KNN Kernel Shift Clustering with Highly Effective Memory Usage

Makoto Hirohata, Tomoyuki Shibata, Kazunori Imoto, and Toshimitsu Kaneko
Toshiba Corporate Research & Development Center
{makoto.hirohata, tomoyuki1.shibata, kazunori.imoto, toshimitsu.kaneko}@toshiba.co.jp

Abstract

This paper presents a novel clustering algorithm with highly effective memory usage. The algorithm, called kNN kernel shift, classifies samples based on underlying probability density function. In clustering algorithms based on density, a local mode of the density represents a cluster center. It is effective to shift each sample to a point having higher density, considering the density gradient. Estimation of density and determination of the shifting are calculated using distance between samples. Large memory is necessary because the number of all combinations of N samples is $O(N^2)$. We propose a mode seeking approach using only neighbor samples in order to save memory. Experimental results show the effectiveness of the proposed algorithm in terms of clustering accuracy, processing time, and memory usage.

1 Introduction

Automatic categorization of images or video shots based on semantics is useful for many applications such as segmentation, indexing and retrieval. Clustering is one of the fundamental solutions for unsupervised data.

Mean shift is a nonparametric clustering algorithm known as a solution to the mode-seeking problem [1], [2], [3]. Mean shift is applied to many applications, such as image categorization, segmentation, and tracking. For seeking the mode of samples, density gradient and density distribution are estimated. The mode is defined as the local maximum of density. In the procedure, the density gradient is calculated by using kernel function and band width. Each sample moves to a point having higher density. Simplistically, each sample moves the average of samples in the neighborhood of the target [1]. After the movements are repeated, each sample converges on a local mode. The number of clusters is settled automatically. Additionally, the robustness to outliers is an advantage compared to k-means algorithm and hierarchical algorithm, such as single link. On the other hand, the repetition of calculation for shifting samples to other points causes an increase of computation.

Medoid shift proposed by Sheikh is one of the algorithms for improving the processing time [4]. In medoid shift, the calculation for the first movement in mean shift is recycled. Each sample moves not to the point having higher density, but to the nearest sample to that. Therefore, each sample converges on a local mode by only repeating the referring to the nearest sample. However, the nearest sample may be the same as each sample, and then fragmentation of clusters is caused. The fragmentation problem and correctness degradation have been reported [5].

Vedaldi proposed quick shift that copes with both computation and clustering accuracy [5]. Quick shift does not calculate the density gradient, but estimates the density for each sample directly by using distances between samples. Each sample shifts the nearest sample that has higher density. Since the computation cost depends on calculation of distances and comparison of densities, the procedure of quick shift is faster than those of mean shift and medoid shift. Quick shift is also effective in terms of clustering accuracy. Additionally, by storing distances between samples, both density estimation and mode seeking are quickly achieved. Therefore, parameter optimization and getting various results by changing parameters are easy. On the other hand, quick shift needs large memory. In order to execute calculation process of density and determination process of samples classified into the same cluster, distance between samples is necessary. Those two processes cannot be executed simultaneously. In order to execute them effectively without overlapping the operation, distance information of all combinations needs to be stored into a memory buffer. In the case of N samples, the memory usage is $O(N^2)$. Even though the procedure executed using only neighbor samples cuts off the memory, the valid number of neighbor samples cannot be previously predicted. Accordingly, a large-capacity memory is essential.

In this paper, we propose an alternative clustering algorithm based on k-nearest neighbor samples (kNN kernel shift algorithm). By extracting neighbor samples under restricted conditions, density is precisely estimated and associations of many samples are considered. The proposed algorithm inherits the advantages of quick shift with small memory requirement.

The remainder of the paper is organized as follows. Section 2 introduces quick shift algorithm investigated in this paper, Section 3 presents the proposed algorithm, and Section 4 describes experimental conditions and results. Finally, Section 5 concludes the paper.

