Mode-seeking by Medoidshifts

Yaser Ajmal Sheikh
Robotics Institute
Carnegie Mellon University
yaser@cs.cmu.edu

Erum Arif Khan
Department of Computer Science
University of Central Florida
ekhan@cs.ucf.edu

Takeo Kanade
Robotics Institute
Carnegie Mellon University
tk@cs.cmu.edu

Abstract

We present a nonparametric mode-seeking algorithm, called medoidshift, based on approximating the local gradient using a weighted estimate of medoids. Like meanshift, medoidshift clustering automatically computes the number of clusters and the data does not have to be linearly separable. Unlike meanshift, the proposed algorithm does not require the definition of a mean. This property allows medoidshift to find modes even when only a distance measure between samples is defined. In this sense, the relationship between the medoidshift algorithm and the meanshift algorithm is similar to the relationship between the k-medoids and the k-means algorithms. We show that medoidshifts can also be used for incremental clustering of growing datasets by recycling previous computations. We present experimental results using medoidshift for image segmentation, incremental clustering for shot segmentation and clustering on nonlinearly separable data.

1. Introduction

Data is being produced at ever increasing rates, recording everything from weather patterns to internet traffic. One approach to discovering the inherent structure of a collected data set is clustering - the task of partitioning a data set $\mathcal{D}$ of N points into different coherent clusters $C_1, \cdots C_J$, such that $C_i \cap C_j = \emptyset$, and $\bigcup_{j=1}^{J} C_j = \mathcal{D}$. Clustering has been widely applied in fields like computer vision, speech processing, psychology, data mining and bioinformatics. In applications where the volume of data continually changes or grows, it is important that clustering algorithms allow for *incremental* clustering - the task of exactly updating a clustering at the incidence of new data samples and the exitance of some existing data samples. In computer vision, analysis of data that is updated incrementally finds natural application in online video processing. For instance, instead of segmenting videos frame by frame or in batch mode, efficient and temporally consistent segmentation can be obtained through incremental clustering.

In this paper, we propose a nonparametric clustering approach called the medoidshift algorithm. It is a mode-seeking procedure that computes shifts towards areas of greater data density using local weighted medoids[1]. The use of medoids to discover structure in data is natural since, locally, the medoid can be considered a good representative of its neighborhood. Unlike means, medoids do not need an explicit feature space and require only a valid distance measure to be defined. Medoids have previously been used in clustering applications in two papers by Kaufman and Rousseeuw, who first proposed PAM (Partitioning Around Medoids) in [11] and then extended this algorithm by using sampling to handle large datasets with CLARA (Clustering Large Applications) also described in [11]. CLARA was further improved by Ng and Han in [15]. Like k-means, these algorithms require the number of clusters to be pre-defined and are not invariant to changes in initializations. The medoidshift algorithm automatically computes the number of clusters and does not need initialization.

The operating concept of the proposed algorithm is similar to the meanshift algorithm which was first proposed by Fukunaga and Hostetler in [8], and further studied by Cheng in [4], Carreira-Perpiñán in [2] and Comaniciu and Meer in [5]. Like meanshift, the medoidshift algorithm computes the number of clusters during execution. However, the medoidshift algorithm has three principal advantages over the meanshift algorithm. First, the computations performed during an earlier clustering do not have to be discarded at the incidence of new samples or the exitance of some existing samples. This allows medoidshift to be used for incremental clustering, for applications like online keyframe selection in video. Second, the proposed algorithm does not require the definition of a mean and can operate directly on a distance matrix, irrespective of the original space in which the samples are distributed. This property allows medoidshift to find modes even when *only* a distance measure (such as the Earth Mover's Distance or the Proportional Transportation Distance) between samples is defined. In this sense, the relationship between the medoidshift algorithm and meanshift algorithm is similar to the k-medoids algorithm ([11]) and the k-means algorithm ([13]). Finally, the need for heuristic terminating conditions in meanshift

[1] The medoid is defined as the the most centrally located point in a set of samples, i.e. it has the minimum distance from all other samples

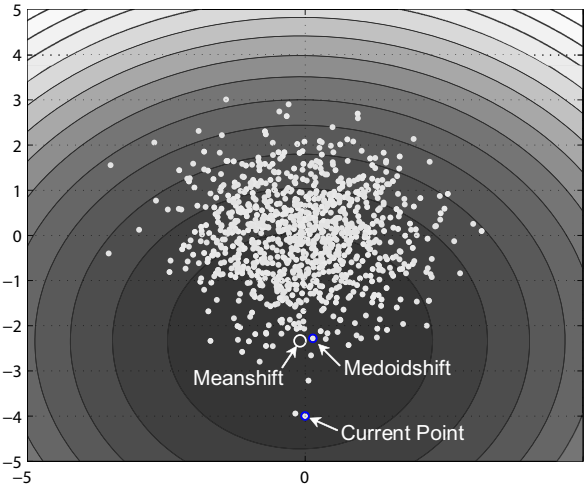


Figure 1. Contour plot of Equation 2 for the current point. Meanshift selects the location that minimizes this function exactly while medoidshift selects the data point that best minimizes it.

is eliminated. One drawback of medoidshift is its higher computational complexity, $\mathcal{O}(N^3)$. To compensate for this higher cost, we cast the core of the algorithm as a matrix multiplication, thereby decreasing complexity to $\mathcal{O}(N^{2.38})$ ([7]).

