A Family of Novel Clustering Algorithms

Conference Paper *in* Lecture Notes in Computer Science · September 2006

DOI: 10.1007/11875581_34 · Source: DBLP

3 authors:

Wesam Ashour
Islamic University of Gaza
64 PUBLICATIONS **518** CITATIONS
SEE PROFILE

Malcolm Crowe
University of the West of Scotland
50 PUBLICATIONS **212** CITATIONS
SEE PROFILE

Colin Fyfe
Universidad de Burgos
95 PUBLICATIONS **986** CITATIONS
SEE PROFILE

A Family of Novel Clustering Algorithms

Wesam Barbakh, Malcolm Crowe, and Colin Fyfe

Applied Computational Intelligence Research Unit,
The University of Paisley,
Scotland
{wesam.barbakh, malcolm.crowe, colin.fyfe}@paisley.ac.uk

Abstract. We review the performance function associated with the familiar K-Means algorithm and that of the recently developed K-Harmonic Means. The inadequacies in these algorithms leads us to investigate a family of performance functions which exhibit superior clustering on a variety of data sets over a number of different initial conditions. In each case, we derive a fixed point algorithm for convergence by finding the fixed point of the first derivative of the performance function. We give illustrative results on a variety of data sets. We show how one of the algorithms may be extended to create a new topology-preserving mapping.

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: a typical program will converge to a local optimum. 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. In this paper, we investigate alternative performance functions and show the effect the different functions have on the effectiveness of the resulting algorithms. We are specifically interested in developing algorithms which are effective in a worst case scenario: when the prototypes are initialised at the same position which is very far from the data points. If an algorithm can cope with this scenario, it should be able to cope with a more benevolent initialisation.

2 Performance Functions for Clustering

The performance function or distortion measure for K-Means may be written as

$$J_K = \sum_{i=1}^{N} \min_{j=1}^{K} \| \mathbf{x}_i - \mathbf{m}_j \|^2 \tag{1}$$

E. Corchado et al. (Eds.): IDEAL 2006, LNCS 4224, pp. 283–290, 2006.

which we wish to minimise by moving the prototypes to the appropriate positions. Note that (1) detects only the centres closest to data points and then distributes them to give the minimum performance which determines the clustering. Any prototype which is still far from data is not utilised and does not enter any calculation that give minimum performance, which may result in dead prototypes, prototypes which are never appropriate for any cluster. Thus initializing centres appropriately can play a big effect in K-Means.

Recently, [7] there have been several investigations of alternative performance functions for clustering algorithms. One of the most effective updates of K-Means has been K-Harmonic Means which minimises

$$J_{HA} = \sum_{i=1}^{N} \frac{K}{\sum_{k=1}^{K} \frac{1}{d(\mathbf{x}_i, \mathbf{m}_k)^2}} \tag{2}$$

for data samples $\{\mathbf{x}_1, , \mathbf{x}_N\}$ and prototypes $\{\mathbf{m}_1, , \mathbf{m}_K\}$. Then we wish to move the centres using gradient descent on this performance function

$$\frac{\partial J_{HA}}{\partial \mathbf{m}_k} = -K \sum_{i=1}^{N} \frac{4(\mathbf{x}_i - \mathbf{m}_k)}{d(\mathbf{x}_i, \mathbf{m}_k)^3 \{\sum_{l=1}^{K} \frac{1}{d(\mathbf{x}_i, \mathbf{m}_l)^2}\}^2} \tag{3}$$

Setting this equal to 0 and "solving" for the $\mathbf{m}_k$'s, we get a recursive formula

$$\mathbf{m}_k = \frac{\sum_{i=1}^{N} \frac{1}{d_{i,k}^3 (\sum_{l=1}^{K} \frac{1}{d_{i,l}^2})^2} \mathbf{x}_i}{\sum_{i=1}^{N} \frac{1}{d_{i,k}^3 (\sum_{l=1}^{K} \frac{1}{d_{i,l}^2})^2}} \tag{4}$$

where we have used $d_{i,k}$ for $d(\mathbf{x}_i, \mathbf{m}_k)$ to simplify the notation. There are some practical issues to deal with in the implementation details of which are given in [7,6].

[7] have extensive simulations showing that this algorithm converges to a better solution (less prone to finding a local minimum because of poor initialisation) than both standard K-means or a mixture of experts trained using the EM algorithm. However we have investigated this algorithm using a number of extreme cases such as when the prototypes are initialised in identical positions and/or are far from the data (see below), and found the algorithm wanting. We thus investigate new algorithms.

