Mach Learn (2011) 82: 71–90
DOI 10.1007/s10994-009-5156-0

Foraging theory for dimensionality reduction of clustered data

Luis Felipe Giraldo · Fernando Lozano · Nicanor Quijano

Received: 30 June 2008 / Revised: 12 June 2009 / Accepted: 10 October 2009 /
Published online: 13 November 2009

Abstract We present a bioinspired algorithm which performs dimensionality reduction on datasets for visual exploration, under the assumption that they have a clustered structure. We formulate a decision-making strategy based on foraging theory, where a software agent is viewed as an animal, a discrete space as the foraging landscape, and objects representing points from the dataset as nutrients or prey items. We apply this algorithm to artificial and real databases, and show how a multi-agent system addresses the problem of mapping high-dimensional data into a two-dimensional space.

Keywords Dimensionality reduction · Visual data exploration · Foraging theory · Prey model

1 Introduction

In many situations, it is useful to visualize high-dimensional data. For example, gene-environment interaction analysis (Chanda et al. 2007), automated detection of pathologies in speech (Carreira-Perpiñán and Renals 1998), and the analysis of neuronal codes in visual perception (McClurkin et al. 1991), involve the manipulation and subjective interpretation of multivariate information. In order to make possible the visual exploration of high-dimensional data, dimensionality reduction techniques deal with the transformation of high dimensional space into a two- or three-dimensional map, while retaining most of its meaningful structure (Jain et al. 2000). Examples of well-known algorithms to perform

Editors: D. Martens, B. Baesens, and T. Fawcett.

L.F. Giraldo (✉) · F. Lozano · N. Quijano
Department of Electrical and Electronic Engineering, Universidad de los Andes, Bogotá, Colombia
e-mail: lf.giraldo404@uniandes.edu.co

F. Lozano
e-mail: flozano@uniandes.edu.co

N. Quijano
e-mail: nquijano@uniandes.edu.co

this task include principal component analysis (PCA) (Pearson 1901), classical multidimensional analysis (MDS) (Torgerson 1958), self-organizing maps (SOM) (Kohonen 2000; Bishop et al. 1998), isometric feature mapping (ISOMAP) (Tenenbaum et al. 2000), locally linear embedding (LLE) (Roweis and Saul 2000), and t-distributed stochastic neighbor embedding (t-SNE) (Hinton and Roweis 2002). Additionally, a variety of methods for dimensionality reduction on data with a clustered structure have been proposed. For example, in Sanguinetti (2008) a latent variable model for linear dimensionality reduction is formulated, where the number of clusters contained in the dataset is known a priori. The Lumer-Faieta (LF) algorithm (Lumer and Faieta 1994; Bonabeau et al. 1999) represents the high-dimensional data into a two-dimensional discrete space, using a nature-inspired model that accounts for the phenomena of sorting and clustering in ants. Other methods can be found in Lisboa et al. (2008), Wen et al. (2006). Most of these methods try to preserve the clustered structure after dimensionality reduction, but they need prior knowledge about the number of clusters, or have a tendency to create spurious clusters.

In this work, we use *foraging theory* (Stephens and Krebs 1986), a branch of behavioral ecology, to address the problem of reducing high-dimensional data into a two-dimensional space. Foraging theory studies the behavior of animals in response to the environment in which the animals live. It states that animals forage in order to maximize their energy intake per unit time, obtaining the nutrient sources to survive and additional time for other important activities such as fighting, reproducing, or shelter-building (Passino 2005). Foraging concepts can be applied for both individual animals (Stephens and Krebs 1986) and social organisms (Giraldeau and Caraco 2000), and they have been successfully applied in several engineering applications. For example, individual foraging concepts are applied to develop a controller for a multizone temperature control problem in Quijano et al. (2006) and to fit an autonomous vehicle control problem in Andrews et al. (2007b). Additionally, social foraging ideas are introduced to create an algorithm that solves a temperature control problem as an optimal resource allocation problem in Quijano and Passino (2007), and to design a multiagent system applied to an autonomous vehicle cooperative search problem in Andrews et al. (2007a). Another interesting application is *information foraging theory* (Pirolli 2007), where an analogy between information searching patterns in humans and animal foraging strategies is elucidated. Here, we propose an algorithm where animals are equivalent to software agents, and the decision strategy is based on the *prey model* (Stephens and Krebs 1986). In our algorithm's decision-making strategy, the agents move around a grid and modify the position of objects which are on it, with the purpose of maximizing the average energy gain of the agents. This average energy gain is directly related to the pairwise distances between datapoints. As a result of this procedure, a two-dimensional discrete representation of the high-dimensional data is provided, revealing the existence of natural groupings or clusters.

The main contribution of this work is to make the connection between some concepts from foraging theory to the field of machine learning. Foraging models the animal's behavior when it makes choices to maximize its long-term rate of energy intake as a sequential optimization procedure (Passino 2005). On the other hand, certain problems found in machine learning involve solving an optimization problem in a similar fashion. For example, in reinforcement learning, an agent attempts to learn a desired behavior by interacting with the environment around him. The learning process is sequential: the agent acts on the environment, which in turn produces a reward signal. The goal for the agent is to learn actions that maximize the sum of rewards in the long run (Sutton and Barto 1998). We believe that there is a natural analogy between rewards and the rate of energy intake that can be exploited to formulate new bioinspired reinforcement learning algorithms. Another related subfield is online learning (see for example Kivinen et al. 2004; Kivinen and Warmuth 1997;

Littlestone and Warmuth 1994), which proceeds in a sequence of trials. In its simplest manifestation, at each trial the learner observes an input provided by the environment, formulates a prediction and then the correct answer is given by the environment. Based on an error signal related to the difference between the prediction and the correct answer, the learner modifies its model with the goal of making as few mistakes as possible over time. We think we can relate the learner to the foraging animal, and utilize the prey model to devise known learning strategies. Recently, several bioinspired algorithms have been developed to deal with machine learning problems such as supervised learning (Martens et al. 2007), reinforcement learning in robots (Ulam and Balch 2003) and data clustering (Ngenkaew et al. 2008), but as far as we know, this work is the first attempt to apply concepts from the foraging prey model to the machine learning field.

We perform experiments on synthetic datasets in order to show the performance of the proposed algorithm. The results of our experiments show that after a certain number of iterations, our algorithm converges to a distribution of objects that remains unchanged, where it typically corresponds to a meaningful representation of the original data in a two-dimensional discrete space. Comparative experiments on real datasets are carried out with the new algorithm and some general methods for dimensionality reduction. Experimental results show the advantages in the performance of the proposed algorithm over the competing algorithms when the principal task is to reveal the presence of clusters in the data.

The remainder of this document is organized as follows. Section 2 reviews the foraging model used in this work. Section 3 describes our algorithm, and the proposed analogies between foraging models and the reduction strategy. In Section 4 we present experimental results, followed by a discussion in Section 5. Finally, some conclusions and future work are considered in Section 6.

2 Foraging prey model

From a biological perspective, a natural environment is comprised of a forager and food items or preys. The forager must make decisions about how to interact with the environment in order to maximize its energy intake from food items, per unit time spent foraging. Mathematical models that deal with this decision-making problem are provided by foraging theory (Stephens and Krebs 1986). In this work we will focus on the *prey model*. The assumption in the prey model is that the forager obtains energy by means of many repetitions of the following sequence: search, encounter, and decide either to attack or to continue searching. Next, we provide an overview of the prey model following a treatment similar to Quijano et al. (2006), Andrews et al. (2007b).

