KNN-kernel density-based clustering for high-dimensional multivariate data

Thanh N. Tran, Ron Wehrens, Lutgarde M.C. Buydens*

Analytical Chemistry, Institute for Molecules and Materials, Radboud University Nijmegen, Nijmegen, The Netherlands

Received 17 December 2004; received in revised form 3 October 2005; accepted 3 October 2005
Available online 24 October 2006

Abstract

Density-based clustering algorithms for multivariate data often have difficulties with high-dimensional data and clusters of very different densities. A new density-based clustering algorithm, called KNNCLUST, is presented in this paper that is able to tackle these situations. It is based on the combination of nonparametric $k$-nearest-neighbor (KNN) and kernel (KNN-kernel) density estimation. The KNN-kernel density estimation technique makes it possible to model clusters of different densities in high-dimensional data sets. Moreover, the number of clusters is identified automatically by the algorithm. KNNCLUST is tested using simulated data and applied to a multispectral compact airborne spectrographic imager (CASI) image of a floodplain in the Netherlands to illustrate the characteristics of the method.

Keywords: Multivariate data; Classification; Clustering

1. Introduction

Clustering of multispectral data (Tran et al., 2005) groups objects, characterized by the values of a set of variables into separate groups (clusters) with respect to a distance or, equivalently, a similarity measure. Its objective is to assign to the same cluster objects that are more close (similar) to each other than to objects from different clusters, which may help to understand relationships that may exist among objects. Examples are exploring of environmental data representing physical and chemical parameters (Smolinski et al., 2002), computational analysis of microarray gene expression profiles (Liang and Kachalo, 2002), electron probe X-ray microanalysis (Bondarenko et al., 1994), or process monitoring (Teppola et al., 1999), and many others. However, the successful application of clustering on multispectral data sets is not a straightforward task. It depends on the understanding of the data set and a good choice of the clustering algorithm.

Several types of clustering methods can be distinguished, among which partitional and hierarchical approaches are the most common (Tran et al., 2005). Density-based clustering methods, such as CLUPOT (Coomans and Massart, 1981), DBSCAN (Ester et al., 1996), and Denclust (Hinneburg and Keim, 1998), form a third clustering type. Density-based clustering uses a local cluster criterion, in which clusters are defined as regions in the data space where the objects are dense, and clusters are separated from one another by low-density regions. Non-parametric density-based clustering...
is based on an estimation of a local non-parametrics density function, proposed by Fukunaga and Hostetler (1975) and been further improved in (Cheng, 1995; Comaniciu and Meer, 1999). Density-based clustering has advantages over partitional and hierarchical clustering methods in discovering clusters of arbitrary shapes, sizes. It is often used in data mining for knowledge discovery.

However, it was shown that current density-based clustering might have difficulties with complex data sets containing clusters with different densities (Comaniciu and Meer, 1999). In this case, it often identifies the very low density classes as noise (Tran et al., 2005). Moreover, the high dimensionality of many multivariate data sets is another problem for density-based clustering. In this case, the volume of the data grows dramatically with the dimension, while the number of objects remains the same. One of the solutions for the dimensionality problem is proposed in Ertoz et al. (2002) and Jarvis and Patrick (1973), using a k-nearest-neighbor (KNN) density estimation technique. Instead of defining a threshold to local density function, low-density regions, “valleys”, separating two clusters can be detected by calculating the number of shared neighbors. If the number of shared neighbors of two adjacent objects is below a threshold (number of objects), then there is a gap, a valley, in between. Hence, the two object belong to two different clusters. In this way, the method does not have to take into account the volume of the high-dimensional search space. However, this clustering method still requires the “density” threshold to be defined, which is very difficult for a real data set (Su et al., 2002).

In this paper, a new density-based clustering algorithm, the so-called KNNCLUST, is developed. The proposed method is based on a combination of nonparametric KNN and kernel density estimation methods (KNN-kernel). It will be shown later in the text that KNN-kernel is not a good solution for estimating the “true” density of a distribution due to an overestimate of density in the tails of the distribution. However, the KNN-kernel has attractive properties to clustering, shown for the first time in this paper.

KNNCLUST has been implemented in MATLAB 6.5 and the toolbox is available on the web: http://www.cac.science.ru.nl/research/software/.

We review nonparametric density estimation techniques in Section 2. The KNN-kernel class-condition rule on clustering and the description of the new KNN-kernel density-based clustering, KNNCLUST, are given in Section 3. In Section 4, its properties are illustrated using a multispectral remote sensing image and compared to the results from DBSCAN. Finally, the work is summarized in Section 5.

