Data Min Knowl Disc
DOI 10.1007/s10618-013-0323-0

Semi-supervised projected model-based clustering

Luis Guerra · Concha Bielza · Víctor Robles · Pedro Larrañaga

Received: 3 September 2012 / Accepted: 7 May 2013

Abstract We present an adaptation of model-based clustering for partially labeled data, that is capable of finding hidden cluster labels. All the originally known and discoverable clusters are represented using localized feature subset selections (subspaces), obtaining clusters unable to be discovered by global feature subset selection. The semi-supervised projected model-based clustering algorithm (SeSProC) also includes a novel model selection approach, using a greedy forward search to estimate the final number of clusters. The quality of SeSProC is assessed using synthetic data, demonstrating its effectiveness, under different data conditions, not only at classifying instances with known labels, but also at discovering completely hidden clusters in different subspaces. Besides, SeSProC also outperforms three related baseline algorithms in most scenarios using synthetic and real data sets.

Responsible editor: Charu Aggarwal.

L. Guerra (✉) · C. Bielza · P. Larrañaga
Computational Intelligence Group, Departamento de Inteligencia Artificial,
Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo,
28660 Boadilla del Monte, Madrid, Spain
e-mail: l.guerra@upm.es

C. Bielza
e-mail: mcbielza@fi.upm.es

P. Larrañaga
e-mail: plarranaga@fi.upm.es

V. Robles
Departamento de Arquitectura y Tecnología de Sistemas Informáticos, Facultad de Informática,
Universidad Politécnica de Madrid, Campus de Montegancedo,
28660 Boadilla del Monte, Madrid, Spain
e-mail: v.robles@fi.upm.es

Published online: 21 May 2013

Keywords Clustering · Subspaces · Semi-supervised · Model-based · Partially labeled data

1 Introduction

Partially labeled data sets are becoming common in machine learning, specifically in classification tasks. Although the available data labels are an important source of information for improving classification, they are still scarce in some domains because they are either expensive or labour intensive to gather. Learning problems using partially labeled data are semi-supervised. Depending on whether the unlabeled instances can be classified according to one of the known labels or there is the possibility of discovering new previously unknown clusters, we can refer to this problem as semi-supervised classification or semi-supervised clustering, respectively. Our approach can be classed as semi-supervised clustering since instances are classified according to known labels, but new clusters can also be found if necessary.

We rely on finite mixture models to tackle this problem. We propose an algorithm that integrates the known labels and the expectation-maximization (EM) algorithm (Dempster et al. 1977) to help estimate the parameters of each mixture component. A localized feature subset selection (LFSS) approach, where each cluster is described using a different subset of features (subspace), is also integrated with the EM. LFSS is a recent trend that is useful for finding hidden data structures. This is different from traditional global feature subset selection (GFSS), where a single subset of features is used to carry out the clustering. Therefore our proposal is a semi-supervised projected model-based clustering (SeSProC) algorithm.

Our algorithm is capable of finding new unknown clusters, the final number of clusters must be then estimated as with many traditional clustering algorithms. We introduce a novel approach, based on an iterative greedy forward search, which uses the known groups as a starting point and tries to introduce a new component at each iteration. Our algorithm is based on the *cluster assumption*, which states that instances in the same cluster are likely to belong to the same class.

The rest of the paper is organized as follows. Section 2 gives an introduction to related work on the topics covered in our proposal, highlighting the advantages of our contribution. In Sect. 3, we first introduce basic model-based clustering theory and then we add the different characteristics of our proposal and detail SeSProC. This section is supported by four appendices at the end of the document. Section 4 outlines the experiment describing the database, experimental process, and results on both, synthetic and real data sets. Finally, Sect. 5 contains a discussion, conclusions, and future work.

2 Related work

Together with partitional, hierarchical, or density-based approaches, which are beyond the scope of this paper, model-based clustering (McLachlan and Basford 1988) is one of the main approaches for clustering. This approach is based on "soft" clustering, i.e., each instance has some probability of belonging to each cluster. This kind of

assignment might lead to more accurate solutions. One advantage of using model-based clustering, is that components and parameter estimations are searched simultaneously, avoiding problems that can arise when this is done separately. Also the problem of selecting the number of clusters (components) in model-based clustering can be considered as a statistical model selection problem, and models with different number of components can be compared using known approaches (Fraley and Raftery 1998). For a recent model-based clustering review and a book focused on this approach, see Maitra and Melnykov (2010) and McLachlan and Peel (2000), respectively.

2.1 Semi-supervised learning

Semi-supervised learning (Chapelle et al. 2006; Chawla and Karakoulas 2005; Zhu and Goldberg 2009) can be located between clustering and supervised classification, and deals with learning either in the presence of both labeled and unlabeled data or some other kinds of constraints, like relationships between instances. Neural networks (Chandel et al. 2009) and nearest neighbors classifiers (Wang et al. 2006) have been proposed for this task.

Semi-supervised learning can be divided into semi-supervised classification and semi-supervised clustering, depending on the input data and the purpose (Zhu 2005). The task related to SeSProC is semi-supervised clustering, since new groups, which are unknown in the input partially labeled data, may be found. Semi-supervised clustering is sometimes known as constrained clustering, since the known information can take the shape of pairwise constraints (must-link and cannot-link constraints). When data are partially labeled, constraints can be directly inferred from the available labels. Metric pairwise constrained K-means (MPCKM) (Basu et al. 2004), is an example of an algorithm based on pairwise constraints. This algorithm penalizes its objective function and trains a metric both by using the available constraints. Reviews of recent works dealing with constrained clustering can be found in Basu et al. (2009).