Let there be n different types of prey items in the environment, described by: e_i, the expected time required to process an item of type i; v_i, the expected energy gain from processing an individual prey item of type i; λ_i, the rate at which the forager encounters items of type i when searching; and p_i, the probability of attacking items of type i if it is found and recognized. Encounters with items of type i are assumed to be sequential and to follow a Poisson process. The average rate of energy gain J for the forager is the expected energy obtained divided by the expected total amount of time spent processing preys. If a forager spends on average T_s time units searching, then we have (Quijano et al. 2006; Andrews et al. 2007b):

$$J = \frac{\sum_{i=1}^{n} p_i \lambda_i T_s v_i}{T_s + \sum_{i=1}^{n} p_i \lambda_i T_s e_i} = \frac{\sum_{i=1}^{n} p_i \lambda_i v_i}{1 + \sum_{i=1}^{n} p_i \lambda_i e_i}.$$

The probability of processing each prey type is the decision variable for the forager. Thus, the goal of the forager is to choose the p_i that maximizes J. We rewrite J as

$$J = \frac{p_i \lambda_i e_i + k_i}{c_i + p_i \lambda_i e_i}$$

where k_i is the summation of all terms in the numerator not involving prey type i and c_i is a similar variable for the denominator. Differentiation with respect to p_i gives:

$$\begin{aligned} \frac{\partial J}{\partial p_i} &= \frac{\lambda_i v_i (c_i + p_i \lambda_i e_i) - \lambda_i e_i (p_i \lambda_i v_i + k_i)}{(c_i + p_i \lambda_i e_i)^2} \\ &= \frac{\lambda_i v_i c_i - \lambda_i e_i k_i}{(c_i + p_i \lambda_i e_i)^2}. \end{aligned} \tag{1}$$

Note that if the numerator in (1) is negative, then J is maximized by choosing the lowest possible p_i. Therefore, because $0 \le p_i \le 1$, the p_i that maximizes J is either $p_i = 1$ or $p_i = 0$ for each i, depending on the sign of $v_i c_i - e_i k_i$. This concept is known as the *zero-one rule*: to maximize its rate of energy gain, a forager must either process a prey of type i every time it encounters it or never process it. Then, the question is, which items the forager should process and which items it should ignore. The answer must account for missed opportunity. If the rate of energy gain that results from processing prey type i is larger than that of searching for and processing preys of other types, then the forager should process the prey of type i. On the other hand, if the forager would gain more energy by searching for other items and processing those, then the item of type i should not be processed. Summarizing, if $\pi_i = v_i / e_i$, then the strategy that maximizes average rate of energy intake is:

$$p_i = \begin{cases} 0, & \text{if } \pi_i < k_i / c_i \\ 1, & \text{if } \pi_i > k_i / c_i \end{cases}$$

where π_i is the *profitability* of prey type i, which is defined as the rate gain that results from processing prey type i, and the alternative rate of gain resulting from searching for and processing other prey types is k_i / c_i. Algorithm 1 (adapted from Andrews et al. 2007b; Pirolli 2007) describes the prey model algorithm in light of the above discussion.

Observe that the left hand side in (2) is the average rate of energy gain obtained by the diet of the j highest profitability prey types. The right hand side of the inequality is the profitability of the prey type $j+1$. The highest j that satisfies (2) is the least profitable prey type in the diet. In other words, if prey types in the environment are ranked according to profitability with $i = 1$ being the most profitable, and if type $j+1$ is the most profitable

Algorithm 1 Prey selection algorithm

1: Rank the types in the environment according to their profitability π_i. Without loss of generality, let the index i be ordered such that $\pi_1 > \pi_2 > \cdots > \pi_n$.
2: Include prey types in the diet, starting with the most profitable type until the average rate of energy gain for a diet of the top j prey types is greater than the profitability of the prey type $j+1$,

$$\frac{\sum_{i=1}^{j} \lambda_i v_i}{1 + \sum_{i=1}^{j} \lambda_i e_i} > \pi_{j+1} \tag{2}$$

type such that the forager will benefit more from searching for and processing types with profitability higher than that of $j+1$, then items of types 1 through j should be processed when encountered and all other items should not. If (2) does not hold for any j, then all prey types should be processed when encountered. The exclusion of type $j+1$ does not depend on the rate of encounter with type $j+1$. This exclusion implies that if the expected missed opportunity gains exceed the immediate gain of processing a particular type, then it does not benefit the forager to process the type, no matter how often the forager encounters it. Equivalently, if a type's rate of encounter exceeds a critical threshold, then less profitable types should be ignored regardless of how common they are in the environment.

3 Prey model for dimensionality reduction

We have a data set with m samples $\mathcal{X} = \{\mathbf{x}_k\}_{k=1}^m$ with $\mathbf{x}_i \in \mathbb{R}^p$, and $D(\mathbf{x}_i, \mathbf{x}_j)$, a dissimilarity measure between $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, with $D \in [0, 1]$. Let us consider a 2-dimensional grid $\mathcal{G} \subset \mathbb{Z}^2$, where $2 \leqslant p$, and $\mathbb{Z}$ denotes the set of all integers. Our goal is to map the original p-dimensional space of features into the 2-dimensional discrete space, preserving possible clusters present in the original space, with closeness given by D. This task is carried out by a set of agents that move around freely on the grid and are able to pick up an object from its current location and drop it at a different empty location. The general idea is that objects with a dissimilar neighborhood should be picked up and dropped at some other location where more similar objects are present.

The algorithm works as follows. At the initialization phase, m objects are randomly scattered on the grid. These objects are associated with the points $\mathbf{x}_k \in \mathcal{X}$. We denote the objects by o_k and denote the sites in the grid where they are placed by $\mathbf{u}_k \in \mathcal{G}$. Subsequently, each agent selects a random site $\mathbf{u}_k$ occupied by an object o_k, and decides to either pick up the object o_k, or move to another occupied site. Then, each agent that has picked up an object selects a random empty site $\mathbf{u}_e \in \mathcal{G}$, and decides to either drop the object o_k, or move to another empty site. The decisions made by the agents rely on the prey model, and are based principally on the average similarity function f. This function is defined as follows:

$$f(o_i, S) = \frac{1}{n_o} \sum_{j=1}^{n_o} w(\mathbf{u}_i, \mathbf{u}_j)[1 - D(o_i, o_j)] \tag{3}$$

where $S = \{o_j\}_{j=1}^{n_o}$ is a squared neighborhood centered at $\mathbf{u}_i$, which grows until it encompasses n_o neighboring objects, $\mathbf{u}_j$ is the position where o_j resides, and $w(\mathbf{u}_i, \mathbf{u}_j)$ is a weighting function that decreases for objects that are more distant from $\mathbf{u}_i$ in the grid. For example we can choose:

$$w(\mathbf{u}_i, \mathbf{u}_j) = e^{-\gamma(\max|\{u_i^{(k)} - u_j^{(k)}\}_{k=1}^q| - 1)} \tag{4}$$

where $\gamma > 0$, and $u_i^{(k)}$ and $u_j^{(k)}$ denote the k-th entry of vectors $\mathbf{u}_i$ and $\mathbf{u}_j$, respectively.

In order to apply the prey model to make the agent's decision, we view an agent as a forager (animal), and relate its energy gain from an individual prey item to the value of the average similarity function f. An agent in the picking up mode will gain more energy if it picks up an object with lower average similarity. On the other hand, an agent in the dropping mode will gain more energy if it drops an object with higher average similarity. According to the prey selection algorithm (see Algorithm 1), we would like to have lower values of i (prey type) to indicate a higher energy gain. Hence, larger values of f correspond to larger

values of i when the agent is picking up, and larger values of f correspond to lower values of i when the agent is dropping. We define type distributions in the following way:

$$i^{(pick)}(f) = \text{round}((n-1)f + 1) \tag{5}$$

$$i^{(drop)}(f) = \text{round}\,(-(n-1)\,f + n) \tag{6}$$

where $i^{(pick)}$ is the prey type when the agent is searching for an object to pick up, $i^{(drop)}$ when the agent is searching for an object to drop, n is the number of prey types (given as a parameter of the algorithm), and "round" is the standard rounding function for converting to an integer. These functions are chosen so that they establish an equivalence between f and one of the n prey types, taking into account that $f \in [0, 1]$ ((3) and (4)). We choose the energy gain according to the function