2 Quick Shift Algorithm

Given a set of samples $\{x_1, x_2, ..., x_N\}$ in the d-dimensional space $\mathbb{R}^d$, quick shift algorithm estimates underlying probability density function for each sample x_i,

$$P(x_i) = \sum_{j=1}^{N} K\left(\left\|\frac{x_j - x_i}{h}\right\|^2\right) \quad (1)$$

where $K(\cdot)$ is a kernel function, and h is the bandwidth. In order to shift each sample to the nearest

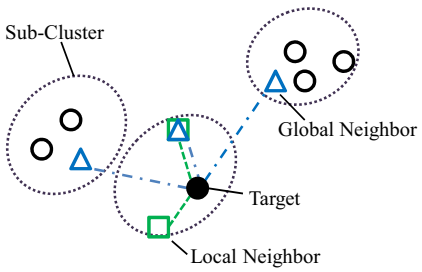

Figure 1. Concept of selection of local neighbor samples and global neighbor samples.

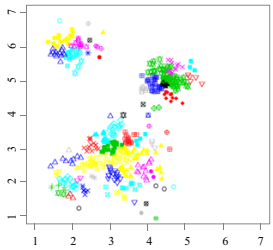

Figure 2. A result of creation of sub-clusters.

mode, each sample x_i simply moves to the nearest neighbor $y(x_i, 1)$ that has higher density. In formulas,

$$y(x_i, 1) = \begin{cases} q(x_i) & \|q(x_i) - x_i\| < \tau \\ x_i & otherwise \end{cases} \quad (2)$$

$$q(x_i) = \arg\max_{j=1,\ldots,N} \frac{sgn\left(P(x_j) - P(x_i)\right)}{\|x_j - x_i\| + 1} \quad (3)$$

where τ is a shift parameter. In the case that there is no sample having higher density, $q(x_i)$ becomes x_i. If the distance between x_i and $q(x_i)$ is smaller than τ, the sample continues to be associated with other samples (e.g. $y(x_i, m+1)$) as follows:

$$y(x_i, m+1) = y(y(x_i, m), 1) \quad (4)$$

where m is the number of movements. When $y(x_i, m+1)$ is the same as $y(x_i, m)$ in Equation 4, each sample x_i is connected to the nearest mode $y(x_i)$.

3 KNN Kernel Shift Algorithm

In quick shift, after density estimation, the nearest sample having higher density is searched for each sample. Distances between samples are essential for not only density estimation but also searching nearest samples. Therefore, quick shift needs to store all distances, and memory requires $O(N^2)$ space.

We propose a clustering algorithm (kNN kernel shift) that needs only k-nearest neighbor samples for each sample. This algorithm is executed within the $O(kN)$ memory requirement. A set of neighbor samples consists of local neighbors and global neighbors. The concept is shown in Figure 1. The local neighbors are nearest samples of all samples. On the other hand, the global neighbors are selected one by one from sub-clusters. Each global neighbor sample is the nearest in the sub-cluster to which it belongs. Here, each sample can refer to farther samples. Accordingly, a few global

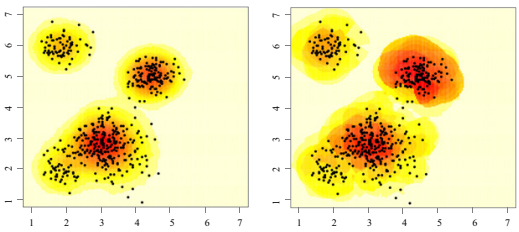

Figure 3. Density distribution of quick shift on the left and kNN kernel shift on the right.

neighbor samples make connection of many samples possible. It is essential to create sub-clusters in the process of referring to distances. The algorithm for creating sub-clusters is described as follows:

[Algorithm for creation of sub-clusters]

Step 1. Store a new sample.

Step 2. Calculate distances between the new sample and the samples already stored. If the minimum distance is smaller than a threshold, distribute the new sample into the sub-cluster that has the stored sample with the minimum distance. Otherwise, create a sub-cluster with the new sample. Afterward, the information of local neighbors and global neighbors is updated using the calculated distances.

Step 3. If all samples were not stored, return to step 2. Otherwise, finish the process.

Figure 2 shows a result of creation of sub-clusters. The example used 500 samples in 2-dimensional space and each sample is described as a symbol based on the sub-cluster ID. Euclidean distance was calculated. The threshold is set at 0.25. Each sub-cluster should be smaller than the clusters desired, and a lower threshold is better.

