An Efficient Clustering Scheme using Support Vector Methods

J Saketha Nath\(^1\) and S K Shevade\(^2\)

\(^1\) Supercomputer Education and Research Center, Indian Institute of Science, Bangalore-560012, INDIA. Saketh.Nath@gmail.com
\(^2\) Department of Computer Science and Automation, Indian Institute of Science, Bangalore-560012, INDIA. shirish@csa.iisc.ernet.in

Abstract. Support vector clustering involves three steps — solving an optimization problem, identification of clusters and tuning of hyper-parameters. In this paper, we introduce a pre-processing step that eliminates data points from the training data that are not crucial for clustering. Pre-processing is efficiently implemented using the R*-tree data structure. Experiments on real-world and synthetic datasets, show that pre-processing drastically decreases the run-time of the clustering algorithm. Also, in many cases reduction in the number of support vectors is achieved. Further, we suggest an improvement for the step of identification of clusters.

1 Introduction

Clustering analysis of data aims at discovering smaller, more homogeneous groups from a large heterogeneous collection of data objects. It is used in many fields including data mining, statistical data analysis, vector quantization and various business applications. Clustering is a challenging task as there is normally little or no a priori information about the data. Agglomerative algorithms, k-Means algorithm, Fuzzy algorithms, BIRCH and CLARANS are a few of the existing clustering methods [1].

Formally, the task of clustering is : given a set of data points (\(m\) in number, denoted by \(\chi = \{x_i \mid x_i \in \mathbb{R}^n \forall i = 1, 2, \ldots, m\}\)) and the number of clusters, \(K\), each \(x_i\) has to be assigned a cluster label \(l_i \in \{1, 2, \ldots, K\}\). In Support Vector Clustering (SVC) [2], data points are mapped from the data space to a high dimensional space called feature space. In feature space, the smallest hypersphere that encloses the images of most of the data points is identified. This hypersphere when mapped back to data space forms a set of disjoint contours which enclose most of the data points. These contours are interpreted as cluster boundaries and the points enclosed by each contour are associated with the same cluster. Points enclosed by none of the contours are interpreted as outliers. The task of identifying the hypersphere is mathematically expressed below:

\[
\min_{R, \mu, \xi_i} \quad R^2 + C \sum_i \xi_i \\
\text{s.t.} \quad \|\phi(x_i) - \mu\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0 \quad \forall \ i
\]
where, $R$ is the radius, $\mu$ is the center of the hypersphere, $\xi_i$ are the slack variables, $\phi$ is the data space to feature space mapping and $C$ is hyper-parameter. Higher the value of $C$, lesser the number of points whose images lie outside the hypersphere and vice-versa. The Wolfe dual [3] of the above optimization problem is:

$$\min_{\alpha_i} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \phi(x_i) \cdot \phi(x_j)$$

s.t. $\sum_i \alpha_i = 1, 0 \leq \alpha_i \leq C$

where, $\alpha_i$, the Lagrange multipliers are the dual variables. Sequential Minimal Optimization (SMO) [4], is the most popular algorithm that efficiently solves the dual problem. It has the ability to solve large optimization problems. The data points with $0 < \alpha_i < C$ are non-bound support vectors (NBSVs), $\alpha_i = C$ are bounded support vectors (BSVs) and $\alpha_i = 0$ are non-support vector points (NSVs). NBSVs always lie on the hypersphere, BSVs lie outside, or on the hypersphere and NSVs lie inside, or on the hypersphere. NBSVs and BSVs together are referred to as support vectors (SVs). In most of the cases, the number of SVs ($\rho \ll m$).

