Clustering with Reinforcement Learning

Wesam Barbakh and Colin Fyfe

The University of Paisley,
Scotland
wesam.barbakh, colin.fyfe@paisley.ac.uk

Abstract. We show how a previously derived method of using reinforcement learning for supervised clustering of a data set can lead to a sub-optimal solution if the cluster prototypes are initialised to poor positions. We then develop three novel reward functions which show great promise in overcoming poor initialization. We illustrate the results on several data sets. We then use the clustering methods with an underlying latent space which enables us to create topology preserving mappings. We illustrate this method on both real and artificial data sets.

1 Introduction

We have previously investigated [2, 4] clustering methods which are robust with respect to poor initialization. In this paper, we investigate using reinforcement learning with reward functions which are related to our previous clustering methods in that the reward functions also allow the reinforcement learning algorithms to overcome the disadvantages of a poor initialization and achieve the globally optimal clustering.

We begin by reviewing a reinforcement learning algorithm which has previously [9] been used to perform clustering of data sets.

2 The Bernoulli Model

[12, 11] investigated a particular form of reinforcement learning in which reward for an action is immediate which is somewhat different from mainstream reinforcement learning [10, 7]. Williams [11] considered a stochastic learning unit in which the probability of any specific output was a parameterised function of its input, $\mathbf{x}$. For the i^{th} unit, this gives

$$P(y_i = \zeta | \mathbf{w}_i, \mathbf{x}) = f(\mathbf{w}_i, \mathbf{x}) \tag{1}$$

where, for example,

$$f(\mathbf{w}_i, \mathbf{x}) = \frac{1}{1 + \exp(- \parallel \mathbf{w}_i - \mathbf{x} \parallel^2)} \tag{2}$$

Williams [11] considers the learning rule

$$\Delta w_{ij} = \alpha_{ij}(r_{i,\zeta} - b_{ij}) \frac{\partial \ln P(y_i = \zeta | \mathbf{w}_i, \mathbf{x})}{\partial w_{ij}} \tag{3}$$

H. Yin et al. (Eds.): IDEAL 2007, LNCS 4881, pp. 507–516, 2007.

where α_{ij} is the learning rate, $r_{i,\zeta}$ is the reward for the unit outputting ζ and b_{ij} is a reinforcement baseline which in the following we will take as the reinforcement comparison, $b_{ij} = \overline{r} = \frac{1}{K}\sum r_{i,\zeta}$ where K is the number of times this unit has output ζ. ([11], Theorem 1) shows that the above learning rule causes weight changes which maximises the expected reward.

[11] gave the example of a Bernoulli unit in which $P(y_i = 1) = p_i$ and so $P(y_i = 0) = 1 - p_i$. Therefore

$$\frac{\partial \ln P(y_i)}{\partial p_i} = \begin{cases} -\frac{1}{1-p_i} & \text{if } y_i = 0 \\ \frac{1}{p_i} & \text{if } y_i = 1 \end{cases} = \frac{y_i - p_i}{p_i(1-p_i)} \tag{4}$$

[9] applies the Bernoulli model to (unsupervised) clustering with

$$p_i = 2(1 - f(\mathbf{w}_i, \mathbf{x})) = 2\left(1 - \frac{1}{1 + \exp(- \parallel \mathbf{w}_i - \mathbf{x} \parallel^2)}\right) \tag{5}$$

The environment identifies the p_{k*} which is maximum over all output units and y_{k*} is then drawn from this distribution. Rewards are given such that

$$r_i = \begin{cases} 1 & \text{if } i = k* \text{ and } y_i = 1 \\ -1 & \text{if } i = k* \text{ and } y_i = 0 \\ 0 & \text{if } i \neq k* \end{cases} \tag{6}$$

where $k*$ represents the winning node, the node that are most similar to the input sample. This is used in the update rule

$$\Delta w_{ij} = \alpha r_i (y_i - p_i)(x_j - w_{ij}) \tag{7}$$

$$= \alpha |y_i - p_i| (x_j - w_{ij}) \text{ for } i = k* \tag{8}$$

which is shown to perform clustering of the data set.