An earlier work by Koontz *et al*. in [12] describes a graph-theoretic approach to clustering that also operates on a similar principle to the medoidshift algorithm. However, compared to the approach proposed in this paper, the algorithm in [12] is sensitive to the order in which points are considered and additional checks are needed to ensure convergence. Spectral clustering methods, such as normalized cuts ([16]), require the number of clusters to be prespecified and another clustering method (e.g. k-means) is needed to finally cluster the data in the spectral domain. Both these issues introduce arbitrariness in design and can introduce problems in practice. Traditional incremental algorithms such as [1] and [14] are not invariant to the order of input patterns. More recently several approaches have been proposed for incremental clustering that are more robust to changes in the order of input including Suffix Tree Clustering in [22], DC-Tree Clustering in [20], incremental hierarchical clustering in [3] and cluster similarity histograms in [9]. For a general overview of clustering methods, several good surveys exist including [10] by Jain *et al.* and more recently [21] by Xu and Wunsch.

2. Medoidshift Clustering

The medoidshift algorithm may be best explained in terms of the mode-seeking behavior of the meanshift algorithm. The N samples are denoted by the set $\{\mathbf{x}_i\} \in \mathbb{R}^d, i = 1, \cdots N$. Given such a set of samples, kernel density estimation can be used to evaluate the underlying distribution function at a point by,

$$f(\mathbf{x}) = c_0 \sum_i \Phi\Big(\Big\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\Big\|^2\Big), \tag{1}$$

where $\Phi(\cdot)$ is a kernel function ([19]) using the profile notation, and c_0 is a positive scalar dependent on the number of samples and the bandwidth, h. In addition, $\Phi(x)$ is the shadow of the kernel $\varphi(x)$ ([4, 5]), i.e. $\varphi(x) = -\Phi'(x)$. During mode-seeking, each point is initially denoted by $\mathbf{y}_0$, and the set of intermediate points traversed during progress towards the solution is denoted by $\{\mathbf{y}_k\} \in \mathbb{R}^d, k = 0, 2, \cdots K$. Each step of meanshift moves along the direction of highest gradient from the current point. For a single step, $\mathbf{y}_k$ denotes the current position of the point and $\mathbf{y}_{k+1}$ denotes the next position of the point. $\mathbf{y}_{k+1}$ is selected from a set of candidate positions (which is the set of samples $\{\mathbf{x}_i\}$), according to,

$$\mathbf{y}_{k+1}^{\text{mean}} = \arg\min_{\mathbf{y}} \sum_i \|\mathbf{x}_i - \mathbf{y}\|^2 \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{y}_k}{h}\Big\|^2\Big). \tag{2}$$

Setting the first derivative to zero and solving for $\mathbf{y}$ we get an estimate for the new position,

$$\mathbf{y}_{k+1}^{\text{mean}} = \frac{\sum_i \mathbf{x}_i \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{y}_k}{h}\Big\|^2\Big)}{\sum_i \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{y}_k}{h}\Big\|^2\Big)}. \tag{3}$$

This can be compared to Equation 20 in [5]. For each data point, this process is repeated until the point converges to the locally maximum density. If the meanshift procedure takes K iterations to converge, the mode corresponding to the original data point $\mathbf{y}_0$ is $\mathbf{y}_K$ at the time of convergence.

In the medoidshift algorithm, instead of computing the new position $\mathbf{y}_{k+1}$, as shown in Equation 3, the weighted *medoid* is used instead. A point $\mathbf{y} \in \{\mathbf{x}_i\}$ is therefore the (weighted) medoid, if

$$\mathbf{y}_{k+1}^{\text{medoid}} = \arg\min_{\mathbf{y} \in \{\mathbf{x}_i\}} \sum_i \|\mathbf{x}_i - \mathbf{y}\|^2 \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{y}_k}{h}\Big\|^2\Big). \tag{4}$$

The relationship between the meanshift update and the medoidshift update is defined by Equations 2 and 4 and illustrated in Figure 1. The point selected by meanshift corresponds to the minimum of the function whereas the point selected by medoidshift corresponds to the *sample* which best minimizes the function. For medoidshift clustering, since $\forall k, \mathbf{y}_k \in \{\mathbf{x}_i\}$, for any $\mathbf{y}_k$, $\mathbf{y}_{k+1}$ will necessarily belong to a point in the sample set. As a result, a medoidshift need only be computed once per data sample. Once a medoidshift has been computed for all points in the data set, the next shift will already exist for all $\mathbf{y}_{k+1}$. A mode is reached if $\mathbf{y}_k$ is the same as $\mathbf{y}_{k+1}$, and therefore unlike meanshift, no threshold is required to specify the terminating condition. The sequence $\{\mathbf{y}_k\}$ is guaranteed to converge.

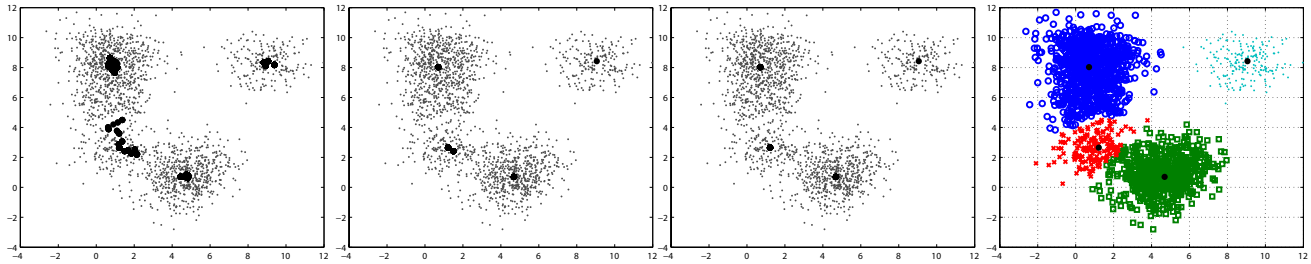


Figure 2. Left to right: Clustering using medoidshift after 1, 2, and 3 iterations, and the final labels for each point.

