Inverse Weighted Clustering Algorithm

Wesam Barbakh and Colin Fyfe,
The University of Paisley,
Scotland.
email:wesam.barbakh,colin.fyfe@paisley.ac.uk

Abstract

We discuss a new form of clustering which overcomes some of the problems of traditional K-means such as sensitivity to initial conditions. We illustrate convergence of the algorithm on a number of artificial data sets. We then introduce a variant of this clustering which preserves some aspects of global topology in the organisation of the centres. We illustrate on artificial data before using it to visualise some standard datasets.

1 Introduction

The K-Means algorithm is one of the most frequently used investigatory algorithms in data analysis. The algorithm attempts to locate K prototypes or means throughout a data set in such a way that the K prototypes in some way best represents the data. The algorithm is one of the first which a data analyst will use to investigate a new data set because it is algorithmically simple, relatively robust and gives 'good enough' answers over a wide variety of data sets: it will often not be the single best algorithm on any individual data set but be close to the optimal over a wide range of data sets. However the algorithm is known to suffer from the defect that the means or prototypes found depend on the initial values given to them at the start of the simulation. There are a number of heuristics in the literature which attempt to address this issue but, at heart, the fault lies in the performance function on which K-Means is based.
A variation on K-means is the so-called soft K-means [7] in which prototypes are allocated according to

$$\mathbf{m}_k = \frac{\sum_n r_{kn}\mathbf{x}_n}{\sum_{j,n} r_{jn}} \quad (1)$$

$$\text{where e.g. } r_{kn} = \frac{\exp(-\beta d(\mathbf{x}_n, \mathbf{m}_k))}{\sum_j \exp(-\beta d(\mathbf{x}_n, \mathbf{m}_j))} \quad (2)$$

and $d(a, b)$ is the Euclidean distance between a and b. Note that the standard K-means algorithm is a special case of the soft K-means algorithm in which the responsibilities, $r_{kn} = 1$ when $\mathbf{m}_k$ is the closest prototype to $\mathbf{x}_n$ and 0 otherwise. However the soft K-Means does increase the non-localness of the interaction since the responsibilities are typically never exactly equal to 0 for any data point-prototype combination.

However there are still problems with soft K-Means.We find that with soft K-means it is important to choose a good value for β; if we choose a poor value we may have poor results in finding the clusters. Even if we choose a good value, we will still find that soft K-means has the problem of sensitivity to the prototypes' initialization. In this paper, we investigate a new clustering algorithm that solves the problem of sensitivity in K-means and soft K-means algorithms. We are specifically interested in developing an algorithm which are effective in a worst case scenario: when the prototypes are initialised very far from the data points. If an algorithm can cope with this scenario, it should be able to cope with a more benevolent initialization.

2 Inverse Weighted Clustering Algorithm (IWC)

Consider the following performance function:

$$J_I = \sum_{i=1}^{N}\sum_{k=1}^{K} \frac{1}{\| \mathbf{x}_i - \mathbf{m}_k \|^P} \tag{3}$$

$$\frac{\partial J_I}{\partial \mathbf{m}_k} = \sum_{i=1}^{N} P(\mathbf{x}_i - \mathbf{m}_k) \frac{1}{\| \mathbf{x}_i - \mathbf{m}_k \|^{P+2}} \tag{4}$$

$$\begin{aligned} \frac{\partial J_I}{\partial \mathbf{m}_k} &= 0 \Longrightarrow \\ \mathbf{m}_k &= \frac{\sum_{i=1}^{N} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^{P+2}} \mathbf{x}_i}{\sum_{i=1}^{N} \frac{1}{\|\mathbf{x}_i - \mathbf{m}_k\|^{P+2}}} \\ &= \frac{\sum_{i=1}^{N} b_{ik}\mathbf{x}_i}{\sum_{i=1}^{N} b_{ik}} \end{aligned} \tag{5}$$

where

$$b_{ik} = \frac{1}{\| \mathbf{x}_i - \mathbf{m}_k \|^{P+2}} \tag{6}$$

The partial derivative of J_I with respect to $\mathbf{m}_k$ will maximize the performance function J_I. So the implementation of (5) will always move $\mathbf{m}_k$ to the closest data point to maximize J_I to ∞, see Figure 1.

