Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), Vol. 2
Lipo Wang, Jagath C. Rajapakse, Kunihiko Fukushima, Soo-Young Lee, and Xin Yao (Editors)

SUPPORT VECTOR CLUSTERING THROUGH PROXIMITY GRAPH MODELLING

Jianhua Yang, Vladimir Estivill-Castro, and Stephan K. Chalup*

School of Electrical Engineering and Computer Science,
The University of Newcastle, Callaghan, NSW 2308, Australia.

ABSTRACT

Support Vector Machines (SVMs) have been widely adopted for classification, regression and novelty detection. Recent studies [1, 2] proposed to employ them for cluster analysis too. The basis of this support vector clustering (SVC) is density estimation through SVM training. SVC is a boundary-based clustering method, where the support information is used to construct cluster boundaries. Despite its ability to deal with outliers, to handle high dimensional data and arbitrary boundaries in data space, there are two problems in the process of cluster labelling. The first problem is its low efficiency when the number of free support vectors increases. The other problem is that it sometimes produces false negatives. In the present paper, we propose a robust cluster assignment method that harvests clustering results efficiently. Our method uses proximity graphs to model the proximity structure of the data. We experimentally analyze and illustrate the performance of this new approach.

KEYWORDS: Clustering, Support Vector Machines, Proximity Graph.

1. INTRODUCTION

Kernel methods have become an increasingly popular tool for machine learning tasks such as classification, regression and novelty detection [5, 10]. Support Vector Machines (SVMs) [3, 4] have been successfully applied to a number of applications, and they exhibit good generalization performance and desirable properties, such as invariance under symmetries and robustness in the presence of noise. In addition to their accuracy, a key characteristic of SVMs is their mathematical tractability and geometric interpretation.

While SVMs have been widely adopted as supervised learning methods with labeled data, they have also been used for the exploration of unlabeled data (cf., [1, 8, 9]). Novelty detection and cluster analysis using SVMs are examples for learning unlabeled data. For many real-world problems, the task is not to classify but to detect novel or abnormal instances. Novelty detection has potential applications in many problem domains such as condition monitoring and medical diagnosis. Recent studies [1, 2] adapted SVMs for cluster analysis. One possible approach of novelty detection is one-class classification where the aim is to model the *support information* of a data distribution. The task is to find a hypersphere with minimal radius R and center $\vec{a}$ that contains most of the data points. Novel test points are then identified as those that lie outside the boundary of this hypersphere. As a by-product of this algorithm, a set of contours that enclose the data points is obtained. These contours can be interpreted as cluster boundaries and linkages between each pair of data items can be estimated. The SVC clustering algorithm [1, 2] is able to detect arbitrary shape clusters with a hierarchical structure in high dimensional data. It provides a way to deal with outliers and by using kernel methods, explicit calculations in feature space are not necessary. Despite these advantages, the algorithm's main drawback is its high time complexity.

According to Ben-Hur et al. [2], there are two main steps in the SVC algorithm, namely SVM training and cluster labeling. The SVM training part is responsible for novelty model training. The cluster labeling part checks the connectivity for each pair of points based on cut-off criteria obtained from the trained SVMs. The time complexity of the cluster labeling part is $O(n^2m)$, where n is the number of data items, and $m \ll n$ is the number of sampling points on each edge which is usually a constant between 10 and 20. With our implementation of the clustering algorithm of [2], we found that the SVM training part takes much less time than the cluster labeling part. Therefore the time complexity of SVM training part is not the main concern of support vector clustering. But, the computation of the cluster labeling part is the critical issue. It has extremely demanding CPU-requirements for large data sets. The heuristic introduced by Ben-Hur et al. [1] lowers the linkage estimations in the cluster labeling part. This approach does not check the linkages between all pairs of data items, but only those between points and support vectors. This improved the efficiency of the cluster labeling part. Its time complexity was lowered to $O((n-n_{bsv})n_{sv}^2m)$, where n_{bsv} is the number of bounded support vectors, and n_{sv} is the number of free sup-

*Current address: School of Computing and Information Technology, Griffith University, Brisbane, QLD 4111, Australia.