Theorem 2.1 *The sequence* $\{\mathbf{y}_k\}_{k=1,2,\cdots}$ *generated by successive medoidshifts converges for all starting locations in* $\{\mathbf{x}_i\}_{i=1,\cdots N}$.

Proof Since $\mathbf{y}_k \in \{\mathbf{x}_i\}_{i=1,\cdots N}$ and N is finite, the series will converge if there are no cycles, i.e. if $\mathbf{y}_k \neq \mathbf{y}_{k+w}$ for all k and for all $w > 0$. From Equation 1,

$$f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k) = c_0 \sum_i \Phi\Big(\Big\|\frac{\mathbf{y}_{k+1} - \mathbf{x}_i}{h}\Big\|^2\Big) - \Phi\Big(\Big\|\frac{\mathbf{y}_k - \mathbf{x}_i}{h}\Big\|^2\Big).$$

Since $\Phi(x)$ is a convex function (Appendix A.4, [5]) we have,

$$\Phi(x_2) - \Phi(x_1) \geq \varphi(x_1)(x_1 - x_2), \tag{5}$$

and therefore,

$$f(\mathbf{y}_{k+1}) - f(\mathbf{y}_k) \geq c_0 \sum_i \varphi\Big(\Big\|\frac{\mathbf{y}_k - \mathbf{x}_i}{h}\Big\|^2\Big)\Big(\|\mathbf{y}_k - \mathbf{x}_i\|^2 - \|\mathbf{y}_{k+1} - \mathbf{x}_i\|^2\Big). \tag{6}$$

From Equation 4, successive points are selected according to the following criterion,

$$\sum_i \|\mathbf{y}_k - \mathbf{x}_i\|^2 \varphi\Big(\Big\|\frac{\mathbf{y}_k - \mathbf{x}_i}{h}\Big\|^2\Big) > \sum_i \|\mathbf{y}_{k+1} - \mathbf{x}_i\|^2 \varphi\Big(\Big\|\frac{\mathbf{y}_k - \mathbf{x}_i}{h}\Big\|^2\Big), \tag{7}$$

which can be re-written as,

$$\sum_i \varphi\Big(\Big\|\frac{\mathbf{y}_k - \mathbf{x}_i}{h}\Big\|^2\Big)\Big(\|\mathbf{y}_k - \mathbf{x}_i\|^2 - \|\mathbf{y}_{k+1} - \mathbf{x}_i\|^2\Big) > 0. \tag{8}$$

From Inequalities 6 and 8, we can conclude strictly that $f(\mathbf{y}_{k+1}) > f(\mathbf{y}_k)$, and therefore for the sequence $\{\mathbf{y}_0, \mathbf{y}_1 \cdots \mathbf{y}_K\}$, the corresponding sequence $\{f(\mathbf{y}_0), f(\mathbf{y}_1) \cdots f(\mathbf{y}_K)\}$ is strictly increasing. As a result, $f(\mathbf{y}_k) < f(\mathbf{y}_{k+w})$ for all $w > 0$, and therefore $\mathbf{y}_k \neq \mathbf{y}_{k+w}$. ∎

Points within the clustered data set may be interpreted as nodes on a graph connected by directed edges, ([12]). For a data set with multiple modes, there will exist several disjoint trees, where each tree consists of nodes (corresponding to points) that belong to the same cluster. Each medoidshift establishes a directed edge between the data point $\mathbf{y}_k$ and its corresponding $\mathbf{y}_{k+1}$. The root of each tree corresponds to the mode of the cluster specified by that tree. A single tree traversal may be used to assign the cluster to all points associated with that tree. It should be noted that the procedure thus far can be considered non-iterative.

To complete the medoidshift algorithm, each data point is moved to the location of its estimated mode and the procedure is repeated until no further change occurs in labeling. Each data point $\mathbf{x}_i$ is moved to the location of its estimated mode $\mathbf{y}_{K_i}$, i.e. $\mathbf{x}_i^{(1)} \leftarrow \mathbf{y}_{K_i}$ where $\mathbf{x}_i^{(t)}$ is the updated location of the point $\mathbf{x}_i$ at iteration $t-1$. The mode-seeking process, as described earlier is repeated on $\{\mathbf{x}_i^{(t)}\}$ until no further change in labeling occurs. Figure 2 shows the clustering result of a data set after each iteration. This iterative process is guaranteed to converge for normal kernels.

