

Boosted Mean Shift Clustering

Yazhou Ren^{1,3}, Uday Kamath², Carlotta Domeniconi³, and Guoji Zhang⁴

¹ School of Comp. Sci. and Eng., South China Uni. of Tech., Guangzhou, China

² BAE Systems Applied Intelligence, Mclean, VA, USA

³ Department of Comp. Sci., George Mason University, Fairfax, VA, USA

⁴ School of Sci., South China Uni. of Tech., Guangzhou, China

yazhou.ren@mail.scut.edu.cn, uday.kamath@baesystems.com,

carlotta@cs.gmu.edu, magjzh@scut.edu.cn

Abstract. Mean shift is a nonparametric clustering technique that does not require the number of clusters in input and can find clusters of arbitrary shapes. While appealing, the performance of the mean shift algorithm is sensitive to the selection of the bandwidth, and can fail to capture the correct clustering structure when multiple modes exist in one cluster. DBSCAN is an efficient density based clustering algorithm, but it is also sensitive to its parameters and typically merges overlapping clusters. In this paper we propose Boosted Mean Shift Clustering (BMSC) to address these issues. BMSC partitions the data across a grid and applies mean shift locally on the cells of the grid, each providing a number of intermediate modes (iModes). A mode-boosting technique is proposed to select points in denser regions iteratively, and DBSCAN is utilized to partition the obtained iModes iteratively. Our proposed BMSC can overcome the limitations of mean shift and DBSCAN, while preserving their desirable properties. Complexity analysis shows its potential to deal with large-scale data and extensive experimental results on both synthetic and real benchmark data demonstrate its effectiveness and robustness to parameter settings.

Keywords: Mean shift clustering, density-based clustering, boosting.

1 Introduction

Clustering aims to partition data into groups, so that points that are similar to one another are placed in the same cluster, and points that are dissimilar from each other are placed in different clusters. Clustering is a key step for many exploratory tasks. In the past decades, many clustering algorithms have been proposed, such as centroid-based clustering (e.g., k -means [21] and k -medoids [18]), distribution-based clustering (e.g., Expectation-Maximization with Gaussian mixture [8]), and density-based clustering (e.g., mean shift [4], DBSCAN [10] and OPTICS [1]).

Most of the existing clustering methods need the number of clusters in input, which is typically unknown in practice. The mean shift algorithm is an appealing and nonparametric clustering technique that estimates the number of clusters

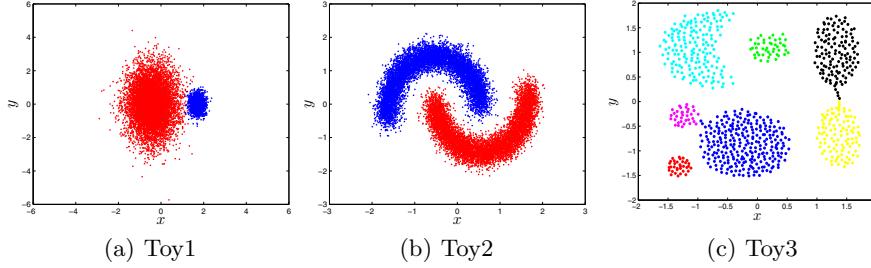


Fig. 1. Toy examples

directly from the data, and is able to find clusters with irregular shapes. It performs kernel density estimation, and iteratively locates the local maxima of the kernel mixture. Points that converge to the same mode are considered members of the same cluster [4]. The key parameter of mean shift is the kernel bandwidth. Its value can affect the performance of mean shift and is hard to set. Furthermore, mean shift may fail to find the proper cluster structure in the data when multiple modes exist in a cluster. As Fig. 1(b) shows, continuous dense regions exist in each cluster, possibly resulting in multiple modes detected by mean shift.

DBSCAN [10] is another popular density-based clustering method that does not require the number of clusters as input parameter. DBSCAN has the drawback of being sensitive to the choice of the neighborhood's radius (called *Eps*) [10]. DBSCAN tends to merge two clusters when an unsuitable *Eps* value is used, especially when the two clusters overlap, since the overlap may result in a contiguous high-density region, as shown in Fig. 1(a) and (c). (Experimental results on the three toy examples are presented in Section 5.)

Recently, a meta-algorithm known as Parallel Spatial Boosting Machine Learner (PSBML) has been introduced as a boosting algorithm for classification [17]. PSBML runs many classifiers in parallel on sampled data. The classifiers are organized in a two dimensional grid with a neighborhood structure. Data which are hard to classify are shared among the neighbor classifiers. PSBML is a robust algorithm that outperforms the underlying classifier in terms of accuracy and is less sensitive to parameter choice or noise [16]. The question we investigate in this research is whether the PSBML algorithm can be adapted to a clustering scenario to overcome the robustness issues related to parameter sensitivity as discussed above. The idea is to have a spatial grid framework as in PSBML, where a clustering algorithm such as mean shift runs at each node of the grid using local sampled data. A boosting process is applied to the local modes, which in turn are shared across the neighbors in the grid.

Specifically, we propose Boosted Mean Shift Clustering (BMSC) to address the aforementioned limitations of mean shift and DBSCAN. BMSC is an iterative and distributed version of mean shift clustering. Specifically, BMSC partitions the data across a grid, and applies mean shift locally on the cells of the grid.