port vectors. However, if n_{sv} is over $0.05n$ for $m = 20$ or $0.10n$ for $m = 10$ (these are common cases), the time complexity of cluster labeling is $O((n - n_{bsv})n)$. On the other hand, our experiments show that such a heuristic for linkage estimation can sometimes fail. The observation is that adjacencies between points are only kept within *neighborhoods* in the same cluster in feature space. It is more likely that a point within a cluster has linkages to its neighbors, but not to the boundary points (support vectors) of the cluster. Checking only adjacencies with support vectors may generate a number of unlinked data points (we refer to these as *false negatives*) and makes the clustering results less meaningful.

In the present paper we propose an efficient cluster assignment method to harvest clusters. Our approach constructs appropriate proximity graphs to model a data set. In these graphs, vertices represent data points and edges connect pairs of points to model their proximity and adjacency. After the SVM training process, we adapt the obtained cutoff criteria (i.e., R) to estimate the edges of a proximity graph. This method avoids redundant checks in a complete graph and also avoids the loss of neighborhood information as it can occur when only estimating adjacencies to support vectors. The experiments demonstrate that our method of cluster labeling can discover meaningful and valid clustering results. The time complexity of our method is only $O(mn \log n)$, where n is the size of data set and m is the number of sampling points on each edge.

The rest of this paper is organized as follows: Section 2 reviews previous work related to proximity graph cluster analysis. Section 3 presents the main ideas of our approach. In Section 4 we discuss experimental results and conclude the paper with final remarks in Section 5.

2. RELATED WORK

Recently, Estivill-Castro and Lee [6, 7] proposed boundary-based clustering methods using proximity graph modelling. Proximity and density information modelling of 2D point data using Delaunay Diagrams is a powerful exploratory and argument free clustering algorithm for geographical data mining [6]. The main idea behind this approach is to detect the sharp density changes at potential cluster boundaries. In their approach, the main principle is modelling proximity and topology in terms of proximity graphs.

Typically, clustering methods use a similarity concept (e.g., Euclidean distance) to measure proximity between data objects. Even in high-dimensions, proximity is critical to cluster analysis. In proximity graphs, vertices represent data points and edges connect pairs of points to model proximity and adjacency. Despite of the fact that the point is the most primitive data object, it is not easy to define point proximity as a discrete relation. To best describe proximity between data points, a common family of proximity graphs was investigated and compared for different modelling considerations [7]. These proximity graphs include for example Delaunay Diagrams (DD), Minimum Spanning Trees (MST) and k-Nearest Neighbors (k-NN). By choosing appropriate underlying proximity graphs the expected time complexity is sub-quadratic for data of all dimensions.

Using SVMs, another boundary-based clustering method was proposed in [1, 2] (we call it SVC). This approach employs support vectors to construct cluster boundaries. Its principle is *novelty detection* [8] which is sometimes also called *data domain description* [9]. Domain description produces a *description* of a given set of objects. This description should cover the class of given objects, and ideally reject other possible objects in the object space. Generally, novelty detection can characterize estimating functions of the data that tell something interesting about the underlying distribution. Rather than finding a real-valued function for estimating the density data, it models the *support* of the data distribution through a binary function, such that most of data will live in the region where the function is non-zero (its support). One approach to domain description which is inspired by SVMs is constructing a hypersphere with minimum volume (or minimum radius) containing all objects. In the following we summarize the main points of the clustering approach proposed in [1, 2]:

Kernel techniques

SVC is an unsupervised learning technique that is kernelized. Assume given is a set of n data points $\{\vec{x_i}\} \subseteq \mathcal{X}$, with $\mathcal{X} \subseteq \Re^d$, the data space. To formulate a support vector description of this data set, a nonlinear mapping ϕ is employed to map $\mathcal{X}$ into some high dimensional feature space. The next step is to find the smallest enclosing hypersphere:

$$\|\phi(\vec{x_i}) - \vec{a}\|^2 \leq R^2 + \xi_i \ (\xi_i \geq 0) \ \forall i, \qquad (1)$$