Implementation

1. Randomly select a sample $\mathbf{x}$ from the data set.
2. For $i = 1, ..., L$ compute the probability p_i
3. Specify the winning unit $k*$ with $p_{k*} = max p_i$, and sample the output y_{k*} from p_{k*}
4. Compute the reinforcement rewards r_{k*} using equation (6)
5. Update the weight vectors w_{k*} using equation (7)
6. Repeat until convergence

2.1 Simulation

We applied the Bernoulli algorithm to the artificial data set shown in Figure 1, left, but the Bernoulli algorithm failed to identify all the clusters successfully as shown in Figure 1, right.

The Bernoulli algorithm is sensitive to the prototypes' initialization which can lead it to finding local optima which often are detectable because of dead prototypes which are not near any data. The main reason for these problems is that we update the winner prototypes only, not all of them.

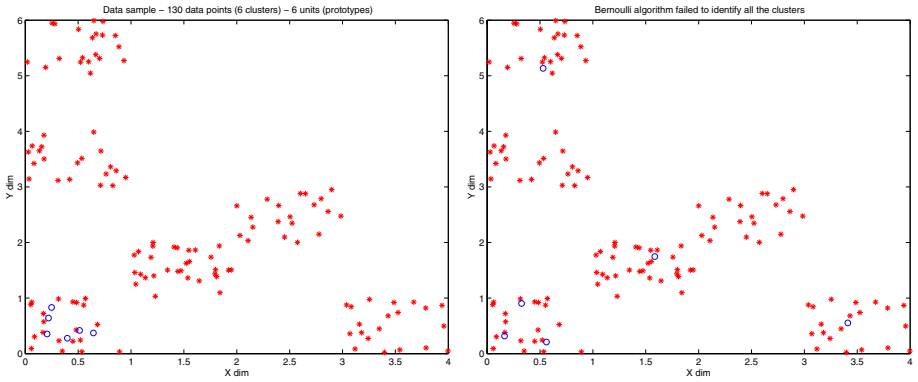

Fig. 1. Left: artificial data set is shown as 6 clusters of red '*'s, and 6 prototypes of blue 'o's. Right: Bernoulli algorithm failed to identify all the clusters successfully.

3 Algorithm RL1

A first new algorithm, RL1, has the following reward function:

$$r_i = \begin{cases} \frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^3}{\|\mathbf{x}-\mathbf{m}_i\|^3} & \text{if } y_i = 1 \\ -\frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^3}{\|\mathbf{x}-\mathbf{m}_i\|^3} & \text{if } y_i = 0 \end{cases} \forall i \tag{9}$$

where

$$k* = \arg\min_{k=1}^{K}(\| \mathbf{x} - \mathbf{m}_k \|)$$

This new reward function has the following features:

1. We apply this equation to all prototypes, not only the winners and thus all prototypes can find the clusters even if they are initialized badly.
2. This reward function allows the prototypes to respond differently to each other, and each prototype before moving to any new location responds to all the other prototypes' position, and hence it is possible for it to identify the free clusters that are not recognized by the other prototypes.
3. This reward function gives the highest value, 1, for highest similarity between the data point and the node (prototype).

3.1 Simulation

Figure 2 shows the result after applying RL1 algorithm to the artificial data set, but with very poor prototypes' initialization.

Figure 2, left shows the prototypes after many iterations but before convergence; in this Figure we can see one prototype still far from the data points while others have spread into data; this distant prototype still has the ability to learn even if it is very far from data, and this is an advantage for this algorithm over the previous algorithms.