However, the implementation of (5) will not identify any clusters as the prototypes always move to the closest data point. But the advantage of this performance function is that it doesn't leave any prototype far from data: all the prototypes join the data.

We can enhance this algorithm to be able to identify the clusters without losing its property of pushing the prototypes inside data by changing b_{ik} in (6) to the following:

$$b_{ik} = \frac{\| \mathbf{x}_i - \mathbf{m}_{k*} \|^{P+2}}{\| \mathbf{x}_i - \mathbf{m}_k \|^{P+2}} \tag{7}$$

where $\mathbf{m}_{k*}$ is the closest prototype to $\mathbf{x}_i$.

With this change, we have an interesting behavior: (7) works to maximize J_I by moving the prototypes to the freed data points (or clusters) instead of the closest data point (or local cluster).

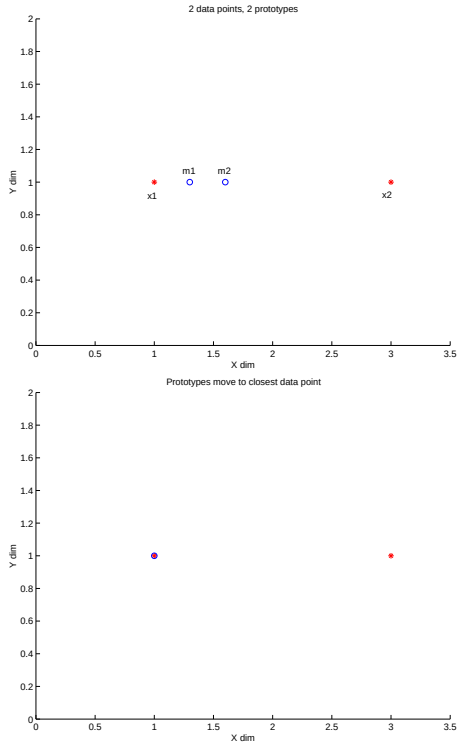

Figure 1: Top: two data points and two prototypes. Bottom: the result after applying (5).

We will call this the Inverse Weighted Clustering Algorithm (IWC).

Note that (6) and (7) never leaves any prototype far from the data even if they are initialized outwith the data. The prototypes always are pushed to join the closest data points using (6) or to join the free data points using (7). But (6) doesn't identify clusters while (7) does.

(7) keeps the property of (6) of pushing the prototypes to join data, and provides the ability of identifying clusters.

Consider we have two data points and two prototypes, so we have the following possibilities:

1. Two prototypes are closest to one data point, as shown in Figure 1, top.

2. One prototype is closest only to one data point,

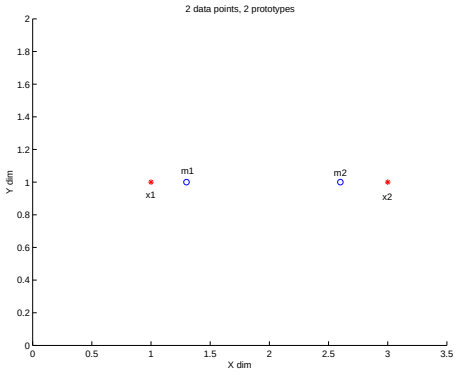

Figure 2: One prototype is closest only to one data point.

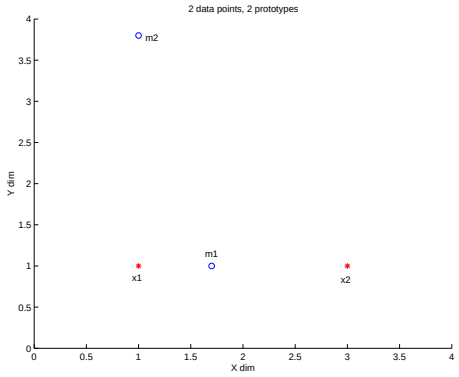

Figure 3: One prototype is closest to both data points.

as shown in Figure 2.