2. KNN-kernel density estimation

An unknown probability density function of a data set can be estimated by a nonparametric kernel density estimation method. Consider a $N \times d$ dimensional data set. The $d$-dimensional space can be partitioned into a number of equal bins (volumes), $V$, e.g. hyper-rectangles. The multivariate kernel density estimate obtained at the object $x$ with kernel $K$ is defined as (Webb, 2002):

$$\hat{f}(x) = \frac{1}{NV} \sum_{i=1}^{N} K \left( \frac{x - x_i}{H} \right).$$  \hspace{1cm} (1)

The size of the bin is given by a scale vector $H = [h_1 \ldots h_d]$ in $d$-dimensional space, and the matrix operation “./” is the element-by-element division of two equal-sized matrices or vectors. The data volume $V$ is $\prod_{i=1}^{d} h_i$. $K$ is a multivariate kernel, defined by a product of univariate kernels $K_i$. $K(z) = K_1(z_1)K_2(z_2)\ldots K_d(z_d)$, where $z_i$ is the value of $z$ on the dimension $i$. In particular, $K \left( \frac{x - x_i}{h} \right) = K_1 \left( \frac{x - x_{i1}}{h_1} \right)K_2 \left( \frac{x - x_{i2}}{h_2} \right)\ldots K_d \left( \frac{x - x_{id}}{h_d} \right)$. A list of common univariate kernels is given in Table 1. A triangular or Gaussian kernel function is normally used (Fig. 1).

In (1), bin $V$ is fixed in size. If bin $V$ is defined just like in KNN (Webb, 2002), where the volume around object $x$, $V_x$, is adjusted to include the KNN objects, the method is called KNN-kernel and given by

$$\hat{f}(x) = \frac{1}{NV_x} \sum_{i=1}^{N} K \left( \frac{x - x_i}{H_x} \right),$$  \hspace{1cm} (2)

where $H_x$ is a scale vector $[h_1^x \ldots h_d^x]$ of the volume $V_x$ in $d$-dimensional space.

The idea of KNN-density is first introduced by Loftsgaarden and Quesenberry (1965) and then generalized by Terrell and Scott (1992), where the Euclidean distance or Mahalanobis distance to the $k$th nearest neighbor is used. Here, we
Table 1
Commonly used univariate kernels; where \( z = (x - x_i) / H \)

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rectangular</td>
<td>( \frac{1}{2} ) if ( z^T z &lt; 1 ), 0 otherwise</td>
</tr>
<tr>
<td>Triangular</td>
<td>( 1 -</td>
</tr>
<tr>
<td>Biweight</td>
<td>( \frac{15}{16} (1 - z^T z)^2 ) if ( z^T z &lt; 1 ), 0 otherwise</td>
</tr>
<tr>
<td>Gaussian</td>
<td>( \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{z^T z}{2} \right) )</td>
</tr>
<tr>
<td>Bartlett–Epanechnikov</td>
<td>( \frac{1}{5} (1 - z^T z / 5)^{\frac{1}{2}} ) if ( z^T z &lt; \sqrt{5} ), 0 otherwise</td>
</tr>
</tbody>
</table>

use the volume \( V_x \), which is more general. KNN-kernel can also be seen as a case of variable kernel density estimation methods (Terrell and Scott, 1992; Silverman, 1986). KNN itself obviously is a simply case of KNN-density estimation where the uniform kernel is used. Readers are referred to Webb (2002) for a complete overview of nonparametric kernel density estimation methods.

The KNN-kernel method has two advantages over other methods for clustering purpose. Without the kernel, the first arises from density estimate is non-smooth; using a kernel makes the KNN-kernel estimator smooth. The second advantage is the result of the application of KNN and allows for an adaptive kernel width: a broader kernel in low-density regions and a narrower kernel in high-density regions. Comparing with fixed kernel width methods, abnormal small-density peaks appear in low-density regions (e.g. in Fig. 3a), which will result in many small clusters found with ordinary density-based clustering. Hence, the KNN-kernel method is useful for clustering, even though it is not better than the fixed kernel scheme for the purpose of estimating a density, due to an overestimate of density in the tails of the distribution (Terrell and Scott, 1992).

These features are demonstrated in Fig. 2, where the KNN and the KNN-kernel methods are applied to a synthetic data set, which includes 500 objects generated from one Gaussian distribution.