where R is the radius, $\vec{a}$ is the center and ξ_i are some slack variables allowing for soft boundaries (some data points can be allowed to lie outside the sphere). The problem (1) is usually solved in its dual by introducing the Lagrangian and a regularization constant C in the penalty term,

$$\begin{aligned} L = R^2 - \textstyle\sum_i (R^2 + \xi_i - \|\phi(\vec{x_i}) - \vec{a}\|^2)\alpha_i \\ - \textstyle\sum \xi_i \mu_i + C \sum \xi_i. \end{aligned} \qquad (2)$$

where $\alpha_i \geq 0$ and $\mu_i \geq 0$ are Lagrangian multipliers, and $C \sum \xi_i$ is a penalty term. Also the Karush-Kuhn-Tucker condition allows the problem to be rewritten as

$$\begin{aligned} &\max L = \textstyle\sum_i \alpha_i \phi(\vec{x_i})^2 - \sum_{i,j} \alpha_i \alpha_j \phi(\vec{x_i}) \phi(\vec{x_j}) \\ &\text{subject to } 0 \leq \alpha_i \leq C, \textstyle\sum \alpha_i = 1, i = 1, \cdots, n. \end{aligned} \qquad (3)$$

Following the SVMs method, we use a *kernel* representation $k(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$. Eq. (3) is now written as:

$$\begin{aligned} &\max L = \textstyle\sum_i \alpha_i k(\vec{x}_i, \vec{x}_i) - \sum_{i,j} \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j) \\ &\text{subject to } 0 \leq \alpha_i \leq C, \textstyle\sum \alpha_i = 1, i = 1, \cdots, n. \end{aligned} \qquad (4)$$

The Lagrangian multipliers α_i can be obtained by optimizing Eq. (4). Only those points with non-zero α_i satisfy Eq. (1) as equality. These points lie on the boundary of the sphere and are called *Support Vectors* (SVs). Points with $\alpha_i = C$ have hit the upper bound for the radius and lie outside the sphere. These points are called *Bounded Support Vectors* (BSVs), and are treated as noise.

One of the key features of kernel methods is that they do not require an explicit calculation of the feature map ϕ but only use the values of the dot products between mapped patterns. For clustering purposes, we followed [1, 2] and used Gaussian kernels $k_q(\vec{x}_i, \vec{x}_j) = e^{q\|\vec{x}_i - \vec{x}_j\|^2}$ with width parameter $q = -1/(2\sigma^2)$. According to [9] polynomial kernels do not yield tight boundaries.

Construction of cluster boundaries

Support vectors can be used to describe the hypersphere in feature space. For each point $\vec{x}$, the distance of its image $\phi(\vec{x})$ to the center of the hypersphere is given by

$$R^2(\vec{x}) = k(\vec{x}, \vec{x}) - 2\sum_i \alpha_i k(\vec{x}, \vec{x}_i) + \sum_{i,j} \alpha_i \alpha_j k(\vec{x}_i, \vec{x}_j).$$

The radius R of the sphere can be obtained by calculating the distances $R_i = \{R(\vec{x}_i) \mid \vec{x}_i \text{ is a support vector}\}$ of the support vectors from the center of the hypersphere (in a practical implementation the average of these distances can be used).

Cluster boundaries can be constructed by a set of contours that enclose the points in data space $\{\vec{x} \mid R(\vec{x}) = R\}$. Thus, SVs lie on cluster boundaries, BSVs are outside, and all other points lie inside clusters.

Clustering-labeling

The cluster description itself does not differentiate between points that belong to different clusters. To do this, an adjacency matrix A_{ij} is defined based on geometric observation: given a pair of data points that belong to different clusters, for any path in data space connecting them, the corresponding path in feature space must have an intersection with the outside of the hypersphere. For each pair of points $\vec{x}_i$ and $\vec{x}_j$, A_{ij} takes a binary value.

$$A_{ij} = \begin{cases} 1, \text{if } R(\vec{x}_i + \lambda(\vec{x}_j - \vec{x}_i)) \le R, \forall \lambda \in [0,1]; \\ 0, \text{otherwise.} \end{cases}$$

Clusters are now defined as the connected components of the graph induced by A. Calculating A_{ij} for points $\vec{x}_i$ and $\vec{x}_j$ is implemented by sampling a number of, say m points on the line segment between the two points (where m is usually chosen between 10 and 20). The two cluster labeling strategies of Ben-Hur et al. [1, 2] are described in Cluster Labeling Strategy 1 and 2 below.