Theorem 2.2 *For any* $\{\mathbf{x}_i\}$ *of finite size* N*, for the sequence*

$$\{\mathbf{x}_i\}, \{\mathbf{x}_i^{(1)}\}, \cdots \{\mathbf{x}_i^{(T)}\},$$

generated by the medoidshift algorithm, T *is always finite.*

Proof Since $\mathbf{x}_i^{(t)} \leftarrow \mathbf{y}_{K_i}^{(t-1)}$ and $\mathbf{y}_{K_i}^{(t-1)} \in \{\mathbf{x}_i^{(t-1)}\}$, the number of unique points will either remain the same or decrease. If the number of unique points remain the same the terminating condition is met provided there is no mode-switching, i.e. $\mathbf{x}_p^t \leftarrow \mathbf{x}_q^{t-1}$ and $\mathbf{x}_q^t \leftarrow \mathbf{x}_p^{t-1}$. We see that if switching occurs,

$$\sum_i \|\mathbf{x}_i - \mathbf{x}_p\|^2 \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{x}_p}{h}\Big\|^2\Big) > \sum_i \|\mathbf{x}_i - \mathbf{x}_q\|^2 \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{x}_p}{h}\Big\|^2\Big),$$

$$\sum_i \|\mathbf{x}_i - \mathbf{x}_q\|^2 \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{x}_q}{h}\Big\|^2\Big) > \sum_i \|\mathbf{x}_i - \mathbf{x}_p\|^2 \varphi\Big(\Big\|\frac{\mathbf{x}_i - \mathbf{x}_q}{h}\Big\|^2\Big),$$

which is a contradiction (Lemma A.1). Therefore the number of unique points remaining the same is a necessary and sufficient condition for the terminating condition to be met. Since the data set has a finite number of points and the number of unique points must keep decreasing (or otherwise terminate), T is at most $N-1$. ∎

This iterative process is similar to the original formulation of meanshift proposed in [8], which has been referred to as "blurring" meanshift. Unlike meanshift, no additional heuristic is required to merge modes of the same cluster together into a single mode, and unlike blurring meanshift no additional heuristic is required to terminate the iterations before all points become coincidental. The complete algorithm is detailed in Figure 3.

In addition to removing the heuristics for a stopping criterion and mode merging, the medoidshift algorithm allows the idea of mode-seeking to be applied to problems where

the meanshift algorithm is inapplicable. When comparing datasets, often distances can be defined without explicitly defining a feature space (and therefore a mean). Examples include the Earth Mover's Distance for Image Retrieval, the Proportional Transportation Distance for Signal Matching and manifold clustering where the manifold cannot be analytically defined. Distances between two points are computed using $d(\mathbf{x}_i, \mathbf{x}_j)$, which satisfies the properties of symmetry, triangular inequality and positivity. As we show later, medoidshift also allows clustering where the data set may evolve over time without requiring a complete re-evaluation at each time instance.

While this algorithm has the above-mentioned advantages over the meanshift algorithm, its clustering behavior is similar to that of meanshift. In fact, as the sample size increases, the intermediate clustering yielded by the non-iterative part (Figure 3, steps 1 and 2) of the medoidshift algorithm can be shown to tend towards the clustering yielded by meanshift. For the underlying distribution function f,

$$f(\mathbf{y}_{k+1}^{\text{mean}}) > f(\mathbf{y}_k^{\text{mean}}), \tag{9}$$

from Theorem 1 in [5]. From the law of large numbers, as N tends to infinity the ratio of number of samples at any $\mathbf{x}$ to the total number of samples will tend towards $f(\mathbf{x})$. Thus, if a sample $\mathbf{x}$ exists then $f(\mathbf{x}) > 0$. Since $\mathbf{y}_0^{\text{mean}} \in \{\mathbf{x}_i\}$ we have $f(\mathbf{y}_0^{\text{mean}}) > 0$ and because of Inequality 9, $\forall k\ f(\mathbf{y}_k^{\text{mean}}) > 0$. If follows that,

$$\forall \mathbf{y}_k^{\text{mean}}\ \ p\Big(\lim_{N\to\infty} \mathbf{y}_{k+1}^{\text{mean}} \in \{\mathbf{x}_i\}\Big) = 1. \tag{10}$$

As a result, since $\mathbf{y}_{k+1}^{\text{medoid}}$ is the sample closest to $\mathbf{y}_{k+1}^{\text{mean}}$,

$$\lim_{N\to\infty} \mathbf{y}_{k+1}^{\text{medoid}} \to \mathbf{y}_{k+1}^{\text{mean}}$$

and therefore, $\lim_{N\to\infty} \mathcal{C}_{\text{medoid}} \to \mathcal{C}_{\text{mean}}$, where $\mathcal{C}_{\text{medoid}}$ and $\mathcal{C}_{\text{mean}}$ are the clusterings yielded by the medoidshift and meanshift algorithms respectively.

3. Implementation

The computations involving medoidshifts can be compactly expressed in terms of a matrix product. The two matrices that are computed are an $N \times N$ symmetric matrix of distances, $\mathbf{D}$ whose $(i,j)^{\text{th}}$ entry is defined as the distance between the i^{th} and j^{th} point,

$$\mathbf{D}(i,j) = d(\mathbf{x}_i, \mathbf{x}_j), \tag{11}$$

and an $N \times N$ weight matrix $\mathbf{K}$,

$$\mathbf{K}(i,j) = \varphi(d(\mathbf{x}_i, \mathbf{x}_j)). \tag{12}$$

Once these matrices have been computed their product matrix, $\mathbf{S} = \mathbf{DK}$, is computed. The next position for the i^{th} data point is denoted by the index with the minimum value in the i^{th} column of $\mathbf{S}$. Once the new position is computed in this manner for all $\mathbf{x}_i$, tree traversal is used to find the unique root (or mode) for all points. In Figure 3, the first step is replaced by steps shown in Figure 4.

Objective

Given N data points estimate the modes of the underlying data distribution function and associate each point with its mode.