Once the dual is solved, identification of clusters (cluster labeling) is done using a simple Graphical Connected-component Method (GCM): let $LS[x_i, x_j]$ denote the line segment joining $x_i$ and $x_j$ in the data space. If the image of this line segment in the feature space does not exit the hypersphere, then the points $x_i$ and $x_j$ are said to be connected else, they are said to be disconnected. This leads to a natural definition of the adjacency matrix $A$: $A_{ij}$ is 1 if the image of $LS[x_i, x_j]$ lies entirely within the hypersphere and 0 otherwise. Note that, the matrix $A$ is built using all points, except the BSVs (since, their images lie outside the hypersphere). Clusters are now defined as the connected components of the graph induced by $A$ and labeled with $l_i \in \{1, 2, \ldots, \kappa\}$, where $\kappa$ is the number of connected components. Each $x_i$ is assigned the label of the connected component it belongs to. The BSVs are assigned the labels of their nearest clusters. The inference function which indicates whether a point $x$ has an image that lies inside, on, or outside the hypersphere ($f(x)$ negative, zero or positive, respectively) is:

$$f(x) = g(x) - g(x_s)$$

where, $x_s$ is any point lying on the hypersphere and

$$g(x) = \phi(x) \cdot \phi(x) - 2 \sum_i \alpha_i \phi(x_i) \cdot \phi(x) + \sum_{i,j} \alpha_i \alpha_j \phi(x_i) \cdot \phi(x_j)$$

Note that, the dual and the inference function $f(x)$ involve inner products in the feature space. Thus, one can use a Gaussian kernel,

$$K(x, y) = e^{-\sigma \|x-y\|^2} = \phi(x) \cdot \phi(y)$$

with width parameter $\sigma$, for evaluation of inner products in the feature space. This eliminates the explicit evaluation of $\phi$. $\sigma$ controls the number and shape of
the clusters. In general, higher the value of \( \sigma \), higher the number of clusters and vice-versa. The hyper-parameters \( \sigma \) and \( C \) need to be tuned to get the required number of clusters. The final clustering obtained through SVC is the clustering obtained with the tuned set of hyper-parameters (denoted by \((C^*, \sigma^*)\)).

Note that, the NSVs do not influence the inference function. Also, it can be shown that the optimum dual solution (for a given \((C, \sigma)\)) will be the same even if one uses the SVs instead of all the data points in solving the optimization. Thus, if the set of NSVs is known a priori, then one can eliminate them from training set and achieve a reduction in the computational effort. But, it is not possible to determine the exact set of SVs without solving the optimization problem. However, one can eliminate some data points that have a high chance of being NSVs. If in this process some SVs are eliminated, then the optimum dual solution will not be the same as that with all data points. However, if the final clustering obtained is not significantly different from that obtained with the entire training set, then such elimination can be employed. In this paper, we present a novel method that efficiently eliminates data points from the training data that are not crucial for determining the cluster labels. The computational effort in clustering (with such pre-processing) will be less because of the reduction in \( m \) and \( \rho \).

Figure 1 shows clustering with SVC using \((C^*, \sigma^*)\) on a synthetic dataset. Note that, NSVs lie inside the cluster boundaries. From the figure it can be seen that the points that are well surrounded by large number of points from all directions have a high chance of being NSVs. These observations suggest that, we can eliminate points from training data that have following properties:

- **P1** very large number of neighbors,
- **P2** neighboring points from all directions.

This elimination process if executed carefully, will not effect the final clustering. The proposed pre-processing method uses these ideas to eliminate redundant data points from the training data (Sect. 2.1). The optimization problem is then solved using the reduced training data. Cluster labeling is then done using a variant of the GCM (Sect. 2.2). The idea is to build the matrix \( A \) using NBSVs only. The points eliminated from the original training data are assigned the labels of their nearest clusters. Tuning of the hyper-parameters \( \sigma \) and \( C \) is done using an algorithm presented in Sect. 2.3.

In Sect. 3, we present the results of experiments done on synthetic and real-world datasets. We conclude the paper in Sect. 4.