3. One prototype is closest to both data points, as shown in Figure 3.

Analysis for first possibility

With (6),

$$\begin{aligned} \mathbf{m}_1 &= \frac{\frac{1}{d_{11}^{(P+2)}}\mathbf{x}_1 + \frac{1}{d_{21}^{(P+2)}}\mathbf{x}_2}{\frac{1}{d_{11}^{(P+2)}} + \frac{1}{d_{21}^{(P+2)}}} \qquad (8) \\ where & \\ d_{ik} &= \| \mathbf{x}_i - \mathbf{m}_k \| \end{aligned}$$

if $d_{11} < d_{21}$ ($\mathbf{m}_1$ is closer to $\mathbf{x}_1$)
$\mathbf{m}_1$ will move toward $\mathbf{x}_1$
else if $d_{11} > d_{21}$ ($\mathbf{m}_1$ is closer to $\mathbf{x}_2$)
$\mathbf{m}_1$ will move toward $\mathbf{x}_2$
else ($\mathbf{m}_1$ located at the mean of data)
$\mathbf{m}_1$ will remain at the mean of data

The same as above for the prototype $\mathbf{m}_2$, $\mathbf{m}_2$ will move independently toward the closest data point without taking into account how the other prototypes respond. There is no way to identify clusters using (6).

With (7), b_{ik} is always in the range [0 1].

$$\mathbf{m}_1 = \frac{\frac{d_{11}^{(P+2)}}{d_{11}^{(P+2)}}\mathbf{x}_1 + \frac{d_{22}^{(P+2)}}{d_{21}^{(P+2)}}\mathbf{x}_2}{\frac{d_{11}^{(P+2)}}{d_{11}^{(P+2)}} + \frac{d_{22}^{(P+2)}}{d_{21}^{(P+2)}}}$$

Normally $\frac{d_{22}^{(P+2)}}{d_{21}^{(P+2)}} < 1$, $\mathbf{m}_1$ will move toward $\mathbf{x}_1$.

(If this value =1, then $\mathbf{m}_1$ will move to the mean.)

$$\mathbf{m}_2 = \frac{\frac{d_{11}^{(P+2)}}{d_{12}^{(P+2)}}\mathbf{x}_1 + \frac{d_{22}^{(P+2)}}{d_{22}^{(P+2)}}\mathbf{x}_2}{\frac{d_{11}^{(P+2)}}{d_{12}^{(P+2)}} + \frac{d_{22}^{(P+2)}}{d_{22}^{(P+2)}}} \qquad (9)$$

Normally $\frac{d_{11}^{(P+2)}}{d_{12}^{(P+2)}} < 1$, $\mathbf{m}_2$ will move toward $\mathbf{x}_2$, although $\mathbf{m}_2$ is closer to $\mathbf{x}_1$.

Notice, if we have two prototypes, one initialized at the mean and the second initialized anywhere between the two data points, we will find each prototype is closer to one data point and hence after the next iteration each data point will move towards a data point. So there is no problem if any prototype moves toward the mean.

Analysis for second possibility

(6) and (7) give the same effect. Each prototype will move toward the closest data point.

Analysis for third possibility

With (6), each prototype moves to the closest data point, so for Figure 3, $\mathbf{m}_1$ and $\mathbf{m}_2$ will move to the same data point (1,1).

With (7), after the first iteration, $\mathbf{m}_1$ will move to the mean of data as it is the closest for both data points, and $\mathbf{m}_2$ will move to a location between the two data points and then we get the first or second possibility for the next iteration.

$$\mathbf{m}_1 = \frac{\frac{d_{11}^{(P+2)}}{d_{11}^{(P+2)}}\mathbf{x}_1 + \frac{d_{21}^{(P+2)}}{d_{21}^{(P+2)}}\mathbf{x}_2}{\frac{d_{11}^{(P+2)}}{d_{11}^{(P+2)}} + \frac{d_{21}^{(P+2)}}{d_{21}^{(P+2)}}}$$

$$\mathbf{m}_2 = \frac{\frac{d_{11}^{(P+2)}}{d_{12}^{(P+2)}}\mathbf{x}_1 + \frac{d_{21}^{(P+2)}}{d_{22}^{(P+2)}}\mathbf{x}_2}{\frac{d_{11}^{(P+2)}}{d_{12}^{(P+2)}} + \frac{d_{21}^{(P+2)}}{d_{22}^{(P+2)}}}$$

From extensive simulations, we can confirm that (7) always push the prototypes toward the data.