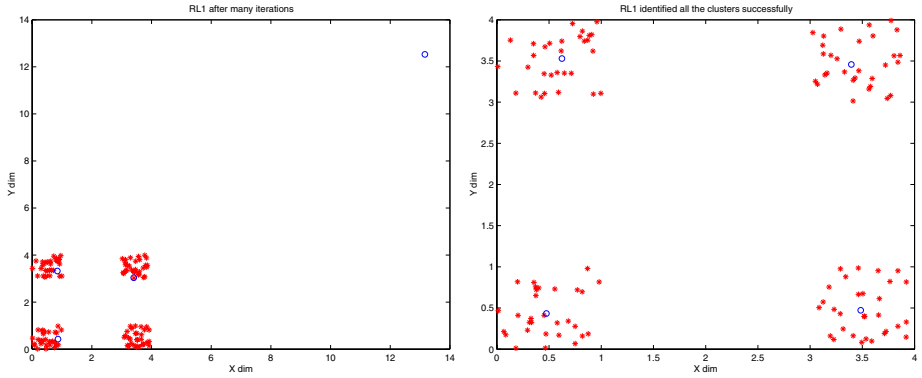

Fig. 2. Left: RL1 result after many iterations but before convergence. Right: RL1 result after convergence.

4 New Algorithm RL2

A second new algorithm, RL2, has the following reward function:

$$r_i = \begin{cases} 1 & \text{if } i = k^* \text{ and } y_i = 1 \\ \frac{1-\exp(-\beta\|\mathbf{x}-\mathbf{m}_{k*}\|^3)}{\|\mathbf{x}-\mathbf{m}_i\|^3} & \text{if } i \neq k^* \text{ and } y_i = 1 \\ -1 & \text{if } i = k^* \text{ and } y_i = 0 \\ \frac{\exp(-\beta\|\mathbf{x}-\mathbf{m}_{k*}\|^3)-1}{\|\mathbf{x}-\mathbf{m}_i\|^3} & \text{if } i \neq k^* \text{ and } y_i = 0 \end{cases} \tag{10}$$

where again $k* = \arg\min_j \parallel \mathbf{x} - \mathbf{m}_j \parallel$.

The reward function (10) has values ranged between 0 and 1. We update the closest prototype (or most similar one) by giving it directly a maximum possible reward value, 1, to allow it to learn more than others and also to a void any division by zero which may happen using the second equation in (10). The second equation in (10) is used for all the other prototypes. Prototypes closer (or more similar) to the input data sample will learn more than others by taking higher reward value, and so on for all prototypes.

5 Algorithm RL3

A third new algorithm, RL3, has the following reward function:

$$r_i = \begin{cases} \frac{1}{\|\mathbf{x}-\mathbf{m}_i\|^4\{\sum_{l=1}^{K}\frac{1}{\|\mathbf{x}-\mathbf{m}_l\|^2}\}^2} & \text{if } y_i = 1 \\ \frac{-1}{\|\mathbf{x}-\mathbf{m}_i\|^4\{\sum_{l=1}^{K}\frac{1}{\|\mathbf{x}-\mathbf{m}_l\|^2}\}^2} & \text{if } y_i = 0 \end{cases} \tag{11}$$

The reward function in (11) has similar principles like the previous new reward functions. It has values ranged between 0 and 1. All the prototypes can learn in an effective way. The prototype that is more similar to the input data sample takes higher reward value. In implementation, to avoid any division by zero we can rewrite (11) as follows:

$$r_i = \begin{cases} \dfrac{\frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^4}{\|\mathbf{x}-\mathbf{m}_i\|^4}}{\left\{1+\sum_{l\neq k*}^{K} \frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^2}{\|\mathbf{x}-\mathbf{m}_l\|^2}\right\}^2} \text{ if } y_i = 1 \\ \dfrac{-\frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^4}{\|\mathbf{x}-\mathbf{m}_i\|^4}}{\left\{1+\sum_{l\neq k*}^{K} \frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^2}{\|\mathbf{x}-\mathbf{m}_l\|^2}\right\}^2} \text{ if } y_i = 0 \end{cases} \tag{12}$$

where $k* = \arg\min_j \| \mathbf{x} - \mathbf{m}_j \|$. Note that $\frac{\|\mathbf{x}-\mathbf{m}_{k*}\|^4}{\|\mathbf{x}-\mathbf{m}_{k*}\|^4}$ is always set to 1.