The idea of using partially labeled data sets together with EM was introduced in Miller and Browning (2003). This was also applied to a leukemia data set in Alexandridis et al. (2004). Examples of the introduction of constraints into the EM algorithm for finite mixture models and model-based clustering can be found in Shental et al. (2003) and Lange et al. (2005). Another approach can be found in Lu and Leen (2005), where clustering constraints were expressed in the prior distribution over assignments of instances to clusters in a Gaussian mixture model. Cluster assignments were then penalized depending on this prior and according to the degree of constraint violation. Finally, a more recent example of semi-supervised mixture modeling can be found in Miller et al. (2009), where standard semi-supervised mixtures and nearest neighbors classification were combined. We can conclude that available information has been successfully introduced into model-based clustering, using different approaches, in many previous works.

2.2 Clustering in subspaces

Two important surveys related to clustering in subspaces were presented in Parsons et al. (2004) and Kriegel et al. (2009). Kriegel et al, who presented another related

and updated review in Kriegel et al. (2012), categorize the subspace approaches more thoroughly, distinguishing between (i) algorithms that are able to find clusters parallel to the axes (axis-parallel algorithms), (ii) algorithms that find clusters in arbitrarily oriented subspaces (correlation clustering), and (iii) clustering algorithms based on patterns, also called biclustering, which is out of the scope of this work. Correlation clustering is a more general case of subspace clustering than (i), but the number of arbitrarily oriented subspaces might become infinite. Examples of correlation clustering algorithms are ORCLUS (Aggarwal and Yu 2000), P3C (Moise et al. 2008), MrCC (Cordeiro et al. 2010), and SSCC (Günnemann et al. 2012). Taking into account the problem with possibly high number of subspaces, it can be enough, depending on the application, to assume the existence of axis-parallel subspaces. Our proposal is therefore focused on this approach.

Axis-parallel algorithms can be further divided, according to how algorithms are created, into bottom-up and top-down algorithms depending on whether the search starts from all one-dimensional subspaces or from the full-dimensional space, respectively. This separation is also suggested by Parsons et al. (2004). However, axis-parallel algorithms can also be divided, according to the made assumptions, into the next types of algorithms:

- Projected clustering algorithms, where disjoint clusters are identified and characterized by different subspaces. Related to this kind of algorithms, soft projected clustering aims at finding subspaces by weighting features instead of by selecting them as relevant/irrelevant. A feature with a high weight is considered more important to define a subspace than another lower weighted feature. Many projected algorithms are also top-down algorithms. PROCLUS (Aggarwal et al. 1999) was the first top-down and projected clustering algorithm, followed by FINDIT (Woo et al. 2004), or COSA (Friedman and Meulman 2004). COSA is an example of algorithm that is also classified as soft projected clustering algorithm.
- Subspace clustering algorithms, which aim at finding all subspaces where clusters can be found. Therefore, instances can be grouped into many different clusters (overlapping is allowed), and each cluster is characterized by a different feature subspace. Most subspace clustering algorithms are bottom-up algorithms. CLIQUE (Agrawal et al. 1998) was the first algorithm of this type, using a grid approach. Other algorithms are ENCLUS (Cheng et al. 1999) and MAFIA (Goil et al. 1999).
- Hybrid algorithms find clusters that may overlap. However, the number of found subspaces is not the full set of possible subspaces, but only a smaller subset according to some criteria. This kind of algorithms can be both top-down and bottom-up algorithms. Some examples of hybrid algorithms are DOC (Procopiuc et al. 2002), which is a density-based algorithm, and HARP (Yip et al. 2004), which uses different distance functions to iteratively merge clusters by minimizing the number of relevant features for each cluster.

Our proposal can be categorized as soft projected soft clustering because subspaces are found by weighting features and clusters may also overlap since model-based clustering is based on soft clustering. Therefore, our proposal is soft at selecting features and at assigning instances to clusters.

An important and related concept is *feature saliency*, that is the probability that a feature is relevant to a cluster. This probability was introduced into the EM algorithm for model-based clustering and GFSS (Law et al. 2004). It is a close approach to soft projected clustering if it is extended to all clusters separately. The extension of Law et al. (2004) to subspaces was applied in Li et al. (2007), which is the groundwork for our research. A variant was later presented by the same authors, estimating parameters with Bayesian variational learning (Li et al. 2009). Other approaches (Boutemedjet et al. 2010; Hoff 2005, 2006), based also on mixture models, used the Dirichlet process. Somehow related, Graham and Miller 2006 presented an approach allowing one shared representation for each feature when appropriate to reduce the complexity of the model. The extension of this work, using multivariate Gaussian mixtures, was presented in Markley and Miller (2010). Finally, MPC (Chen et al. 2012) is a related projected model-based clustering algorithm. MPC requires the correct number of clusters to be entered as an input parameter and is sensitive to initialization.

2.3 Semi-supervised and subspaces

Another recent review (Sim et al. 2012) identifies the relationship between semi-supervised clustering and subspace-based approaches. Sim et al deal with subspace clustering from an enhanced point of view, classifying approaches into two groups: *handling complex data* and *improving clustering results*. Approaches in the first group handle streaming (Aggarwal et al. 2004; Kriegel et al. 2011) or categorical data (Müller et al. 2009). The second group covers three aspects: finding significant subspace clusters (instead of all subspace clusters), relaxing the parameter-sensitivity problem, and introducing knowledge as semi-supervised subspace clustering (see Fig. 1).

In this respect, different knowledge can be introduced to enhance clustering results. For example, ASCLU (Günnemann et al. 2010) finds alternative solutions considering previous subspace clustering results using different features. However, knowledge is commonly related to constraints or known labels. The algorithm SSPC (Yip et al. 2005) is based on a partitional method similar to K-medoids algorithms to find projected clusters, where the available domain knowledge is related to class labels rather than instance-level constraints. Cheng et al. (2008) presented an extension of LAC, called CLWC, incorporating pairwise constraints (must-link and cannot-link constraints) into the local weighting scheme. A main limitation of these algorithms is that all final clusters must be represented by some of the beforehand known labels or constraints. Fromont et al. (2009) presented a common framework for subspace clustering, called SC-MINER. This work proposed integrating pairwise constraints

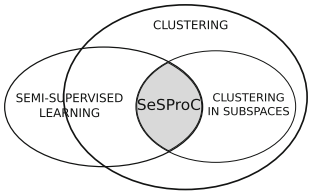

Fig. 1 SeSProC is a semi-supervised soft projected model-based clustering algorithm that combines the search for subspaces and class labels information

into the mining step of subspace clustering algorithms. SISC (Ahmed and Khan 2009) is another closely related approach that deals with binary features only. Zhang et al. (2010) pointed out that feature correlation and distance divergence are important considerations for subspace clustering, and, consequently, used constraints for clustering. Zhang et al. (2011) recently exploited constraint inconsistency for dimension selection in subspaces.