2.1 Simulation

In Figure 4, the prototypes have all been initialized within a single cluster. As shown in the figure, while K-means failed to identify clusters, middle, IWC based on (7) identified all of them successfully, bottom diagram.

Figure 5 shows the result of applying IWC algorithm to the same artificial data set but with bad initialization of the prototypes. As shown in the figure, Inverse Weighted Clustering algorithm succeeds in identifying the clusters under this bad initialization, bottom, while K-means failed, middle.

In general initializing prototypes far from data is an unlikely situation to happen, but it may be that all the prototypes are in fact initialized very far from a particular cluster.

In Figure 6, we have 40 data points, each of which represents one cluster. All the prototypes are initialized very close together. The IWC algorithm, bottom, gives a better result than K-means, middle. Figure 7 shows the result of applying IWC algorithm to the same artificial data set, 40 clusters, but with bad initialization of the prototypes. As shown in the figure, K-means failed to identify clusters and there are 39 dead prototypes due to the bad initialization, middle, while the Inverse Weighted Clustering algorithm succeeded in identifying the clusters under this bad initialization, bottom.

3 A Topology Preserving Mapping

In this part we show how it is possible to extend Inverse Weighted Clustering algorithm (IWC) to provide a new algorithm for visualization and topology-preserving mappings.

3.1 Inverse weighted Clustering Topology-preserving Mapping (ICToM)

A topographic mapping (or topology preserving mapping) is a transformation which captures some structure in the data so that points which are mapped close to one another share some common feature while points which are mapped far from one another do not share this feature. The Self-organizing Map (SOM) was introduced as a data quantisation method but has found at least as much use as a visualisation tool.

Topology-preserving mappings such as the Self-organizing Map (SOM) [6] and the Generative Topographic Mapping(GTM) [4] have been very popular for data visualization: we project the data onto the map which is usually two dimensional and look for structure in the projected map by eye. We have recently investigated a family of topology preserving mappings [5] which are based on the same underlying structure as the GTM.

The basis of our model is K latent points, $t_1, t_2, \cdots, t_K$, which are going to generate the K prototypes, $\mathbf{m}_k$. To allow local and non-linear modeling, we map those latent points through a set of M basis functions, $f_1(), f_2(), \cdots, f_M()$. This gives us a matrix Φ where $\phi_{kj} = f_j(t_k)$. Thus each row of Φ is the response of the basis functions to one latent point, or alternatively we may state that each column of Φ is

the response of one of the basis functions to the set of latent points. One of the functions, $f_j()$, acts as a bias term and is set to one for every input. Typically the others are gaussians centered in the latent space. The output of these functions are then mapped by a set of weights, W, into data space. W is $M \times D$, where D is the dimensionality of the data space, and is the sole parameter which we change during training. We will use $\mathbf{w}_i$ to represent the i^{th} column of W and Φ_j to represent the row vector of the mapping of the j^{th} latent point. Thus each basis point is mapped to a point in data space, $\mathbf{m}_j = (\Phi_j W)^T$.

We may update W either in batch mode or with online learning: with the Topographic Product of Experts [5], we used a weighted mean squared error; with the Inverse Exponential Topology Preserving Mapping [1], we used Inverse Exponential K-means, with the Inverse-weighted K-means Topology-preserving Mapping (IKToM) [3, 2], we used Inverse Weighted K-means (IWK). We now apply the Inverse Weighted Clustering (IWC) algorithm to the same underlying structure to create a new topology preserving algorithm.

3.2 Simulation

3.2.1 Artificial data set

We create a simulation with 20 latent points deemed to be equally spaced in a one dimensional latent space, passed through 5 Gaussian basis functions and then mapped to the data space by the linear mapping W which is the only parameter we adjust. We generated 500 two dimensional data points, (x_1, x_2), from the function $x_2 = x_1 + 1.25\sin(x_1) + \mu$ where μ is noise from a uniform distribution in [0,1]. Final result from the ICToM is shown in Figure 8.

3.2.2 Real data set

Iris data set: 150 samples with 4 dimensions and 3 types.

Algae data set: 72 samples with 18 dimensions and 9 types

Genes data set: 40 samples with 3036 dimensions and 3 types

Glass data set: 214 samples with 10 dimensions and 6 types

We show in Figure 9 the projections of the real data sets onto a two dimensional grid of latent points using ICToM. The results are comparable with others we have with these data sets from a variety of different algorithms.