The KNN-kernel is smoother than that provided by the KNN method and gives an overestimate of density in the tails of the distribution. This feature is not good for the purpose of estimating a “true” density function but it will be useful for the classification purpose, in which it provides smoother functions with higher density values for low-density areas and lower-density values for high-density regions.

The other simple example in Fig. 3 shows the advantage of the KNN-kernel on a data set containing two classes of different densities. Class one is a high density class containing 500 objects generated from one Gaussian distribution (mean = 0 and \( s = 1 \)). Class two is a low density class containing 150 objects generated from one Gaussian distribution (mean = 100 and \( s = 10 \)). The kernel-based estimation method (1) with bin-size = 1.3 provides a smooth estimate for the first class but a bad estimate for the second class, showing many sharp peaks, due to the aforementioned (the “equal- and fixed volume-size”) problem of the kernel-based method (Fig. 3a). In contrast, the KNN-kernel method (2) with \( k = 100 \) provides a smooth density estimate for both classes (Fig. 3b).

In general, nonparametric methods are sensitive to the choice of the smoothing parameter. If it is too small, the density estimate is too detailed, showing many sharp peaks (as in Fig. 3a, the kernel method for cluster 2). If it is too
Fig. 2. KNN and KNN-kernel estimation on the sample data set containing 500 objects generated from one Gaussian distribution (mean $= 0$ and $s = 1$), with $k = 100$. The dotted line is the theoretical pdf function for the data set.

Fig. 3. Density estimation functions for the data set of two classes of different densities.

large, the structure of the density function is lost. Hand (1981) showed that the smoothing parameter can be estimated from the average distance of $k$ nearest neighbors. The KNN-kernel method, on the other hand, forms a flexible way to deal with a complex data set, where densities can be very different between clusters. Then, the smoothing parameter values are adapted locally for different clusters.
3. KNN-kernel density-based clustering

3.1. Classification rule based on KNN-kernel density estimates

The most common ways to assign objects to clusters, also called classification rules, are based on Bayes’ decision rule:

\[ p(x | \omega_i) p(\omega_i) > p(x | \omega_j) p(\omega_j), \quad \forall j \neq i, \]  

(3)

where \( p(x | \omega_i) \) is the class-conditional density function at \( x \) of each class \( \omega_i \) and \( p(\omega_i) \) is the prior probability function. The class-conditional density function can be estimated by the nonparametric KNN-kernel method, mentioned earlier:

\[ \hat{p}(x | \omega_i) = \frac{1}{n_i V_x} \sum_{x_j \in \omega_i} K((x - x_j) \cdot H_x), \]  

(4)

where \( n_i \) is the size of cluster \( \omega_i \), and \( \sum_i n_i = N \). Bayes’ KNN-kernel class-condition can be rewritten as

\[ \frac{1}{n_i V_x} \left( \sum_{x_j \in \omega_i} K((x - x_i) \cdot H_x) \right) p(\omega_i) > \frac{1}{n_j V_x} \left( \sum_{x_l \in \omega_j} K((x - x_l) \cdot H_x) \right) p(\omega_j), \quad \forall j \neq i. \]  

(5)

The prior probability functions \( p(\omega_i) \) and \( p(\omega_j) \) are normally estimated by \( n_i / N \) and \( n_j / N \), respectively. Then, the KNN-kernel Bayes’ class-condition can be simplified:

\[ \sum_{x_j \in \omega_i} K((x - x_i) \cdot H_x) > \sum_{x_l \in \omega_j} K((x - x_l) \cdot H_x), \quad \forall j \neq i. \]  

(6)

Thus, the decision rule used here is the same as the one in the KNN classifier (Webb, 2002) in the supervised classification method, but the density estimation is replaced by the KNN-kernel. The advantage of this for clustering is illustrated in the following section.

3.2. The KNNCLUST algorithm

We propose in this section KNNCLUST as a “hard” clustering algorithm, which assigns each object \( x_i \) to one and only one cluster. Just like partitional clustering (Tran et al., 2005), which “seeks an organization of objects which optimizes a target function”, KNNCLUST forms clusters in order to maximize the total class-conditional density function for all objects defined by

\[ D = \sum_{i=1}^{N} \hat{p}(x_i | c), \]  

(7)

where the point \( x_i \) is assigned to cluster \( c \).