$$v_i = n + 1 - i. \tag{7}$$

The encounter rate λ_i corresponds to the occurrence rate of prey type i. Here, we calculate λ_i on the whole object distribution at the time when the agent is making a decision. With this purpose, we define the set of average similarities for all occupied sites, denoted by

$$F_{pick} = \{f\,(o_k, S_k)\}_{k:\mathbf{u}_k \text{ is occupied}}. \tag{8}$$

Similarly, the set of average similarities between an object o_i carried by an agent and objects surrounding each empty site of the grid is denoted by

$$F_{drop} = \{f\,(o_i, S_k)\}_{k:\mathbf{u}_k \text{ is empty}}. \tag{9}$$

Then, encounter rate is calculated using the following expressions:

$$\lambda_i^{(pick)} = \frac{1}{|F_{pick}|} \sum_{f \in F_{pick}} I_{\{i^{(pick)}(f)=i\}} \tag{10}$$

$$\lambda_i^{(drop)} = \frac{1}{|F_{drop}|} \sum_{f \in F_{drop}} I_{\{i^{(drop)}(f)=i\}} \tag{11}$$

where I_A is the indicator function of predicate A, i.e., $I_A = 1$ if A is true, and $I_A = 0$, otherwise. Processing times for each prey type e_i are assumed to be equal to 1. Once the parameters are calculated, each agent makes its decision using Algorithm 1, and this process is repeated. A high-level description of the prey model-based algorithm is shown in Algorithm 2. In this conceptual algorithm, the function `preymodel` receives a set of average similarities and the average similarity where the agent is positioned, and returns the decision to do or not the action of picking up or dropping an object made using (2).

4 Experiments and results

In this section, we evaluate the performance of the algorithm on synthetic and real datasets. First, we generate artificial datasets for which we c an perform a visual evaluation of how well the algorithm preserves the clusters present in the data. Next, we use real world datasets to compare the performance of our algorithm with the performance of two commonly used

Algorithm 2 Prey model-based algorithm

```
//Initialization
Define t_max, p-dimensional grid
for Every item o_i do
   place o_i randomly on grid
   assign attribute x_i to the object o_i
end for

//Main loop
for t = 1 to t_max do
   for all agents do
      if (agent unladen) then
         place agent randomly at a site u_i occupied by some o_i
         Compute F_pick
         d = preymodel(F_pick, f(o_i, S_i)) //Decision
         if d = true then
            Pick up item o_i
         end if
      else if (agent carrying item o_i) then
         place agent randomly at an empty site u_e
         Compute F_drop
         d = preymodel(F_drop, f(o_i, S_e)) //Decision
         if d = true then
            Drop item o_i at site u_e
         end if
      end if
   end for
end for
```

algorithms (ISOMAP (Tenenbaum et al. 2000) and t-SNE (Van der Maaten and Hinton 2008)), as well as a bioinspired algorithm called Lumer-Faieta (LF) (Lumer and Faieta 1994). t-SNE is a variation of the technique SNE (Hinton and Roweis 2002), that performs a dimensionality reduction for data visualization. It is a probabilistic approach that represents the similarities between datapoints in the low-dimensional space as conditional probabilities computed using a Student-t distribution. In the case of ISOMAP, it is assumed that data points are samples from a Riemannian manifold. The basic idea of ISOMAP is to find a mapping that best preserves the geodesic distances between any two points on the manifold, defining the geodesic distance as the shortest curve between two points measured over the manifold. The LF algorithm is a nature-inspired heuristic for dimensionality reduction. It was first introduced in Deneubourg et al. (1990), where a clustering algorithm was conceived by observing the clustering of corpses and larval-sorting activities in real ant colonies. In our experiments we implement the improved version of LF in Lumer and Faieta (1994), which allows the handling of numerical data. The LF algorithm, and the prey model-based algorithm use the normalized Euclidean distance as the dissimilarity function, i.e.,

$$D(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}}{C_t}$$

where C_t is a constant that corresponds to the maximum value of

$$\sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T (\mathbf{x}_i - \mathbf{x}_j)}$$

for all points in the dataset, ensuring that the property $D \in [0, 1]$ mentioned before, is satisfied. In the case of the prey model-based algorithm, 3 agents take part in the process and $n = 100$ prey types are assumed. In order to ensure a tradeoff between the space of free and occupied sites, we use an $M \times M$ grid, where M is given by

$$M = \text{ceil}\left(\sqrt{10m}\right)$$

where m is the number of samples, and "ceil" is the standard ceiling function for converting to an integer. This rule is obtained experimentally and previously recommended by Handl et al. (2006), where an extension of the LF algorithm is proposed. This extension has some similarities with the prey model-based algorithm. The maximum number of iterations and number of neighboring objects present in the average similarity function, depend on the dataset.

4.1 Synthetic datasets

Here, we generate clustered datasets with a variety of structures and observe the ability of the prey model-based algorithm to handle them. We generate databases with normally distributed clusters with different covariance matrices and means. In order to quantify the performance of the algorithm on each database, we consider some measures about the quality of the resulting mapping. For each database, we indicate the number of points per cluster (no/c), and perform several runs of the algorithm in order to calculate the following two statistics: the percentage of correct number of resulting clusters ($\%cc$), which is a measure that indicates if the resultant data distribution has the same number of clusters as the original one; and the average percentage of incorrectly located objects ($\overline{\%ilo}$), which gives us information about objects that are located at incorrect clusters in the distribution provided by the algorithm. For this last measure, we use a clustering technique known as the complete-link algorithm (King 1967), to partition and label each two-dimensional datapoint. We selected this algorithm because it produces a partition that is independent of initial conditions, and it performs well in these types of scenarios (Jain et al. 1999). This partition is compared to the original one (which is known in advance), and the percentage of incorrectly located objects is computed.

Table 1 Experimental results on datasets with different number of clusters. Each algorithm's run includes 30000 iterations, where 9 neighboring objects are used. All clusters have a standard deviation $\sigma = 1$ and the distance between clusters lies within the interval [10, 30]. The column no/c indicates the number of points per cluster in the dataset

Number of clusters	$\overline{\%cc}$	$\overline{\%ilo}$	no/c
2	100	0	50
3	100	0	33
4	100	0	25
5	100	0	20

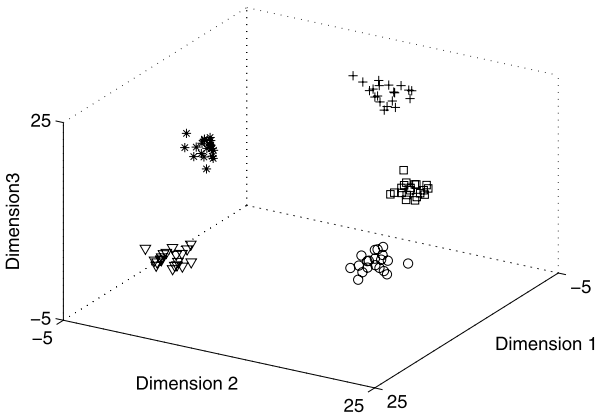

Fig. 1 Dataset with 5 clusters normally distributed. Each cluster is marked with a different symbol

The first group of experiments is performed on four datasets comprising three-dimensional points, with different number of clusters. In order to ensure the formation of separated clusters, for all datasets the minimum distance between means is 10 and the maximum is 30, and each cluster has an isotropic distribution with standard deviation $\sigma = 1$. Table 1 shows the results for these datasets after 100 runs of the algorithm. As an example, Figs. 1 and 2 show the original data distribution of the 5-clustered dataset, and the results of the algorithm at different stages of its execution.

The second group of experiments is performed on 6 datasets with *two*-dimensional points, containing two clusters, each with different covariance matrices, and distance between means D_μ. These datasets are more difficult to handle than the previous ones because they have clusters with different eccentricities and orientations. Table 2 shows the results after 100 runs of the algorithm. Figures 3 and 4 show the resulting clusters for typical runs of the algorithm. An example of the algorithm's behavior when it is dealing with this kind of datasets is illustrated in Figs. 5 and 6.