Each cell outputs a set of intermediate modes (iModes in short), which represent the denser regions in the data. A mode-boosting technique is used to assign larger confidence values to those data points which are closer to the iModes. Points are then sampled with a probability that is proportional to the corresponding confidence. In successive iterations, BMSC progressively chooses data points in denser areas. Furthermore, at each iteration, DBSCAN is applied to partition all the iModes obtained so far. When DBSCAN results become stable in successive iterations, the algorithm stops. The accumulated iModes provide the “skeleton” of the data clusters and can be leveraged to group the entire data. The main contributions of this paper are summarized as follows:

- We introduce Boosted Mean Shift Clustering (BMSC) to overcome the disadvantages of mean shift and DBSCAN, while preserving their nonparametric nature. Our technique has the ability to identifying the essential structure (skeleton) of the clusters through the boosting of points around the modes.
- We present a complexity analysis to show the potential of BMSC to solve large-scale clustering tasks efficiently.
- Extensive experiments demonstrate the effectiveness and robustness of our proposed approach.

The rest of this paper is organized as follows. We review related work in Section 2 and introduce our methodology in Section 3. Section 4 presents the empirical evaluation and Section 5 discusses the experimental results. A final comment and conclusions are provided in Section 6 and 7, respectively.

2 Related Work

Mean shift [4,12] is a nonparametric feature space analysis technique that has been widely used in many machine learning applications, such as clustering [3], computer vision and image processing [4], and visual tracking [2]. It iteratively estimates the density of each point and computes the mean shift vector, which always points toward the direction of maximum increase in the density [4]. This defines a path leading to a stationary point (mode). The set of original data points that converge to the same mode defines a cluster. Mean shift uses a global fixed bandwidth, while the adaptive mean shift [5] sets different bandwidths for different data points. The convergence of mean shift procedure is guaranteed [4].

Density-based clustering methods [20] define a cluster as a set of points located in a contiguous region of high density, while points located in low-density areas are considered as noise or outliers. DBSCAN [10] is a popular clustering algorithm that relies on a density-based notion of clusters. It has only one parameter Eps , provided that the minimum number of points ($Minpts$) required to form a cluster is fixed. OPTICS [1] replaces the parameter Eps in DBSCAN with a maximum search radius and can be considered as a generalization of DBSCAN.

Adaboost [11] is the most popular boosting algorithm. It iteratively generates a distribution over the data in such a way that misclassified points by previous

classifiers are more likely to be selected to train the next weak classifier. Adaboost is an ensemble algorithm that combines these weak classifiers to form a strong classifier that has shown to be more robust than the single classifier. Parallel spatial boosting machine learning (PSBML) [16] is a recent boosting algorithm which combines concepts from spatially structured parallel algorithms and machine learning boosting techniques. Both Adaboost and PSBML solve classification problems. The technique we introduce in this work is inspired by the PSBML framework. Unlike PSBML, though, our focus here is unsupervised learning, and in particular density-based clustering.

3 Boosted Mean Shift Clustering

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ denote the data set, where n is the number of points and d is the dimensionality of each point $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})^T, i = 1, 2, \dots, n$. A hard clustering $C = \{C_1, C_2, \dots, C_{k^*}\}$ partitions \mathcal{X} into k^* disjoint clusters, i.e., $C_i \cap C_j = \emptyset$ ($\forall i \neq j, i, j = 1, 2, \dots, k^*$), and $\cup_{k=1}^{k^*} C_k = \mathcal{X}$.

3.1 Preliminary

In this section, we first give a brief review of the mean shift technique [4,5]. When using one global bandwidth h , the multivariate kernel density estimator with Kernel $K(\mathbf{x})$ ¹ is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \quad (1)$$

The *profile* of a kernel K is defined as a function $\kappa : [0, +\infty) \rightarrow \mathbb{R}$ such that $K(\mathbf{x}) = c \cdot \kappa(\|\mathbf{x}\|^2)$, where the positive constant c makes $K(\mathbf{x})$ integrate to one. Then, the sample point estimator (1) becomes

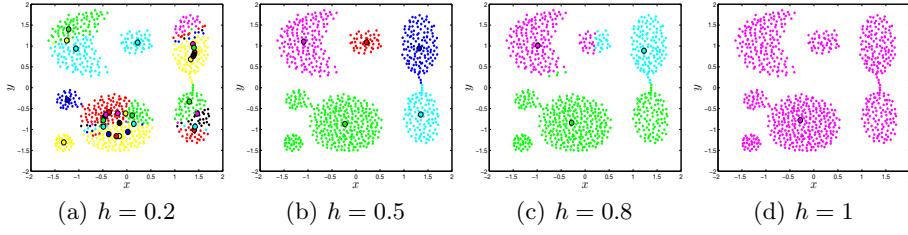
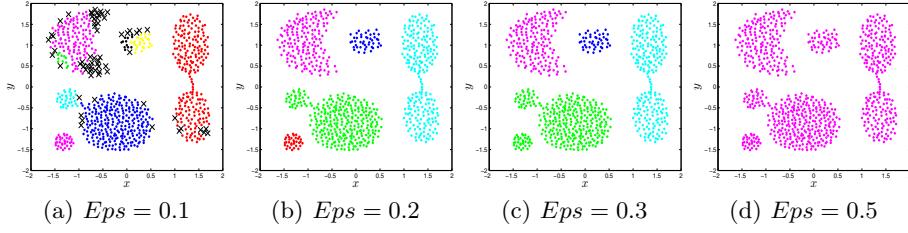
$$\hat{f}(\mathbf{x}) = \frac{c}{nh^d} \sum_{i=1}^n \kappa\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \quad (2)$$

By taking the gradient of $\hat{f}(\mathbf{x})$ we obtain

$$\nabla \hat{f}(\mathbf{x}) = \frac{2c}{nh^{d+2}} \left[\sum_{i=1}^n g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \right] \times \underbrace{\left[\frac{\sum_{i=1}^n \mathbf{x}_i g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^n g\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)} - \mathbf{x} \right]}_{\text{mean shift vector}} \quad (3)$$

where $g(x) = -\kappa'(x)$, provided that the derivative of κ exists. The first part of Eq. (3) is a constant, and the factor in bracket is the mean shift vector, which

¹ We use a Gaussian kernel in this paper.