In sum, our work focuses on a projected soft clustering approach based on semi-supervised clustering in subspaces. An adaptation of the EM algorithm is used to fit the finite mixture models, including both the search for subspaces by weighting the feature relevance and the labeled data as known information. The labeled data are used to initialize the model, whereas new components are added depending on how instances fit the known components. A novel greedy process estimates the number of mixture components. This process is based on both instance fitting to the components and a quality measure.

Our proposal contributes to projected clustering and semi-supervised learning, providing a single algorithm that covers all the three aspects presented in Sim et al. (2012) to improve clustering results:

- We guide the clustering process using the available data labels. They are used to initialize the known groups and as a starting point for selecting the final number of clusters. Note that our algorithm does not need labels of all the final clusters as input.
- We relax the number of tuning parameters. Users have to choose only one parameter. This parameter does not have a big impact on the results and is easily adjustable.
- Our work is based on projected clustering. Therefore, we not enumerate all but only the subspace clusters considered as relevant by the algorithm.

3 Proposed algorithm: SeSProC

This section introduces our algorithm SeSProC for semi-supervised subspace model-based clustering using the EM algorithm, and estimating the final number of mixture components. It is based on the traditional concept of finite mixture models (Bishop 2007).

3.1 Notation

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of instances described by continuous features in a space of dimension F, that is, $\mathbf{x}_i \in \Re^F$, $\forall i \in \{1, \ldots, N\}$. Besides, the class[1] information of some instances is available in this partially labeled data set. Thus, $\mathcal{X}$ can be divided into $\mathcal{X} = \mathcal{X}^L \cup \mathcal{X}^U$, where $\mathcal{X}^L = \{\mathbf{x}_1, \ldots, \mathbf{x}_L\}$ is the subset of instances with an associated known class label and $\mathcal{X}^U = \{\mathbf{x}_{L+1}, \ldots, \mathbf{x}_N\}$ are the instances with

[1] Note that "class", "component", and "cluster" are equivalent concepts at the end of the classification, but each concept will be used here to refer, respectively to a priori knowledge about instances (classes), mixture components (components) or identified groups (clusters).

unknown labels. This information is gathered in the mixture using a latent variable $\mathcal{Z} = \mathcal{Z}^L \cup \mathcal{Z}^U = \{\mathbf{z}_1, \ldots, \mathbf{z}_L\} \cup \{\mathbf{z}_{L+1}, \ldots, \mathbf{z}_N\}$, separating known from unknown class labels, respectively, where $\mathbf{z}_i = (z_{i1}, \ldots, z_{iC}, z_{i,C+1}, \ldots, z_{iK})$, with $z_{im} = 1$ if instance i belongs to component m and with all other elements $z_{im'} = 0, \forall\, m' \neq m$. Note that $z_{im} = 0$ if $m > C$ for $\mathbf{z}_i \in \mathcal{Z}^L$, C being the number of known classes in our semi-supervised setting. This constraint does not apply to $\mathcal{Z}^U$ because the instances with unknown class labels can belong to a known class or any other discoverable class. For this reason, the final number of classes is K, with $K \geq C$.

The feature relevance for each mixture component is indicated in the set $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$, being $\mathbf{v}_m = (v_{m1}, \ldots, v_{mF})$ with $v_{mj} = 1$ if feature j is relevant to component m, and $v_{mj} = 0$ otherwise, $\forall\, m \in \{1, \ldots, K\}$, $j \in \{1, \ldots, F\}$. The values of each $\mathbf{v}_m$ are unknown, and, therefore, set $\mathcal{V}$ is a new set of latent variables. The aim of the algorithm is to find the correct group component for each instance in $\mathcal{X}^U$, which is either a known class or a new unknown cluster.

3.2 Basic theory

The underlying theory of mixture modeling is the groundwork for SeSProC. There is no kind of subset selection or knowledge about the data to start with in the basic theory. For this reason, $\mathcal{V}$ does not exist, and $\mathcal{Z}^L$ is not a priori known. In a finite mixture model, an instance is assumed to be generated by a probabilistic model given by a finite mixture of distributions. Assuming that the mixture has K components, the density function of an instance $\mathbf{x}_i$ is (see details in Appendix 1)

$$p(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m\, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m), \tag{1}$$

where $p(\cdot)$ is the density function, and $\boldsymbol{\theta}_m$ is the parameter set and π_m is the mixing probability of component m, with $\pi_m \geq 0$ and $\sum_{m=1}^{K} \pi_m = 1$. The full parameter set of the mixture is $\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K, \pi_1, \ldots, \pi_K\}$. This parameter set could be directly estimated using the maximum likelihood method if both the latent variables ($\mathcal{Z}$) and the observable data ($\mathcal{X}$) were known , i.e., if the complete-data were available to compute the log-likelihood function. This function would be (see details in Appendix 1)

$$\log L(\Theta \mid \mathcal{X}, \mathcal{Z}) = \sum_{i=1}^{N} \sum_{m=1}^{K} z_{im} \left(\log \pi_m + \log p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)\right). \tag{2}$$

Nevertheless, the latent variables are unknown and we cannot use this function directly. However, we can obtain the expectation of this log-likelihood function with respect to the posterior distribution of the latent variables. This expectation is calculated in iteration t, having fixed the parameters from the previous iteration[2] $t-1$, in the E-step