4 Conclusion

We have discussed a new form of clustering which has been shown to be less sensitive to poor initialisation than the traditional K-means algorithm. We have discussed the reasons for this insensitivity using simple two dimensional data sets to illustrate our reasoning.

We have also created a topology-preserving mapping with the Inverse Clustering Algorithm as its base and shown its convergence on an artificial data set. Finally we used this mapping for visualising some of our standard data sets.

The methods of this paper are not designed to replace those of other clustering techniques but to stand alongside them as alternative means of enabling data analysts to understand high dimensional complex data sets. Future work will compare these new algorithms with the results of our previous algorithms

References

[1] W. Barbakh. The family of inverse exponential k-means algorithms. *Computing and Information Systems*, 11(1):1–10, February 2007. ISSN 1352-9404.

[2] W. Barbakh, M. Crowe, and C. Fyfe. A family of novel clustering algorithms. In *7th international conference on intelligent data engineering and automated learning, IDEAL2006*, pages 283–290, September 2006. ISSN 0302-9743 ISBN-13 978-3-540-45485-4.

[3] W. Barbakh and C. Fyfe. Performance functions and clustering algorithms. *Computing and Infor-*

mation Systems, 10(2):2–8, May 2006. ISSN 1352-9404.

[4] C. M. Bishop, M. Svensen, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 1997.

[5] C. Fyfe. Two topographic maps for data visualization. *Data Mining and Knowledge Discovery*, 2006.

[6] Tuevo Kohonen. *Self-Organising Maps*. Springer, 1995.

[7] D. J. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press., 2003.

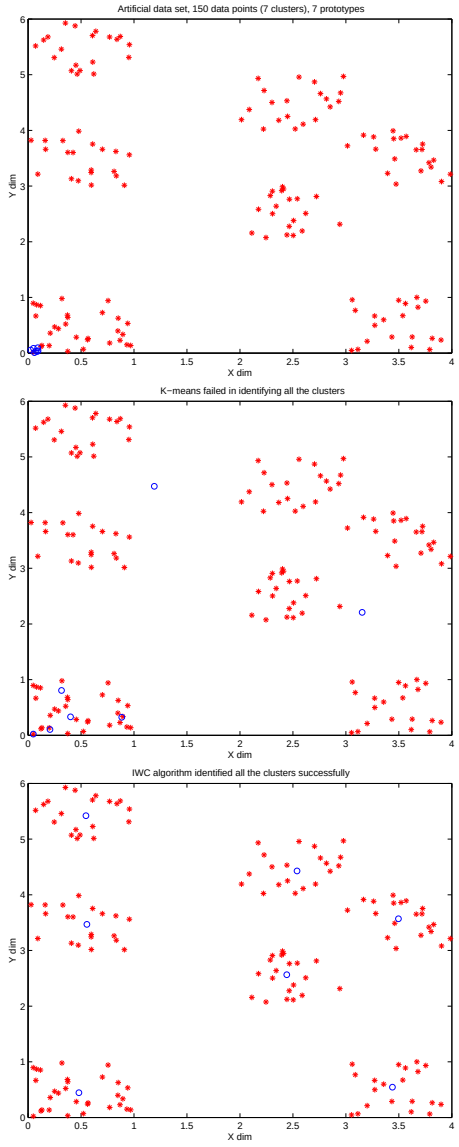

Figure 4: Top: Artificial data set: data set is shown as 7 clusters of red '*'s, prototypes are initialized to lie within one cluster and shown as blue 'o's. Middle: K-means result. Bottom: IWC algorithm result.