4.2 Real datasets

We consider three well-known real-world datasets to perform a comparison between some standard algorithms for dimensionality reduction and the prey model algorithm. Unlike synthetic datasets, we cannot give a visual assessment of their structure before mapping, because of their high dimensionality. However, we have prior knowledge about the properties related to the structure in those high dimensions, such as their partition into classes. In the t-SNE and LF algorithms, we set the parameter values as in Van der Maaten and Hinton (2008) and Bonabeau et al. (1999) respectively. The first two real datasets that we consider, are the Iris and Wine datasets.[1] They are widely used as a benchmark for data visualization and classifier testing. The Iris dataset contains 3 classes of 50 four-dimensional real valued patterns each. The Wine dataset is comprised of 178 real valued points with dimension 13, and it is divided into 3 classes. Figures 7 and 8 show the reduction of the Iris and Wine datasets to 2 dimensions by the four algorithms, respectively.

Our next dataset is the Olivetti database.[2] It contains a 400 grayscale images of size 64×64, ten images per person. For our experiments, we select 100 images, ten per person,

[1] Iris and Wine datasets are available at http://archive.ics.uci.edu/ml.

[2] Olivetti face dataset are available at http://mambo.ucsc.edu/psl/olivetti.html.

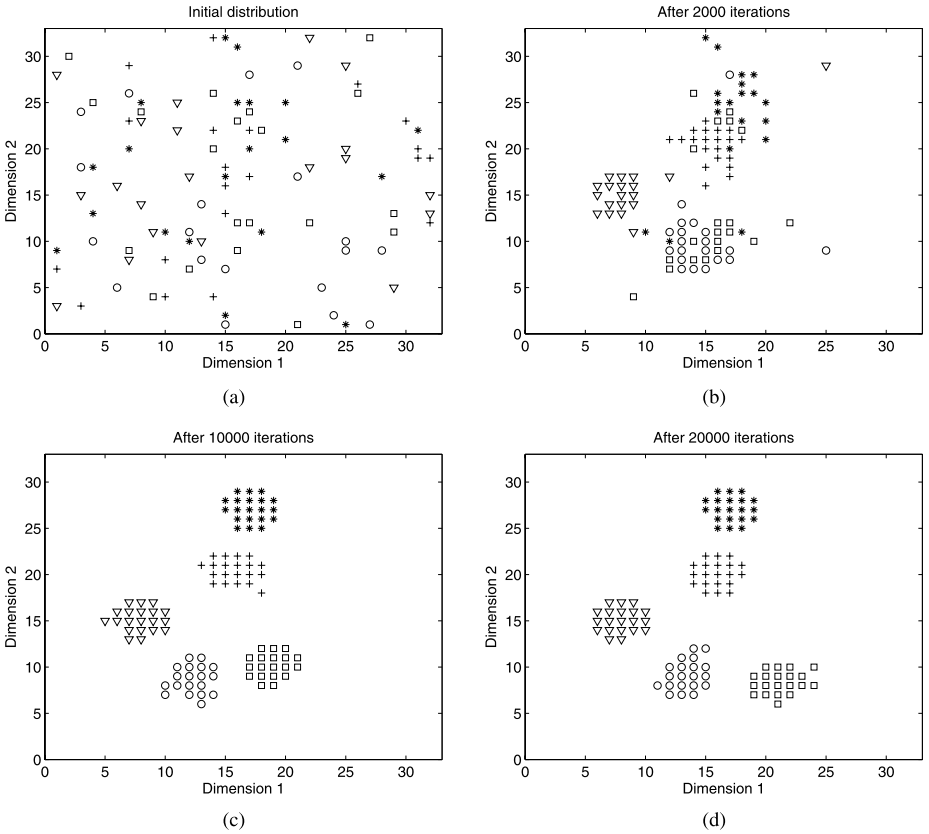

Fig. 2 (**a**) Initial distribution of 100 objects on a 32×32 grid. (**b**) Distribution after 2000 iterations. (**c**) Distribution after 10000 iterations. (**d**) Distribution after 20000 iterations

Table 2 Experimental results on datasets with two clusters, and distance between clusters D_μ. Every cluster has 100 two-dimensional points. Each algorithm's run includes 30000 iterations, where 25 neighboring objects are used

Type of clusters	D_μ	$\overline{\%cc}$	$\overline{\%ilo}$
	20	100	0
1	10	100	0
	5	100	2.1
	28.28	100	0
2	21.21	100	0.35
	14.14	100	4.5

and we convert them to 4096 dimensional vectors, in which each entry is a pixel's gray level. Figure 9 shows the reduction to 2 dimensions by the four algorithms.

In order to quantify the comparative results, we estimate the percentage of objects that are located at incorrect clusters in the two-dimensional representation, assuming that the

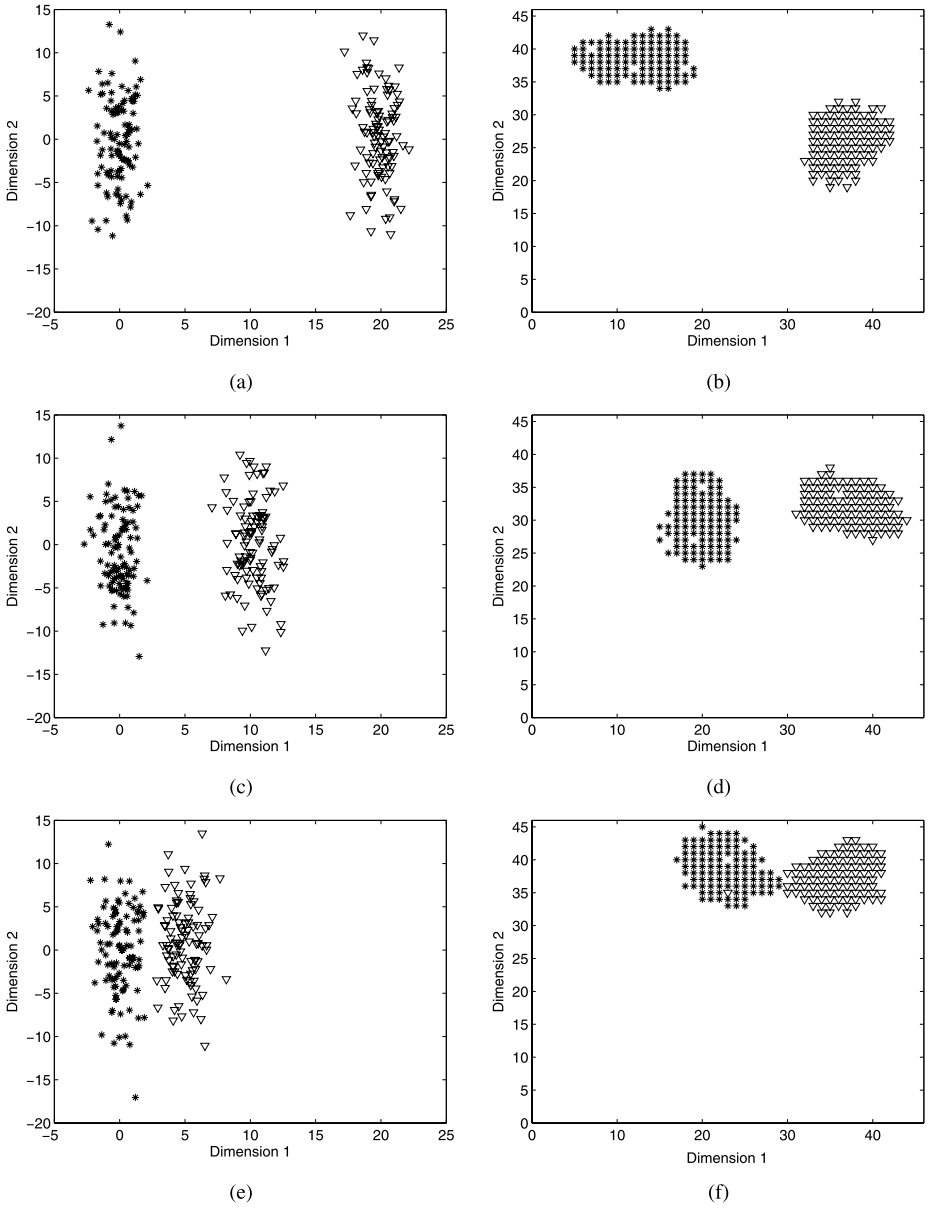

Fig. 3 (**a**), (**c**) and (**e**) correspond to the datasets with type of clusters 1 (according to Table 2) with different distance between means D_μ, and (**b**), (**d**) and (**f**) their corresponding mappings

number of formed clusters corresponds to the number of classes in the dataset. Here, we calculate this statistic using the same procedure as in the synthetic datasets. Table 3 shows the experimental results for the competing algorithms on real datasets.