**Fig. 2.** Toy3: Clustering results of mean shift for different values of h **Fig. 3.** Toy3: Clustering results of DBSCAN for different values of Eps

always points towards the direction of the greatest increase in density. Using the mean shift vector, a sequence of estimation points $\{\mathbf{y}_t\}_{t=1,2,\dots}$ is computed

$$\mathbf{y}_{t+1} = \frac{\sum_{i=1}^n \mathbf{x}_i g(\|\frac{\mathbf{y}_t - \mathbf{x}_i}{h}\|^2)}{\sum_{i=1}^n g(\|\frac{\mathbf{y}_t - \mathbf{x}_i}{h}\|^2)} \quad (4)$$

The starting point \mathbf{y}_1 is one of the points \mathbf{x}_i . The point that $\{\mathbf{y}_t\}_{t=1,2,\dots}$ converges to is considered as the mode of \mathbf{y}_1 . The points that converge to the same mode are considered members of the same cluster. Please refer to [4] for more details.

3.2 The Algorithm

The performance of mean shift is sensitive to the choice of the bandwidth h . To demonstrate this fact, Fig. 2 shows the clustering results of mean shift for different values of h on a two-dimensional dataset containing clusters of different shapes (called Toy3 in our experiments). The dark circles in Fig. 2 correspond to the (global) modes generated by mean shift. For any given mode, the points that converge to it are marked with the same color, and they define a cluster. As shown in Fig. 2 (a), when the value $h = 0.2$ is used, mean shift finds several modes and therefore detects a large number of clusters. Larger values of h lead to fewer modes, and to the merging of separate clusters.

DBSCAN is another popular density-based clustering algorithm (refer to [10] for more details) which is also sensitive to its input parameters and is likely to merge overlapping clusters. Fig. 3 gives the results of DBSCAN on Toy3 for different values of the parameter Eps . Here $Minpts$ is set to 4. Points of the

same color belong to the same cluster. In Fig. 3(a) the points marked as “ \times ” are classified as noisy points by DBSCAN. More clusters and more noisy points are found by DBSCAN when $Eps = 0.1$. The detected noisy points are actually members of a cluster, and should not be considered as outliers. The larger Eps becomes, the more clusters are merged by DBSCAN. Eventually, for $Eps = 0.5$, DBSCAN detects only one cluster. The two rightmost clusters are also merged in Fig. 3(a), when $Eps = 0.1$ is used.

To overcome these limitations of mean shift and DBSCAN, while retaining their nonparametric nature, we propose the Boosted Mean Shift Clustering (BMSC) algorithm. We seek to capture the underlying group structure of the data by selecting the subset of data that provides the *skeleton* of the clusters. To achieve this goal, we iteratively compute modes relative to sampled data in a distributed fashion, and boost points proportionally to their distance from the modes. To achieve this goal, BMSC partitions the original data across a grid, and applies mean shift locally on the cells of the grid to search for the denser regions iteratively. The details are described below.

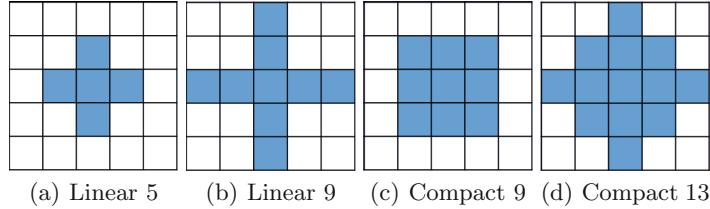


Fig. 4. Various neighborhood structures

Mode-boosting. BMSC first partitions the data uniformly across the cells of a two dimensional grid structure², as depicted in Fig. 4 [16,17,22] (Line 1 of Algorithm 1). The cells of the grid interact with the neighbors, where the neighborhood structure is user defined, as shown in Fig. 4³. Each cell applies a mean shift clustering algorithm on its local data. The mean shift algorithm outputs intermediate modes (iModes), which are located within dense regions (Line 6). Each cell uses its own iModes to assign confidence values to points assigned to the cell itself and to its neighbors. Specifically, given a set of iModes generated at $cell_j$, i.e., $iModes^{(j)} = \{iMode_1^{(j)}, \dots, iMode_L^{(j)}\}$, we assign each local point (in $cell_j$ itself or in any neighboring cells) to the nearest iMode in $iModes$. For the points assigned to the same $iMode_l^{(j)}$ ($l = 1, \dots, L$), we compute the confidence value of point i w.r.t. $cell_j$ as

$$conf_i^{(j)} = 1 - \frac{dis_i - min_dis}{max_dis - min_dis} \quad (5)$$

² The dimensionality of the grid affects the size of the neighborhood, and therefore the speed at which data is propagated through the grid. Note that the dimensionality d of the data can be arbitrary.

³ In this paper, we use the ‘Linear 5’ structure.