[2] Note that, for legibility, the notation related to iterations is used with Θ, but not with $\boldsymbol{\theta}$ throughout the paper.

of the EM algorithm. After this, the parameters of the distributions are recalculated to maximize this expectation (M-step). These two steps are repeated until a convergence criterion is reached. Hence, using the expectation of z_{im},

$$\begin{aligned}\mathbb{E}_{z_{im}|\mathbf{x}_i,\boldsymbol{\theta}_m}[z_{im}] &= \gamma(z_{im}) \\ &= \frac{\pi_m\, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)}{\sum_{m'=1}^{K} \pi_{m'}\, p(\mathbf{x}_i \mid \boldsymbol{\theta}_{m'})} \\ &= p(z_{im} = 1 \mid \mathbf{x}_i, \boldsymbol{\theta}_m),\end{aligned}$$

the expectation of the complete-data log-likelihood function is given by (see details in Appendix 1)

$$\mathbb{E}_{\mathcal{Z}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z})] = \sum_{i=1}^{N}\sum_{m=1}^{K} \gamma(z_{im})\,(\log \pi_m + \log p(\mathbf{x}_i \mid \boldsymbol{\theta}_m))\,. \quad (3)$$

The set of parameters is estimated in the M-step to maximize the expectation of the complete-data log-likelihood, presented in Eq. (3). They are obtained by computing the partial derivatives with respect to the different parameters and equaling to zero. These derivatives will be presented as part of the proposed solution for our specific problem.

3.3 Introduction to SeSProC

We apply the above mixture model theory to a clustering problem with two specific characteristics:

1. The groups of instances can be hidden in different feature subspaces. Therefore, an LFSS is required in each mixture component. This way we can identify data structures that would remain undiscovered using all features or GFSS.
2. The class information of some instances is available. This knowledge is used during the EM process to improve the final clustering; therefore, this is a semi-supervised clustering task.

The possibilities for adapting basic model-based clustering to the characteristics of our problem are described next. First, LFSS is presented in Sect. 3.4, which leads on to the development for maximizing the above expectation of the complete-data log-likelihood function. Section 3.5 details how the knowledge about some instance labels is taken into account to improve the final clustering. Section 3.6 discusses how to estimate the final number of components using all the above characteristics, based on an iterative forward greedy approach. Finally, both the E and M procedural steps are detailed in Sects. 3.7 and 3.8, respectively. Therefore, SeSProC adds subspaces and available label instance information to model-based clustering, plus a novel proposal for the estimation of the final number of clusters.

3.4 Adding subspaces to EM

We have to know which features are relevant to each mixture component to find the subspaces that best describe the components. As previously mentioned, each component's feature relevance is indicated in the set $\mathcal{V}$, which is a new set of latent variables. First, we can define, for each component and feature, $\rho_{mj} = p(v_{mj} = 1)$, the probability that feature j is relevant to component m. A feature is relevant to a component following the concept of *feature saliency* indicated in Law et al. (2004), but extending it to each cluster. We assume that features are independent given the component label, which is a common choice in many models and obtains reasonable results, as also commented in Law et al. (2004). Then, the new density function is (see details in Appendix 2)

$$p(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj})\, p(x_{ij} \mid \lambda_{mj}) \Big),$$

where θ_{mj} indicates the parameters for the density function if feature j is relevant to component m, whereas λ_{mj} indicates the parameters for the density function if feature j is not relevant to component m. With the inclusion of subspaces, a whole new set of parameters has to be estimated: $\Theta = \{\theta_{mj}, \lambda_{mj}, \rho_{mj}, \pi_m\}_{m=1,\ldots,K;\, j=1,\ldots,F}$.

The new complete-data includes the new set of latent variables ($\mathcal{V}$). Therefore, including it in Eq. (2), the new complete-data log-likelihood function is given by (see details in Appendix 2)

$$\begin{aligned}
&\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V}) \\
&= \sum_{i=1}^{N} \sum_{m=1}^{K} \Big(z_{im} \log \pi_m \\
&\quad + \sum_{j=1}^{F} \big(z_{im} \big[v_{mj} (\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + (1 - v_{mj}) (\log(1 - \rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})) \big] \big) \Big).
\end{aligned}$$

This function indicates how necessary the two sets of latent variables are: z_{im} indicates instance i's membership of component m, whereas v_{mj} indicates feature j's relevance to a component m. Obviously, the problem again is that these variables are not available and must be estimated. For this reason, and as in Eq. (3), the expectation of the complete-data log-likelihood function must be calculated in the E-step of the EM algorithm, taking into account the two sets of latent variables $\mathcal{Z}$ and $\mathcal{V}$. Hence, using

$$\begin{aligned}
&\mathbb{E}_{v_{mj}, |\mathbf{x}_{ij}, \boldsymbol{\theta}_{mj}} [v_{mj}] = \gamma(v_{mj}) \\
&= \frac{\rho_{mj}\, p(x_{ij} \mid \theta_{mj})}{\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj})\, p(x_{ij} \mid \lambda_{mj})} \\
&= p(v_{mj} = 1 \mid x_{ij}, \theta_{mj}),
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{E}_{z_{im}|\mathbf{v}_m,\mathbf{x}_i,\boldsymbol{\theta}_m}[z_{im}] = \gamma(z_{im}) \\
&= \frac{\pi_m \prod_{j=1}^{F}[\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj})]}{\sum_{m'=1}^{K} \pi_{m'} \prod_{j=1}^{F}[\rho_{m'j}\, p(x_{ij} \mid \theta_{m'j}) + (1-\rho_{m'j})\, p(x_{ij} \mid \lambda_{m'j})]} \\
&= p(z_{im} = 1 \mid \mathbf{v}_m, \mathbf{x}_i, \boldsymbol{\theta}_m),
\end{aligned}$$

defining

$$\begin{aligned}
\gamma(u_{imj}) &= \gamma(z_{im})\gamma(v_{mj}), \\
\gamma(w_{imj}) &= \gamma(z_{im})(1-\gamma(v_{mj})),
\end{aligned}$$

and separating each parameter into different addends, the expectation of the new complete-data log likelihood function is (see details in Appendix 3)

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z},\mathcal{V}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V})] \\
&= \sum_{i=1}^{N}\sum_{m=1}^{K} \gamma(z_{im}) \log \pi_m \\
&\quad + \sum_{i=1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(u_{imj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + \sum_{i=1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(w_{imj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})).
\end{aligned} \tag{4}$$

Before detailing how to obtain the updated parameters of SeSProC in the M-step, the expectation of the complete-data log-likelihood function is adapted to introduce the available instance label information.