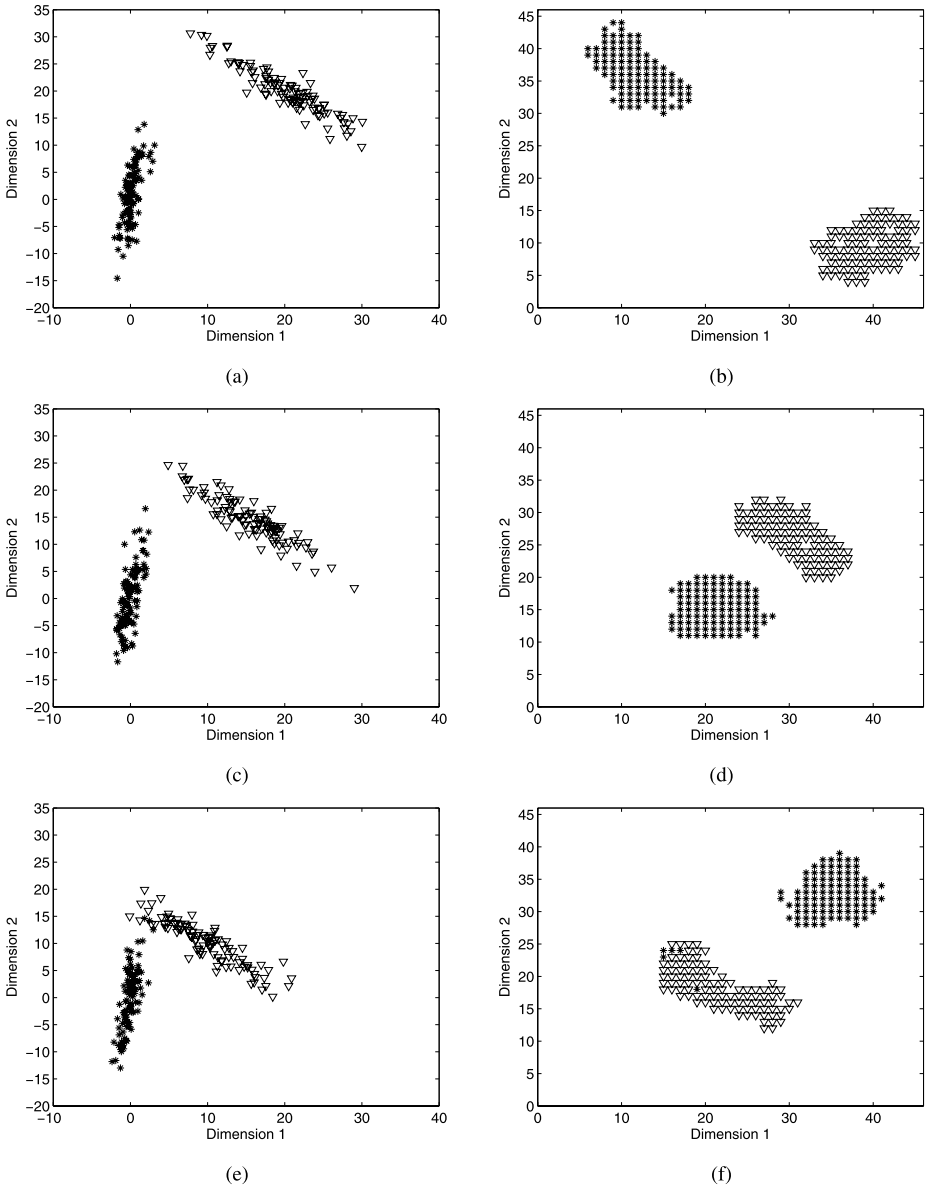

Fig. 4 (**a**), (**c**) and (**e**) correspond to the datasets with type of clusters 2 (according to Table 2) with different distance between means D_μ, and (**b**), (**d**) and (**f**) their corresponding mappings

5 Discussion

Experiments on synthetic and real datasets show that the mapping obtained by the prey model-based algorithm preserves the clustered structure of the dataset, for an appropriate selection of the number of neighboring objects n_o. This is true for a variety of cluster shapes and number of clusters. For instance, Table 1 shows that for all experiments in those datasets,

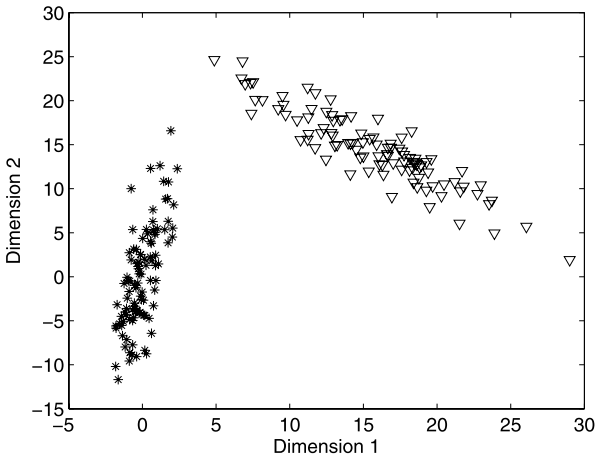

Fig. 5 Dataset with 2 clusters generated by a non-isotropic Gaussian distribution. Each cluster is marked with a different symbol

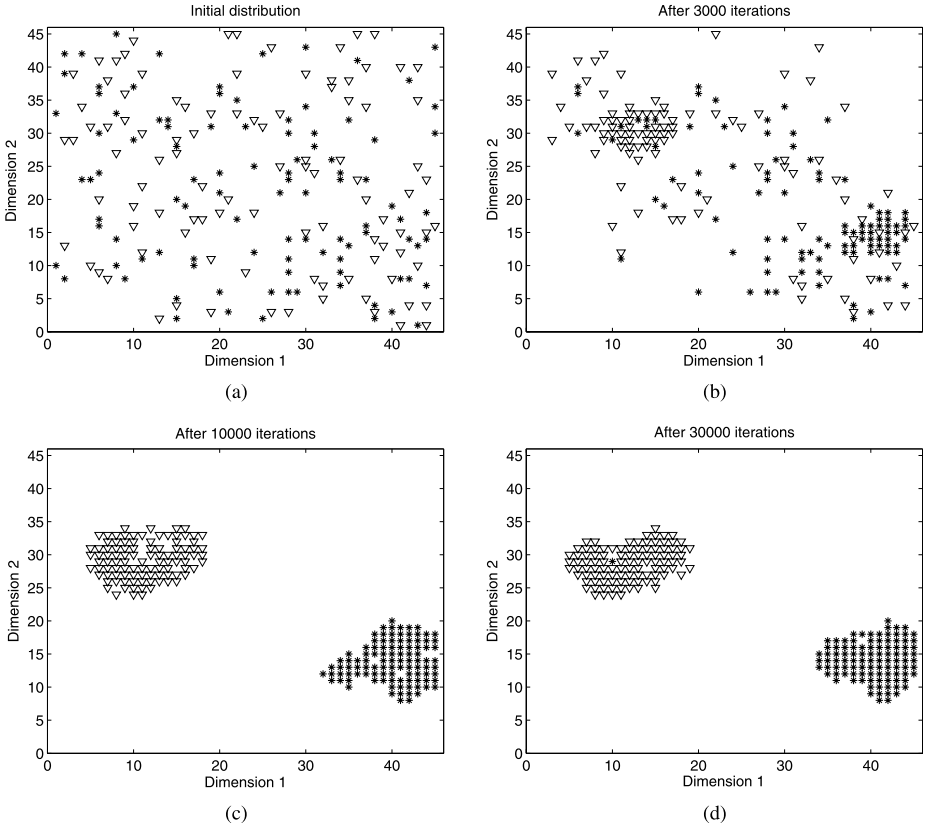

Fig. 6 (**a**) Initial distribution of 200 objects on a 45×45 grid. (**b**) Distribution after 3000 iterations. (**c**) Distribution after 10000 iterations. (**d**) Distribution after 30000 iterations