5.1 Simulation

Figure 3 shows the results after applying Bernoulli algorithm, top right, RL2, bottom left, and RL3, bottom right, to the artificial data set shown in Figure 3, top left. RL2 and RL3 succeeded to identify the clusters successfully while Bernoulli model failed.

6 A Topology Preserving Mapping

In this section, we show how we can extend RL1 and RL2 to provide new algorithms for visualisation and topology-preserving mappings.

6.1 RL1 Topology-Preserving Mapping (RL1ToM)

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) [8] and the Generative Topographic Mapping(GTM) [5] 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 [6] 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

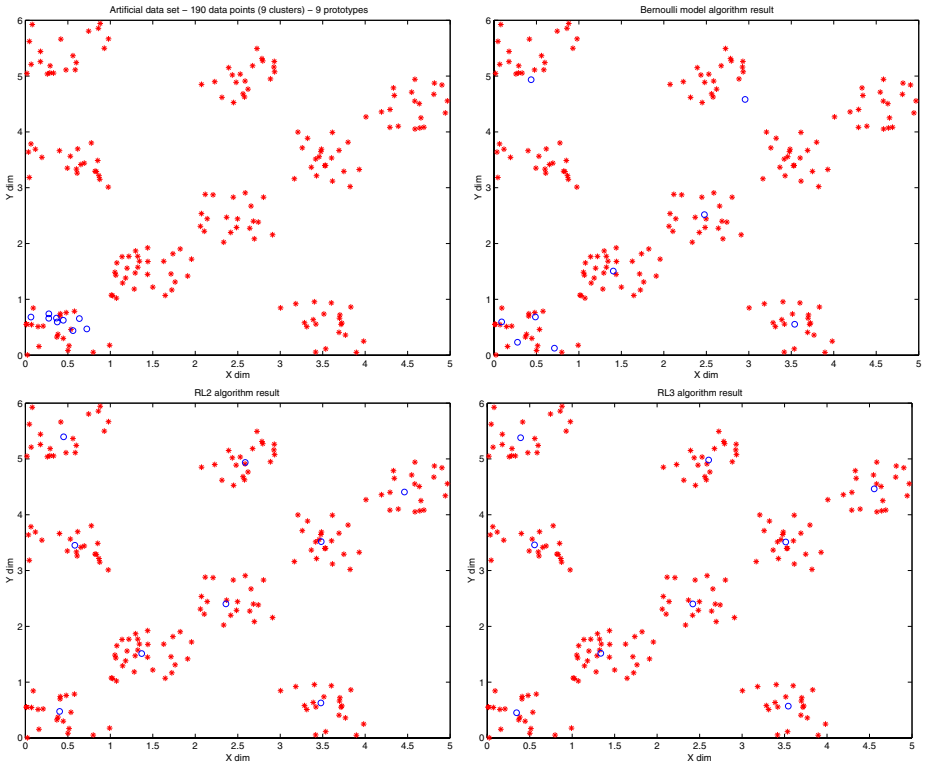

Fig. 3. Top left: Artificial data set with poor prototypes' initialization. Top right: Bernoulli algorithm result. Bottom left: RL2 algorithm result. Bottom right: RL3 algorithm result.

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 [6], 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 RL1 algorithm to the same underlying structure to create a new topology preserving algorithm.

Each data point is visualized as residing at the prototype on the map which would win the competition for that data point. However we can do rather better by defining the responsibility that the j^{th} prototype has for the i^{th} data point as

$$r_{ji} = \frac{\exp(-\gamma \parallel \mathbf{x}_i - \mathbf{w}_j \parallel^2)}{\sum_k \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{w}_k \parallel^2)} \tag{13}$$

We then project points taking into account these responsibilities: let y_{ij} be the projection of the i^{th} data point onto the j^{th} dimension of the latent space; then

$$y_{ij} = \sum_k t_{kj} r_{ki} \tag{14}$$

where t_{kj} is the j^{th} coordinate of the k^{th} latent point.