Algorithm 1. Boosted Mean Shift Clustering

Input: \mathcal{X} , width, height, h , Eps.
Output: The final clustering result cl_final .

```

1: INITIALIZEGRID( $\mathcal{X}$ , width, height); //Distribute  $\mathcal{X}$  over  $I = \text{width} \times \text{height}$  cells.
2:  $iModes \leftarrow \emptyset$ ; //Initialize the set of intermediate modes.
3:  $counter \leftarrow 1$ ;
4: repeat
5:   for  $j \leftarrow 1$  to  $I$  do
6:      $newiModes \leftarrow \text{MEANSHIFT}(cellData_j, h)$ ;
7:      $iModes.\text{APPEND}(newiModes)$ ;
8:   end for
9:   CONFIDENCEASSIGNMENT(); // Assign confidence values to points in each cell
   via Eqs. (5) and (6).
10:  for  $j \leftarrow 1$  to  $I$  do
11:     $CollectedData \leftarrow \text{COLLECTNEIGHBORDATA}(j) \cup cellData_j$ ;
12:     $cellData_j \leftarrow \text{WEIGHTEDSAMPLING}(CollectedData)$ ; // Update  $cellData_j$ .
13:  end for
14:   $[cl\_iModes, \text{numberOfClustersDetected}] \leftarrow \text{DBSCAN}(iModes, Eps)$ ;
   // $cl\_iModes$  is the clustering result of  $iModes$ .
15:  if ( $\text{numberOfClustersDetected} == \text{lastnumberOfClustersDetected}$ ) then
16:     $counter++$ ;
17:  else
18:     $counter \leftarrow 1$ ;
19:  end if
20: until  $counter == 3$ 
21:  $cl\_final \leftarrow \text{DATAASSIGNMENT}(\mathcal{X}, iModes, cl\_iModes)$ ; //Assign points in  $\mathcal{X}$ .
22: return  $cl\_final$ .

```

where dis_i is the distance between point i and $iMode_l^{(j)}$, min_dis and max_dis are the minimum and maximum distances between the corresponding points and $iMode_l^{(j)}$, respectively. $conf_i^{(j)} \in [0, 1]$. Intuitively, points near $iMode_l^{(j)}$ obtain larger confidence values, while those far away from $iMode_l^{(j)}$ are assigned smaller confidence values. Since a point \mathbf{x}_i is a member of the neighborhood of multiple cells, an ensemble of confidence's assessments is obtained. We set the final confidence to the maximum confidence value obtained from any cell:

$$conf_i = \max_{j \in N_i} conf_i^{(j)} \quad (6)$$

where N_i is a set of indices of the neighbors of the cell to which point \mathbf{x}_i belongs (Line 9).

The confidence values are used to select a sample of the data, locally at each cell, via a weighted sampling mechanism. Specifically, for each $cell_j$, all points in the cell and in its neighbors are collected. The larger the confidence value credited to a point \mathbf{x}_i is (i.e., the closer \mathbf{x}_i is to some iMode), the larger is the probability for \mathbf{x}_i to be selected (Lines 11-12). As such, copies of points with larger confidence values will have higher probability of being selected, while points with low confidence will

have a smaller probability of being selected. The sample size at each cell is kept constant at each epoch. Note that duplicate points may appear in a given cell, and the same point may appear in different cells.

Stopping Criterion. At each epoch, BMSC applies mean shift locally at each cell, thus generating a set of new iModes. We combine the new iModes with the iModes generated during the previous iterations, and apply DBSCAN on such updated set of iModes (Line 14). This process is repeated until the number of detected clusters by DBSCAN does not change for three consecutive iterations. At each iteration, the set of iModes produced so far gives a representation of the original data. A stable partition (i.e., a consistent number of detected clusters is obtained for three iterations) of iModes indicates a stable partition of the original data. In practice we have observed that the number of distinct points at each epoch quickly decreases, and BMSC always stops in less than 20 iterations in our experiments. The convergence of BMSC is empirically shown.

When BMSC stops, DBSCAN gives a partition of all the iModes. We then assign each original data point to the cluster to which its nearest iMode belongs. This gives the final clustering of the original data (Line 21). The pseudo-code of BMSC is given in Algorithm 1.

Computational Complexity. The computational complexity of estimating the density and computing the mean shift vector for one data point is $O(n)$, where n is the total number of data. Let T_1 be the maximum number of iterations it takes to compute the mode of any point in \mathcal{X} . Then the complexity of mean shift on the whole data is $O(T_1 n^2)$. The running time of BMSC is driven by the complexity of mean shift running on the cells of the grid and of DBSCAN running on the obtained iModes. BMSC applies mean shift locally on every cell, each with complexity $O(T_1 m^2)$, where $m = \frac{n}{I}$ and $I = \text{width} \times \text{height}$ is the number of cells in the spatial grid. The runtime complexity of DBSCAN on \mathcal{X} is $O(n^2)$ and it can be reduced to $O(n \log n)$ if one uses R*-tree to process a range query [20]. Let s be the number of obtained iModes and T_2 be the number of iterations when BMSC stops, then the total computation complexity of BMSC is $O(T_2(I T_1 m^2 + s \log s))$. T_2 is empirically proved to be small, and $m \ll n$, $s \ll n$ when n is large. Thus, the computational complexity of BMSC is lower than mean shift and DBSCAN when dealing with large scale data. BMSC can be further speeded up with a parallel implementation. The complexity can be reduced to $O(T_2 \max\{T_1 m^2, s \log s\})$ with an I multi-thread process, which makes BMSC available for large-scale clustering tasks.