3.5 Adding instance label information

SeSProC uses the available instance label information to guide the clustering of the unlabeled instances. Based on this information, the model learning process can be divided into two learning parts: the labeled instances are correctly classified into known classes $\{1, \ldots, C\}$ (classification term) and the unlabeled instances can be grouped either in those known or in other unknown components $\{C+1, \ldots, K\}$ (clustering term). Thus, the expectation of the log-likelihood presented in Eq. (4), separating the two learning steps, is

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z},\mathcal{V}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V})] \\
&\quad = \mathbb{E}_{\mathcal{Z}^L,\mathcal{V}|\mathcal{X}^L,\Theta^{t-1}}[\log L_1(\Theta \mid \mathcal{X}^L, \mathcal{Z}^L, \mathcal{V})] \\
&\qquad + \mathbb{E}_{\mathcal{Z}^U,\mathcal{V}|\mathcal{X}^U,\Theta^{t-1}}[\log L_2(\Theta \mid \mathcal{X}^U, \mathcal{Z}^U, \mathcal{V})],
\end{aligned}$$

where the expectation corresponding to the classification term is

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z}^L,\mathcal{V}|\mathcal{X}^L,\Theta^{t-1}}[\log L_1(\Theta \mid \mathcal{X}^L, \mathcal{Z}^L, \mathcal{V})] \\
&= \sum_{i=1}^{L}\sum_{m=1}^{C} z_{im} \log \pi_m \\
&\quad + \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(u_{imj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(w_{imj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})).
\end{aligned} \tag{5}$$

Note that when z_{im} is known, $\gamma(u_{imj})$ and $\gamma(w_{imj})$ are obtained using the value of z_{im} instead of $\gamma(z_{im})$. The expectation related to the clustering term is

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z}^U,\mathcal{V}|\mathcal{X}^U,\Theta^{t-1}}[\log L_2(\Theta \mid \mathcal{X}^U, \mathcal{Z}^U, \mathcal{V})] \\
&= \sum_{i=L+1}^{N}\sum_{m=1}^{K} \gamma(z_{im}) \log \pi_m \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(u_{imj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(w_{imj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})).
\end{aligned} \tag{6}$$

Another feature of SeSProC is the procedure for estimating the final number of clusters, which is described next.

3.6 Selecting the final number of clusters

The number of components of the finite mixture is an unknown parameter that must be estimated. A top-down approach is usually taken to estimate this parameter, i.e., establishing a maximum number of components, K_{max}, and iteratively deleting one component in each step, depending on a quality measure, as the (regularized) log-likelihood function.

As we have knowledge about some instance labels, we know that C is the minimum number of mixture components (assuming one component per each known class). From this number of components we propose a bottom-up approach (see Algorithm 1) to estimate the final number of clusters using a greedy forward search. This begins at an initial level ($l = 0$), where a model $\mathcal{M}^0$ is built using a finite mixture with C components, each in a different feature subspace. Then, in $l = 1$, a new model $\mathcal{M}^1$ with $C + 1$ components is built. $\mathcal{M}^1$ tries to find a new component in a new feature subspace. This model is evaluated and compared with $\mathcal{M}^0$. If $\mathcal{M}^1$ is better than $\mathcal{M}^0$,

then a new component is added to the group of known components, and we have $C+1$ known components to begin the next level. The convergence criterion is reached when $\mathcal{M}^l$ is better than $\mathcal{M}^{l+1}$, returning the $C+l$ known components as the clustering solution. Note that the labeled instances can only belong to the C known components, whereas the unlabeled instances can belong to any component $C+l$ in a level l.

Algorithm 1 General code for model order selection. l is the level of the algorithm, K is the number of components of a model, C is the number of known classes at the beginning of the execution, BIC_{old} and BIC_{new} are the evaluation values of a model obtained with BIC. A model is represented by $\mathcal{M}^l$.

```
l = 0
K = C
Build M^l
BIC_old, BIC_new = Evaluation of M^l
repeat
  BIC_old = BIC_new
  l = l + 1
  K = K + 1
  Initialize K
  Build M^l
  BIC_new = Evaluation of M^l
until BIC_old ≥ BIC_new
```

Two models from consecutive levels and with a different number of components are compared by penalizing the log-likelihood with a term related to model complexity. We use the Schwartz criterion (Schwarz 1978), also known as Bayesian information criterion (BIC). This term must be minimized. If R is the number of free parameters of the model and N is the number of instances, its definition is

$$BIC = -2\log L + R\log N.$$

At level 0, the parameters of the C known components must be initialized. This is carried out using only the labeled instances. On the other hand, the initialization of the new component of each $\mathcal{M}^l$, with $l \geq 1$, must be explained in detail, because it is performed using those instances that worse fit the components of level $l-1$. We assume that the instances that worse fit the known components of our model are candidates for belonging to another component. Based on this assumption we rank the unlabeled instances, taking into account the sum of all the membership values of the $C+(l-1)$ components. This is calculated, from Eq. (15) and for an instance i, as

$$\sum_{m=1}^{C+(l-1)} \pi_m \prod_{j=1}^{F} \left(\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj})\right).$$

The top ranked instances are the ones that worse fit the known components. Then, starting with the top-ranked instance, Cth (*candidates threshold*) instances are selected to initialize the new subspace of the next level.

