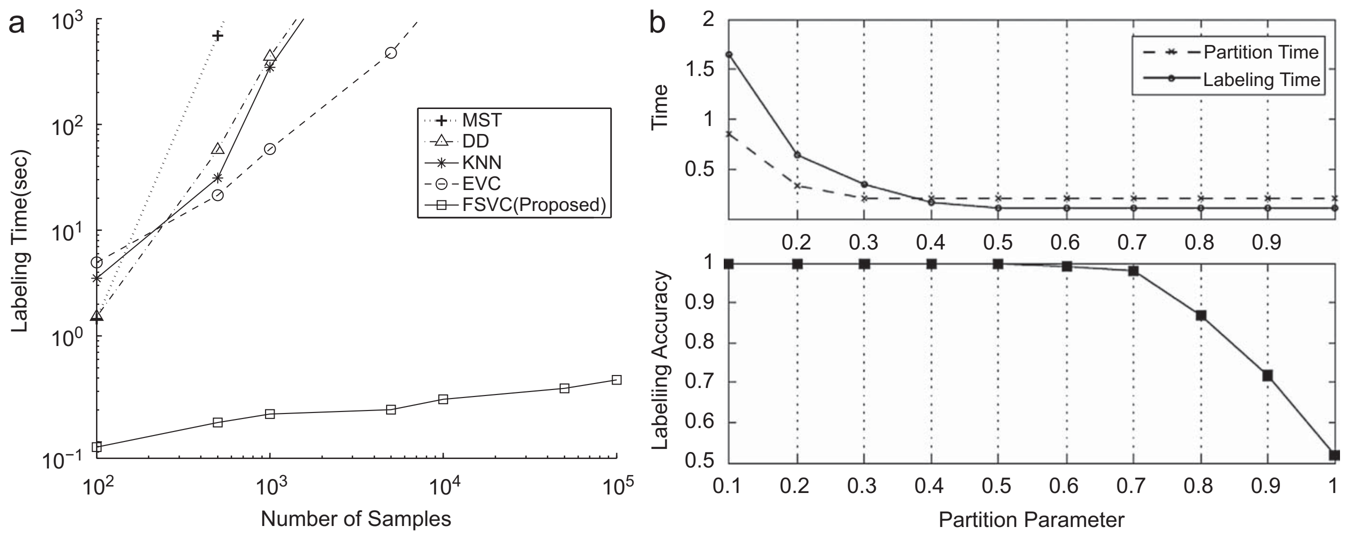

Fig. 2. Benchmark results. (a) Comparison of clustering time with respect to various sample size. (b) Time and accuracy of the proposed method with respect to a merging parameter.

Table 1
Benchmark result of clustering time and accuracy.

Dataset			Spectral		K-SVC		EVC		FSVC (Proposed)	
Category	Name	N	Time	RI_{adj}	Time	RI_{adj}	Time	RI_{adj}	Time	RI_{adj}
Small scale	four−Gaussians	100	0.56	0.86	4.09	0.82	0.78	1.00	0.08	1.00
	wine	118	0.77	0.77	7.42	0.45	18.43	1.00	0.54	0.98
	iris	150	0.75	0.87	3.36	0.86	2.46	0.94	0.29	0.94
	ring	160	1.18	1.00	7.21	0.75	1.48	1.00	0.19	1.00
	sunflower	210	1.38	0.89	6.56	0.85	2.87	1.00	0.49	1.00
	delta61	320	3.43	1.00	5.56	0.38	3.14	1.00	0.37	1.00
	oxours	400	4.34	0.84	12.36	0.38	6.14	0.99	0.12	0.98
Large scale	car	1728	9.29	0.25	12.42	0.10	823.90	0.57	12.89	0.57
	abalone	4177	15.56	0.15	447.22	0.10	67.43	0.20	4.27	0.20
	five−Gaussians	25 000	90.31	0.90	152.56	0.90	954.42	0.93	32.42	0.92
	shuttle	30 450	122.36	0.27	336.45	0.20	225.66	0.59	3.14	0.58

and CBSVM [19], which significantly reduces the evaluation of (2) in large-scale problems.