6.2 RL2 Topology-Preserving Mapping (RL2ToM)

RL2ToM algorithm like RL1ToM has the same structure as the GTM, with a number of latent points that are mapped to a feature space by M Gaussian functions, and then into the data space by a matrix W. Each latent point t indexed by k is mapped, through a set of M fixed basis functions $\phi_1()$, $\phi_2()$,...,$\phi_M()$ to a prototype in data space $m_k = W\phi(t_k)$. But the similarity ends there because the objective function is not a probabilistic function like the GTM neither it is optimised with the Expectation-Maximization (EM) algorithm. Instead, the RL2ToM uses the well proved clustering abilities of the K-means algorithm, improved by using RL2 to make it insensitive to initialisation.

6.3 Simulation

6.3.1 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

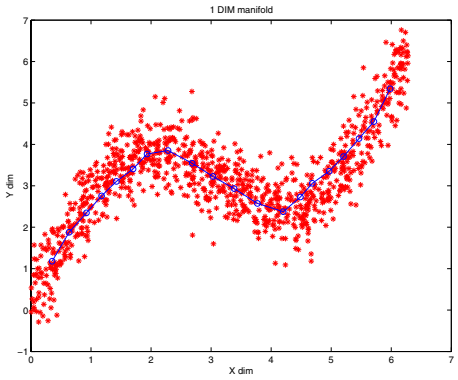

Fig. 4. The resulting prototypes' positions after applying RL1ToM. Prototypes are shown as blue 'o's.