Finally, to generalize for a level l of SeSProC, Eqs. (5) and (6), for $\mathcal{M}^l$, are updated as

$$\begin{aligned}
&\mathbb{E}^l_{\mathcal{Z}^L,\mathcal{V}|\mathcal{X}^L,\Theta^{t-1}}[\log L_1(\Theta \mid \mathcal{X}^L, \mathcal{Z}^L, \mathcal{V})] \\
&= \sum_{i=1}^{L}\sum_{m=1}^{C} z_{im} \log \pi_m \\
&\quad + \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(u_{imj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + \sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F} \gamma(w_{imj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})),
\end{aligned} \tag{7}$$

and

$$\begin{aligned}
&\mathbb{E}^l_{\mathcal{Z}^U,\mathcal{V}|\mathcal{X}^U,\Theta^{t-1}}[\log L_2(\Theta \mid \mathcal{X}^U, \mathcal{Z}^U, \mathcal{V})] \\
&= \sum_{i=L+1}^{N}\sum_{m=1}^{C+l} \gamma(z_{im}) \log \pi_m \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{C+l}\sum_{j=1}^{F} \gamma(u_{imj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + \sum_{i=L+1}^{N}\sum_{m=1}^{C+l}\sum_{j=1}^{F} \gamma(w_{imj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})),
\end{aligned} \tag{8}$$

respectively. The E and the M steps of the EM algorithm are detailed next. For simplicity's sake, it is assumed that $l = 1$ in the following.

3.7 E-step

The aim in the E-step of the EM algorithm is to calculate the expectation of the complete-data log-likelihood function, having fixed the parameters. These parameters must be initialized before the first iteration of the EM algorithm, whereas they are recalculated in the M-step for subsequent iterations.

The value of the expected complete-data log-likelihood function for $\mathcal{M}^1$ is the sum of Eqs. (7) and (8). The expectations of the latent variables, $\gamma(z_{im})$ and $\gamma(v_{mj}$, can be found in Appendix 3.

3.8 M-step

The parameters are recalculated in the M-step to maximize the value of the expectation of the complete-data log-likelihood function. As already mentioned, these updates are

obtained by computing the partial derivatives of this expectation and equaling to zero. The parameter updates for the Gaussian case are (see details in Appendix 4)

$$\pi_m = \frac{\sum_{i=1}^{L} z_{im} + \sum_{i=L+1}^{N} \gamma(z_{im})}{N},$$
$$\mu_{\theta_{mj}} = \frac{\sum_{i=1}^{N} \gamma(u_{imj}) x_{ij}}{\sum_{i=1}^{N} \gamma(u_{imj})},$$
$$\sigma^2_{\theta_{mj}} = \frac{\sum_{i=1}^{N} \gamma(u_{imj})(x_{ij} - \mu_{\theta_{mj}})^2}{\sum_{i=1}^{N} \gamma(u_{imj})}.$$

The same development is valid for $\lambda_{mj} = (\mu_{\lambda_{mj}}, \sigma^2_{\lambda_{mj}})$ but using $\gamma(w_{imj})$ instead of $\gamma(u_{imj})$ to indicate that feature j is irrelevant for component m:

$$\mu_{\lambda_{mj}} = \frac{\sum_{i=1}^{N} \gamma(w_{imj}) x_{ij}}{\sum_{i=1}^{N} \gamma(w_{imj})},$$
$$\sigma^2_{\lambda_{mj}} = \frac{\sum_{i=1}^{N} \gamma(w_{imj})(x_{ij} - \mu_{\lambda_{mj}})^2}{\sum_{i=1}^{N} \gamma(w_{imj})},$$
$$\rho_{mj} = \frac{\sum_{i=1}^{N} \gamma(u_{imj})}{\sum_{i=1}^{L} z_{im} + \sum_{i=L+1}^{N} \gamma(z_{im})},$$

$\forall m = 1, \ldots, C+1$; $j = 1, \ldots, F$, for all the equations.

4 Experimental results

4.1 Experiments and evaluation

We have used two experimental setups. The first evaluates the proposed algorithm with different configurations (created by changing the value of Cth) and data conditions: using different data distributions, different cluster overlap levels, or clusters with a different number of dimensions. Synthetic data are used for the first set of experiments (see Sect. 4.2). These data have been randomly generated using the Weka data generator tool (Witten et al. 2011). The specific characteristics of each data set are detailed in each experiment.

The second setup compares SeSProC with three related baseline algorithms: Mclust (Fraley and Raftery 2012), which is an implementation of model-based clustering, and HARP and SSPC (see Sect. 2.2), which are algorithms based on subspace clustering without and with known labeled instances, respectively. We have used synthetic and known real data sets with different levels of noisy injections. For details, see Sect. 4.3.

All the used data are fully labeled. However, as the input of SeSProC is expected to be a partially labeled data set, some of the labels will be hidden to the algorithm in each experiment. Instances that retain their original labels (OL) are randomly selected

and, on this ground, each scenario is executed five times (for five random selections). Only the unknown labels will be used as ground truth to evaluate the output clusters.

SeSProC is based on soft clustering, but for the purposes of validation and comparison, we translate the output of the algorithm using the group with highest posterior probability in each case, to assign only one cluster to each instance (hard clustering). We use the Adjusted Rand Index (ARI) (Hubert and Arabie 1985) to compare the post-processed output (PO) with the OL. This measure of similarity between two partitions can be defined as

$$\text{ARI} = \frac{\binom{N}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{N}{2}^2 - [(a+b)(a+c)+(c+d)(b+d)]}, \tag{9}$$

where N is the number of instances, a is the number of pairs of instances located in the same group in PO and in OL, b is the number of pairs of instances in the same group in PO but not in OL, c is the number of pairs of instances in the same group in OL but not in PO, and d is the number of pairs of instances in different groups in both partitions PO and OL. A perfect match between two partitions receives an ARI of 1.