The density for each sample is calculated by using only global neighbor samples. Equation 1 becomes

$$P(x_i) = \sum_{x_j \in G_i} c(x_j) K\left(\left\|\frac{x_j - x_i}{h}\right\|^2\right) \quad (5)$$

where G_i is the set of global neighbor samples of x_i, and $c(x_j)$ is the number of samples in the sub-cluster including x_j. Figure 3 shows the density distribution examples of quick shift and kNN kernel shift. Samples are the same in Figure 2 and each dot represents a sample. We used Gaussian kernel and the band width h is set at 0.7. In Figure 3, deep color means high density. The density calculated using only global neighbor samples was similar to that of quick shift.

The mode seeking in kNN kernel shift consists of 4 processes: local index detection, local parent detection, global index detection, and global parent detection. Figure 4 shows the process flow diagram and the concept of each process. Firstly, a local index $l(x_i)$ is defined at each sample x_i. The local index is the nearest mode in the local neighbor samples.

$$l(x_i) = \arg\max_{x_j \in L_i \vee x_j = x_i} \frac{sgn\left(P(x_j) - P(x_i)\right)}{\|x_j - x_i\| + 1} \quad (6)$$

Here, L_i is the set of local neighbor points of x_i. Each sample x_i moves to the sample $y_L(x_i, 1)$ as follows:

$$y_L(x_i, 1) = \begin{cases} l(x_i) & \|l(x_i) - x_i\| < \tau \\ x_i & otherwise \end{cases} \quad (7)$$

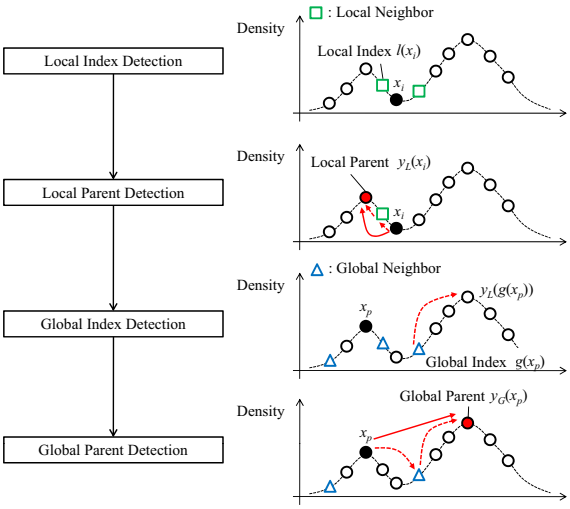

Figure 4. Process flow diagram and concept of each process.

x_i is associated with the mode $y_L(x_i)$ (local parent) in the same way as for quick shift (see Equation 4). Secondly, a global index $g(x_i)$ is defined at each local parent x_p. The local parent of the global index $g(x_p)$ has higher density than $y_L(x_p)$. $g(x_p)$ is calculated as follows:

$$g(x_p) = \underset{x_j \in G_p \vee x_j = x_p}{\arg\max} \frac{sgn\left(P\left(y_L(x_j)\right) - P(x_p)\right)}{\|x_j - x_p\| + 1} \quad (8)$$

where G_p is the set of global neighbor samples of x_p. x_p moves to the sample $y_G(x_p, 1)$ and is associated with the mode $y_G(x_p)$ (global parent) in the same way as for local parents.

$$y_G(x_p, 1) = \begin{cases} y_L(g(x_p)) & \|g(x_p) - x_p\| < \tau \\ x_p & otherwise \end{cases} \quad (9)$$

Finally, each sample x_i shifts to the local parent and then shifts to the global parent.

$$y(x_i) = y_G(y_L(x_i)) \quad (10)$$

4 Experiments

4.1 Experimental Conditions

We applied clustering algorithms to a set of 1000 images of 5 different categories: sky, sea, forest, autumn leaves, and snow. The examples are shown in Figure 5. Each category consists of 200 images. We resized them into 240 x 240 pixel size and extracted co-occurrence features [6]. The feature, which is co-occurrence of a color matching result and a pair of edge directions, is effective for object classification. The feature was extracted from the whole region of the resized image and concatenated into a single feature vector. The vector was normalized by constant value and the number of dimensions was 1194. Figure 6 shows the histogram of Euclidean distances of images. Four clustering algorithms were evaluated: mean shift (Mean), medoid shift (Medoid), quick shift (Quick), and kNN kernel shift (kNN). We used Gaussian kernel. The parameters h and τ were controlled for each algorithm, and the algorithms were compared using the best conditions. For improving fragmentation of clusters, single-link hierarchical clustering was applied for each algorithm as post-processing in which the threshold for the

Figure 5. Image samples of experimental data set (from SOZAIJITEN^{TM}).