4.3. Characteristics of the fast support-based clustering

By investigating theoretical and empirical characteristics, we observe that following properties distinguish fast support-based clustering from others.

(i) *Fast inductive clustering*: When unseen data sample is provided, the proposed method is able to instantly label this new sample just finding the closest center when the center is closer than the partition parameter ρ (which is most probable case). Even in the case when the new sample itself should be a new center, the numerical integration to find its corresponding SEV could be performed much faster by continuously checking the distance from existing centers or SEVs.

(ii) *Robustness*: Since the SEVs are relatively distant in the data space by nature, the proposed method is robust in its labeling accuracy with small perturbations of the merging parameter, which is a necessary property to machine learning algorithm and is supported conceptually by the principle of low density separation and empirically as shown in Section 5.

(iii) *Parallelization*: When we skip the intermediate merging step, each center point converges to the SEV separately without any information or function value of other center points involved. This means that we can significantly speed up the proposed method by parallelizing the most time consuming part in a distributive computing manner. Each machine only needs the kernel support function and its centers to be processed. This is a very important feature when we applied to large-scale problems that exceeds the capacity of one machine. By contrast, most of the state-of-the-art clustering methods such as spectral clustering ([20]), k-means, SVC ([2,3]), kernel clustering ([4]) are not directly parallelizable because labeling process of each point is dependent on those of other points.

In the next section, we show some experimental result on benchmark dataset and application to image segmentation for demonstrating the performance of proposed method.

5. Experimental results and discussions

5.1. Benchmark results

To demonstrate the effectiveness and performance of the proposed method, simulations on toy and benchmark datasets are conducted with Pentium IV 3.0 GHz machine. For toy problems, we generated 2D samples from mixture of four Gaussians in various sample size. We compared labeling time of the proposed method (FSVC) with four different labeling methods; Delaunay Diagram (DD) ([21]), Minimum Spanning Tree (MST), K-nearest Neighbors (KNN) ([9]), and Equilibrium Vector-based Clustering (EVC) ([5]). We used a trained kernel radius function (2) of SVDD as the support function of the proposed framework and we call it as the fast support vector clustering (FSVC) in the experiment.

As shown in Fig. 2(a), FSVC dominates the other methods in its scalability. Here the labeling time for FSVC includes merging time of original data and the merging parameter is unchanged for all sample size. This empirical time complexity results show that DD, MST, KNN are quadratic and EVC is linear to the sample size, but they all exceeded the capacity of machine when the sample is large. On the contrary, FSVC showed a log-liner time complexity and stably works for large-scale samples.

To examine the effect of merging parameter, we conducted experiments with various merging parameters. The toy data was also generated with 2D four Gaussian generator and the sample size is 10^4. As shown in Fig. 2(b), the merging and labeling time are large as the merging parameter is small because the number of center is relatively large in this case. When the parameter is larger than some threshold value, it becomes insensitive to the parameter change while the accuracy of labeling is decreased. This is straightforward because when the parameter is too large, the initial balls contain too many samples and hence are likely to have non-homogeneous samples from other clusters. Therefore, the accuracy and the labeling time are in trade-off relationship. However, we could observe empirically that if the merging parameter is within practical range, the performance showed insensitiveness to the parameter change and therefore tuning merging parameter was not critical issue.