Algorithm

1. For each data point, $\mathbf{x}_i$, $i = 1, 2, ..., N$, compute its shift:
   - For each candidate data point $\mathbf{x}_j$, $j = 1, 2, ..., N$, compute the sum of its weighted distance from all other data points:
   $$score\,(\mathbf{x}_j) = \sum_{k=1}^{N} d\,(\mathbf{x}_j, \mathbf{x}_k)\,\varphi\,(d\,(\mathbf{x}_i, \mathbf{x}_k))\,.$$
   - Select $\mathbf{x}_j$ with the minimum distance from all other data points as the next position.
2. Perform tree traversals to associate each point with its corresponding root node (which is its mode).
3. Replace each $\mathbf{x}_i$ with its mode, and perform Steps 1 and 2 with the new $\mathbf{x}_i$ as initial data points. Repeat this step until no further change occurs.

Figure 3. The Medoidshift Algorithm

Implementation

1. Compute $N \times N$ distance matrix $\mathbf{D}$ and the $N \times N$ weight matrix $\mathbf{K}$, e.g. $\mathbf{K} = \exp(-\mathbf{D}/2)$.
2. Compute $\mathbf{S} = \mathbf{DK}$.
3. Calculate the data point indices that correspond to the minimum along each column of $\mathbf{S}$. These indices correspond to $\mathbf{y}^+$ for all data points $\mathbf{y}^-$.

Figure 4. Compact Implementation of Step 1 of the Medoidshift Algorithm

To see why this works, we can consider each entry in $\mathbf{S}$ as an evaluation of the summand in Equation 4. For $\mathbf{x}_i$, the N values of the i^{th} column of $\mathbf{S}$ contain the eligibility score according to

$$\mathbf{S}(i,j) = \sum_{k=1}^{N} d(\mathbf{x}_j, \mathbf{x}_k)\varphi(d(\mathbf{x}_i, \mathbf{x}_k)), \tag{13}$$

of each of the N data points of being the weighted medoid given $\mathbf{x}_i$. By taking the minimum of a column of $\mathbf{S}$, we choose that data point which satisfies the criterion in 4. This approach is straightforward to implement (a few lines of code in MATLAB®), and has a computational complexity of $\mathcal{O}(N^{2.38})$, [7].

For the iteration step, the same procedure needs to be performed on data points that have been replaced by their modes. This procedure has a lot of redundancy since the N original points are moved to a smaller set of locations. Mul-

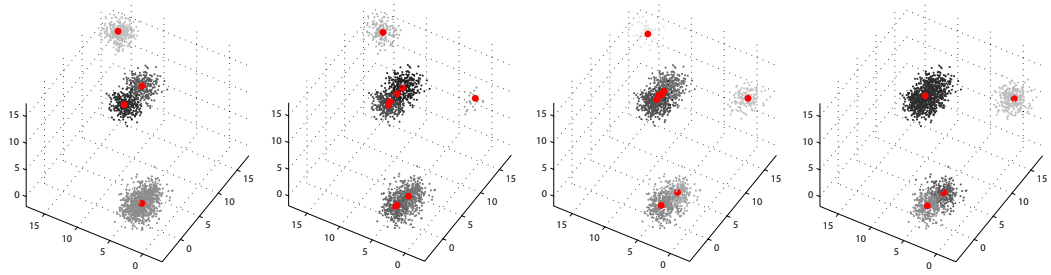


Figure 5. Incremental clustering. In the leftmost image, medoidshift algorithm was applied to 4 clusters. From left to right, the remaining images illustrate the algorithm's response to the data after: 2 original clusters were merged, 1 original cluster was split into 2, 1 original cluster was destroyed and a new one was created.

tiple identical computations will be performed when the distance of a point at a mode is being computed from all other points. This redundancy may be removed by applying the procedure on the modes alone, and premultiplying each column of $\mathbf{D}$ with the number of points at the mode associated with that column. Thus Equation 13 becomes

$$\mathbf{S}^{(t)}(i,j) = \sum_{m=1}^{M} \lambda_m d\Big(\mathbf{x}_j^{(t)}, \mathbf{x}_m^{(t)}\Big) \varphi\Big(d\Big(\mathbf{x}_m^{(t)}, \mathbf{x}_i^{(t)}\Big)\Big), \quad (14)$$

where λ_m is the number of points currently at location $\mathbf{x}_m$, M is the current number of unique positions (corresponding to the number of modes of points from the previous iteration). Both matrices $\mathbf{D}^{(t)}$ and $\mathbf{K}^{(t)}$ are of size $M \times M$, where M denotes the number of modes computed in the previous iteration. Instead of multiplying the matrices together directly, the product $\mathbf{D}^{(t)}\mathbf{\Lambda}^{(t)}\mathbf{K}^{(t)}$ is computed, where matrix $\mathbf{\Lambda}$ is a diagonal matrix. Each diagonal element $\mathbf{\Lambda}^{(t)}(m,m) = \lambda_m$ denotes the number of points associated with the m^{th} mode. As the number of modes is typically very small compared to the number of original data points, the computations saved in each iteration after the first one are significant.