4 Experimental Setup

Datasets. We conducted experiments on three toy examples and ten real-world data sets to evaluate the performance of BMSC and comparing methods. Table 1 provides the characteristics of all the datasets used in our experiments. The toy examples are shown in Fig. 1. The two classes of toy

Table 1. Datasets used in the experiments

Data	#points	#features	#classes
Toy1	10000	2	2
Toy2	14977	2	2
Toy3	788	2	7
2D2K	1000	2	2
8D5K	1000	8	5
Letter	1555	16	2
Satimage	4220	36	4
Symbols	1020	398	6
KDD99	25192	38	22
Banknote	200	6	2
Chainlink	400	3	2
Image_Seg	990	19	3
Wall	5456	2	4

example 1 (Toy1), which were generated according to multivariate Gaussian distributions, consist of 8,000 and 2,000 points, respectively. The mean vector and the covariance matrix of the left class are (-10,0) and (10 0; 0 10), while those of the right class are (2,0) and (1 0; 0 1), respectively. Toy example 2 (Toy2, two-moons data), Banknote, and ChainLink contain two classes and are available at <http://www.mathworks.com/matlabcentral/fileexchange/34412-fast-and-efficient-spectral-clustering>. 400 points (200 points per class) of ChainLink were randomly chosen in the experiments. Toy3 (Aggregation data) was downloaded from <http://cs.joensuu.fi/sipu/datasets/> and was used in [13]. 2D2K and 8D5K were two datasets used in [23] and were downloaded from <http://strehl.com/>. Symbols is a UCR time series data [19]. NSL-KDD data set retains all the important statistical characteristics of KDDCup-99. A subset which contains 25192 instances of 22 classes (1 normal class and 21 attack types) was downloaded from <http://ns1.cs.unb.ca/NSL-KDD/> and the 38 numerical features were used for the experiments. Letter, Satimage, Image_Seg (Image segmentation) and Wall (Wall-following robot navigation data) are all available from the UCI repository (<http://archive.ics.uci.edu/ml/index.html>). The letters ‘A’ and ‘B’ were selected from the Letter database. The first 4 classes of Satimage and the first three classes of Image_Seg were used in our experiments. For each dataset, each feature was normalized to have zero mean value and unit variance.

Evaluation Criteria. We chose Rand Index (RI) [14], Adjusted Rand Index (ARI) [14], and Normalized Mutual Information (NMI) [23] as evaluation criteria since the label information of data are known. The label information is only used to measure the clustering results, and is not used during the clustering process. Both RI and NMI range from 0 to 1, while ARI belongs to [-1,1]. A value 1 of RI/ARI/NMI indicates a perfect clustering result.

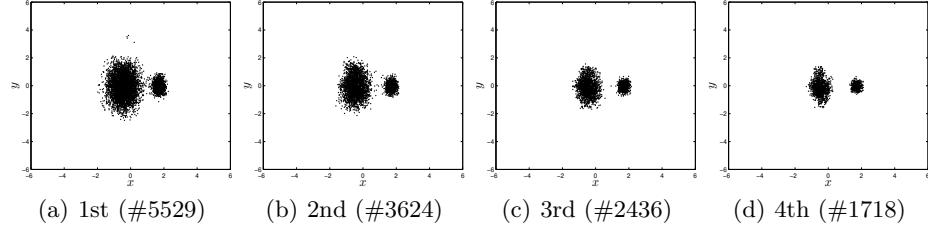
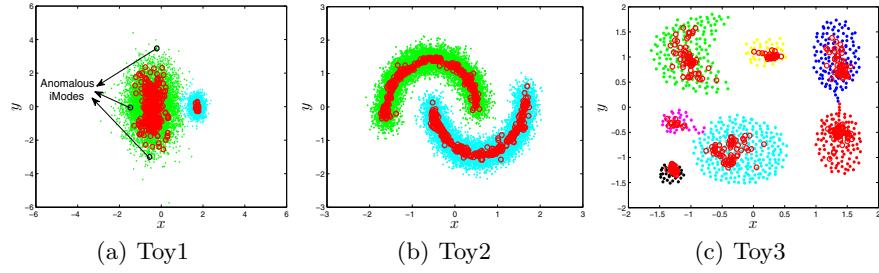
Experimental Settings. As shown in Algorithm 1, BMSC requires four parameters in input: *width*, *height*, the bandwidth *h* for mean shift, and *Eps* for DBSCAN. We used a 5×5 spatial grid for Toy1, Toy2, and KDD99, and a 3×3 grid for all the other data sets. We set *h* to the average distance between each point and its *k*-th nearest neighbor, where $k = \alpha\sqrt{n_Data}$. Here *n_Data* is the size of the sample assigned to a cell of a grid for BMSC, or the size of the whole dataset when we run mean shift on the entire collection. The value of α also affects the bandwidth; a larger α value corresponds to a larger global bandwidth *h*. α is always set to 0.5 for both BMSC and mean shift in our experiments. We set *Eps* = 0.5 for BMSC and DBSCAN on the three 2-dimensional toy datasets. Sensitivity analysis of parameters α and *Eps* is discussed in Section 5.4. To set the *Eps* parameter on real data, we consider the iModes generated after the first epoch of BMSC, and compute the distance to the 4-th nearest iMode [10] for each of the iModes. We then choose the median value of all such distances as the value of *Eps* for DBSCAN in all the successive iterations. When DBSCAN is run on the whole data, the 4-th nearest neighbor distances are computed with respect to the entire collection of data, and *Eps* is again set to the median of those values. The *MinPts* value of DBSCAN is always set to four in our experiments, as done in [10].