Cluster Labeling Strategy 2 is faster than Strategy 1. However, there are two issues with Strategy 2. First, when

Cluster Labeling Strategy 1 (CG)

Calculate A_{ij} for each pair of points $\vec{x}_i$ and $\vec{x}_j$ in data space. This results in using the complete graph, denoted by CG, to model adjacency A_{ij}. It takes $O(n^2 m)$ time.

Cluster Labeling Strategy 2 (SVG)

Calculate A_{ij} only for pairs of points $\vec{x}_i$ and $\vec{x}_j$, where $\vec{x}_i$ or $\vec{x}_j$ is a support vector. This results in a subgraph of CG, which is referred to as SVG. It takes $O((n - n_{bsv})n_{sv}^2 m)$ time, where n_{bsv} is the number of BSVs and n_{sv} is the number of free SVs.

n_{sv} is greater than $0.05n - 0.1n$, its time complexity is still quadratic. Second, points in data space that are close neighbors are more likely to belong to the same cluster. However, checking only linkages to support vectors does not necessarily take this into account and can produce unlinked data points of close proximity. This can make clustering results less meaningful.

In the next section, we will extend the above two types of boundary-based clustering methods, and present our approach to SVC through proximity graph modeling. In our method, cluster boundaries are described by a set of SVs, and cluster labeling is based on proximity graph modelling. The time complexity of our method is only $O(mn \log n)$. Characteristics of several different proximity graphs will be analyzed and contrasted.

3. SUPPORT VECTOR CLUSTERING THROUGH PROXIMITY GRAPH MODELLING

SVC through proximity graph modelling extends the clustering method of [1, 2] with the concept of proximity graph modelling (cf. e.g., [6, 7]). Our method consists of three steps, that will now be explained in detail.

3.1. SVM-training for detecting cluster structure

The number of SVs and BSVs affects the cluster structure, which therefor can be controlled by the SVM training parameters q and C. As the width q of the Gaussian kernels increases, the number of SVs (n_{sv}) increases, the shape of the cluster boundaries becomes rougher, and the contours tend to split up (cf., the boundaries of white regions in Fig. 1(a)-(d) below). On the other hand, the number of BSVs (n_{bsv}) can be controlled by C, more precisely by $n_{bsv} < 1/C$. That is, if $C \geq 1$, there are no BSVs. To allow for BSVs, one should set $C < 1$. Instead of using C it is more naturally to work with the parameter $p = 1/nC$, which represents an upper bound for the fraction of BSVs. The parameter q determines the scale at witch the data is probed, and p decides the softness of the boundary [1]. Fig. 1 shows a 285-points data set and the changes of cluster boundaries in dependence to different settings of q and p, which are selected experimentally as in [1].

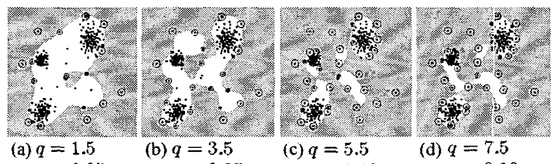
(a) $q = 1.5$ $p = 0.05$ $n_{sv} = 22$ $n_{bsv} = 10$
(b) $q = 3.5$ $p = 0.05$ $n_{sv} = 26$ $n_{bsv} = 6$
(c) $q = 5.5$ $p = 0.10$ $n_{sv} = 37$ $n_{bsv} = 19$
(d) $q = 7.5$ $p = 0.10$ $n_{sv} = 40$ $n_{bsv} = 17$

Figure 1: The clusters are represented by white regions. Their boundaries and number vary with q and p. Encircled points are the resulting SVs and BSVs.