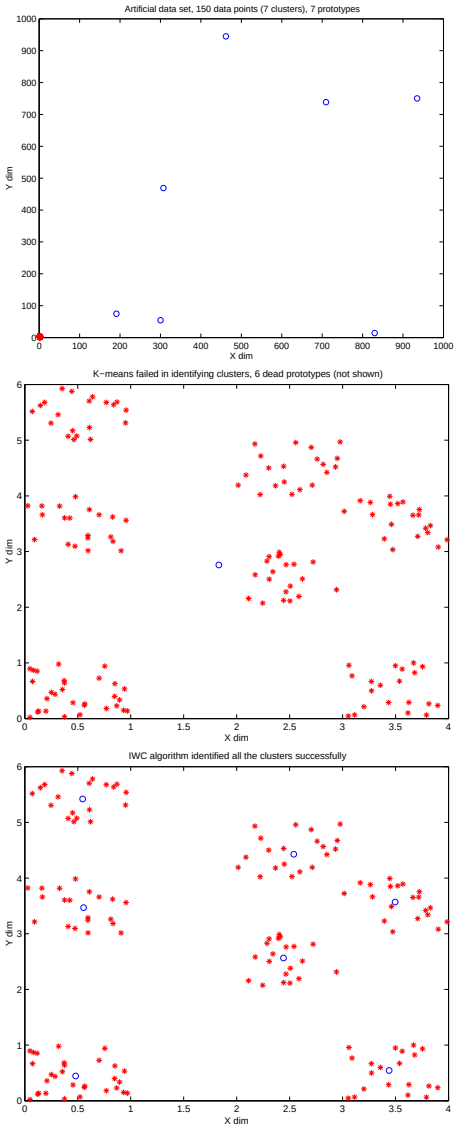

Figure 5: Top: Artificial data set: data set is shown as 7 clusters of red '*'s, prototypes are initialized very far from data and shown as blue 'o's. Middle: K-means result. Bottom: IWC algorithm result.

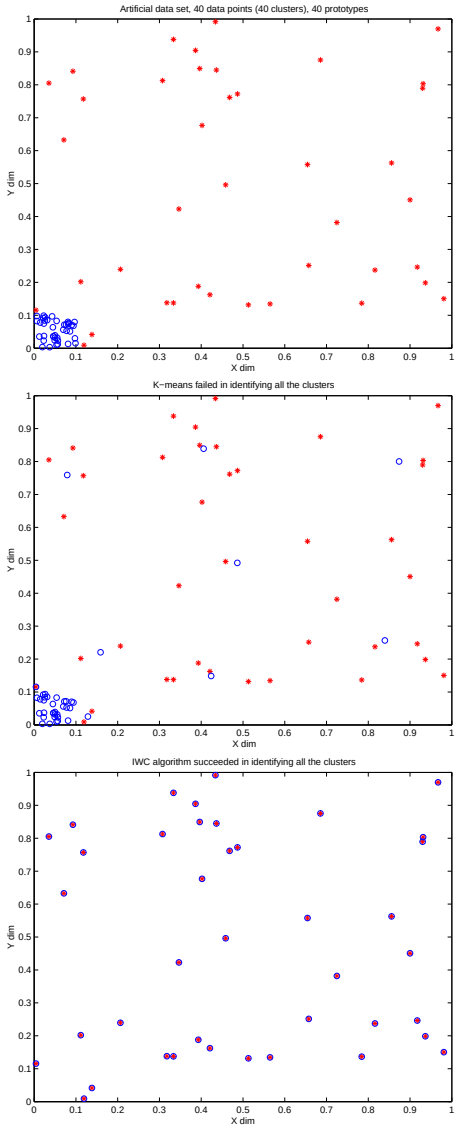

Figure 6: Top: Artificial data set: data set is shown as 40 clusters of red '*'s, 40 prototypes are initialized close together and shown as blue 'o's. Middle: K-means result. Bottom: IWC algorithm result.