The framework of KNNCLUST is as follows:

**Steps of the algorithm:**

1. **Start:** \( N \) singleton clusters, the number neighbors \( k \), and the KNN table \( T \) of size \( (N \times k) \), the list of KNN of all samples.
2. **Iteration:** re-calculate cluster memberships of all points using the class-condition (6) in order to maximize the function \( D \).
   
   **STOP:** if no, or only a few cluster memberships change (stop-condition). Otherwise LOOP and start new iteration (Step 2).

Using the KNN-kernel Bayes’ class-condition (6), in Step 2, \( \hat{p}(x_i | c) \) is replaced by \( \hat{p}(x_i | d) = \max(\hat{p}(x_i | j) \forall j \in C) \) for all points. The old membership \( c \) is replaced by new membership \( d \) of object \( x_i \). At the end of
iteration, there may be an empty cluster because all points were moved to other clusters. This cluster is removed from the system and the total number of clusters is decreased by one. The algorithm ends if the stop-condition is fulfilled. Note that \( \hat{p}(x_i | c) \) never decreases at any stage. Therefore, eventual convergence is assured.

In KNNCLUST, only the triangular kernel is recommended for the kernel function \( K \) in KNN-kernel Bayes’ class-condition (6) to reduce computation time. Using the Gaussian kernel gives similar results but is more time consuming. The rectangular kernel (equivalent to the well-known KNN class-condition, often-used in supervised classification) is not used here. It leads to problems in the initial state where the KNN estimated density values at any point are equal for all clusters.

A simple example in Fig. 4 shows how KNNCLUST performs on a simple data set of eight object values in a 1D space with \( k = 2 \). Each row plots objects in one particular step of the process when the membership is changed. For example, iteration one starts with object one, \( x_1 \). Because \( k = 2 \), the width of the bin around \( x_1 \) is given by

\[
H_{x_1} = x_3 - x_1.
\]

By applying the triangular kernel we obtain

\[
\hat{p}(x_1 | \ast) = 1 - \left| (x_1 - x_2) / H_{x_1} \right|,
\]

\[
\hat{p}(x_1 | o) = 1 - \left| (x_1 - x_3) / H_{x_1} \right|.
\]

It is obvious that (the class-condition Eq. (6)), so \( x_1 \) is assigned to the cluster of object \( x_2 \), indicated with symbol \( \ast \). The process is repeated to all other objects in turn; this concludes one iteration. The order in this case is random, e.g., object six is considered at step three of iteration one. Only two iterations are needed for clustering the data set into two clusters (o and \( \Delta \)).
In general, the object order, in which the objects are considered, may influence the result of the algorithm. One may order objects by their densities, in which higher density object is taken before the lower. However, density values are changed during iteration and the re-ordering at every step takes a lot of the computation time. In practice, objects may be processed in any convenient order. We have not seen any performance degradation.

3.2.1. Computational complexity

The computational complexity of KNNCLUST depends mainly on the calculation of KNN table, the list of KNN of all objects, which is very expensive. For example, if we acquire KNN query for each object independently, the simplest way is to order all distances from this object to other objects, which leads to a complexity of $O(N \log(N))$. However, there are many ways to make it more efficient; e.g. integrate information on all queries; see (Dasarathy, 1991) for a summary. The R-tree indexing technique is often utilized, e.g. in DBSCAN.

3.2.2. User-defined parameters

Apart from the choice of the kernel, the algorithm requires only one parameter, the number of neighborhood points, $k$. The smaller $k$, the more detail there is in the clustering and the more clusters can be obtained. In contrast, with a higher value for $k$, the clustering result is "smoother" and a smaller number of clusters is obtained. In all cases, $k$ should be smaller than the size of the smallest cluster, because this cluster will otherwise be missed. It may be difficult to find an optimal value of $k$ for a data set which has clusters of very different size. It is recommended to use several values of $k$, and to pick the one that captures the relevant features of data set. However, as will be shown below, in practice there will be a range of $k$ values that give quite similar results.

