Mode Seeking Clustering by KNN and Mean Shift Evaluated

Robert P.W. Duin[1], Ana L.N. Fred[2], Marco Loog[1], and Elżbieta Pękalska[3]

[1] Pattern Recognition Laboratory,
Delft University of Technology, The Netherlands
r.duin@ieee.org,m.loog@tudelft.nl

[2] Department of Electrical and Computer Engineering, Instituto Superior Técnico
(IST - Technical University of Lisbon), Portugal
afred@lx.it.pt

[3] School of Computer Science,
University of Manchester, United Kingdom
pekalska@cs.man.ac.uk

Abstract. Clustering by mode seeking is most popular using the mean shift algorithm. A less well known alternative with different properties on the computational complexity is kNN mode seeking, based on the nearest neighbor rule instead of the Parzen kernel density estimator. It is faster and allows for much higher dimensionalities. We compare the performances of both procedures using a number of labeled datasets. The retrieved clusters are compared with the given class labels. In addition, the properties of the procedures are investigated for prototype selection.

It is shown that kNN mode seeking is well performing and is feasible for large scale problems with hundreds of dimensions and up to a hundred thousand data points. The mean shift algorithm may perform better than kNN mode seeking for smaller dataset sizes.

1 Introduction

The mean shift clustering procedure is based on a 1975 paper by Fukunaga [7]. It has been made most popular by Cheng [2] and by Comaniciu and Meer [3]. They showed how the idea of finding the modes of a non-parametrically estimated probability density function based on the Parzen kernel could be implemented sufficiently fast such that it can be used for segmenting images. As for reliable estimates many data points are needed, this became only feasible when sufficiently large memories and fast CPUs entered the market after 2000. The mean shift algorithm however suffers from the fact that determining and tracking the gradient in high dimensional spaces is still computationally heavy. Its use is thereby mainly restricted to applications with small sets of features, e.g. three color images.

There exists an interesting alternative for the Parzen kernel in mode seeking: the k-Nearest Neighbor (kNN) rule. This has also been shown by Fukunaga and his colleagues Koontz and Narendra in 1976 [9], but it did not receive much attention. An early version of the algorithm has been included in the PRTools

Matlab package [4] about 15 years ago. We renewed this implementation to make it feasible for 10^4 - 10^5 data points and 10^2 - 10^3 dimensions. It is the purpose of this paper to compare the two mode-seeking algorithms with each other. We will show that the performance of kNN mode seeking is reasonable, sometimes worse, sometimes better than mean shift, but it has the advantage of a significantly better computational efficiency. This will be shown and explained.

Comparing procedures for data analysis in general and for cluster analysis in particular is a mining field. It is very difficult to make general statements and one can easily be deceived. For that reason, we include a discussion (Section 2) on our philosophy on benchmarking cluster procedures and give arguments for the choices we have made. In Section 3, the algorithms will be discussed and our version of kNN mode seeking will be specified. In Section 4, the algorithms are compared on a number of datasets for two performance criteria and their computing times. The paper is finished with a discussion, summarizing the main properties and differences of both algorithms, see Section 5.

2 Comparing Cluster Procedures

The aim of clustering is to find an interesting structure in data, e.. sensor data collected in some scientific study. What is interesting is usually not pre-defined. Any structure that makes sense for building an understanding of the observations may give a hint to the researcher to think in a particular direction. For this reason, clustering is necessarily ill defined. Attempts to specify 'interesting' in terms of numerically well defined criteria may limit the analyst in his exploration. Consequently, a vast number of procedures has been developed, partially as diversity is essential, but also because there is no way inside clustering to estimate performances.

A way out of this dilemma is to use datasets that have already been analyzed by experts and for which they defined labels to identify objects that belong to the same structure (classes). This may be done by inspecting the data itself, or by other means, outside the given data. It is thereby possible to find in retrospect the cluster procedures yielding results that are consistent with expert supplied class labels. In this way, examples of procedures can be found that make sense in real world problems.

Cluster procedures give the following two types of results. They may group the objects, i.e. they define subsets of objects that are in one way or another similar. Some procedures determine in addition, or sometimes just instead, small sets of prototypes or examples of objects that are representative for the whole set. Procedures that don't deliver both can be extended with an additional step to determine the missing information. If just clusters are obtained and no prototypes, then for every cluster its centre (or medoid) will define a prototype (i.e. the object for which the maximum (mean) distance to the other objects in the cluster is minimized). If just prototypes are found, clusters can be defined by applying the nearest mean classifier trained by the prototypes to all objects.