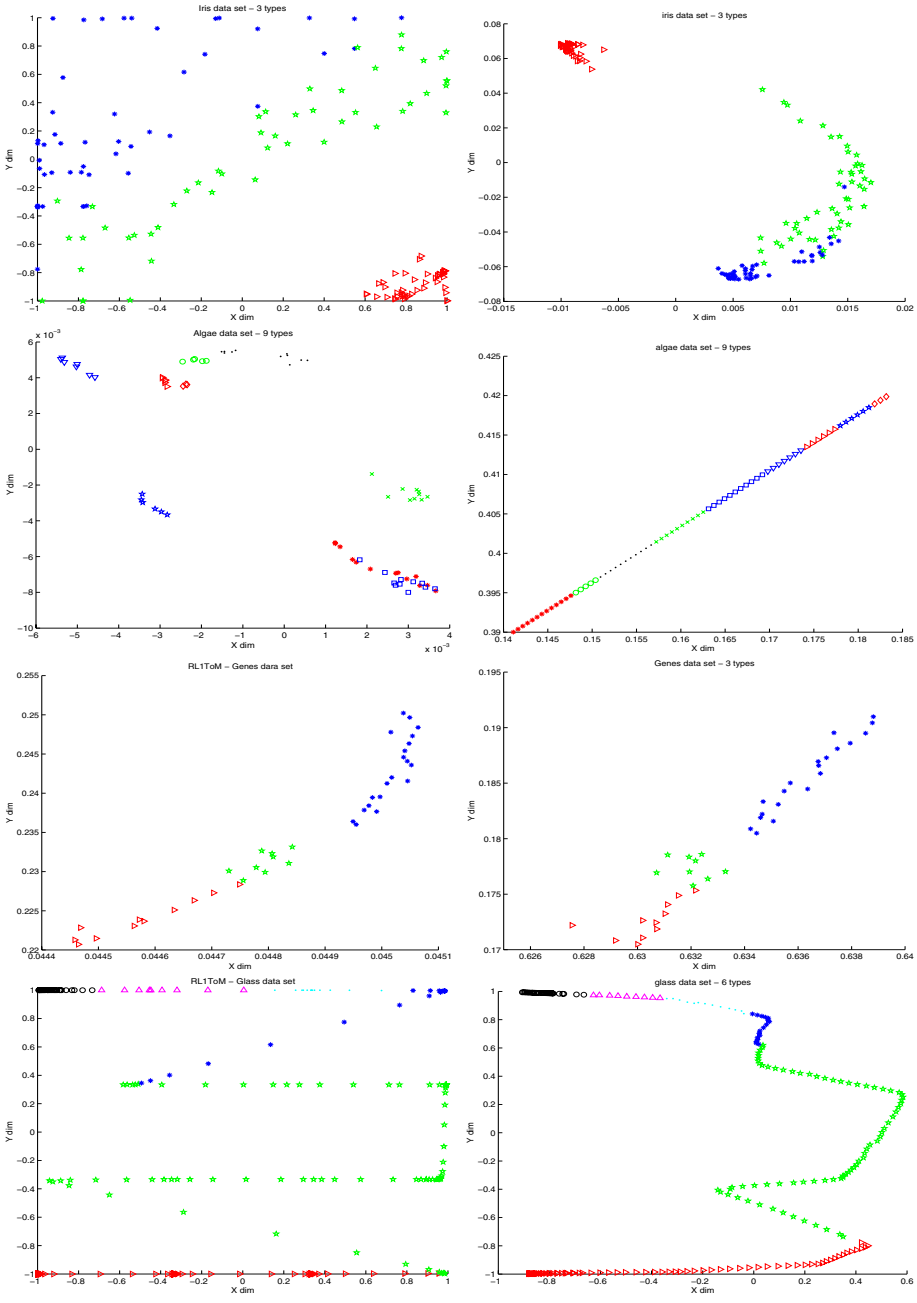

Fig. 5. The first column shows the results of using RL1ToM; the second column shows the results of using RL2ToM. The top line is the projection of the iris data set; the second line shows the algae data set; the third line shows the genes data set; the bottom line shows the glass data set.

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 RL1ToM is shown in Figure 4.

6.3.2 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 5, left and right, the projections of the real data sets onto a two dimensional grid of latent points using RL1ToM and RL2ToM, respectively. The results are comparable with others we have with these data sets from a variety of different algorithms.

7 Conclusion

We have shown how reinforcement learning of cluster prototypes can be performed robustly by altering the reward function associated with finding the clusters. We have illustrated three different reward functions which clearly have a family resemblance. Most importantly all three overcome the disadvantages of poor initialization in that they do not succumb to local minima as the existing Bernoulli algorithm does.

We have also illustrated how a topology preserving mapping can be created by using these algorithms with an underlying fixed latent space. Future work will compare these methods with our existing methods of creating robust optimal clusters.

References

[1] Barbakh, W.: The family of inverse exponential k-means algorithms. Computing and Information Systems 11(1), 1–10 (2007)
[2] Barbakh, W., Crowe, M., Fyfe, C.: A family of novel clustering algorithms. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 283–290. Springer, Heidelberg (2006)
[3] Barbakh, W., Fyfe, C.: Performance functions and clustering algorithms. Computing and Information Systems 10(2), 2–8 (2006)
[4] Barbakh, W., Fyfe, C.: Tailoring local and global interactions in clustering algorithms. Technical Report 40, School of Computing, University of Paisley (March 2007), ISSN 1461-6122
[5] Bishop, C.M., Svensen, M., Williams, C.K.I.: Gtm: The generative topographic mapping. Neural Computation (1997)
[6] Fyfe, C.: Two topographic maps for data visualization. Data Mining and Knowledge Discovery 14, 207–224 (2007)

[7] Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
[8] Kohonen, T.: Self-Organising Maps. Springer, Heidelberg (1995)
[9] Likas, A.: A reinforcement learning approach to on-line clustering. Neural Computation (2000)
[10] Sutton, R.S., Barto, A.G.: Reinforcement Learning: an Introduction. MIT Press, Cambridge (1998)
[11] Williams, R.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8, 229–256 (1992)
[12] Williams, R.J., Pong, J.: Function optimization using connectionist reinforcement learning networks. Connection Science 3, 241–268 (1991)