To demonstrate the performance of the proposed method in the entire clustering procedure, we conducted simulation on various dataset : `ring`, `four-Gaussians`, `delta61`, `oxours` and `five-Gaussians` are widely used in the literatures [4,7] and `wine`, `iris`, `car`, `abalone` and `shuttle` are from UCI repository [22]. We compared the performance of the proposed method with equilibrium vector-based clustering (EVC, [6]), spectral clustering based on Nyström method (Spectral, [23]) and kernel clustering (K-SVC, [4]). To evaluate the clustering accuracy, we used adjusted rand index (RI_{adj}) which is a widely used similarity measure between two data partitions ([24]) where given true labels given from [22] and predicted cluster labels are used. In measuring clustering time of FSVC, we included training time of a support function, merging time, and labeling time. As shown in Table 1, EVC and FSVC outperform the other methods in the accuracy, while FSVC significantly reduces the clustering time than EVC with marginal accuracy loss. Specifically, the time reduction becomes apparent in large-scale dataset because it tends to be dense locally, and therefore the merging step of FSVC significantly reduces the computation. Even though the spectral method shows moderate performance with faster speed than EVC, it also suffers from the scalability in the large-scale case because the number of points ($n \ll N$) to be sampled in Spectral should be increased to obtain meaningful accuracy and Spectral has $O(n^3)$ complexity to obtain an approximated solution. K-SVC is inferior both in accuracy and clustering time. Specifically it becomes much slower when the number of clusters, K, is increased.

To verify the applicability of the proposed framework to other support-based clustering algorithms we've conducted simulations on Gaussian Process clustering (GPC) which uses the variance function of Gaussian process as the support function and we call it as fast Gaussian Process Clustering (FGPC). As is shown in Table 2, the proposed framework can also be effectively applied to GPC and FGPC significantly reduces labeling time while maintaining the original clustering performance. However, the original GPC have to use all of data points to evaluate support function values compared with SVDD that is a sparse model and use only support vectors, and require us to calculate inverse of covariance matrix which is very expensive in terms of both memory capacity efficiency and computational complexity. These intrinsic drawbacks prevent the GPC from applying to large-scale problems and

Table 2
Benchmark result of clustering time and accuracy using GPC.

Dataset			GPC		Fast-GPC	
Name	N	Dim.	RI_{adj}	Time	RI_{adj}	Time
`four−Gaussians`	100	2	1.0000	0.7272	1.0000	0.0975
`ring`	160	2	1.0000	1.4625	1.0000	0.2490
`sunflower`	210	2	0.9942	2.0094	0.9942	0.4438
`delta61`	320	2	1.0000	4.3635	1.0000	0.2864
`oxours`	400	2	0.9857	4.9954	0.9857	0.2778

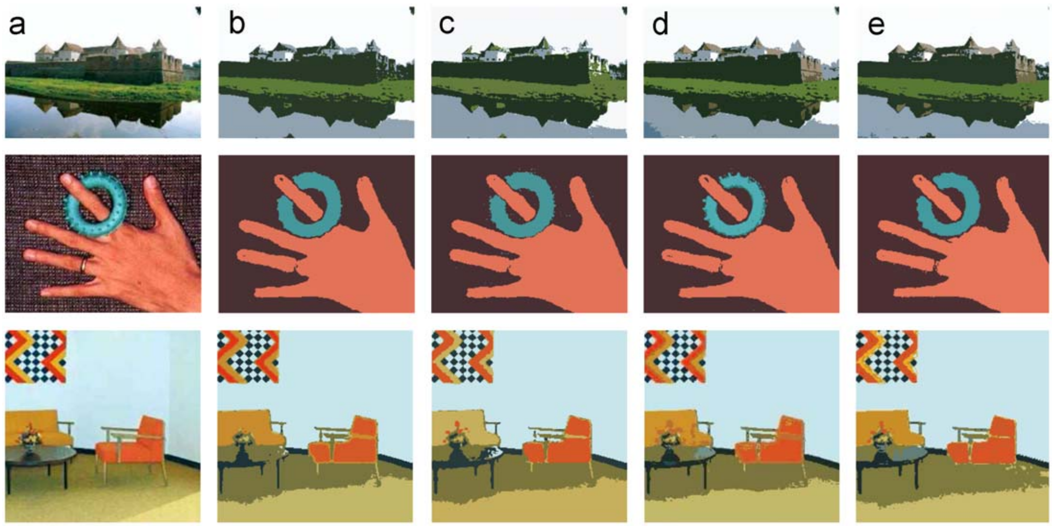

Fig. 3. Experimental results on benchmark image segmentation. Segmentation time for each image is provided in Table 3. (a) Original image. (b) Spectral. (c) K-SVC. (d) EVC. (e) FSVC.

thereby the corresponding support function is not a proper choice as the support function for our purpose. Therefore, we exclude the FGPC from applying to image segmentation problems below.