3.2. Cluster labeling using proximity graphs

After SVM training, the radius R of the hypersphere can be used as a cut-off criterion to check the connectivity between data points. Under certain cut-off conditions, some points will become non-connected. The clusters are the connected components induced by A_{ij}. Note that a connected component can also be identified by a spanning tree, which produces a much smaller number of edges than the CG. Consequently, Cluster Labeling Strategy 1 results in testing many redundant edges. The sparse graph SVG in Cluster Labeling Strategy 2 does not decode neighborhood information exactly, and sometimes yields trivial clusters. Therefore, we propose a new cluster labeling strategy below to overcome the disadvantages of Cluster Labeling Strategy 1 and 2.

Cluster Labeling Strategy 3 (Proximity graphs)

We model the data with an appropriate proximity graph that reflects the data distribution and incorporates proximity and topology information. The idea is to calculate coefficients of the adjacency matrix A_{ij} only for pairs of $\vec{x}_i$ and $\vec{x}_j$, where $\vec{x}_i$ and $\vec{x}_j$ are linked by an edge E_{ij} in a proximity graph. Actually, the adjacency matrix A_{ij} is not held explicitly in the memory, but encoded in the proximity graph. The problem is to find the connected components in the graph by exploring the edges induced by A_{ij}. We perform the same sampling strategy for computation of A_{ij} as in [1, 2]. All edges in the current proximity graph are called *candidate edges*. We refer to an edge E_{ij} as *active edge* if $A_{ij} = 1$, and as *passive edge* if $A_{ij} = 0$. An *active path* in the proximity graph will be formed if every edge in the path is an active edge. A connected component is equivalent to an active path.

In a proximity graph, points are connected by edges if they are close to each other according to some proximity measure. Near-by points are naturally more likely to be in the same cluster than points that are far away. Thus, cluster labeling with a proximity graph strategy is a good heuristic to reduce the time of testing linkages. We will discuss three types of proximity graphs for cluster assignment. These are Delaunay Diagram (DD), Minimum Spanning Tree (MST), and k-Nearest Neighbors (k-NN) [6, 7]. They can be derived by considering different aspects of proximity and topology. DD represents a "is-neighbor" relation. The MST is based on the local closeness of data points. It is a subgraph of DD, and encodes less proximity information. k-NN is based on distance concepts. Fig. 2 shows the construction of a variety of proximity graphs for cluster assignment. We used the Leda[1] and ANN[2] libraries to construct these proximity graphs.

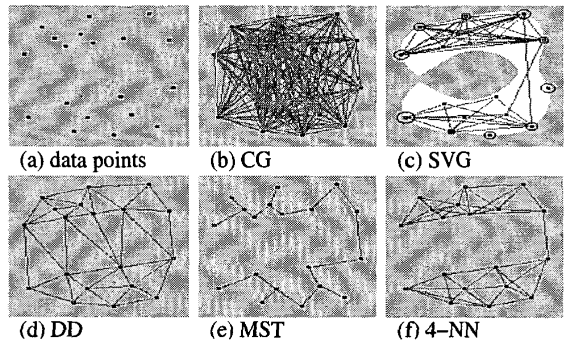
(a) data points (b) CG (c) SVG
(d) DD (e) MST (f) 4–NN

Figure 2: Various proximity modelling graphs. In (c), encircled points are SVs and the white region is the pre-image of the hypersphere in feature space.

3.3. Cluster harvest

Without going into the mathematical details, the cluster harvest procedure is justified by the empirical observation that clusters correspond to connected components of edges, i.e. active paths. Passive edges are not of interest, and they will be removed from the proximity graph. After the removal of passive edges, the task of cluster harvest becomes recognizing all active paths formed. Once active edges have been determined, a classical algorithm like Depth-First-Search (DFS) can be used for collecting connected components. Note that DFS has complexity proportional to the number of edges in the proximity graph. Pseudo code for cluster harvest is shown in Algorithm 4. To avoid trivial clusters, some clean-up work is necessary. We treat a cluster with a small number of points under a threshold (say 3) as noise. BSVs

[1]http://www.mpi-sb.mpg.de/LEDA/
[2]http://www.cs.sunysb.edu/~algorith/implement/ANN/implement.shtml

are included in the closest cluster as suggested in [1, 2]. Fig. 3 demonstrates the whole procedure of SVM clustering for the case of DD modelling.