Besides mean shift and DBSCAN [7,10], we also performed comparisons against several other clustering algorithms: OPTICS [1,6], *k*-means [21], LAC [9], Aver-l (average-linkage clustering) [15], and EM (with a Gaussian mixture) [8]. OPTICS is a density based clustering algorithm which creates an augmented ordering of the data representing its clustering structure, and then retrieves DBSCAN clusters as the final clustering result. When OPTICS uses DBSCAN to extract clusters, the parameters were set as in DBSCAN itself. Both DBSCAN and OPTICS may output noisy clusters. *k*-means, LAC, Aver-l, and EM require the number of clusters in input, which we set equal to the number of classes in the data. LAC requires an additional parameter (weight of the regularization term; see [9] for details), which we set to 0.2 throughout our experiments. Mean shift, DBSCAN, OPTICS, and Aver-l are deterministic for fixed parameter values. For the remaining methods, the reported values are the average of 20 independent runs. One-sample *t*-test and paired-samples *t*-test were used to assess the statistical significance of the results at 95% significance level.

5 Results and Analysis

5.1 Results on Toy Examples

To illustrate the effectiveness of BMSC, we first conducted experiments on the three toy datasets. Fig. 5 shows the data selected by BMSC at different epochs on Toy1 in one independent run. For this run, BMSC stops at the fourth iteration. ‘(#5529)’ in Fig. 5(a) means that at this epoch 5529 points are selected. The number of (distinct) points in each iteration greatly decreases, and points around the densest regions are more likely to survive. At successive iterations, the data becomes better separated, even though the original two classes overlap. Fig. 6

**Fig. 5.** Data selected by BMSC on Toy1 at different epochs**Fig. 6.** BMSC clustering results on the three toy examples

gives the final clustering results of BMSC on the three toy examples for a given run. The red circles represent the iModes accumulated in successive iterations when BMSC stops. Points with the same color belong to the same cluster. The iModes successfully capture the structure of the different clusters. As a result, BMSC achieves a close to perfect performance on these datasets. When two clusters overlap (like in Toy1), the corresponding iModes are still well separated, and therefore easy to partition. When performing DBSCAN on the set of iModes, some iModes may be classified as noise. For example, three such iModes occur in Fig. 6(a). iModes detected as anomalous are discarded and not used to partition the whole data.

Table 2. Results on toy examples

Data		BMSC	MS	DBSCAN	OPTICS	<i>k</i> -means	LAC	Aver-l	EM
Toy1	RI	0.9955	0.3977	0.6792	0.6796	0.9294	0.9299	0.6798	0.9978
	ARI	0.9897	0.0755	-0.0009	-0.0004	0.8432	0.8443	-0.0001	0.9949
	NMI	0.9709	0.3846	0.0027	0.0018	0.7614	0.7626	0.0010	0.9833
Toy2	RI	0.9995	0.5380	0.5001	0.5001	0.8201	0.8185	0.8608	0.6584
	ARI	0.9990	0.0763	0.0000	0.0000	0.6401	0.6370	0.7216	0.3168
	NMI	0.9970	0.4495	0.0000	0.0000	0.5311	0.5280	0.6832	0.3675
Toy3	RI	0.9891	0.8697	0.2165	0.2165	0.9096	0.9006	0.9971	0.9063
	ARI	0.9686	0.5096	0.0000	0.0000	0.7061	0.6781	0.9913	0.7121
	NMI	0.9711	0.7925	0.0000	0.0000	0.8316	0.8115	0.9869	0.8379

Table 3. Results on real data (RI)

Data	BMSC	MS	DBSCAN	OPTICS	<i>k</i> -means	LAC	Aver-l	EM
2D2K	0.9560	0.9522	0.6962	0.4995	0.9531	0.9538	0.9250	0.9078
8D5K	1.0000	1.0000	0.9094	0.8164	0.9359	0.9448	1.0000	0.9488
Letter	0.8928	0.8909	0.6479	0.5133	0.8762	0.8767	0.8875	0.7054
Satimage	0.8216	0.8044	0.7305	0.5271	0.8030	0.8051	0.5370	0.7975
Symbols	0.9081	0.9007	0.8102	0.3922	0.8842	0.8773	0.6275	0.8193
KDD99	0.7699	0.7416	0.6843	0.6029	0.6777	0.6755	0.3966	0.7106
Banknote	0.9694	0.9510	0.8009	0.5934	0.9228	0.9261	0.4975	0.8892
Chainlink	0.7475	0.5626	0.5378	0.5264	0.5431	0.5410	0.5550	0.7350
Image_Seg	0.9073	0.8695	0.6978	0.7036	0.8091	0.8112	0.3461	0.7446
Wall	0.7244	0.7131	0.6609	0.5905	0.7055	0.7092	0.3656	0.6530

Table 4. Results on real data (ARI)

Data	BMSC	MS	DBSCAN	OPTICS	<i>k</i> -means	LAC	Aver-l	EM
2D2K	0.9119	0.9043	0.3922	0.0000	0.9062	0.9075	0.8499	0.8156
8D5K	1.0000	1.0000	0.6977	0.4010	0.8269	0.8446	1.0000	0.8558
LetterAB	0.7856	0.7817	0.2957	0.0265	0.7524	0.7533	0.7749	0.4109
Satimage	0.5631	0.5452	0.3508	0.0831	0.5234	0.5293	0.2242	0.5201
Symbols	0.7042	0.6645	0.4818	0.0071	0.6339	0.6186	0.2369	0.4566
KDD99	0.4684	0.3933	0.2979	0.0164	0.2229	0.2160	-0.0003	0.3154
Banknote	0.9387	0.9020	0.6011	0.1850	0.8456	0.8521	0.0000	0.7788
Chainlink	0.4944	0.1233	0.0737	0.0542	0.0865	0.0822	0.1110	0.4701
Image_Seg	0.7843	0.6775	0.2271	0.2078	0.6273	0.6218	0.0012	0.4812
Wall	0.2893	0.2437	0.1434	0.0532	0.3697	0.3761	0.0240	0.2656

Table 2 shows the results of the different algorithms on the toy examples using the three evaluation measures. In each row, the significantly best and comparable results are highlighted in boldface. On these datasets, BMSC improves upon mean shift, DBSCAN, and OPTICS by a large margin. As expected, EM gives the best performance on Toy1, which is a mixture of two Gaussians. Aver-l works quite well on Toy3. But both EM and Aver-l require the number of clusters in input, and their performance degrades on the other data. BMSC significantly outperforms all the comparing methods on Toy2, and it's the only approach that works well on all three toy datasets. The poor performance of *k*-means and LAC is mainly caused by the unbalanced data in Toy1, and the irregular shapes of clusters in Toy2 and Toy3.