These two results give two different ways to determine how consistent a cluster result is with a given class labeling. It depends on the number of clusters or prototypes that have been found. Almost every clustering procedure has a parameter that controls this number. In many studies, attempts are made to determine an optimal number of clusters. In comparing clustering results with given class labeling, this is not always appropriate. The expert may have used a varying resolution in distinguishing and naming classes. To remove this problem, we decided to use in this study a series of clusterings obtaining $k = 1$ up to (e.g.) $k = 50$ clusters. Next a performance measure is computed relating the result of all these clusterings with the given, true class labels. We used two performance measures, one that focuses on the obtained clusters and one on the prototypes.

An obtained clustering differs from a given class labeling in two ways. Objects in the same cluster may belong to different classes and objects in different clusters may belong to the same class. Let the cluster index of an object x be given by $C(x) \in \{1, 2, ..., k\}$ and let its object label be $\lambda(x) \in \{1, 2, ..., n\}$. The following two probabilities for an arbitrarily selected pair of objects $\{x_i, x_j\}$ are a measure for the consistency of the clustering with the true class labeling:

$$\epsilon_1 = Prob(C(x_i) \neq C(x_j) | \lambda(x_i) = \lambda(x_j)) \tag{1}$$

$$\epsilon_2 = Prob(C(x_i) = C(x_j) | \lambda(x_i) \neq \lambda(x_j)) \tag{2}$$

A clustering is consistent with the class labelling if $\epsilon_1 = 0$ in case $k <= n$ and $\epsilon_2 = 0$ in case $k >= n$. Both should be zero if $k = n$ in case of consistency. For a set of clusterings with varying k a set of $(\epsilon_1^k, \epsilon_2^k)$ pairs is obtained for every value of k, constituting a curve in the (ϵ_1, ϵ_2)-plane. An example is given in Fig. 1 for the two procedures, kNN mode seeking and mean shift, applied to the Iris dataset, see Section 3 and Section 4 for more details.

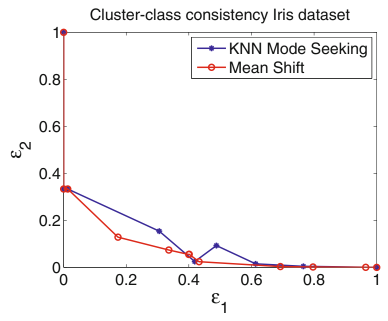

Fig. 1. Cluster-class consistency plot for the Iris dataset

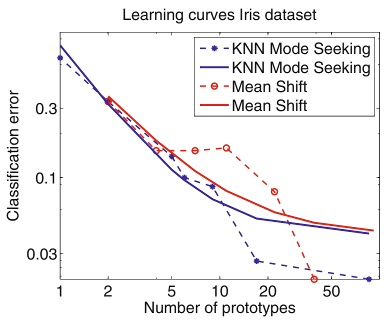

Fig. 2. Learning curves for the Iris dataset. Dashed curves as estimated, solid ones are approximated by Eq. 4.

Points on the horizontal axis relate to clusterings in which every cluster contains just objects that belong to the same class. Points on the vertical axis relate to clusterings in which all objects of the same class are taken by a single cluster.

A point in the origin corresponds to a clustering that fully coincides with the class labeling. The previous two types of clustering are still consistent with this one as either a merge or a split of clusters may produce this origin clustering.

As the two curves in Fig. 1 describing the two clustering procedures cross each other it is not clear whether one of the two is better than the other for the Iris dataset. A solution to judge this can be found be reducing such a curve to a single number. A possible option is to find the point on the curve that is most close to the origin, i.e. use $\epsilon_1 + \epsilon_2$ as a criterion. As this depends on just a single point, we prefer the following. Like in judging ROC curves that describe the trade-off between two types of classification errors [1], we may use the Area Under the Curve (AUC) for judging it. In this case, the lower the better. If the AUC is zero the set of clusterings is perfect as all clusterings are consistent with the class labels.