3 A Family of Algorithms

We have previously investigated this initialization effect in detail in [1]. We might consider the following performance function:

$$J_A = \sum_{i=1}^{N} \sum_{j=1}^{K} \| \mathbf{x}_i - \mathbf{m}_j \|^2 \tag{5}$$

which provides a relationship between all the data points and prototypes, but it doesn't provide useful clustering at minimum performance since

$$\frac{\partial J_A}{\partial \mathbf{m}_k} = 0 \Longrightarrow \mathbf{m}_k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i, \forall k \tag{6}$$

Minimizing the performance function groups all the prototypes to the centre of data set regardless of the intitial position of the prototypes which is useless for identification of clusters.

We wish to form a performance function with following properties:

- Minimum performance gives an intuitively 'good' clustering.
- It creates a relationship between all data points and all prototypes.

(5) provides an attempt to reduce the sensitivity to prototypes' initialization by making a relationship between all data points and all prototypes while (1) provides an attempt to cluster data points at the minimum of the performance function. Therefore it may seem that what we want is to combine features of (1) and (5) to make a performance function such as:

$$J_1 = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} \| \mathbf{x}_i - \mathbf{m}_j \| \right] \min_{k=1}^{K} \| \mathbf{x}_i - \mathbf{m}_k \|^2 \tag{7}$$

We derive the clustering algorithm associated with this performance function by calculating the partial derivatives of (7) with respect to the prototypes. We call the resulting algorithm Weighted K-Means (though recognising that other weighted versions of K-Means have been developed in the literature). The partial derivatives are calculated as

$$\frac{\partial J_{1,i}}{\partial \mathbf{m}_r} = -(\mathbf{x}_i - \mathbf{m}_r)\{\| \mathbf{x}_i - \mathbf{m}_r \| + 2\sum_{j=1}^{K} \| \mathbf{x}_i - \mathbf{m}_j \|\} = -(\mathbf{x}_i - \mathbf{m}_r)a_{ir} \tag{8}$$

when $\mathbf{m}_r$ is the closest prototype to $\mathbf{x}_i$ and

$$\frac{\partial J_{1,i}}{\partial \mathbf{m}_k} = -(\mathbf{x}_i - \mathbf{m}_k)\frac{\| \mathbf{x}_i - \mathbf{m}_r \|^2}{\| \mathbf{x}_i - \mathbf{m}_k \|} = -(\mathbf{x}_i - \mathbf{m}_r)b_{ik} \tag{9}$$

otherwise.

We then solve this by summing over the whole data set and finding the fixed point solution of

$$\frac{\partial J_1}{\partial \mathbf{m}_r} = \sum_{i=1}^{N} \frac{\partial J_{1,i}}{\partial \mathbf{m}_r} = 0 \tag{10}$$

which gives a solution of

$$\mathbf{m}_r = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}} \tag{11}$$

We have given extensive analysis and simulations in [1] showing that this algorithm will cluster the data with the prototypes which are closest to the data points being positioned in such a way that the clusters can be identified. However there are some potential prototypes which are not sufficiently responsive to the data and so never move to identify a cluster. In fact, these points move to (a weighted) centre of the data set. This may be an advantage in some cases in that we can easily identify redundancy in the prototypes however it does waste computational resources unnecessarily.

3.1 A New Algorithm

Consider the performance algorithm

$$J_2 = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} \frac{1}{\| \mathbf{x}_i - \mathbf{m}_j \|^p} \right] \min_{k=1}^{K} \| \mathbf{x}_i - \mathbf{m}_k \|^n \tag{12}$$

Let $\mathbf{m}_r$ be the closest centre to $\mathbf{x}_i$. Then

$$\begin{aligned} J_2(\mathbf{x}_i) &= \left[\sum_{j=1}^{K} \frac{1}{\| \mathbf{x}_i - \mathbf{m}_j \|^p} \right] \| \mathbf{x}_i - \mathbf{m}_r \|^n \\ &= \| \mathbf{x}_i - \mathbf{m}_r \|^{n-p} + \sum_{j \neq r} \frac{\| \mathbf{x}_i - \mathbf{m}_r \|^n}{\| \mathbf{x}_i - \mathbf{m}_j \|^p} \end{aligned} \tag{13}$$

Therefore

$$\begin{aligned} \frac{\partial J_2(\mathbf{x}_i)}{\partial \mathbf{m}_r} &= -(n-p)(\mathbf{x}_i - \mathbf{m}_r) \| \mathbf{x}_i - \mathbf{m}_r \|^{n-p-2} \\ &\quad -n(\mathbf{x}_i - \mathbf{m}_r) \| \mathbf{x}_i - \mathbf{m}_r \|^{n-2} \sum_{j \neq r} \frac{1}{\| \mathbf{x}_i - \mathbf{m}_j \|^p} \\ &= (\mathbf{x}_i - \mathbf{m}_r) a_{i,r} \end{aligned} \tag{14}$$

$$\frac{\partial J_2(\mathbf{x}_i)}{\partial \mathbf{m}_j} = p(\mathbf{x}_i - \mathbf{m}_j) \frac{\| \mathbf{x}_i - \mathbf{m}_r \|^n}{\| \mathbf{x}_i - \mathbf{m}_j \|^{p+2}} = (\mathbf{x}_i - \mathbf{m}_j) b_{i,j} \tag{15}$$