4.2 Synthetic scenarios

Different scenarios have been created to simulate different configurations and data conditions. These scenarios cover many situations that can arise when using our algorithm. An important characteristic for the data sets is the cluster overlap level. We specify the degree of interaction between mixture components using an overlap measure introduced in Melnykov and Maitra (2010). The average cluster overlap level among clusters of a data set ($\hat{\omega}$) is calculated as the average value of the overlap level between each pair of components, where the overlap level between two components i and j is calculated as $\omega_{ij} = \omega_{i|j} + \omega_{i|j}$, $\omega_{i|j}$ being the misclassification probability of an instance originated from the ith component but assigned to the jth component. An implementation of this measure can be found in the R-project (R Core Team 2012) `MixSim` package (Melnykov et al. 2012). For more details, see Melnykov and Maitra (2010).

4.2.1 Candidates threshold

Cth is, as indicated in Sect. 3.6, the only parameter that must be fixed beforehand in SeSProC. Figure 2 shows the results of fixing different values for Cth with two synthetic data sets. Both synthetic data sets were composed of 100 instances in four balanced clusters and described by eight features. The Gaussian distribution was used to generate the data. The differences between the two data sets were the number of relevant features for each cluster and the cluster overlap level: each cluster in the first data set (*cth1*) was described by four features with $\hat{\omega} = 0.15$; the second data set (*cth2*) contained two clusters described by four features, and other two clusters described by only two features, all of them with $\hat{\omega} = 0.03$. We fixed 15 % of labeled instances for this experiment.

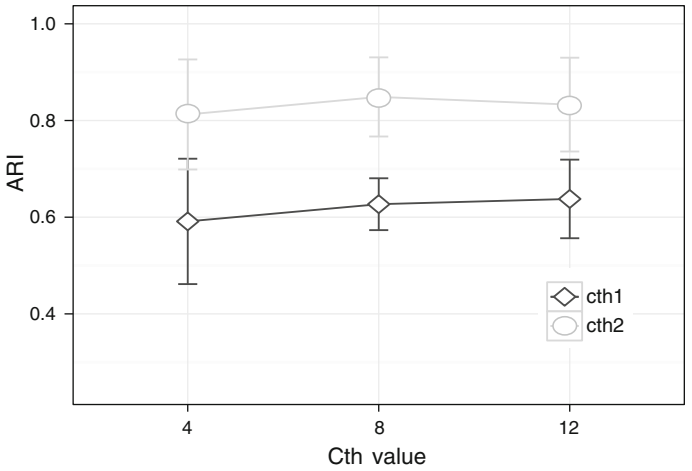

Fig. 2 ARI results with different Cth values and data sets

The results illustrated in Fig. 2 yield a higher stability of SeSProC with $Cth = 8$ than with $Cth = 4$ and $Cth = 12$, according to the lower values of the standard deviations. Regarding the average values, the results for $Cth = 8$ were higher than for $Cth = 4$ with both data sets, and also than for $Cth = 12$ with *cth2*. The results for a value of $Cth = 12$ were slightly better than for $Cth = 8$ with only *cth1*. According to these results, we fix $Cth = 8$ for the next experiments. However, the most interesting finding was that the performance of SeSProC was relatively similar regardless of the Cth value, and there were no significant differences. This is one of the main advantages of SeSProC over other related algorithms, which require extensive parameter tuning in order to achieve any, let alone valuable, results.

4.2.2 Different data distributions

This experiment aims at evaluating the behaviour of SeSProC when features are generated using different distributions. We generated four different data sets, each with 100 instances and four balanced clusters. Each dataset contained six relevant features, whereas four irrelevant features were injected to each data set. Each cluster had from two to four relevant features and $\hat{\omega} = 0.02$ for all data sets. The differences between data sets were the distributions used to generate the data. Specifically, we considered Gaussian and uniform distributions. The proportion of labeled instances for this experiment was fixed to 15 %. Details and results for each data set are shown in Table 1.

The main conclusion is that there were no significant differences in the results regardless of the distribution used to generate both relevant and irrelevant features, even when SeSProC is built to model all features following Gaussian distributions. The results show that the similarities between the data sets regarding the number of features and, mainly, the cluster overlap level, are more important for achieving similar results than using different distributions to generate the data sets. The next experiment covers this scenario, showing results for data sets with different overlap levels taking into account different percentages of labeled instances.

Table 1 Details of the distributions used to generate relevant and irrelevant features for the study of different data distributions, together with average (av) and standard deviation (sd) ARI results

Data	Relevant		Irrelevant		ARI
	Gaussian	Uniform	Gaussian	Uniform	av ± sd
dist1	√		√		0.85 ± 0.12
dist2	√			√	0.81 ± 0.11
dist3		√		√	0.84 ± 0.08
dist4		√	√		0.83 ± 0.07

Table 2 Differences between data sets for the overlap level study

Data	F	p (%)	$\hat{\omega}$
over1	12	25	0
over2	10	50	0.005
over3	10	50	0.01
over4	5	75	0.2

Differences are related to the number of features (F), the average percentage of relevant features for each cluster (p), and the overlap level ($\hat{\omega}$)

4.2.3 Overlap level

The cluster overlap level is one of the most important features for determining how hard it is to cluster a data set. We generated four data sets with similar numbers of instances and features, but with different cluster overlap levels. The aim is to study the performance of SeSProC when dealing with data sets that are increasingly harder to cluster. All data sets were composed of 100 instances in four balanced clusters. The differences between data sets are detailed in Table 2.

We use different percentages of labeled instances in order to check the evolution of the performance of SeSProC. Figure 3 shows results depending on the used data set and the percentage of labeled instances. As expected, the ARI value was generally more accurate, the higher the percentage of labeled instances was. These results show the importance of the cluster overlap level for obtaining accurate results. SeSProC classification was near perfect in scenarios with *over1* and *over2*, with average ARI values ranging from 0.95 to 1 from 10 % of labeled instances. However, the results were lower when the overlap level was increased with *over3*, and average ARI values were from around 0.85 to 0.87 from 5 to 20 % of labeled instances. And, predictably, with a higher overlap level with *over4*, the average ARI results only reached around 0.7 on average with 20 % of labeled instances.