5.2. *Application to image segmentation*

For image segmentation, three images taken from [25] are used for comparison. We transformed each pixel of the original image into 3D vector in the (l,u,v) color space and to include spatial information, we augmented the 3D vector to 5D, by adding (X,Y) pixel position. Because each pixel is therefore a sample point in 5 dimensional space, image segmentation problem even for moderate image size is large-scale task and hence conventional support-based clustering algorithms require excessive computation time on this kind of application. The segmentation results are shown visually in Fig. 3 while the image size and segmentation time is provided in Table 3. As we can observe in Fig. 3, the images are generally well segmented into regions of homogeneous color and the resulting images are well simplified. However, the segmentation time taken by K-SVC and EVC is intolerable while FSVC takes only a few seconds and spectral method is significantly slower than FSVC. This result shows that in image segmentation task, FSVC is the only practical choice of use among the compared methods. To examine the sensitivity of segmentation result to the merging parameter setting, we performed segmentation with same images used above with different parameters. As shown in Fig. 4, the images tend to be more simplified with larger value of merging parameter while the segmentation time can be significantly reduced using larger merging parameter as shown in Table 4. However, using too large merging parameter leads to oversimplification of the image and hence loses some meaningful information contained in the images while using too small merging parameter needs considerable segmentation time. Therefore, we need to control the visual quality and segmentation time with merging parameter value and from the simulations, we observed that the value within range from 0.1 to 0.3 is a proper choice of merging parameter in most of the cases assuming that the sample points are normalized. In Fig. 5, we give various images obtained by boundary detection procedure. The images are taken from well-known image segmentation database [26]. The image size is normalized to 200×300. The boundary detection procedure is as follows: (i) conduct the FSVC algorithm and find cluster label for each pixel. (ii) find homogeneous regions with the same cluster. (iii) find a region boundary by finding pixels that are surrounded by the pixels from different clusters. (iv) remove regions with less number of pixels than pre-specified smoothing parameter. As shown in the figure, the boundaries of the main objects are well detected and clearly separated from the background. In summary, through the simulation results we can observe that the proposed FSVC algorithm effectively reduces the clustering time of the conventional support-based clustering algorithms especially for large-scale problem while maintaining the accuracy. We also observe that the FSVC can be successfully applied to real large scale applications such as image segmentation where many support-based clustering algorithms is impractical due to the labeling time bottleneck.

6. Conclusion

In this paper, we proposed a fast support-based clustering algorithm for large-scale data problems. Starting from an empirical support function trained from data, we first construct a dynamical system associated with the support function for

Table 3
Benchmark result on image segmentation time for various images.

Description		Segmentation time			
Image	Size	Spectral	K-SVC	EVC	FSVC
`castle`	300×201	21.23	123.94	686.23	2.96
`hand`	250×201	38.20	256.38	904.00	4.12
`livingroom`	246×250	23.91	174.69	771.05	3.53

Table 4
Image segmentation time for different merging parameter setting.

Description		Segmentation time			
Image	Size	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.3$	$\rho = 0.4$
`castle`	300×201	6.19	1.23	0.68	0.58
`hand`	250×201	15.96	3.05	1.07	0.74
`livingroom`	246×250	12.29	1.69	0.90	0.63