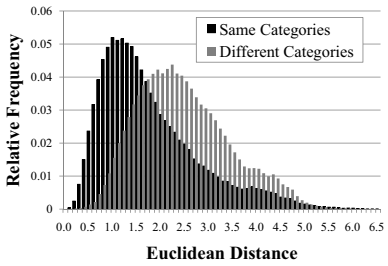

Figure 6. Histogram of the Euclidean distances between images.

single link is 0.2. Those experiments were executed with Xeon 2.67 GHz CPU.

4.2 Evaluation Criteria

Clustering results were evaluated by recall, precision and F-measure, which is a harmonic average of recall and precision. Those criteria are useful for retrieval, which is an important application. Recall and precision are defined as follows:

$$Recall = \frac{1}{A}\sum_{a=1}^{A}\sum_{i=1}^{I}\frac{n(i,a)}{n(*,a)} \cdot \frac{n(i,a)}{n(*,a)} \quad (11)$$

$$Precision = \frac{1}{A}\sum_{a=1}^{A}\sum_{i=1}^{I}\frac{n(i,a)}{n(*,a)} \cdot \frac{n(i,a)}{n(i,*)} \quad (12)$$

where A is the number of categories, I is the number of clusters, $n(i,a)$ is the number of images that belong to category a and cluster i, $n(*,a)$ is the number of images in category a, and $n(i,*)$ is the number of images in cluster i. In the right side of Equation 11 and 12, the first item $n(i,a)/n(*,a)$ is the probability that an image of category a is classified into cluster i. In the right side of Equation 11, the second item $n(i,a)/n(*,a)$ is the coverage ratio of images of cluster i in category a. On the other hand, in the right side of Equation 12, the second item $n(i,a)/n(i,*)$ is the coverage ratio of images of category a in cluster i.

4.3 Clustering Performance

Table 1 shows the results for each algorithm with the best parameter. In kNN, the threshold for creating sub-clusters was set at 0.3. The processing time was calculated except for the process for extracting features. We evaluated quick shift using only 20 nearest neighbor features for each feature (Quick(20)). In kNN(X,Y), the number of neighbors is $X + Y$. For instance, kNN(5,15) uses 20 neighbors. X represents the number of local neighbor features, and Y represents the number of global neighbor features. Quick got a better score than Mean and Medoid. However, Quick(20) did not maintain its performance. The neighbor features whose numbers were limited did not correctly associate features with the mode of density. In kNN, the computation speed was equivalent to or faster than Quick, and F-measure was improved. The memory usage in

Table 1. Results for each algorithm.

Algorithm	F-measure	Time[s]	Memory[KB]
Mean	0.550	13.0	1951
Medoid	0.477	2.1	1951
Quick	0.623	0.8	1951
Quick(20)	0.552	0.6	156
kNN(5,5)	0.667	0.6	86
kNN(5,10)	0.647	0.6	125
kNN(5,15)	0.647	0.7	164

Table 2. F-measure for each algorithm.

Algorithm	X=5	X=10	X=15
Quick(X)	0.148	0.305	0.504
kNN(X,5)	0.667	0.659	0.668
kNN(X,10)	0.647	0.675	0.688
kNN(X,15)	0.647	0.675	0.681

Table 1 means the storage for distances of all combinations of samples in Mean, Medoid, and Quick. In contrast, Quick(20) and kNN store distances and IDs of neighbor samples. Additionally, in the process of creating sub-clusters, kNN stores a sub-cluster ID for each sample, and the number of samples that belong to the sub-cluster. The memory usage of kNN was one-tenth of those of Mean, Medoid, and Quick or less.