At convergence, $E(\frac{\partial J_2}{\partial \mathbf{m}_r}) = 0$ where the expectation is taken over the data set. If we denote by V_j the set of points, $\mathbf{x}$ for which $\mathbf{m}_j$ is the closest, we have

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{m}_r} = 0 \Longleftrightarrow & \int_{\mathbf{x} \in V_r} \{(n-p)(\mathbf{x}_i - \mathbf{m}_r) \| \mathbf{x}_i - \mathbf{m}_r \|^{n-p-2} \\ & + n(\mathbf{x} - \mathbf{m}_r) \| \mathbf{x} - \mathbf{m}_r \|^{n-2} \sum_{j \neq r} \frac{1}{\| \mathbf{x} - \mathbf{m}_j \|^p} P(\mathbf{x}) \} d\mathbf{x} \\ + & \sum_{j \neq r} \int_{\mathbf{x} \in V_j} p(\mathbf{x} - \mathbf{m}_j) \frac{\| \mathbf{x} - \mathbf{m}_r \|^n}{\| \mathbf{x} - \mathbf{m}_j \|^{p+2}} P(\mathbf{x}) d\mathbf{x} = 0 \end{aligned} \tag{16}$$

where $P(\mathbf{x})$ is the probability measure associated with the data set. This is, in general, a very difficult set of equations to solve. However it is readily seen that, for example, in the special case that there are the same number of prototypes as there are data points, that one solution is to locate each prototype at each data point (at which time $\frac{\partial J}{\partial \mathbf{m}_r} = 0$). Again solving this over all the data set results in

$$\mathbf{m}_r = \frac{\sum_{i \in V_r} \mathbf{x}_i a_{ir} + \sum_{i \in V_j, j \neq r} \mathbf{x}_i b_{ir}}{\sum_{i \in V_r} a_{ir} + \sum_{i \in V_j, j \neq r} b_{ir}} \tag{17}$$

From (16), we see that $n \geq p$ if the direction of the first term is to be correct and $n \leq p + 2$ to ensure stability in all parts of that equation. We compare results with $n = p + 1$ and $n = p + 2$ below.

In practice, we have found that a viable algorithm may be found by using (15) for all prototypes (and thus never using (14) for the closest prototype). We will call this the Inverse Weighted K-Means Algorithm.

4 Comparative Study

In order to compare the convergence of these algorithms, we have created a number of data sets with varying degrees of difficulty (see Table 1). The data sets are all two dimensional to make visual identification of the convergence easy. Each data set consists of 4 clusters each of 10 data points drawn from uniform distributions in the four disjoint squares, $\{(x, y) : 0 \leq x < 1, 0 \leq y < 1\}, \{(x, y) : 3 \leq x < 4, 0 \leq y < 1\}, \{(x, y) : 0 \leq x < 1, 3 \leq y < 4\}$, and $\{(x, y) : 3 \leq x < 4, 3 \leq y < 4\}$.

Example 1. We begin with 4 prototypes, initialised in the first square cluster and success is having each of the 4 prototypes identify one of the clusters.

Example 2. Again 4 prototypes but now each prototype is initialised to the same position very distant from the data, the point (100,100).

Example 3, 4. We have 40 prototypes all initialised in the first cluster. Now successful convergence requires each prototype to identify a single data point. This measures the responsiveness of the algorithm to the data.

Example 5. 40 prototypes all initialised to the same position (100,100) far from the data and success is as for Example 3.

Example 6, 7. 40 prototypes randomly initialised throughout $\{(x, y) : 0 \leq x < 4, 0 \leq y < 4\}$ and success is as for Example 3.

We see the results in Table 1. K-Means fails to converge to successful solutions on all these simple data sets. K-Harmonic Means fails in two cases, both of which correspond to the situation in which all prototypes are initialised to the same position far from the data. Weighted K-Means always converges to successful solutions but some simulations take many iterations to converge. Inverse Weighted K-Means also always converges to successful solutions and does so very much faster than Weighted K-Means. In general, it appears that convergence with $n = p + 1$ is faster than when $n = p + 2$, though not in the case of data in Example 5.

Table 1. Number of iterations to convergence on 7 data sets (see text) using the various algorithms. Top: K-Means fails on all 7 data sets, Second Row: K Harmonic Means fails on 2 data sets. Third Row: Weighted K-Means algorithm. Next 5 rows; the Inverse Weighted K-Means with $n = p + 1$. Last 5 rows show results with the the Inverse Weighted K-Means with $n = p + 2$.