## 2 Algorithm

The major steps in the proposed clustering algorithm are:

- **Pre-processing**: Eliminates redundant points and gives reduced training set.
- **Tuning of hyper-parameters**: gives the value of \((C^*, \sigma^*)\).
- **Optimization**: Solving dual for Lagrange multipliers (with \((C^*, \sigma^*)\)).
- **Cluster Labeling**: Determining cluster labels for entire training set using modified GCM.
2.1 Pre-Processing using R*-tree

In order to rank points according to $P_1$: for each point, $k_1$ nearest neighbors are selected and the distance $d$, between the point and the farthest of the $k_1$ neighbors is calculated. Greater the value of $d$, lesser the neighbors it has. In order to rank points according to $P_2$: for each point, the resultant, $\tau$, of unit vectors drawn from the point to its $k_2$ nearest neighbors is calculated. The lesser the value of $\tau$, greater is the possibility that it has neighboring points from all directions. All nearest neighbor queries are processed using R*-tree data structure [5], which is the most efficient structure known for nearest neighbor queries.

For each point in the training set, the values of $d$ and $\tau$ are calculated. These values are then normalized, to lie in $[0, 1]$. The “weight” of each point (quantifies $P_1$ and $P_2$ of a point) is calculated as $\eta d + (1 - \eta) \tau$, where $\eta$ is a factor in $[0, 1]$. The points are then listed in decreasing order of weight. Thus, the head of the list will be dominated by points who have very few neighbors or neighbors in few directions, whereas, tail will be dominated by redundant points. Thresholds $\theta_1$, $\theta_2$ ($> \theta_1$) are selected. Points listed above threshold $\theta_1$ are considered outliers, those between $\theta_1$ and $\theta_2$ are used as training set (non-redundant points) and those below $\theta_2$ are considered as redundant points.

But, as mentioned in Sect. 1, a point with $P_1$ and $P_2$ is redundant only in presence of the neighbors. Thus, in the above method some non-redundant points may get eliminated because of the static nature of weight assignment.
Ideally, after each point is eliminated from the training set, weight assignment must be redone. However, this dynamic weight assignment method will be inefficient. In view of this, the above method is modified as follows: Weight assignment for all points is done and outliers, reduced training set and redundant points are identified as given above. $k_3$ nearest neighbors of every non-redundant point are computed. A redundant point is selected from these set of $k_3$ neighbors and sent along with the non-redundant point to the training set (none will be sent if no such point exists). This reduces the possibility of eliminating non-redundant points from the training set. It is easy to see that, the reduced training set size with this method is $2(\theta_2 - \theta_1)$. In the remainder of the text, this pre-processing method will be referred to as “Heuristic for Redundant point Elimination” (HRE). Figure 2 shows the result of pre-processing using HRE on a synthetic dataset ($m = 500$, $n = 2$, $K = 5$, $\theta_1 = 0$, $\theta_2 = 100$). Note that, the HRE works well even when clusters are arbitrary shaped.

The pre-processed data is used as the training set in optimization (solved by SMO). The extra effort required for pre-processing using R*-tree is $O(m \ln m \exp(n))$. Note that, the heuristic needs to be run only once (as a pre-processing step), and need not be run again during the entire tuning stage or later stages. Thus, increase in effort due to pre-processing is negligible when compared to decrease in effort at later stages. However, due to the $O(m \ln m \exp(n))$
effort in pre-processing, the HRE may not be feasible for very high dimensional datasets.

2.2 Cluster Labeling

The GCM described in [2] checks if every pair of points in the training set is connected or not. Checking the connectivity for all pairs of points is not necessary. Suppose the number of clusters formed as result of SVC is $\kappa$. Suppose we build the matrix $A$ using $\kappa$ points, where each point belongs to different cluster and determine the cluster labels for these $\kappa$ points. The other points can be assigned the label of one of these $\kappa$ points to which they are connected to. Now, determining the set of $\kappa$ points as mentioned above cannot be done prior to clustering. Thus, we propose to build the matrix $A$ using NBSVs only. Note that, there will be no cluster that does not contain an NBSV. It is easy to see that with this cluster labeling the computational effort needed for cluster labeling is brought down from $O(m^2 \rho n)$ to $O(m \rho^2 n)$ and the storage from $O(m^2)$ to $O(\gamma^2)$, where $\gamma$ is the number of NBSVs. Typically $\gamma, \rho \ll m$. The points eliminated from the original training data and the BSVs are assigned the labels of their nearest clusters. In the remainder of the text, this method of cluster labeling will be referred to as “Modified Graphical Connected-component Method” (MGCM).