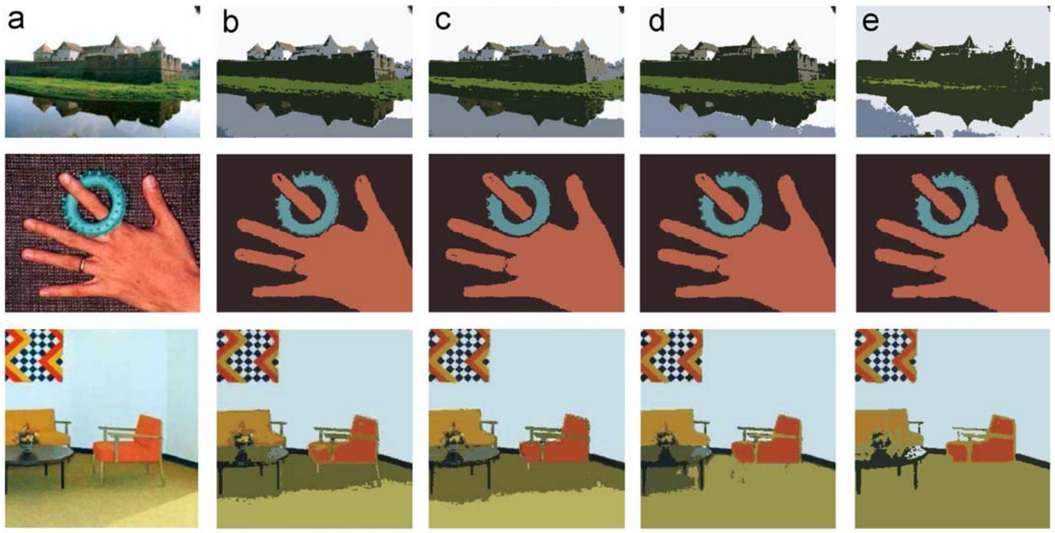

Fig. 4. Experimental results with different merging parameter setting. Segmentation time for each parameter is provided in Table 4. (a) Original image. (b) $\rho = 0.1$. (c) $\rho = 0.2$. (d) $\rho = 0.3$. (e) $\rho = 0.4$.

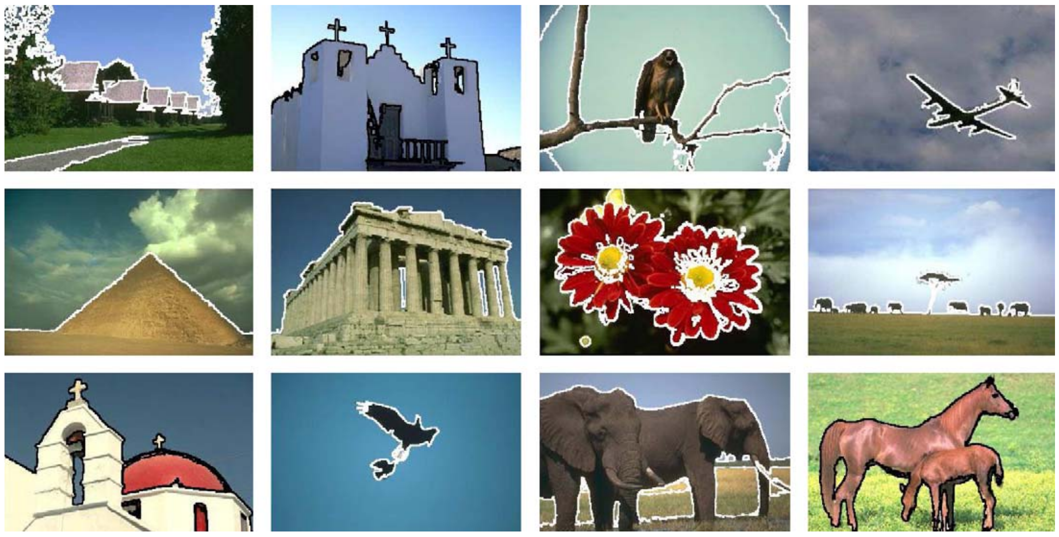

Fig. 5. Various images with region boundary delineated. Each homogeneous region consists of pixels that are convergent to the same SEV. See the text for more detail.

inductive clustering. Then utilizing the dynamical properties of the system, we decompose similar points into several small balls and apply the system to only their candidate points iteratively until we locate the SEVs from which we can label the whole data sample according to its original membership information. Experimental results show that the proposed method significantly reduces the clustering time while maintaining the accuracy. We expect that the method will provide a way to efficiently analyze large-scale clustering problems especially with a few number of modes in nature such as computer vision problem, leaving a further investigation.