4. Incremental Clustering

Incremental clustering is the problem of updating the clustering of N earlier points on the incidence of P new points and the removal of Q existing points. Not only does the incoming data need to be clustered, but earlier clustering may also need to be revised to make it equivalent to clustering from scratch or clustering within a window of observation. When a set of N data points, $\mathcal{D}$, has already been clustered as $\mathcal{C}_\mathcal{D}$, incremental clustering is the task of efficiently updating the clustering on the incidence of $\mathcal{D}_{\text{new}}$, a set of P new data points that have not yet been clustered, and the exitance of $\mathcal{D}_{\text{old}} \subseteq \mathcal{D}$, a set of Q data points that have already been clustered. In general, there are four events that may occur as data enters and exits the current set: (1) cluster creation, (2) cluster destruction, (3) cluster splitting and (4) cluster merging. Methods that require the number of clusters to be predefined (like k-means or normalized cuts) are inapplicable here since each of the four events involve a change in the number of clusters and it is unlikely that the number of clusters is provided at each time instant. Figure 5 illustrates this concept. With medoidshift clustering, earlier computation can be recycled making incremental clustering viable. The values in the original product matrix $\mathbf{S}_{\text{old}}$, distance matrix $\mathbf{D}_{\text{old}}$ and kernel matrix $\mathbf{K}_{\text{old}}$ contain useful information that can be reused in the evaluation at the next time instant.

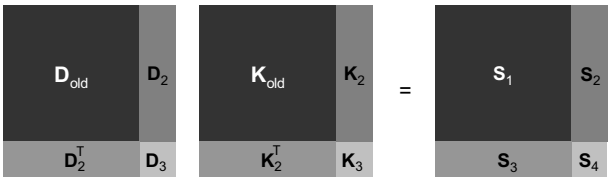


Figure 6. Incremental Clustering. The matrices $\mathbf{S}_{\text{old}}$, $\mathbf{D}_{\text{old}}$ and $\mathbf{K}_{\text{old}}$ can be recycled when computing $\mathbf{S}_{\text{new}}$.

If P points are being added to the data set and no points are being discarded, we can compute the different submatrices $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{S}_4$ (see Figure 6) as follows

$$\mathbf{S}_1 = \mathbf{S}_{\text{old}} + \mathbf{D}_2\mathbf{K}_2^T, \quad (15)$$

$$\mathbf{S}_2 = [\mathbf{D}_{\text{old}} \ \mathbf{D}_2][\mathbf{K}_2^T \ \mathbf{K}_3^T]^T, \quad (16)$$

$$\mathbf{S}_3 = [\mathbf{D}_2^T \ \mathbf{D}_3][\mathbf{K}_{\text{old}} \ \mathbf{K}_2]^T, \quad (17)$$

$$\mathbf{S}_4 = [\mathbf{D}_2^T \ \mathbf{D}_3][\mathbf{K}_2^T \ \mathbf{K}_3^T]^T. \quad (18)$$

If the data set were to be completely re-evaluated the number of operations would be $(N+P)^3$. When the process described is applied, the number of computations drop to $P^3 + 3N^2P + 3P^2N$, a savings of N^3 computations (the number of computations required to estimate the original N data points). Once again the index of the minimum of each column gives the medoid for the data point corresponding to that column. The resulting clustering is exactly equivalent to clustering from scratch.

In a similar manner, previous computations may also be recycled if Q existing points are being discarded from the data set. The columns in $\mathbf{D}$ and rows in $\mathbf{K}$, corresponding to the Q points, are first selected. The rows and columns corresponding to the Q points are then removed from the modified $\mathbf{D}$ and $\mathbf{K}$ respectively. Both rows and columns corresponding to the Q points are removed from $\mathbf{S}_{\text{old}}$. Next,

the modified $\mathbf{D}$ and the modified $\mathbf{K}$, of size $(N-Q)\times Q$ and $Q\times(N-Q)$ respectively, are multiplied together to form matrix $\mathbf{S}_\Delta$ of size $(N-Q)\times(N-Q)$. This matrix consists of computations that are due to the Q points, and is therefore subtracted from $\mathbf{S}_{\text{old}}$ to give $\mathbf{S}_{\text{new}}$,

$$\mathbf{S}_{\text{new}} = \mathbf{S}_{\text{old}} - \mathbf{S}_\Delta. \tag{19}$$

The minimum value is computed along each column of $\mathbf{S}_{\text{new}}$ as before, to find the (possibly new) modes of the remaining points.

As may be seen, all computations used to generate $\mathbf{S}_{\text{new}}$ are reused from $\mathbf{S}_{\text{old}}$. Only the extra computations (due to the Q points) need to be removed. The matrix multiplication involved to determine the contribution of these points requires $Q\times(N-Q)^2$ computations, instead of $(N-Q)^3$. The number of computations saved are therefore $N^3-4N^2Q+5NQ^2-2Q^3$. The savings are not always positive as may be seen in the following inequality.

$$(N-Q)^3 > Q\,(N-Q)^2\,, \tag{20}$$

which upon simplification gives

$$Q < N/2. \tag{21}$$

It is therefore beneficial to reuse previous calculations when the number of discarded points is less than half the total number of points in the data set.

In the case where P points are being added and Q points are being discarded from the data set, the following inequality needs to hold to make reuse of computations beneficial.

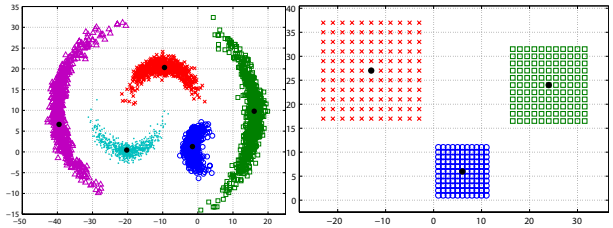
$$(N-Q+P)^3 > (Q+3P)\,(N-Q)^2+P^3+3P^2(N-Q). \tag{22}$$

This inequality also simplifies to Inequality 21, suggesting that incremental clustering should be used regardless of how many points are being added to the set.