In case the clustering procedure results in prototypes, another way of judging its performance is possible. By simulating an active learning scenario, the labels of these prototypes may be used to label the clusters they belong to. By repeating such a procedure as a function of the number of prototypes, a learning curve, is created, see Fig. 2 for an example. To convert such a curve into a single performance measure, the learning curve, the classification error ϵ as a function of the number of training samples k, can be approximated by the following function:

$$\epsilon(k) = \epsilon_0 * k^{-\alpha} + \epsilon_\infty \tag{3}$$

in which ϵ_0 is the starting value and ϵ_∞ is the asymptotic value that equals here the nearest neighbor error for an infinite training set. The important parameter is the *learning speed* α that can be considered as a measure for the quality of the prototypes. We estimate α by

$$\alpha = argmin_\alpha \sum_k (\hat{\epsilon}_k - \epsilon(k))^2 \tag{4}$$

in which k takes the values determined by the set of clustering procedures and $\hat{\epsilon}_k$ is the observed error based on classifying the out-of-prototypes objects according to the k true prototype labels. Objects are classified by assigning the clusters to the label of the corresponding prototype, or by the nearest mean rule in case of a prototype selection procedure that does not generate clusters in addition.

The advantage of using a supervised criterion for judging cluster performances is that, by definition, such a criterion cannot be used by any cluster procedure itself. This contrasts the unsupervised criteria. Any choice, e.g. based on within and/or between cluster distances would result in a bias towards specific cluster algorithms.

Before discussing algorithms and datasets, we like to emphasize that a comparative study is not a match between procedures. The target is not to find a winner in one way or the other. As algorithms differ, they are good for different datasets. Any cluster algorithm can be considered as an estimator of some statistics of the data defined by the performance measure. Different algorithms relate to different estimators. Any estimator is biased, as it is based on some assumptions or a model. It will be better if these assumptions hold for a considered dataset. What is done in a comparative study over a collection of datasets is that one tries to find out for which problems, which estimators are better, or at least, whether for different estimators different datasets can be found for which they are useful. Any voting or averaging of results over a collection is arbitrary, unless one is sure that the collection is very representative for the problems to be studied in future.

3 The Algorithms

Mode seeking clustering can be considered as an agglomerative approach. First, a density function is estimated for the dataset. In general, it has several local maxima: the modes. In the clustering phase, for every object it is decided to which mode it belongs by following the density gradient from that object until a mode is found. Objects that end up in the same mode are considered to belong to the same cluster.

By this procedure, the number of clusters is identical to the number of modes. Here this approach differs from the Mixture-of-Gaussians (MoG) procedures as for these every component may be related to a cluster, but not every component constitutes its own mode in the mixture. The two mode seeking procedures discussed in this paper are not based on a mixture of Gaussians, but on non-parametric density estimates based on the Parzen kernel (the mean shift procedure) and the k-nearest-neighbor estimator (kNN mode seeking). Both have a width parameter that influences the number of modes in the density estimate. This number may thereby vary between one, for a very wide kernel or a large k, and m for a narrow kernel or $k = 1$. We do not try to optimize the width parameter, but instead consider the clustering results as a function of this parameter.

The use of the Parzen kernel for mode seeking clustering can be traced back to a 1975 paper by Fukunaga [7]. It resulted in the mean shift algorithm, which uses the observation that the shift of the mean of a kernel of a single object after weighing it with the neighboring objects inside the kernel points into the direction of the gradient. It has been made most popular by Cheng [2] and by Comaniciu and Meer [3]. They showed how this idea could be implemented sufficiently fast such that it can be used for segmenting images. For our study, we used the Matlab implementation by Bart Finkston [5]. It is not deterministic as it depends on the order in which objects are considered. Instead of the kernel width we used the number of nearest neighbors like in the kNN mode seeking (see below). The kernel width was set to the average distance to the k-th nearest neighbor over the entire dataset.

kNN mode seeking is originally described by Koontz [9]. It is related to an algorithm studied by Kittler [8] for which Shaffer et al. [10] stated that although it is based on another idea, its results may be very similar to single-linkage hierarchical clustering. Our experiments have shown that this is not true for our version. In our definition of kNN mode seeking, the density of every object is proportional to the distance to its k-th neighbor. We define for every object a pointer to the object with the highest density in its neighborhood. Finally, these pointers are followed to the object that points to itself. It stands for a mode in the density as it is itself the object with the highest density in its neighborhood.