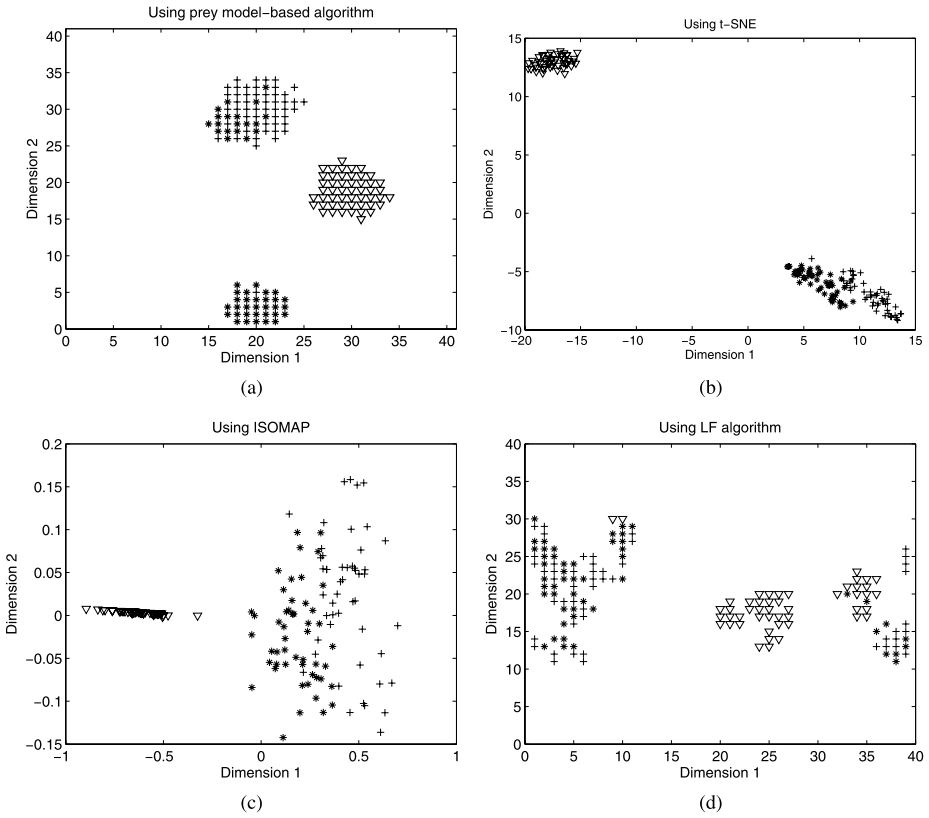

Fig. 7 Two-dimensional reduction for the Iris dataset using (**a**) prey model-based algorithm with 25 neighboring objects and 50000 iterations; (**b**) t-SNE with perplexity of 40 and 1000 iterations; (**c**) ISOMAP with a neighborhood of size 30; and (**d**) LF algorithm after 1000000 iterations. Each class is marked with a different symbol

Table 3 Mean and standard deviation of the percentage of incorrectly located objects after 50 runs of each competing algorithm. Best results in each database are in bold

Database	Prey	ISOMAP	t-SNE	LF
Iris	$\mathbf{10.7 \pm 2.2}$	16.3 ± 0	11.5 ± 3.1	25.3 ± 3.4
Wine	20.1 ± 2.4	22.6 ± 0	$\mathbf{10.7 \pm 5.2}$	33.9 ± 5.3
Olivetti	$\mathbf{0.73 \pm 0.5}$	6.0 ± 0	2.2 ± 1.2	13.4 ± 1.8

objects are grouped into the correct number of clusters. In the second group of synthetic datasets, Table 2 shows that the algorithm has a similar performance, in spite of the asymmetry of the clusters.

For the real world data sets, we have data points that belong to a number of different classes and our results show that the projections obtained by the prey model-based algorithm have clusters that preserve the original classification, even if the classes have different sizes. In the experiments summarized in Table 3, we observe that with respect to our metric, the prey model-based algorithm outperforms the other algorithms in two out of three cases.

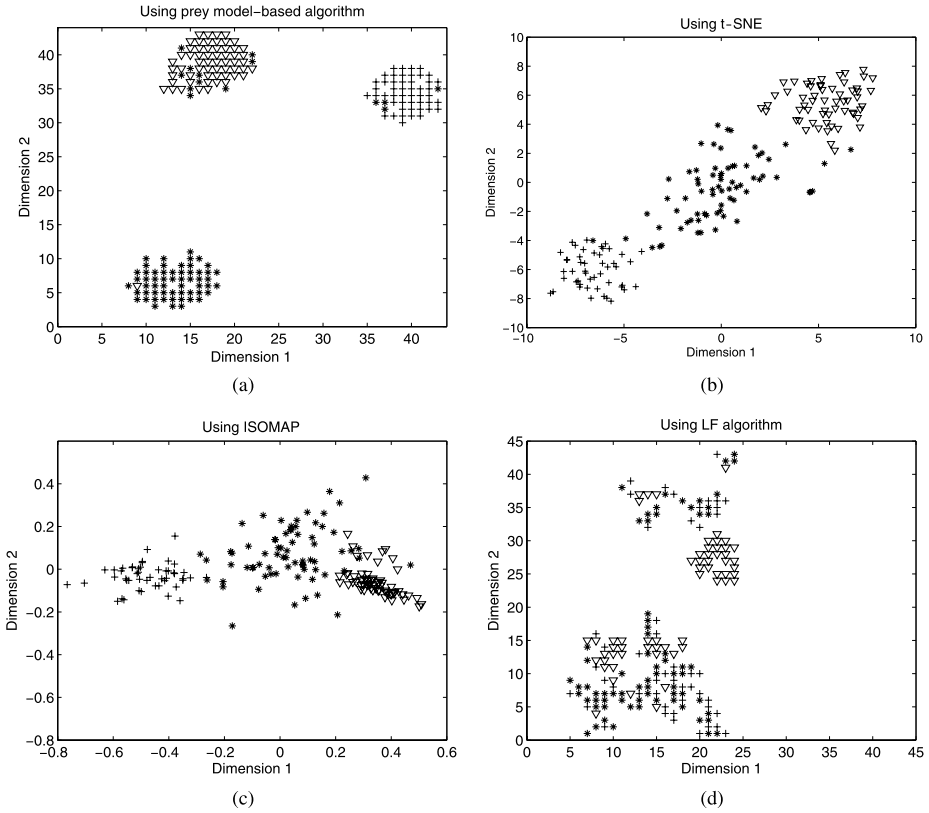

Fig. 8 Two-dimensional reduction for the Wine dataset using (**a**) prey model-based algorithm with 9 neighboring objects and 50000 iterations; (**b**) t-SNE with perplexity of 40; (**c**) ISOMAP with a neighborhood of size 10; and (**d**) LF algorithm after 1000000 iterations. Each class is marked with a different symbol

In general, we observe that t-SNE and ISOMAP may give an idea about the structure of the data, but they do not make evident the differences between classes, as the prey model-based algorithm does. For example, Fig. 7 shows that t-SNE and ISOMAP generate a projection where two different classes are not distinguishable. Figures 8 and 9 show a similar situation, where most of the data points that belong to different classes are not overlapped, but there is not a clear distinction between classes. The dimensionality reduction by the LF algorithm has similar limitations. The three competing algorithms try to group data points that belong to the same class, but the resulting object distribution does not have a defined structure. Compared to the LF algorithm, in our experiments the prey-model based algorithm achieves a better solution using less computational resources: the prey model-based algorithm achieves a correct distribution using 3 agents and up to 50000 iterations, while the LF algorithm demands more than 10 agents and 1000000 iterations to reach the best distribution that it can generate. An advantage of ISOMAP over the prey model-based algorithm is that it is a one-pass algorithm, i.e., they use only one iteration to achieve a solution, while the prey model-based algorithm requires a lot of iterations to arrive to a high-quality projection.