5. Experiments

The first experiment demonstrates that the algorithm proposed in this paper does not require linearly separable clusters. In Figure 7(a) we show the clustering of four crescent shaped distributions each of differing radius and orientation, each containing 1000 data points. The bandwidth h was set to 4 for this experiment. Clustering with grid data is shown for 100 data points evenly spaced at three different scales in Figure 7(b). The bandwidth h was set to 8 for this result.

We compared the results of meanshift and medoidshift by using them to cluster 4 swiss-roll type data distributions in Figure 8. Each swirl has 500 data points that are normally distributed on each manifold. If we naively use the Euclidean distance the resulting clustering does not respect the manifold on which the data is distributed. Instead, we used the distance measure defined by Tenenbaum *et al.* in [18]. Note that the meanshift algorithm will not work well here since there is no direct way to compute the mean or to ensure that the mean would lie on the manifold. In contrast, since every mode is contained in the set of observations in medoidshift, each mode must lie on the manifold. Recently, a method to apply meanshift on nonlinear manifolds has been proposed in [17], however an analytic manifold is required.



(a) (b)

Figure 7. Data Clustering. (a) Clustering on Nonlinearly Separable Data. (b) Clustering on uniformly distributed data.

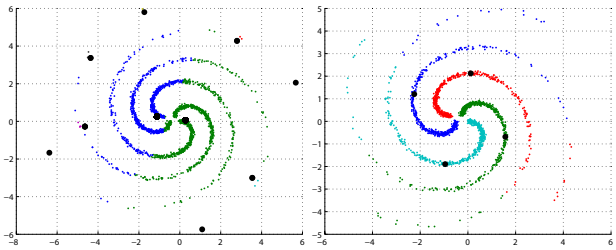


Figure 8. Clustering results obtained by using meanshift (left) in Euclidean space and medoidshift (right) using the manifold distance function described in [18].

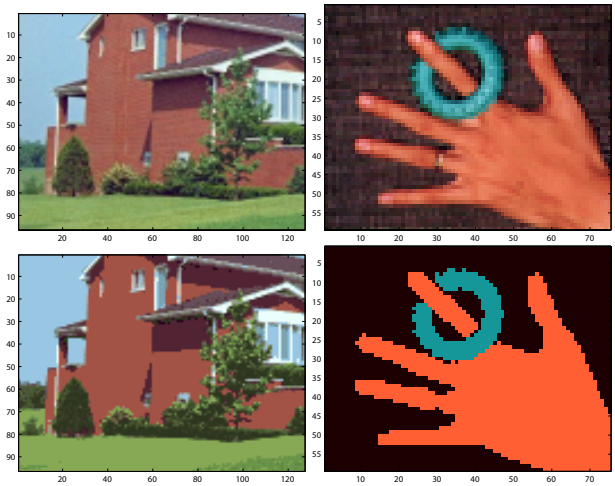


Figure 9. Image Segmentation using Medoidshift.

Medoidshift clustering was applied to image color segmentation and results are shown in Figure 9. A 5 dimensional feature space was used (including color and spatial features), and the bandwidth matrix for both images was set to $5\mathbf{I}_{5\times5}$. The result is comparable to the segmentation achieved in [5].

We have also used medoidshift to cluster a collection of images. We applied the algorithm to a set of 162 images of 11 different scenes. To compute the distance between two

Figure 10. Clustering a collection of images. Each row shows images selected from a single cluster found by medoidshift

Figure 11. Key-frames selected from a video of 1500 frames. Each key-frame corresponds to a mode as estimated by our algorithm.

images, we used the modified Bhattacharya coefficient (as described in [6]) on a 5D histogram of the $Labxy$ values in the images. The Bhattacharya coefficient between two distributions p and q is defined as $\rho = \int (p_{\mathbf{x}} q_{\mathbf{x}})^{\frac{1}{2}} d\mathbf{x}$. In order to give this coefficient a metric structure, Comaniciu *et al*. proposed the modified Bhattacharya coefficient, $d = \sqrt{1-\rho}$. For two histograms, $\hat{p}$ and $\hat{q}$, each with b bins, the estimator is,

$$\hat{d} = \left(1 - \sum_{i=1}^{b} (\hat{p}_i \hat{q}_i)^{\frac{1}{2}}\right)^{\frac{1}{2}}. \tag{23}$$

A coarse histogram was used, where the space was quantized into $[5\ 5\ 5\ 2\ 2]$ bins. The algorithm clustered these images into 11 clusters, 4 of which are shown in Figure 10.

To demonstrate incremental clustering, we show an example on key-frame selection in a video. The problem is to find a representative set of frames from the video in an online fashion. The challenge in this problem is that as the video progresses the number of modes change. Furthermore, it is difficult to define a feature space on which meanshift may be applied. Using the modified Bhattacharya coefficient on a coarse space-color histogram once again, we define these key-frames as the modes of clusters formed from the frames of the video. The procedure was tested on a 1500 frame video sequence. The bandwidth h was set to 1 for this experiment. Figure 11 shows the final set of 9 key-frames.