3.3. Comparison of KNNCLUST to other clustering methods

KNNCLUST is not an agglomerative hierarchical clustering algorithm (Tran et al., 2005), where a pair of clusters is merged based on the similarity between pairs of clusters. KNNCLUST is more like partitional clustering (Tran et al., 2005), where the probability density function (pdf) is used instead of normally used distances, e.g., Euclidean or Mahalanobis distances. In this type of clustering, objects are allowed to be reassigned to other clusters. However, the number of clusters needs to be defined in partitional clustering methods, whereas it is automatically determined by KNNCLUST. Partitional clustering, such as Fuzzy C-means or mixture modeling by expectation maximization (EM) is sensitive to the initial choice of cluster centers and noise/outliers present in the data set. This is not the case for KNNCLUST. Moreover, different from mixture model clustering EM, KNNCLUST does not require clusters to have a certain statistical distribution; e.g. the Gaussian distribution is often used in EM. KNNCLUST also differs from ordinary density-based clustering by constructing the class-condition instead of using a density estimation function for detecting separation density valleys between clusters. As a consequence, KNNCLUST is less suited for finding very elongated clusters or clusters with strange shapes, something that is possible with ordinary density-based clustering. On the other hand, it can be used in cases when clusters have very different densities where other density-based methods cannot. Last but not least, KNNCLUST can work well with data in high-dimensional feature spaces which is difficult for many clustering algorithms, such as the EM method.

4. Results

In this section, we demonstrate the effectiveness of KNNCLUST on two data sets, a simulated data set and a remote sensing compact airborne spectrographic imager (CASI) image.

The 2D simulated data set in Fig. 5 contains four classes having sizes of 600, 400, 200 and 200 objects. To make the simulated data set more realistic, class one is constructed from two overlapping Gaussians. The other three are generated from three single Gaussian distributions with very different cluster densities; the variances of clusters three and four are 10-times smaller than cluster one and two, respectively. The Gaussians are illustrated by ellipses, shown in Fig. 5. In the plot, classes two and three (in the middle-right of Fig. 5) are located in very small areas, and are difficult to distinguish.

Using KNNCLUST, the four-cluster results can be obtained using $k$ values in the range [180, . . . , 220] with total accuracy more than 95% (by counting the misclassified objects). The evaluation of the clustering result can be done by
Fig. 5. The simulated data set. Class one is a mixture of two Gaussians and the other three are generated from three single Gaussian distributions with very different in cluster densities.

Fig. 6. Clustering result by KNNCLUST with $k = 180$; the total accuracy is 95.9%.

a post-processing scheme, which assigns a cluster to the most similar class in reference data. As an example, the result of KNNCLUST by $k = 180$ is given in Fig. 6.

The often-used density-based clustering, DBSCAN (Ester et al., 1996), is applied to the data set as well. The clustering result (in Fig. 7) is very poor, as expected since clusters have very different densities. The best results of DBSCAN on two situations are discussed hereafter. In order to recognize classes three and four, a very high density threshold with min-points = 10 and $\varepsilon = 20$ is set, leading to objects of class one and two to be classified as noise (Fig. 7a). In the opposite situation, using a low dense threshold with min-points = 20 and $\varepsilon = 950$, classes three and four are merged (Fig. 7b).

We also compared KNNCLUST with the state-of-the-art mixture model clustering by EM on this data set. The EM algorithm is very sensitive to initialization (Seidel et al., 2000; McLachlan and Peel, 2000); a random initialization strategy is normally used. We performed EM to four clusters 100 times and the best clustering result in terms of the maximal likelihood criterion is shown in Fig. 8a. Gaussian mixture model clustering assumes clusters to have normal distribution. Because of the mixture of two Gaussians in class one, EM needs two Gaussians to describe the class and the class three and four are merged together. EM works better when working with five clusters, the class three and four can be recognized. However, cluster one still divided to two parts (Fig. 8b). Together with the difficulty of the initialization of and the identifying the number of clusters, KNNCLUST works better than EM for this data set.
The second experiment is done on a multispectral remote sensing satellite image recorded by a CASI scanner from the natural environment research council (NERC). The image was taken at 1536 m over an area in the Klompenwaard, The Netherlands, during August 2001. The data set for this study contains 10 bands from 437 to 890 nm, with bandwidths of 10 nm, except for band 9 with 8 nm. The study area has size of $30 \times 255$ pixels with 3 m resolution, covering 68 850 $m^2$.

Principal components analysis (PCA) is used for reducing the complexity and visualization of the results. The original multispectral data were mean zero and unit variance and compressed via a PCA to the first four principal components, which account for more than 99.8% of the spectral variance. KNNCLUST was applied on both the original 10-bands data set and the four-component compressed data set. The result shows no difference between the two cases. For convenience, the results shown in this paper are shown in PCA space.

Fig. 9 shows the gray-scale images of the first and second principal components, explaining 71% and 27% of the variance in the data, respectively. Six main object patterns have been estimated for the area from the work of Van den Berg (v/d Berg, 2001).