Further, one can observe the following: except in a rare case, the nearest NBSV for any NSV is one of the NBSV in the same cluster. Thus, one can label an NSV point with the nearest NBSV’s label. When the nearest NBSV is not an NBSV of same cluster, one can then check the rest of the NBSVs for connectivity. This reduces the computational effort further to $O(m \rho n)$ (neglecting the rare cases). Employment of MGCM for cluster labeling thus reduces a lot of computational and storage effort in clustering.

2.3 Hyper-parameter Tuning

As mentioned in Sect. 1, the hyper-parameters that need to be tuned are $C$ and $\sigma$. The following method can be used for automatic tuning of these parameters: for each pair $(C, \sigma)$ ($C \in [C_{\min}, C_{\max}]$, $\sigma \in [\sigma_{\min}, \sigma_{\max}]$), the entire clustering algorithm is run to give a clustering of the given training set. Of all the clusterings thus formed, the clusterings that gave exactly $K$ clusters are picked. Let $\zeta_i$ denote the number of points in the smallest cluster (cluster with least number of points) formed in the $i^{th}$ clustering (with $K$ clusters). The tuning algorithm picks those clusterings that give maximum $\zeta_i$. This procedure ensures that we are left with clusterings of the training set that have $K$ well-formed clusters. Out of these, the algorithm picks that clustering which has the least number of BSVs. This ensures that the $\rho$ is as less as possible. The pair of $(C^*, \sigma^*)$ that gave this final clustering is the required tuned value for the hyper-parameters.
3 Numerical Experiments

The clustering method described above was implemented in C++. All experiments were carried out on an Intel Pentium 4 (3GHz) Linux computer. The optimization problem is solved using the routines in LIBSVM library [6]. The nearest neighbor queries are processed using the R*-tree routines in DBSCAN software [7]. Let \( m, n, K \) and \( \rho \) denote the number of data points in the training set, dimensionality of the data points, the desired number of clusters to be found in the training set and the number of SVs respectively.

The performance of the pre-processing method, HRE, was evaluated by doing clustering experiments on some synthetic datasets (Sect. 3.1), on a spatial dataset (Sect. 3.2) and on a real-world dataset (Sect. 3.3). We compare it with the performance of an SVC scheme that uses the MGCM for cluster labeling (and no pre-processing), the reduction in computational effort solely being due to the pre-processing with HRE. We do not present the numerical experiments that explicitly show the reduction in effort achieved by employing MGCM instead of GCM for cluster labeling. This is because, we have theoretically shown that the computational effort decreases from \( O(m^2 \rho n) \) to \( O(mn) \) and the storage from \( O(m^2) \) to \( O(\gamma^2) \). Let M-SVC denote the SVC scheme that uses MGCM for cluster labeling and no pre-processing. Let HRE-SVC denote the SVC scheme that uses heuristic HRE for pre-processing and MGCM for cluster labeling. All experiments use \( k_1 = 4, k_2 = 10, k_3 = 3 \) and \( \eta = 0.25 \). These values were chosen in order to give \( P_2 \) (refer Sect. 2.1) more weightage than \( P_1 \). Any other such choice of values for these parameters will also give similar results. The value of \( \theta_1 \) can be fixed based on the number of outliers in the dataset. If no a priori information about outliers is available it can be assigned the value 0. Good candidates for \( \theta_2 \) are the positions in the weight ordered list (see Sect. 2.1), at which there is a sudden decrease of weight. The accuracy and \( \rho \) shown in all the results are for the tuned set of hyper-parameters.