Finally, we demonstrate application on block-diagonalizing distance matrices. A random permutation of points is applied to the distance matrix shown in Figure 12(a) to produce a re-ordered distance matrix in Figure 12(b). Note that darker values specify small distances while brighter values specify larger distances between points. The distances were computed between samples from four bivariate normal distributions. 50 samples each were taken from the first and fourth distributions, 150 from the second and 100 from the third. The data-points were clustered using medoidshift and the points were ordered into a new distance matrix shown in Figure 12(c). Points were reordered so that points belonging to the same cluster were placed together. The ordering of the clusters themselves is arbitrary, but the association of data points to clusters is completely accurate.

6. Conclusion and Future Work

In this paper we have proposed a new clustering approach that inherits most of the desirable properties of meanshift clustering while allowing the idea of mode-seeking to be applied in problems where meanshift cannot. The following observations can be made about medoidshift clustering: First, the clustering achieved by the medoidshift algorithm is not restricted to spherical or ellipsoidal clusters, i.e. medoidshift does not require linearly separable clusters. Second, data points distributed in any metric space can be clustered. Third, an evolving data set can be incrementally clustered efficiently. Finally, the number of clusters do not need to be pre-specified.

In future work, we intend to investigate a further speed up. The requirement that for the current location $\mathbf{y}_k$, $\mathbf{y}_{k+1}$ be the data point that minimizes Equation 2 is not strictly required for convergence. The sufficient condition in Equation 7 in Theorem 2.1 is simply that the new point $\mathbf{y}_{k+1}$ have a score *better* than $\mathbf{y}_k$. If this condition is used instead of the exact condition the computation can be terminated early. An implementation showed that the computational saving obtained from this was roughly 80% greater than that of the exact algorithm. Further investigation is needed to determine the degree of approximation error. The extension

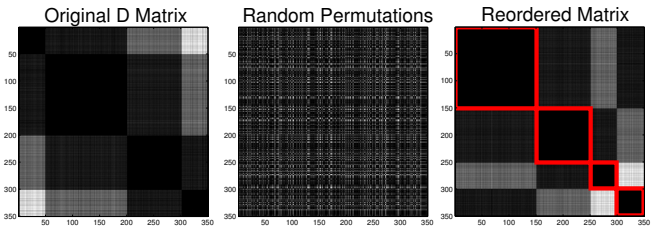


Figure 12. Clustering distance matrices. (a) Original Distance Matrix, (b) Rearranged by random permutation and (c) Reconstructed distance matrix after medoidshift clustering

of medoidshift to a k-nearest neighbor type estimator and the use of variable bandwidth are other future directions.

Acknowledgments

We thank Matthew Zucker for comments and discussion. We also thank Arif Zaman for supplying the proof of Lemma A.1.

References

[1] G. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing*, 1987.

[2] M. Carreira-Perpiñán. Mode-finding for mixtures of gaussian distributions. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2000.

[3] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *ACM Symposium on Theory of Computing*, 1997.

[4] Y. Cheng. Mean shift, mode seeking and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1995.

[5] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2002.

[6] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2000.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 1990.

[8] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function with application in pattern recognition. *IEEE Transaction on Information Theory*, 1975.

[9] K. Hammouda and M. Kamel. Incremental document clustering using cluster similarity histograms. *IEEE/WIC International Conference on Volume*, 2003.

[10] A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 1999.

[11] L. Kaufman and P. J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis Based on the L_1 Norm*, 1987.

[12] W. Koontz, P. Narendra, and K. Fukunaga. A graph theoretic approach to nonparameteric cluster analysis. *IEEE Transactions on Computers*, 1976.

[13] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[14] B. Moore. Art I and pattern clustering. *1988 Connectionist Models Summer School*, 1989.

[15] R. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 2002.

[16] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2000.

[17] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2006.

[18] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

[19] M. Wand and M. Jones. Kernel smoothing. *Chapman and Hall*, 1995.

[20] W. Wong and A. Fu. Incremental document clustering for web page classification. *International Conference on Information Society*, 2000.

[21] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 2005.

[22] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. *International ACM SIGIR Conference*, 1998.

A. Appendix

Lemma A.1 *If*

$$\sum_i a_i \mathrm{e}^{-a_i} > \sum_i b_i \mathrm{e}^{-a_i} \tag{24}$$

then,

$$\sum_i a_i \mathrm{e}^{-b_i} > \sum_i b_i \mathrm{e}^{-b_i}. \tag{25}$$

Proof Let $\alpha_i = (a_i - b_i)\mathrm{e}^{-a_i}$ and $\beta_i = (a_i - b_i)\mathrm{e}^{-b_i}$, then

$$\frac{\alpha_i}{\beta_i} = \mathrm{e}^{-(a_i - b_i)}. \tag{26}$$

The sign of $(a_i - b_i)$ determines three possibilities,

1. $(a_i - b_i) > 0$: Here α_i and β_i are both positive, and $\frac{\alpha_i}{\beta_i} < 1$, thus $\alpha_i < \beta_i$.
2. $(a_i - b_i) < 0$: Here α_i and β_i are both negative, and $\frac{\alpha_i}{\beta_i} > 1$, thus $\alpha_i < \beta_i$.
3. $(a_i - b_i) = 0$: α_i and β_i are equal, i.e. $\alpha_i = \beta_i$.

In all cases, $\alpha_i \leq \beta_i$ and therefore, $\sum_i \alpha_i \leq \sum_i \beta_i$. From Inequality 24, $\sum_i \alpha_i > 0$ and we can infer $\sum_i \beta_i > 0$ which is a rearrangement of Inequality 25. ∎