The main difference between the two mode seeking procedures, kNN and mean shift, is that the latter uses a kernel with a neighborhood size that, in terms of distances, is constant over the entire space. kNN on the other hand uses a fixed neighborhood size in terms of the number of objects and thereby adapts itself to areas with higher or lower densities. In addition, from the implementation point of view, it is much faster to jump from object to object than to compute and follow a gradient.

The below description of kNN mode seeking is good for very large data set sizes m (e.g. $m = 10^5$) objects. All pairwise distances are needed twice and as $m^2 = 10^{10}$ distances cannot be stored, they are computed twice. This is done for a set of n neighborhood sizes k (e.g.$n = 25$) in parallel, by which it is needed to store mn densities and mn pointers. These are used to compute mn cluster indices for the n clusterings.

```
 1. Define a set of target neighborhood sizes $K = k_1, k_2, ..., k_n$.
 2. Repeat for all $m$ objects $x_i$.
 3.   Compute its distances $d_{ij}$ to all other objects $x_j$.
 4.   Sort them: $s_{ir} = sort_j(d_{ij})$
 5.   Store density estimates $\forall k \in K : f_{ij} = 1/s_{ik}$.
 6. Next $i$
 7. Repeat for all $m$ objects $x_i$.
 8.   Compute its distances $d_{ij}$ to all other objects $x_j$.
 9.   Rank them: $q_{ir} = argsort_j(d_{ij})$
10.   Store for all $k \in K$ a pointer $p_i = argmax_{r=1,k}(q_{ir})$
11. Next $i$
12. Repeat for all neighborhood sizes $k \in K$
13.   Repeat until no change $\forall i : p_i = p_{p_i}$
14.   Store clustering for neighborhood size $k$: $C_k = [p_1, p_2, ...p_m]$
15. Next $k$
```

4 Experiments

In this section it is shown that the two procedures defined in Section 3 are both useful. For both of them, datasets can be found for which one is better than the other. The datasets belong to the standard distribution of PRTools [4]. Most of them originate from the UCI repository [6]. Table 1 presents the area-under-the-curve values for the cluster-class consistency plots as defined is Section 2 and Eq. 1-2. The underlined values are the best for a dataset.

In Table 2, the learning speeds (Equation 4) are shown for the same datasets and algorithms. They show how valuable the procedures are for selecting prototypes to be used for labeling an entire dataset. As the mean shift algorithm does not find modes exactly on the position of objects, we used the medoids of the clusters it finds. In comparing Tables 1 and 2 it can be concluded that although

Table 1. The cluster-class-consistency AUC values for a collection of datasets (the lower the better). Underlined are the best results per dataset. The datasets Aviris* and MNIST* are sampled versions (10%) of the originals.

Dataset	m	d	c	kNN	MS
Hepatitis	155	19	2	0.52	0.47
Wine	178	13	3	0.31	0.28
Biomed	194	5	2	0.43	0.28
Glass	214	9	4	0.40	0.34
Malaysia	291	8	20	0.45	0.46
Ecoli	336	7	8	0.20	0.08
Auto-mpg	398	6	2	0.25	0.39
Arrhythmia	420	278	13	0.47	0.29
Breast	699	9	2	0.33	0.05
Diabetes	768	8	2	0.49	0.47
Car	1728	6	4	0.50	0.50
mfeat-fou	2000	76	10	0.27	0.16
Aviris*	2109	200	17	0.37	0.39
MNIST*	6006	784	10	0.35	0.29
Satellite	6435	36	6	0.21	0.17
Ringnorm	7400	20	2	0.50	0.34
Twonorm	7400	20	2	0.10	0.07
ChromoBands	12000	30	24	0.28	0.23

Table 2. The Learning Speed values of the cluster algorithms (the higher the better)

Dataset	kNN	MS
Hepatitis	1.73	0.09
Wine	0.66	0.29
Biomed	0.62	0.47
Glass	0.34	0.16
Malaysia	0.07	0.09
Ecoli	0.32	0.48
Auto-mpg	0.84	0.99
Arrhythmia	1.99	-0.70
Breast	0.76	27.27
Diabetes	0.75	0.26
Car	0.14	0.36
mfeat-fou	0.59	0.47
Aviris*	-0.03	-0.09
MNIST*	0.30	-0.05
Satellite	0.58	0.44
Ringnorm	0.00	0.37
Twonorm	51.76	0.71
ChromoBands	0.32	0.17

our collection of datasets contains more problems for which mean shift has a better AUC value than the kNN procedure, the latter has more problems with a lower learning speed. So for these examples mode shift finds clusters that are more consistent with the classes, but kNN finds better prototypes.