3.1 Results on Synthetic Datasets

The synthetic datasets were produced using the synthetic dataset generation ideas given in [8]. In all cases, datasets were partitioned into two disjoint sets: one is used in training (to form clusters) and the other is the testset. Since, the cluster labels for the testset are known (from data generation), calculation of accuracy can easily be done. The experiments compare the testset accuracy and run-times associated with M-SVC and HRE-SVC schemes and the results on three synthetic datasets are reported in Tables 1 - 3. The column “Time” in the tables reports the sum of pre-processing, hyper-parameter tuning and clustering times in seconds. All experiments on synthetic datasets assume 1% outliers, hence \( \theta_1 = 0.01m \). The results show that the method HRE-SVC achieves similar accuracies as M-SVC, but with lesser number of SVs (\( \rho \)) and computational effort. Figures 3 and 4 show clusterings obtained (with tuned parameters) by M-SVC and HRE-SVC methods for the first and the third synthetic datasets (2-dimensional datasets). Plot A and plot B in these figures show the scatter plots
Table 1. Results comparing the SVC methods on a dataset with $m = 2000$, $n = 2$, $K = 5$ and $\theta_1 = 20$.

<table>
<thead>
<tr>
<th>SVC scheme</th>
<th>$\theta_2$ (sec)</th>
<th>$\rho$ (sec)</th>
<th>Accuracy (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-SVC</td>
<td>408</td>
<td>99.33</td>
<td>928</td>
<td></td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>90</td>
<td>52</td>
<td>64.67</td>
<td>1</td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>140</td>
<td>87</td>
<td>99.67</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2. Results comparing the SVC methods on a dataset with $m = 4000$, $n = 20$, $K = 5$ and $\theta_1 = 40$.

<table>
<thead>
<tr>
<th>SVC scheme</th>
<th>$\theta_2$ (sec)</th>
<th>$\rho$ (sec)</th>
<th>Accuracy (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-SVC</td>
<td>276</td>
<td>100.00</td>
<td>103 020</td>
<td></td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>180</td>
<td>116</td>
<td>100.00</td>
<td>59</td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>280</td>
<td>148</td>
<td>100.00</td>
<td>681</td>
</tr>
</tbody>
</table>

Table 3. Results comparing the SVC methods on a dataset with $m = 8000$, $n = 2$, $K = 5$ and $\theta_1 = 80$.

<table>
<thead>
<tr>
<th>SVC scheme</th>
<th>$\theta_2$ (sec)</th>
<th>$\rho$ (sec)</th>
<th>Accuracy (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-SVC</td>
<td>98</td>
<td>99.96</td>
<td>16 456</td>
<td></td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>160</td>
<td>99.96</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>360</td>
<td>65</td>
<td>100.00</td>
<td>1</td>
</tr>
</tbody>
</table>
of the dataset clustered with M-SVC and HRE-SVC methods respectively. Plot C and plot D in these figures show the actual shape of clusters as identified by the methods. The darkly shaded regions in plots C and D correspond to $f(x) \leq 0$ (and hence represent the clusters). These figures show that the method HRE-SVC achieves a good reduction in $\rho$, without losing generalization. Also one can observe that the clusters are visually well separated. The results show that in such cases one can achieve huge reduction in computational effort and number of support vectors.

![Plot A](image1.png) ![Plot C](image2.png) ![Plot B](image3.png) ![Plot D](image4.png)

**Fig. 3.** Clustering on the synthetic dataset with $m = 2000$, $n = 2$, $K = 5$. Points marked ‘.’ are NSVs, ‘□’ are BSVs and ‘×’ are NBSVs.

### 3.2 Results on Spatial Dataset

Table 4 shows the comparison of performance of the M-SVC and HRE-SVC methods on a synthetic spatial dataset. Since, the labels of this dataset were known, the tuning of hyper-parameters was based on the accuracy obtained (in all other cases the tuning method presented in Sect. 2.3 was used). Figure 5 shows the clustering obtained on the dataset by HRE-SVC method. This figure shows that HRE-SVC works well on this spatial dataset. Note that though the clusters are visually separated from each other, the separation is not huge as in case of synthetic datasets. The results show that in such situations the reduction in computational effort is not as high as that in case of well separated clusters. However, a good reduction in number of support vectors can be achieved.
Fig. 4. Clustering on the synthetic dataset with $m = 8000$, $n = 2$, $K = 5$. Points marked ‘·’ are NSVs, ‘□’ are BSVs and ‘×’ are NBSVs.