5.2 Results on Real Data

This section evaluates the performance of the comparing methods on several real datasets. The RI and ARI values are shown in Tables 3 and 4, respectively. The best and comparable results are shown in boldface. In general, a better result on RI indicates a better result on ARI and NMI. But this is not always the case. Lets consider the Wall data for example. BMSC gives the best RI value,

while LAC gives the best ARI value. BMSC, mean shift, and Aver-l do a perfect job in clustering 8D5K. For the remaining datasets, BMSC significantly outperforms all the other comparing methods. It's worth observing that, in terms of the ARI measure, BMSC outperforms mean shift by a considerable margin on Symbols, KDD99, Banknote, Chainlink, Imag_Seg, and Wall. Similar results were obtained on NMI and are not reported due to the limited space.

5.3 Performance Analysis of BMSC

The average number of iterations of BMSC and the average number of detected clusters by BMSC are shown in Table 5. The table shows that BMSC stops after a small number of iterations. Comparing the number of detected clusters (#clusters) and the actual number of classes (#classes), we can see that a larger number of clusters is detected by BMSC on KDD99, Chainlink, and Wall⁴. For this reason, lower values of RI and ARI are obtained on these three datasets, as shown in Tables 3 and 4. The number of clusters detected by BMSC is similar to the number of classes on Toy3, Satimage, Symbols, and Image_Seg. BMSC always detects a number of clusters that matches the number of classes on Toy1, Toy2, 2D2K, 8D5K, Letter, and Banknote datasets. This indicates that BMSC is capable of automatically finding a reasonable number of clusters.

Table 5. Performance Analysis of BMSC

Data	#iterations	#clusters	#classes
Toy1	3.30	2±0.00	2
Toy2	3.00	2±0.00	2
Toy3	4.45	6.80±0.42	7
2D2K	3.75	2±0.00	2
8D5K	3.15	5±0.00	5
Letter	3.35	2±0.00	2
Satimage	4.60	4.10±0.79	4
Symbols	5.50	6±1.49	6
KDD99	14.55	41.45±5.15	22
Banknote	3.05	2±0.00	2
Chainlink	4.15	4.65±1.23	2
Image_Seg	4.45	3.75±0.55	3
Wall	6.65	12.20±1.61	4

5.4 Sensitivity Analysis of the Parameters α and Eps

We tested the sensitivity of BMSC w.r.t. the parameters α and Eps on three datasets, namely Toy1, Toy2, and Satimage. We first tested the sensitivity of α which controls the kernel bandwidth in BMSC and mean shift. $Eps = 0.5$ was set for BMSC. The test range of α is [0.05, 0.6] and Fig. 7 gives the results.

⁴ This is not surprising since in practice there may not be a one-to-one correspondence between classes and clusters.

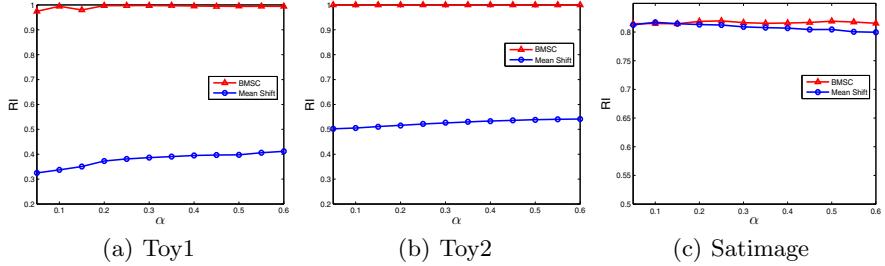


Fig. 7. Sensitivity analysis of the α parameter (RI)

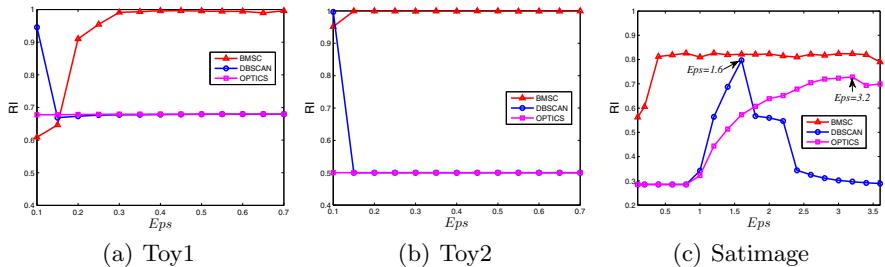


Fig. 8. Sensitivity analysis of parameter Eps (RI)

Mean shift performs slightly better as α increases on Toy1 and Toy2, and a general reduction in performance is observed with a larger α on Satimage. BMSC is stable and achieves good performance throughout. A larger α value leads to fewer modes found by mean shift, while a smaller α value results in more modes, thus generating more clusters in general. BMSC is more robust to the different values of α because α only affects the number of iModes generated by each cell, and such iModes are then linked together by DBSCAN.