Algorithm 4 (Cluster Harvest)

```
function $clusterHarvest(G$:Graph; S:Sets)
    {input $G$: Subgraph after cluster labeling}
    {output $S$: Set of clusters}
    var
        $v$: Node;
        $C$: Set; {A set of points in one cluster}
    begin
        while ($v := G.chooseNode()$)
          $C.clear()$;
          $DFS(G, v, C)$; {$v$ is moved from $G$ to $C$}
          $S.add(C)$;
        end while
    end
```

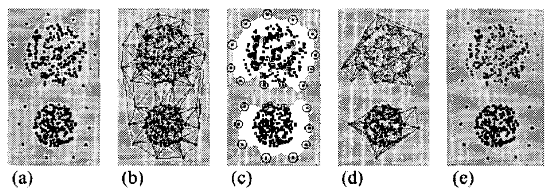

Figure 3: Procedure of SVC using DD modelling. (a) A set of points. (b) DD modelling. (c) SVM training result. (d) Active paths after labeling. (e) Final result after cluster harvest.

4. PERFORMANCE EVALUATION

In this section, we present results of experiments which compare and evaluate the performance of different cluster labelling methods. We demonstrate empirically the robustness and efficiency of our approach of cluster labeling with proximity graphs. The LibSVM[3] library has been used in the performance evaluation.

4.1. Time complexity

Given is a data set of n points $\{\vec{x_i}\} \subseteq \mathcal{X}$, with $\mathcal{X} \subseteq \Re^d$, the data space. The number of edges in DD is linear in size n for 2D ($3n$-6 at most), but it is quadratic in size for 3D. Other proximity graphs are linear in size for all dimensions. The number of edges in MST is $n - 1$. For k-NN, $O(kn)$ is the upper bound. Thus, walking on edges for cluster assignment and cluster harvest takes $O(n)$ time. On the other hand, the field of computational geometry has developed $O(n \log n)$ time algorithms to construct DD, MST and k-NN in spaces of various dimensions. Therefore, all proximity graphs presented here can perform 2D cluster labeling and harvest in $O(n \log n)$ time. This is remarkably efficient compared to the quadratic time requirements of CG and of SVG. The experimental results displayed in Fig. 4 illustrate that our approach outperforms CG and SVG cluster labeling. We first generated a data set with 1,000 points based on a mixture model. From this data set, we sampled five subsets with 100, 200, 400, 600 and 800 points, respectively. Then we performed SVC on these subsets using various cluster labeling mechanisms. After empirical test with different values, we set parameters $q = 0.20$ and $p = 0.10$ to train the SVMs. The results shown in Fig. 4 include the overall time requirements for proximity graph construction, cut-off criterion computation, cluster labeling and harvest.

When we move to high-dimensional data sets, k-NN and MST graphs are more attractive, since the number of edges of these graphs remains linear in n. This makes proximity graph modelling scalable to high-dimensional data for support vector cluster analysis.

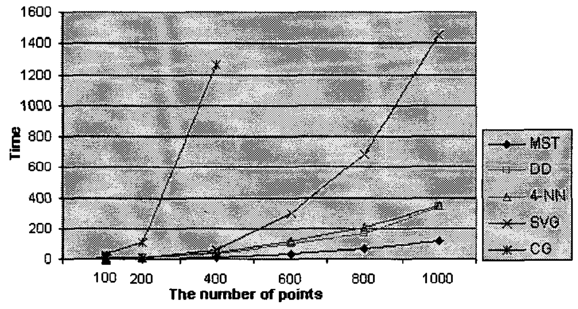

Figure 4: CPU time comparison of various cluster labeling mechanisms. Slowest is the result using CG which is quadratic in the number of points n and fastest is the result using MST which is $n \log n$.