On every execution, the prey model-based algorithm reaches an equilibrium state, that is, after a certain amount of iterations, the distribution of the objects on the grid remains unchanged. This property is illustrated in Figs. 2 and 6. In order to observe this emergent

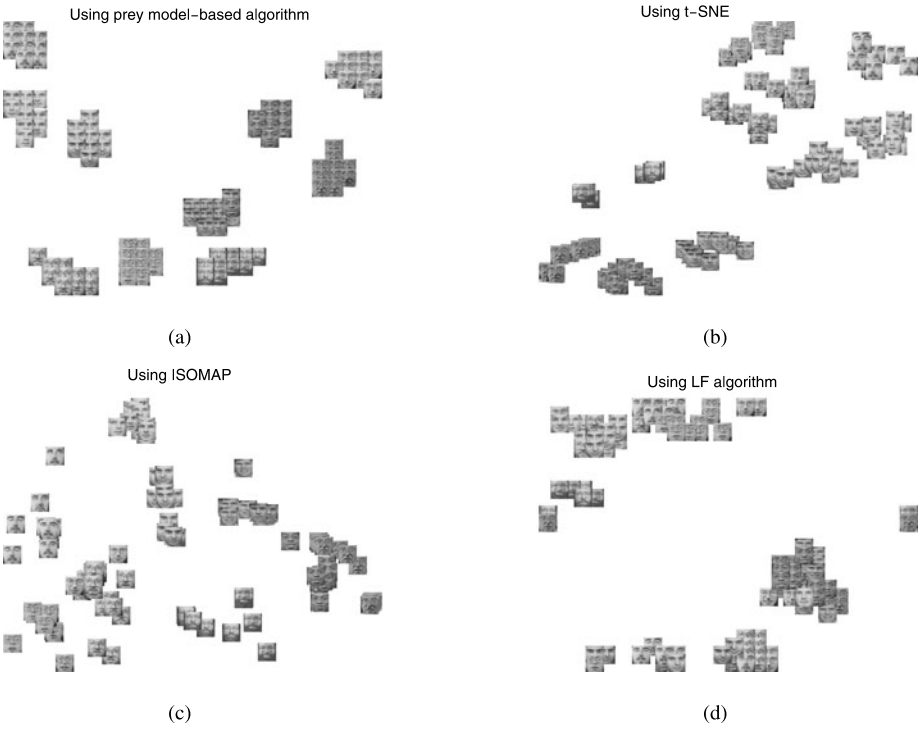

Fig. 9 Two-dimensional reduction for the Olivetti dataset using (**a**) the prey model-based algorithm with 5 neighboring objects and 10000 iterations; (**b**) t-SNE with perplexity of 40; (**c**) ISOMAP with a neighborhood of size 35; and (**d**) LF algorithm after 1000000 iterations. Each data point is represented with its corresponding image

behavior and how fast the algorithm achieves this equilibrium, we examine the spatial entropy (Bonabeau et al. 1999; Kuntz et al. 1999) and the profitability threshold π_{j+1} of the prey selection model (see Algorithm 1), at each iteration of the prey model-based algorithm. To that end, we perform several runs of the prey model-based algorithm with a single agent and 100 prey types, on a toy dataset comprised of 5 clusters with 13 points each. The profitability threshold π_{j+1} is examined when the agent is in the picking and dropping mode, in order to observe when the pool of prey items remains approximately constant. In the algorithm, the profitability of the i-th prey type π_i is equal to the energy gain v_i, since the processing time e_i is assumed to be equal to 1. On the other hand, spatial entropy measures how fast an algorithm groups the objects at different spatial scales. Let $\mathcal{G}_1, \ldots, \mathcal{G}_r$, be a partition of the grid $\mathcal{G}$ into r sub-grids of the same size. Then, the spatial entropy E_s associated with this partition, is defined by

$$E_s = -\sum_{i=1}^{r} p_t\left(\mathcal{G}_i\right) \log_r\left(p_t\left(\mathcal{G}_i\right)\right) \tag{12}$$

where $p_t(\mathcal{G}_i)$ is the fraction of all objects that are found in the sub-grid $\mathcal{G}_i$ at iteration t. We have that $E_s = 0$ when there exists a sub-grid which contains all the objects, $E_s = 1$ when there is a uniform object distribution, i.e., $p_t(\mathcal{G}_1) = \cdots = p_t(\mathcal{G}_r) = 1/r$, and E_s decreases when grouping proceeds. Figure 10 shows π_{j+1} when the agent is picking up and dropping

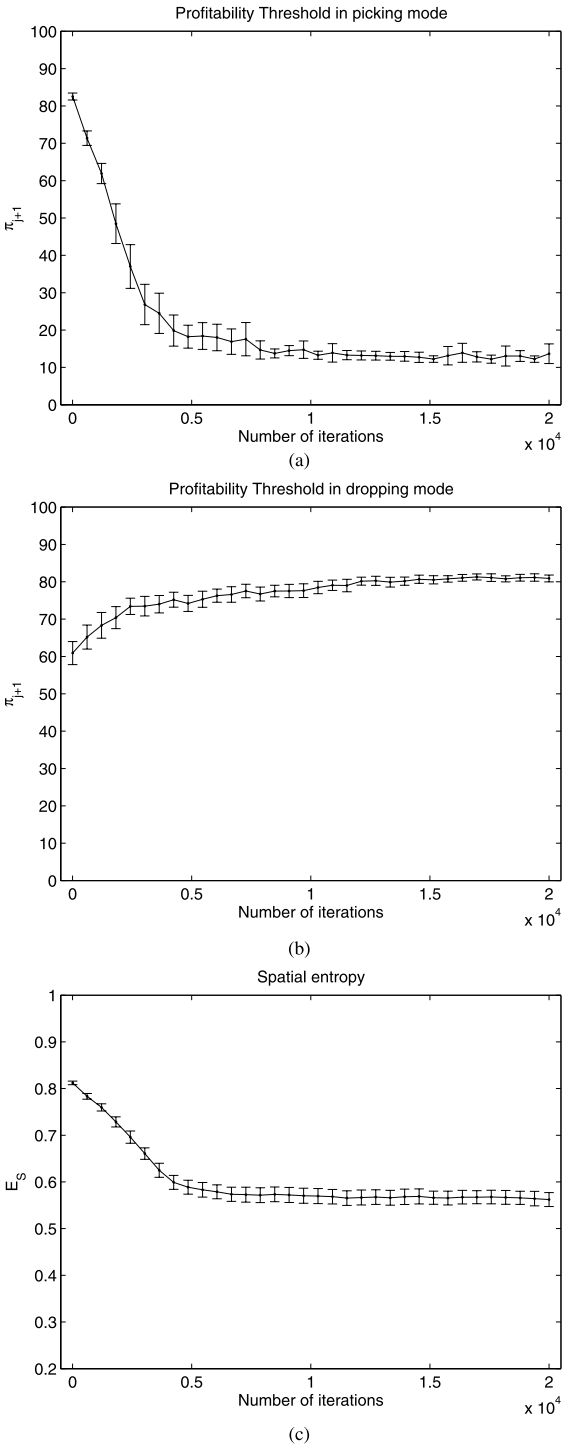

Fig. 10 Variation of the mean and standard deviation of (**a**) the profitability threshold when the agent is picking up objects; (**b**) the profitability threshold when the agent is dropping objects; (**c**) the spatial entropy with a 28×28 grid split into 16 sub-grids of size 7×7. We perform 20000 runs of the prey model-based algorithm for the toy dataset

objects, and the spatial entropy, throughout the execution of the algorithm. These variables indicate that the algorithm converges in 10000 iterations for the toy dataset, approximately. The speed of this convergence can be increased by adding agents to the process, but this inclusion increases the execution time of the algorithm. For the data sets selected in the experiments, we consider that 3 agents is a good trade-off between the number of agents and computation time.