If the modes of the density function have to be used to define clusters, good density estimates should be available. This points to the direction of large datasets in comparison with the (intrinsic) dimensionality. The mean shift algorithm can handle large datasets for low dimensional spaces, but has problems for tracking the density gradient in high dimensions. Our implementation of kNN mode seeking can handle both, large numbers of objects and high dimensions.

It is interesting that the kNN procedure is significantly faster than mean shift, see Table 3. kNN mode seeking can easily handle datasets that are even an order larger than the ones studied here. The mean shift algorithm then fails, either due to intolerable computing times or because of the need to handle too large distance matrices.

5 Discussion

Mode seeking seems a natural procedure for cluster analysis. It is, however, necessary to have a sufficiently large dataset to obtain good density estimates.

Table 3. The total computing times (over about 25 clusterings per problem) in seconds needed for the various experiments

Dataset	m	d	c	kNN	MS
Aviris*	2109	200	17	2	381
ChromB	12000	30	24	72	5976
MNIST*	6006	784	10	43	12974
Ringnorm	7400	20	2	28	3014
Twonorm	7400	20	2	29	1469
Satellite	6435	36	6	22	886

Formally the mean shift procedure should be better able to handle this as the Parzen kernel takes care for a smooth estimate. The kNN procedure, however, offers interesting possibilities for a feasible and fast implementation.

Except for speed the restriction to use densities located in the objects offers a few additional advantages. As it is well defined, there is no inaccuracy in the exact position of the mode. In the mean shift algorithm, thresholds have to be set to determine whether a newly found mode is really new. We studied this in a small experiment based on the mfeat-fou dataset that has 2000 objects. In Tables 4 and 5, the sizes of the largest clusters are given for the two procedures for a set of values for k between 25 and 200. The mean shift procedure has the tendency to select one or a few large clusters and many with a size of one or two objects. kNN mode seeking finds more balanced cluster sizes.

Table 4. The number of objects in the 7 largest clusters found by kNN mode seeking with $k = \{25, ..., 200\}$ for the mfeat-fou dataset (2000 objects)

k							
25	316	245	207	180	159	146	146
50	381	339	303	209	188	142	141
75	595	486	355	253	214	97	
100	746	580	448	226			
125	704	560	502	234			
150	752	577	433	238			
175	741	621	398	240			
200	1086	914					

Table 5. The number of objects in the 7 largest clusters found by mean shift with $k = \{25, ..., 200\}$ for the mfeat-fou dataset (2000 objects)

k							
25	447	327	294	189	184	166	144
50	1410	323	165	67	2	1	1
75	1658	324	2	1	1	1	1
100	1739	252	1	1	1	1	1
125	1734	261	1	1	1	1	1
150	1752	244	1	1	1	1	
175	1996	1	1	1	1		
200	1997	1	1	1			

Another advantage of the kNN mode seeking procedure is that it is based on the object distances only. No computations in the feature space are needed. Consequently, it may operate on given distance matrices and after conversion on similarity matrices as well.

Finally, we showed that for the given collection of datasets kNN mode seeking is a better prototype selector, while the mean shift algorithm found often clusters that are more consistent with given class labelings.

References

1. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)
2. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Trans. Pattern Anal. Mach. Intell. 17(8), 790–799 (1995)
3. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24(5), 603–619 (2002)
4. Duin, R., Juszczak, P., de Ridder, D., Paclík, P., Pękalska, E., Tax, D., Verzakov, S.: PRTools 4.1, a Matlab toolbox for pattern recognition, `http://prtools.org`
5. Finkston, B.: Mean shift clustering
6. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`
7. Fukunaga, K., Hostetler, L.D.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans. Information Theory 21(1), 32–40 (1975)
8. Kittler, J.V.: A locally sensitive method for cluster analysis. Pattern Recognition 8(1), 23–33 (1976)
9. Koontz, W.L.G., Narendra, P.M., Fukunaga, K.: A graph-theoretic approach to nonparametric cluster analysis. IEEE Trans. Computer 25, 936–944 (1976)
10. Shaffer, E., Dubes, R.C., Jain, A.K.: Single-link characteristics of a mode-seeking clustering algorithm. Pattern Recognition 11(1), 65–70 (1979)