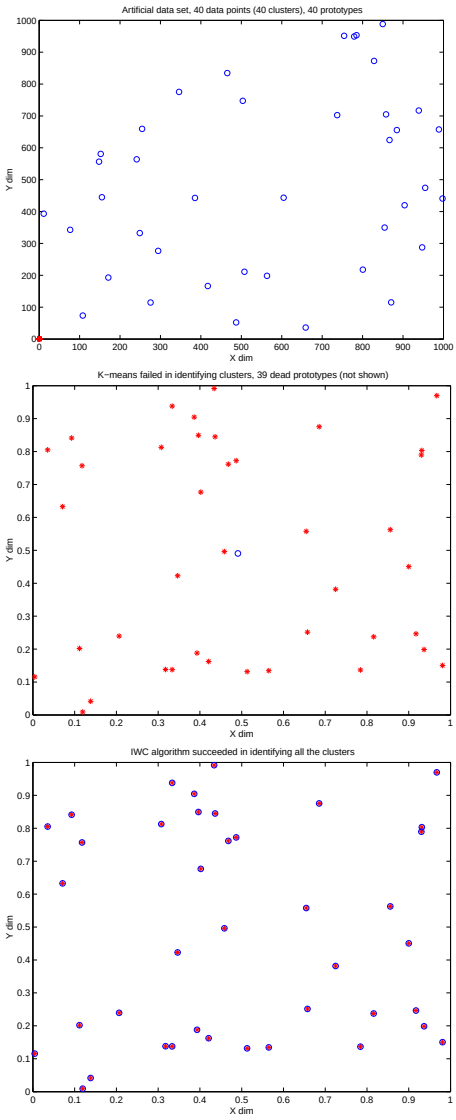

Figure 7: Top: Artificial data set: data set is shown as 40 clusters of red '*'s, 40 prototypes are initialized very far from data and shown as blue 'o's. Middle: K-means result. Bottom: IWC algorithm result.

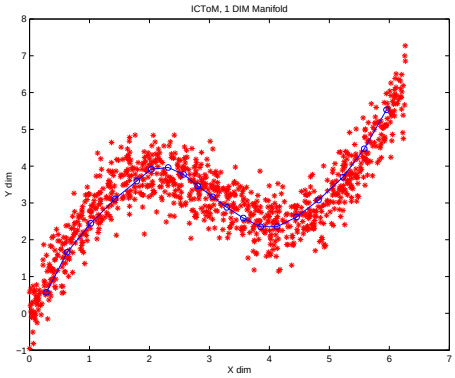

Figure 8: The resulting prototypes' positions after applying ICToM. Prototypes are shown as blue 'o's.

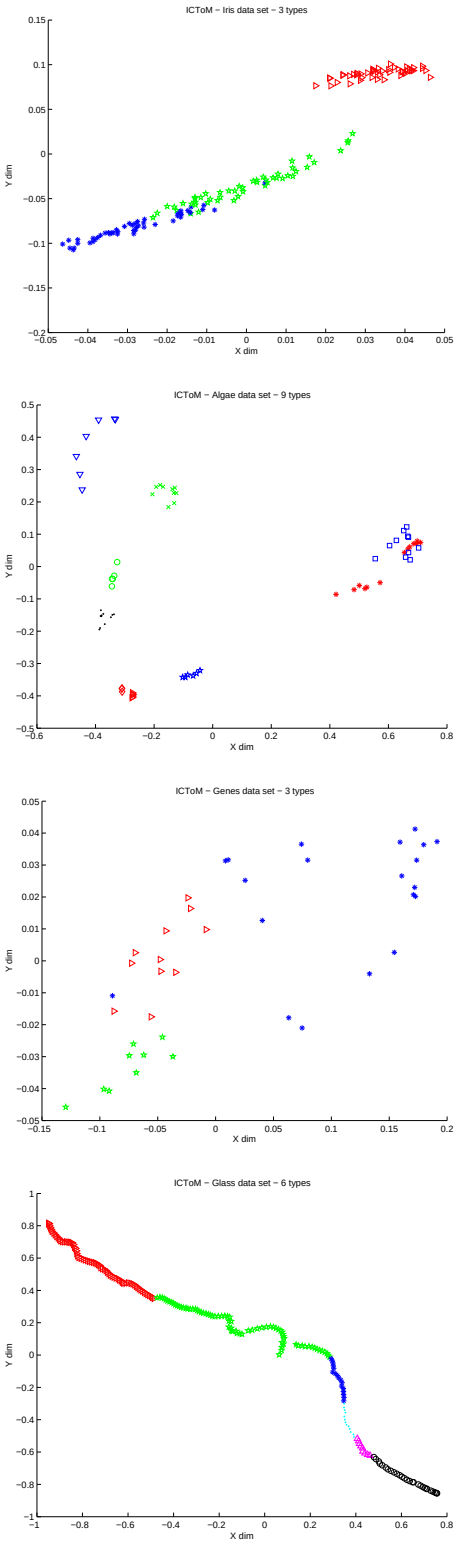

Figure 9: Visualisaton using the ICToM on 4 real data sets.