Table 4. Results comparing the SVC methods on the spatial dataset ($m = 9000$, $n = 2$, $K = 9$, $\theta_1 = 0$, $\theta_2 = 3000$).

<table>
<thead>
<tr>
<th>SVC scheme</th>
<th>$\rho$</th>
<th>Accuracy (%)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-SVC</td>
<td>4.146</td>
<td>99.82</td>
<td>42.461</td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>3.066</td>
<td>99.49</td>
<td>21.601</td>
</tr>
</tbody>
</table>
3.3 Results on Real-world Dataset

The real world application of clustering presented here is the identification of land cover on the earth’s surface using multi-wavelength images from satellite (land cover is the discernible vegetation, geologic, hydrologic or anthropogenic features). The image used here is freely downloadable from the website of Global Land Cover Facility (GLCF, http://glcfapp.umiacs.umd.edu:8080/esdi/index.jsp). Each pixel of the image provided has four brightness values corresponding to green, red and near-infra-red wavelengths. The training dataset is formed by representing each pixel as a data point in $\mathbb{R}^4$, where the four dimensions correspond to the four brightness values. The original resolution of the image is $4128 \times 3768$, which was brought down to $172 \times 157$ in order to reduce the training set to a convenient size. The clusters in this dataset then correspond to regions with different land cover.

The M-SVC scheme was run on the dataset and the hyper-parameters were tuned to give four clusters. The accuracy of HRE-SVC method was determined by comparing the clustering results to that of M-SVC. Table 5 shows the comparison of performance of the methods on this real-world dataset. The results clearly show that HRE-SVC method (with $\theta_1 = 4$, $\theta_2 = 10504$) achieves almost the same clustering as M0 (98.02% accuracy), but with significantly less computational effort. However, there is no reduction in $\rho$, which suggests that there are not many redundant support vectors in the solution. The results confirm that even in such a case, HRE-SVC achieves reduction in computational effort,
primarily due to reduction in \( m \). The clustering given by HRE-SVC and the actual land cover (as given by GLCF) of the region are shown in Fig. 6. One can observe that the 3 major land covers ((A) mountain region, (B) flat land region and (C) ocean region) and (D) background portion in the image are well identified by the HRE-SVC scheme.

Visualization of this dataset showed that the clusters identified are not visually separated. In the clustering achieved with HRE-SVC 17,841 SVs of 17,858 SVs were BSVs. This high number of BSVs also suggests that the clusters are not separated. The results show that in such cases, one does not achieve any reduction in the number of support vectors. However, a reduction in computational effort can be achieved, though not as high as in case of well separated clusters.

**Fig. 6.** Figure on the left is the land cover as given by GLCF and on the right is that given by HRE-SVC.

**Table 5.** Results comparing the SVC methods on the real-world dataset (\( m = 27004, n = 4, K = 4, \theta_1 = 4, \theta_2 = 10504 \)).

<table>
<thead>
<tr>
<th>SVC scheme</th>
<th>( \rho )</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-SVC</td>
<td>17,561</td>
<td>165,062</td>
</tr>
<tr>
<td>HRE-SVC</td>
<td>17,858</td>
<td>97,630</td>
</tr>
</tbody>
</table>
4 Conclusions

In this paper, we presented a novel approach that increases the efficiency of the SVC scheme. The geometry present in the clustering problem was exploited to reduce the training set size. Whenever redundant SVs are present in the training set, the proposed method (HRE-SVC) achieves a reduction in the number of SVs. A simplification for the GCM that is computationally and storage-wise efficient was presented.

References