	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6	Ex. 7
K-Means	-	-	-	-	-	-	-
KHM	6	-	23	19	-	22	25
WKM	3	7	81	106	142	45	67
IWKM, p=1,n=2	5	7	25	29	100	27	21
IWKM, p=2,n=3	7	6	34	17	88	17	21
IWKM, p=3,n=4	5	8	27	15	83	22	17
IWKM, p=4,n=5	8	8	25	18	92	24	19
IWKM, p=5,n=6	5	6	26	14	86	23	17
IWKM, p=1,n=3	4	6	44	23	93	29	24
IWKM, p=2,n=4	5	9	33	37	88	29	22
IWKM, p=3,n=5	5	7	44	42	81	42	18
IWKM, p=4,n=6	6	7	50	77	85	49	20
IWKM, p=5,n=7	6	13	50	38	88	35	47

Note that the reason which the current algorithm succeeds in these extreme cases when other algorithms fail is because we have available two sets of updates rather than a single update for all prototypes. This is a symmetry-breaking factor which enables local minima to be avoided.

5 A Topology Preserving Mapping

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) [4] and the Generative Topographic Mapping(GTM) [2] 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 [3] 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 [3], we used a weighted mean squared error; with the Harmonic Topographic Mapping [5], we used Harmonic K-Means. We now apply the Inverse Weighted K-Means algorithm to the same underlying structure to create a new topology preserving algorithm.

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 60 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 results from the algorithm are shown in Figure 1. The left diagram shows results with $n = p+1, p = 1$, the right with $n = p+2, p = 1$. We see that, in each case, a topology-preserving mapping has been found.

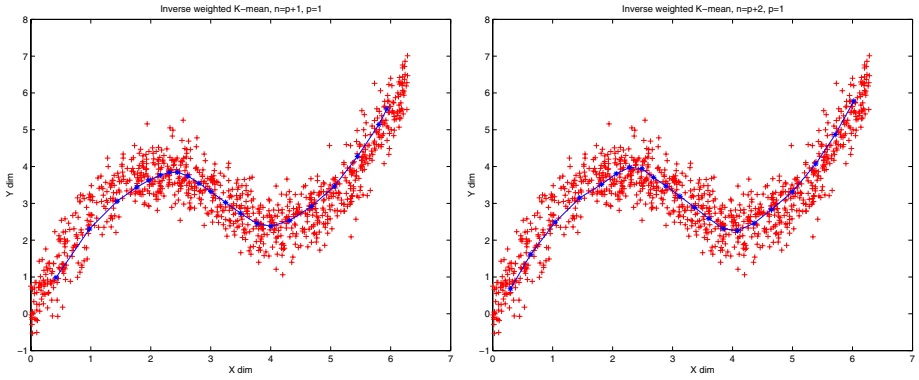

Fig. 1. In both diagrams, the data are shown as red '+'s and the prototypes as blue '*'s. The prototypes have been joined in their natural order. Left: The Inverse Weighted Topolographic Mapping, $n = p + 1, p = 1$. Right: Same, $n = p + 2, p = 1$.

6 Conclusion

We have reviewed K-Means and K-Harmonic Means and their associated performance functions. We have developed new performance functions and derived fixed point algorithms based on their derivatives. We have shown that the Inverse Weighted K-Means is a better algorithm in terms of convergence to the most informative solutions over a variety of artificial data sets. We emphasise again that

the reason which the current algorithm succeeds in extreme cases when other algorithms fail is because we have available two sets of updates rather than a single update for all prototypes. This is a symmetry-breaking factor which enables local minima to be avoided. We have also shown how one of these algorithms can be used as a base for effective topology preserving algorithms as was done with K-Harmonic Means in [3]. Future work will investigate these mappings on real data sets.

References

1. W. Barbakh and C. Fyfe. Performance functions and clustering algorithms. *Computing and Information Systems*, 10(2):2–8, 2006. ISSN 1352-9404.
2. C. M. Bishop, M. Svensen, and C. K. I. Williams. Gtm: The generative topographic mapping. *Neural Computation*, 1997.
3. C. Fyfe. Two topographic maps for data visualization. *Data Mining and Knowledge Discovery*, 2006.
4. Tuevo Kohonen. *Self-Organising Maps*. Springer, 1995.
5. M. Peña and C. Fyfe. Model- and data-driven harmonic topographic maps. *WSEAS Transactions on Computers*, 4(9):1033–1044, 2005.
6. B. Zhang. Generalized k-harmonic means – boosting in unsupervised learning. Technical report, HP Laboratories, Palo Alto, October 2000.
7. B. Zhang, M. Hsu, and U. Dayal. K-harmonic means - a data clustering algorithm. Technical report, HP Laboratories, Palo Alto, October 1999.