Table 2 shows the relationship between F-measure and the number of local neighbor features in Quick and kNN. The addition of local neighbor features contributed to improvement in the F-measure. The mode seeking was improved by using neighbor features belonging to a same cluster. Additionally, in kNN, the range of the seeking was expanded effectively using global neighbor features. Therefore, setting the number of local neighbors and those of global neighbors is easy and high performance is expected.

Figure 7 shows the relationships between recall and precision in Quick and kNN when the parameters h and τ were changed. KNN achieved higher recall and precision than Quick.

In kNN, since the appropriateness of the result of sub-clusters influences the accuracy of clustering, the robustness to the threshold for sub-clusters is important. Figure 8 shows the relationships between F-measure and the threshold, and Figure 9 shows the coverage ratios of samples in sub-clusters having two or more samples and the largest sub-cluster. The F-measure became robust to the threshold when using more global neighbor features. On the other hand, the F-measure was worse in the case that the threshold was low such as 0.1 or high such as 0.6. This is because, in the former case, there were few sub-clusters including two or more samples, and also, in the latter case, a sub-cluster included most of samples. Hence, kNN(5,15) is effective for achieving high performance in terms of F-measure, processing time, and memory usage.

5 Conclusion

This paper has presented a novel clustering algorithm with highly effective memory usage and applied it to image categorization. The proposed algorithm, executed using fewer samples than for quick shift, precisely estimates relationships between samples in density distribution. We showed that the proposed algorithm achieved the best performance in terms of clustering accuracy, processing time, and memory usage.

Subjects for future research include evaluation of other tasks, such as documents and human images, and investigation of other applications, such as image segmentation. In the proposed algorithm, since the threshold for creating sub-clusters is sensitive to the accuracy, automatic optimization of the threshold is important.

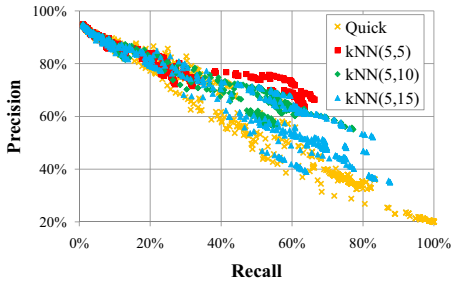

Figure 7. Relationships between recall and precision in Quick and kNN.

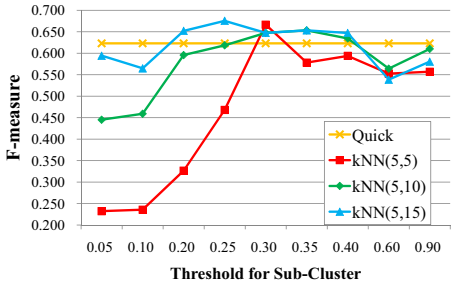

Figure 8. Relationships between F-measure and thresholds for sub-clusters.

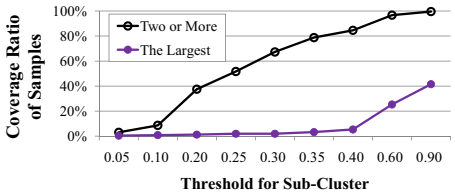

Figure 9. Coverage ratios of samples in sub-clusters having two or more samples (Two or More) and the largest sub-cluster (The Largest).

References

[1] K. Fukunaga, et al.,"The Estimation of the Gradient of a Density Function, with Applications in Pattern - Recognition," *IEEE Trans. Information Theory*, 21(1):32-40, 1975.

[2] Y. Cheng, "Mean Shift, Mode Seeking, and Clustering," *IEEE Trans. PAMI*, 17(8):790-799, 1995.

[3] D. Comaniciu, et al.,"Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. PAMI*, 24(5):603-618, 2002.

[4] Y. A. Sheikh, et al., "Mode-seeking by Medoidshifts," *In Proc. ICCV*, 2007.

[5] A. Vedaldi, et al.,"Quick Shift and Kernel Methods for Mode Seeking," *In Proc. ECCV*, 705-718, 2008.

[6] S. Ito, et al.,"Object Classification Using Heterogeneous Co-occurrence Features," *In Proc. ECCV*, 209-222, 2010.