In foraging theory, the *principle of lost opportunity* states that decisions about attacking prey items can be assessed by comparing potential gains from attack, with the potential loss of opportunity to do better (Stephens and Krebs 1986). This principle is illustrated in Figs. 10(a) and 10(b), where the selectivity of an agent is examined using the profitability threshold when the agent is picking up and dropping objects. In the case when the agent is picking up objects, the profitability threshold decreases and arrives at a stable point during the execution of the algorithm. This is due to the fact that at the initial stage of the object distribution (when the objects are randomly scattered on the grid), the encounter rate of prey items that provide higher energy gain is larger. Therefore, the agent will be more selective (high profitability threshold) since the potential gain from attack those type of preys is high. Then, when the distribution of objects gradually adopts a clustered form, prey items that provide higher energy gain become scarce. Hence, the agent has to be less selective (low profitability threshold) since the potential gain from attacking is low. Similarly, the principle of lost opportunity is appreciable in an opposite way, when the agent is dropping objects.

6 Conclusions and future work

We present a novel algorithm that provides a two-dimensional discrete representation of multivariate information, revealing the existence of clusters in the data. The proposed technique is based on behavioral ecology ideas, where an optimal decision-making system is designed to follow the prey model, a concept from foraging theory. A software agent is thought as an animal searching for prey items, a discrete space as the environment where the animal lives, and the nutrients are related to objects representing datapoints and the pairwise distances between them. As a result of this decision-making strategy, the final discrete distribution of objects brings out the presence of natural groupings in the data. Experimental results on a variety of datasets validate the performance of the proposed algorithm, and show that it outperforms the competing algorithms in most of cases when the main task is to allow the exploration of clustered data.

Future work includes applications of additional foraging theory concepts, such as extensions of the prey model (Dukas 1998), the patch model (Stephens and Krebs 1986), and social foraging (Giraldeau and Caraco 2000). Additionally, we would like to develop mathematical insight into the relationship between the data distribution produced by the algorithm and the underlying theoretical framework behind the prey model. We would like to explore alternative stopping criteria where the equilibrium state is taken into account. We believe that this can lead to important savings in the execution time of the algorithm.

Acknowledgements The authors would like to thank the editors of this special issue, as well as the anonymous referees for their constructive comments.

References

Andrews, B. W., Passino, K. M., & Waite, T. A. (2007a). Social foraging theory for robust multiagent system design. *IEEE Transactions on Automation Science and Engineering*, *4*(1), 74–86.

Andrews, B. W., Passino, K. M., & Waite, T. A. (2007b). Foraging theory for autonomous vehicle decision-making system design. *Journal of Intelligent and Robotic Systems*, *49*(1), 39–65.

Bishop, C. M., Svensén, M., & Williams, C. K. (1998). Generative topographic mapping. *Neural Computation*, *10*(1), 215–234.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial swarm intelligence*. Oxford: Oxford University Press.

Carreira-Perpiñán, M. Á., & Renals, S. (1998). Dimensionality reduction of electropalatographic data using latent variable models. *Speech Communication*, *26*(4), 259–282.

Chanda, P., Zhang, A., Brezeau, D., Sucheston, L., Freudenheim, J. L., Ambrosone, C., & Ramanathan, M. (2007). Information-theoretic metrics for visualizing gene-environment interactions. *American Journal of Human Genetics*, *81*(5), 939–963.

Deneubourg, J.-L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1990). The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior: From animals to animats* (pp. 356–363). Cambridge: MIT Press.

Dukas, R. (1998). *Cognitive ecology: The evolutionary ecology of information processing and decision making*. Chicago: University of Chicago Press.

Giraldeau, L.-A., & Caraco, T. (2000). *Monographs in behavior and ecology*. *Social foraging theory*. Princeton: Princeton University Press.

Handl, J., Knowles, J. D., & Dorigo, M. (2006). Ant-based clustering and topographic mapping. *Artificial Life*, *12*(1), 35–61.

Hinton, G., & Roweis, S. (2002). Stochastic neighbor embedding. In *Advances in neural information processing systems* (pp. 833–840). Cambridge: MIT Press.

Jain, A. K., Narasimha Murty Murty, M., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, *31*(3), 264–323.

Jain, A. K., Duin, R. P. W., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(1), 4–37.

King, B. (1967). Setp-wise clustering procedures. *Journal of the American Statistical Association*, *69*, 86–101.

Kivinen, J., & Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, *132*(1).

Kivinen, J., Smola, A. J., & Williamson, R. C. (2004). Online learning with kernels. *IEEE Transactions on Signal Processing*, *52*(8), 2165–2176.

Kohonen, T. (2000). *Self-organizing maps* (3rd ed.). Heidelberg: Springer.

Kuntz, P., Snyers, D., & Layzell, P. (1999). A stochastic heuristic for visualising graph clusters in bi-dimensional space prior to partitioning. *Journal of Heuristics*, *5*(3), 327–351.

Lisboa, P. J., Ellis, I. O., Green, A. R., Ambrogi, F., & Dias, B. (2008). Cluster-based visualization with scatter matrices. *Pattern Recognition Letters*, *29*(13), 1814–1823.

Littlestone, N., & Warmuth, M. K. (1994). The weighted majority algorithm. *Information and Computation*, *108*(2), 212–261.

Lumer, E. D., & Faieta, B. (1994). Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on simulation of adaptive behavior: From animals to animats 3* (pp. 501–508). Cambridge: MIT Press.

Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., & Baesens, B. (2007). Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, *11*(5), 651–665.

McClurkin, J. W., Optican, L. M., Richmond, B. J., & Gawne, T. J. (1991). Concurrent processing and complexity of temporally encoded neuronal messages in visual perception. *Science*, *253*(5020), 675–677.

Ngenkaew, W., Ono, S., & Nakayama, S. (2008). The deposition of multiple pheromones in ant-based clustering. *International Journal of Innovative Computing Information and Control*, *4*(7), 1583–1593.

Passino, K. M. (2005). *Biomimicry for optimization, control and automation*. London: Springer.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, *2*(6), 559–572.

Pirolli, P. (2007). *Information foraging theory*. Oxford: Oxford University Press.

Quijano, N., & Passino, K. M. (2007). Honey bee social foraging algorithms for resource allocation, Part ii: application. In *Proceedings of the American control conference* (pp. 3389–3394), New York, USA, July 2007.

Quijano, N., Andrews, B. W., & Passino, K. M. (2006). Foraging theory for multizone temperature control. *IEEE Computational Intelligence Magazine*, *1*(4), 18–27.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*(5500), 2323–2326.

Sanguinetti, G. (2008). Dimensionality reduction of clustered data sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(3), 535–540.

Stephens, D. W., & Krebs, J. R. (1986). *Monographs in behavior and ecology*. *Foraging theory*. Princeton: Princeton University Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge: MIT Press.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Torgerson, W. S. (1958). *Theory and methods of scaling*. New York: Wiley.

Ulam, P., & Balch, T. (2003). Niche selection in foraging tasks in multi-robot teams using reinforcement learning. In *Second international workshop on the mathematics and algorithms of social insects*, 2003.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

Wen, G., Jiang, L., Wen, J., & Shadbolt, N. R. (2006). Clustering-based nonlinear dimensionality reduction on manifold. In Q. Yang & G. Webb (Eds.), *PRICAI 2006: Trends in artificial intelligence* (pp. 444–453).