We further compared BMSC against DBSCAN and OPTICS for different values of Eps . In these experiments, $\alpha = 0.5$ for BMSC. Tested ranges of Eps are [0.1, 0.7] for Toy1 and Toy2, and [0.1, 3.6] for Satimage. The results are given in Fig. 8. The performance of BMSC increases as the value of Eps increases, and it is stable for a wide range of Eps values. The main reason for this behavior is that BMSC applies DBSCAN on the iModes, rather than on the whole data. iModes of different clusters are well separated, even though the original clusters may overlap. DBSCAN works well for $Eps = 0.1$, and OPTICS always performs poorly on Toy1 and Toy2. On Satimage, the performance of both DBSCAN and OPTICS increases for larger Eps values, and reaches its peak at $Eps = 1.6$ and $Eps = 3.2$, respectively. After that, the performance drops. This shows that DBSCAN and OPTICS are sensitive to the choice of values for Eps , while our BMSC technique is robust to parameter settings. This provides insight to the superior performance of BMSC against DBSCAN and OPTICS obtained in the previous experiments.

6 Discussion

To overcome some of the limitations of mean shift, one may run mean shift on the whole data with a small bandwidth, thus generating a large number of modes. The modes can then be merged to obtain the final clustering results. But this approach has two major disadvantages: (1) Running mean shift on a large-scale data is of high complexity; and (2) With a fixed bandwidth, only one mode (the local maxima) will be found in dense areas. In contrast, our BMSC is able to find contiguous intermediate modes in dense areas. It achieves this through the partitioning of the data across a spatial grid and through the iterative process of collecting the iModes.

7 Conclusion

In this work we have introduced Boosted Mean Shift Clustering (BMSC), a nonparametric clustering method that overcomes the limitations of mean shift and DBSCAN, namely the sensitivity to parameters' values and the difficulty of handling overlapping clusters. At the same time, BMSC preserves the ability of automatically estimating the number of clusters from the data and of handling clusters of irregular shapes. The effectiveness and stability of BMSC are demonstrated through extensive experiments conducted on synthetic and real-world datasets. We are interested in extending the framework introduced here to other clustering methodologies, e.g. centroid-based, as well as to a semi-supervised scenario. As mentioned earlier in our discussion on computational complexity, BMSC can be easily parallelized via a multi-thread implementation (one thread per cell). We will proceed with such implementation in our future work and test the achieved speed-up for the big data clustering problems.

Acknowledgement. This paper is in part supported by the China Scholarship Council (CSC).

References

1. Ankerst, M., Breunig, M.M., Peter Kriegel, H., Sander, J.: OPTICS: Ordering points to identify the clustering structure. In: SIGMOD, pp. 49–60. ACM Press (1999)
2. Avidan, S.: Ensemble tracking. TPAMI 29(2), 261–271 (2007)
3. Cheng, Y.: Mean shift, mode seeking, and clustering. TPAMI 17(8), 790–799 (1995)
4. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. TPAMI 24(5), 603–619 (2002)
5. Comaniciu, D., Ramesh, V., Meer, P.: The variable bandwidth mean shift and data-driven scale selection. In: ICCV, pp. 438–445 (2001)
6. Daszykowski, M., Walczak, B., Massart, D.: Looking for natural patterns in analytical data. Part 2: Tracing local density with OPTICS. Journal of Chemical Information and Computer Sciences 42(3), 500–507 (2002)

7. Daszykowski, M., Walczak, B., Massart, D.L.: Looking for natural patterns in data. Part 1: Density based approach. *Chemometrics and Intelligent Laboratory Systems* 56(2), 83–92 (2001)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. *Journals of the Royal Statistical Society, Series B* 39(1), 1–38 (1977)
9. Domeniconi, C., Gunopoulos, D., Ma, S., Yan, B., Al-Razgan, M., Papadopoulos, D.: Locally adaptive metrics for clustering high dimensional data. *DMKD* 14(1), 63–97 (2007)
10. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD*, pp. 226–231 (1996)
11. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55, 119–139 (1997)
12. Fukunaga, K., Hostetler, L.D.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory* 21(1), 32–40 (1975)
13. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. *TKDD* 1(1), 1–30 (2007)
14. Hubert, L., Arabie, P.: Comparing partitions. *Journal of Classification* 2(1), 193–218 (1985)
15. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. *ACM Computing Surveys* 31(3), 264–323 (1999)
16. Kamath, U., Domeniconi, C., Jong, K.A.D.: An analysis of a spatial ea parallel boosting algorithm. In: *GECCO*, pp. 1053–1060 (2013)
17. Kamath, U., Kaers, J., Shehu, A., De Jong, K.A.: A spatial EA framework for parallelizing machine learning methods. In: Coello, C.A.C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) *PPSN 2012, Part I*. LNCS, vol. 7491, pp. 206–215. Springer, Heidelberg (2012)
18. Kaufman, L., Rousseeuw, P.: Clustering by Means of Medoids. Faculty of Mathematics and Informatics (1987)
19. Keogh, E., Zhu, Q., Hu, B., Hao, Y., Xi, X., Wei, L., Ratanamahatana, C.A.: The UCR Time Series Classification/Clustering Homepage (2011), http://www.cs.ucr.edu/~eamonn/time_series_data/
20. Kriegel, H.-P., Kroger, P., Sander, J., Zimek, A.: Density-based clustering. *DMKD* 1(3), 231–240 (2011)
21. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297. University of California Press (1967)
22. Sarma, J., Jong, K.: An analysis of the effects of neighborhood size and shape on local selection algorithms. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) *PPSN 1996*. LNCS, vol. 1141, pp. 236–244. Springer, Heidelberg (1996)
23. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *JMLR* 3, 583–617 (2002)