4.2. Clustering quality

When applying different proximity graphs to model data for support vector clustering, they may produce different results. We take the result of CG as reference against which we compare the other cluster labeling methods. Our experiments show, DD works as good as CG. This is because, as a good spanner, DD holds sufficient interconnections. In contrast, as a subgraph of DD, MST encodes much less proximity information. Some linkages between neighbors in DD have been lost in MST. Despite its speed, MST is a fragile clustering method. Interestingly, k-NN with appropriate k

[3]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

also reports good results. The experiments indicate that k-NN (with $k > 3$) captures satisfactory proximity information for cluster labeling. Here the argument k does not need to be tuned as carefully as in [7], because the purpose of proximity graph modelling in our study is only for cluster labeling, but not for the computation of the cut-off criterion. As mentioned before, SVG is a heuristic to produce a sparse graph to simplify cluster labeling. However, it sometimes can produce false negatives and make the clustering result less meaningful. This situation can occur when the parameter p is high. In this case, most SVs turn out to be BSVs and there are only a few free SVs left. The data points lacking of support information (i.e. they can not connect to SVs), become false negatives. Fig. 5 illustrates cluster labeling results from different labeling methods for a data set shown in Fig. 5(a). Fig. 5(b) is the result of CG. Fig. 5(c) shows that SVG strategy produces false negatives. In contrast, with consideration of neighborhood relationship, the proximity graph modeling approaches, as shown in Fig. 5(d, e, f), do not yield false negatives.

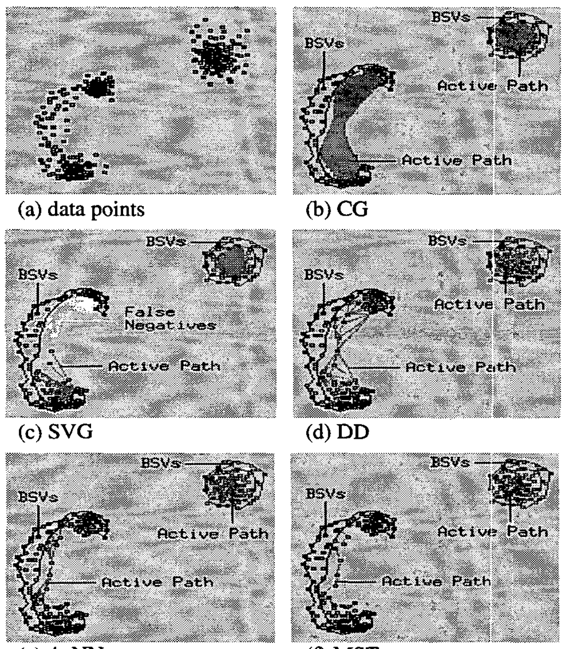

Figure 5: Comparison of clustering results using different labeling methods with parameters $p = 0.5$, $q = 2.75$ for SVM training. Note that, for better recognition, the two clouds of BSVs have been highlighted by additional boundary lines. The cloud in the upper right corner has approximately the shape of an annulus.

5. CONCLUSION

We extended the support vector clustering method, and applied proximity graph modelling to the cluster labeling part. Our experiments demonstrated that the approach robustly and efficiently performs cluster assignment, and makes support vector clustering scalable to large data sets.

The impact of our improvement from quadratic to $n \log n$ time is also reflected if we attempt parallel implementation. For example, such parallel implementation of our approach would be logarithmic in n processors, while the previous approaches would only be linear on n processors.

6. REFERENCES

[1] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[2] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik. A support vector method for hierarchical clustering. In *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[3] K. P. Bennett and C. Campbell. Support vector machines: Hype or hallelujah. *SIGKDD Explorations*, 2(2):1–13, 2000.

[4] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] C. Campbell. Kernel methods: A survey of current techniques. http://lara.enm.bris.ac.uk/cig/.

[6] V. Estivill-Castro and I. Lee. Amoeba: Hierarchical clustering based on spatial proximity using delaunay diagram. In *Proc. of the 9th Int. Symposium on Spatial Data Handling*, pages 7a.26–7a.41, 2000.

[7] V. Estivill-Castro, I. Lee, and A. T. Murray. Criteria on proximity graphs for boundary extraction and spatial clustering. *LNCS*, 2035:348–347, 2001.

[8] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1472, 2001.

[9] D. M. J. Tax and R. P. W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999.

[10] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, Heidelberg, DE, 1995.