Acknowledgments

This work was supported partially by the Korea Research Foundation Grant funded by the Korean Government (KRF-2008-314-D00483) and partially by the KOSEF under the Grant number R01-2007-000-20792-0.

References

[1] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.
[2] D.M.J. Tax, R.P.W. Duin, Support vector domain description, Pattern Recognition Letters 20 (1999) 1191–1199.
[3] A. Ben-Hur, D. Horn, H.T. Siegelmann, V. Vapnik, Support vector clustering, Journal of Machine Learning Research 2 (2001) 125–137.
[4] F. Camastra, A. Verri, A novel kernel method for clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005) 461–464.
[5] J. Lee, D. Lee, An improved cluster labeling method for support vector clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005) 461–464.
[6] J. Lee, D. Lee, Dynamic characterization of cluster structures for robust and inductive support vector clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 1869–1874.
[7] H.-C. Kim, J. Lee, Clustering based on gaussian processes, Neural Computation 19 (11) (2007) 3088–3107.
[8] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, R. Williamson, Estimating the support of a high-dimensional distribution, Neural Computation 13 (7) (2001) 1443–1471.
[9] J. Yang, V. Estivill-Castro, S.K. Chalup, Support vector clustering through proximity graph modelling, in: Proceedings, 9th International Conference on Neural Information Processing (ICONIP'02), 2002, pp. 898–903.
[10] T. Ban, S. Abe, Spatially chunking support vector clustering algorithm, in: Proceedings of International Joint Conference on Neural Networks, 2004, pp. 414–418.
[11] W.J. Puma-Villanueva, G.B. Bezerra, C.A.M. Lima, F.J.V. Zuben, Improving support vector clustering with ensembles, in: Proceedings of International Joint Conference on Neural Networks, 2005.
[12] D. Lee, J. Lee, Equilibrium-based support vector machine for semi-supervised classification, IEEE Transactions on Neural Networks 18 (2) (2007) 578–583.
[13] P. Fletcher, Practical Methods of Optimization, Wiley, New York, 1987.
[14] H.Th. Jongen, P. Jonker, F. Twilt, Nonlinear Optimization in $\Re^n$, Peter Lang Verlag, Frankfurt, 1983.
[15] J.N. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, Berlin, 1999.
[16] L. Perko, Differential Equations and Dynamical Systems, Springer-Verlag, New York, 1991.
[17] H. Späth, Cluster Analysis Algorithms for Data Reduction and Classification of Objects, Ellis Horwood Publishers, Chichester, 1980.
[18] I.W. Tsang, J.T. Kwok, P. Cheung, Core vector machines: fast svm training on very large data sets, Journal of Machine Learning Research 6 (2005) 363–392.
[19] S. Asharaf, M.N. Murty, S.K. Shevade, Cluster based core vector machine, in: IEEE International Conference on Data Mining, IEEE Computer Society, 2006, pp. 1038–1042.
[20] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: analysis and algorithm, in: Adavances in Neural Information Processing Systems, vol. 14, 2002.
[21] R. Jensen, D. Erdogmus, J.C. Principle, T. Eloft, The laplacian pdf distance: a cost function for clustering in a kernel feature space, in: Adavances in Neural Information Processing Systems, vol. 17, 2005, pp. 625–632.
[22] D. Newman, S. Hettich, C. Blake, C. Merz, UCI repository of machine learning databases, 1998, URL ⟨ http://www.ics.uci.edu/∼mlearn/MLRepository.html⟩.
[23] F.C.C. Fowlkes, S. Belongie, J. Malik, Spectral grouping using the nystrom method, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 214–225.
[24] P.A.L. Hubert, Compairing partitons, Journal of Classification 2 (1985) 193–218.
[25] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.
[26] The berkeley segmentation dataset and benchmark (2008). URL ⟨http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/⟩.