The clusters are different in density, as can be seen clearly in Fig. 10; e.g., the clusters of the river (A) and the lake (B) are very dense, with a long narrow shape containing approximately 1000 points, compared to the large cluster corresponding to sand and vegetation (D) of 1440 points.

First, we apply an often-used density-based clustering, DBSCAN (Ester et al., 1996). DBSCAN clustering is a spanning process, grouping points connected by high-density cells and dividing points separated by low-density cells. The threshold is a user-settable parameter, $\varepsilon$. The second parameter that should be set is min-points, the minimum
Fig. 9. The gray-scale images of the first two principal components (PC1 on the left, PC2 on the right), and the six main object classes that have been identified in the area.

number of objects in the neighborhood. The number of clusters is found automatically by DBSCAN. For this data set, many values for both user parameters have been used but none of them gave good results. Some examples are shown in Fig. 11(a–d). This is caused by the absence of a global threshold of the density for the whole data set. If the density parameter is adjusted to identify low density regions such as the sand and the vegetation cluster (D), then it is too high to distinguish between clusters B and C, as well as between clusters E and F in Fig. 11c and d. In other settings, cluster D could not be recognized due to its low density (Fig. 11a and b).

KNNCLUST was applied using the following values of $k$: [450, 500, 550, 600, 650]. In all cases, six clusters are found. The score-plots of two first PCs, showing the clusters obtained with $k = 550$, is given in Fig. 12c. The cluster sizes range from 950 to 1800 points. Seven and five clusters are found when values of $k$ are 300 and 700, respectively.
Fig. 11. Score-plots of two first PCs by DBSCAN with parameter min-points is 25 (a) 8 clusters found by $\varepsilon = 300$, (b) 6 clusters found by $\varepsilon = 400$, (c) 4 clusters found by $\varepsilon = 500$ and (d) 2 clusters found by $\varepsilon = 900$.

Fig. 12. Score-plots of two first PCs and result images of six clusters obtained by (a) $K$-means (the best of 100 runs); (b) EM (the best of 100 runs) and (c) KNNCLUST with $k = 550$.

The method is also compared with $K$-means, and EM and the best results after 100 runs by randomly initialization are shown in Fig. 12a and b, respectively. In this case, the image result of the KNNCLUST (Fig. 12c) and EM are comparable and look much smoother than the one obtained by $K$-means, mainly because of the vegetation area (D). $K$-means incorrectly joins the lake (B) and the river (A), and divides the vegetation area D into two clusters.

The stability and the compactness of the clustering result also can be studied by using an index which measures the ratio of within-cluster variation and between-cluster variation (Brereton, 1992). A lower value indicates a higher
compactness. This index is not designed for a data set with clusters of different shapes. Nevertheless, it might provide an idea about the stability and the compactness of the clustering results.

Fig. 13 shows compactness index values of 100 replicated runs for K-means. It shows that K-means is not stable with a minimum value of the compactness index of 0.2063 and a maximum value of 0.3756. Also in the figure are the smallest and largest compactness values for KNNCLUST using all five values of $k$ leading to six clusters. The smallest value for the index for KNNCLUST is 0.2077 when $k = 650$, and the largest value is 0.2081 when $k = 450$.

They are comparable to the best case obtained by K-means. The small variance of the compactness index indicates that KNNCLUST is not very sensitive to the values of $k$ in the selected range.

5. Summary and conclusion

Many clustering algorithms for multivariate data, such as, EM or most density-based methods, suffer from the problem of clusters in a high-dimensional feature space with different densities. This is not the case for our new proposed algorithm, KNNCLUST, making use of a KNN-kernel density estimator using the triangular kernel. For a given kernel function, KNNCLUST has only one parameter, $k$, the number of neighbors. In most cases, it is not difficult to find a range of $k$ for which clustering results are stable. The number of clusters is automatically determined by the algorithm upon convergence. The computational complexity for the algorithm is quite high, mainly caused by the calculation of the KNN distance matrix. However, indexing techniques (Dasarathy, 1991) could be used to improve the situation for a larger data set. KNNCLUST is less suited for finding very elongated clusters or clusters with strange shapes, something that is possible with ordinary density-based clustering. However, KNNCLUST can detect more “natural” clusters that are required to follow any type of statistical distributions like in mixture model clustering. In conclusion, it is a very good tool to cluster moderately-sized multivariate data set where the clusters are very different in densities.

Acknowledgement

We thank Gertjan Geerling, Department of Environmental Studies, for sharing the data and stimulating discussions.

References