4.2.4 Scalability

This experiment aims at checking the behaviour of SeSProC when data sets with different numbers of instances and features are clustered. Six different synthetic data sets were generated, combining 100, 500, and 1,000 instances, with 10 (four irrelevant)

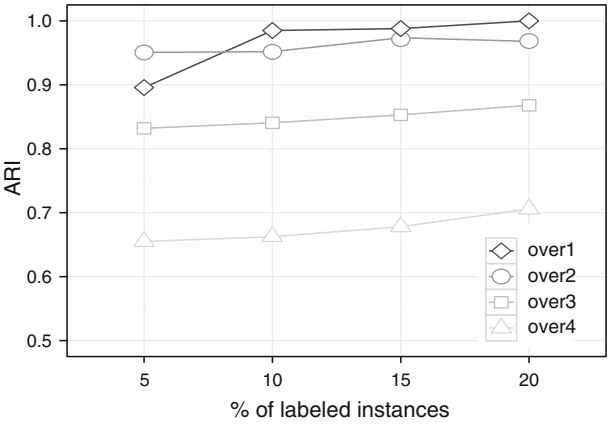

Fig. 3 ARI values depending on the used data set and the percentage of labeled instances for the overlap level study

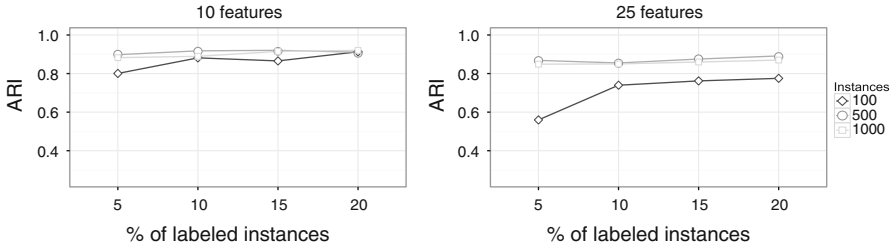

Fig. 4 ARI values depending on the number of features of each data set, percentage of labeled instances, and number of instances

and 25 (seven irrelevant) features. The percentage of relevant features for each cluster was from 30 to 40 %. Data were generated in four balanced clusters with $\hat{\omega} = 0.015$.

Results are presented in Fig. 4 and show that the algorithm behaves well with data sets with 500 and 1,000 instances. Regarding the data set with 100 instances, results depended on the percentage of labeled instances. With the data set with 25 features and only 5 % of labeled instances, the resulting ARI was around 0.55 on average. However, this value rose considerably with a higher percentage of labeled instances, from 10 % onwards, reaching an average of almost 0.8. There was no such rise with the data sets with 500 and 1,000 instances, where the initial result with 5 % of labeled instances was higher. Regarding the number of features, and although results were better for data sets with 10 features than for data sets with 25 features when the number of instances was 100, these differences decreased for data sets with 500 and 1,000 instances. Therefore, the results for these data sets demonstrated the scalability of SeSProC.

SeSProC is a computationally demanding algorithm. EM-based algorithms are known to be computationally expensive (Watanabe and Yamaguchi 2003), and the iterative process used to estimate the relevance of each feature and the final number of clusters further increases this cost in this case. The computational cost for completing one level of SeSProC is detailed in Fig. 5. The experiment was run on a multi-core

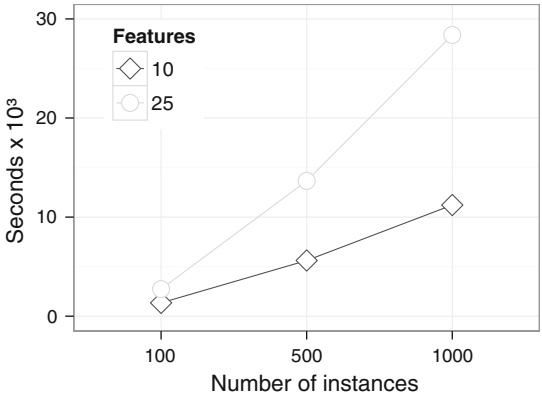

Fig. 5 Average computational time per level of SeSProC depending on the number of instances and features

machine with eight Intel(R) Xeon(R) CPU E5320 @1.86GHz and 12GB of RAM. Values were calculated on average since a level with fewer components takes less time to complete than a level with more components. The difference in computational cost between a level with 10 features and another level with 25 features was very low with 100 instances. However, this difference was very noticeable with 1,000 instances. According to these results, run time depends on the number of both instances and features, and more specifically, on the combination of both.

4.3 Comparison with other algorithms

Mclust, HARP and SSPC are used for comparison purposes. Mclust and HARP are unsupervised algorithms, although one of the input parameters required by HARP is the number of final clusters. Therefore this value was fixed to the correct number of clusters beforehand. We also fixed the correct number of clusters for SSPC, since this algorithm will not work unless the input contains at least one labeled instance from each final cluster. Mclust is freely available from the R-project, whereas the HARP and SSPC source code was provided by the author. We compare these algorithms using synthetic and real data sets.

4.3.1 Synthetic data

We generated a synthetic data set (*hidden*) with the following characteristics: 75 instances in three balanced clusters characterized by eight relevant and two irrelevant features, generated with Gaussian and uniform distributions, respectively. The percentage of relevant features per cluster was 40 % and $\hat{\omega} = 0.02$. The aim is to compare the performance of SeSProC with three related algorithms and also to study the behaviour of our proposal when there were different numbers of hidden clusters in the algorithm input.

Figure 6 shows the results depending on the algorithm and the percentage of labeled instances. Note that all clusters are represented by some labeled instances for SeSProC and SSPC within this scenario. The ARI values for our proposal were higher than for

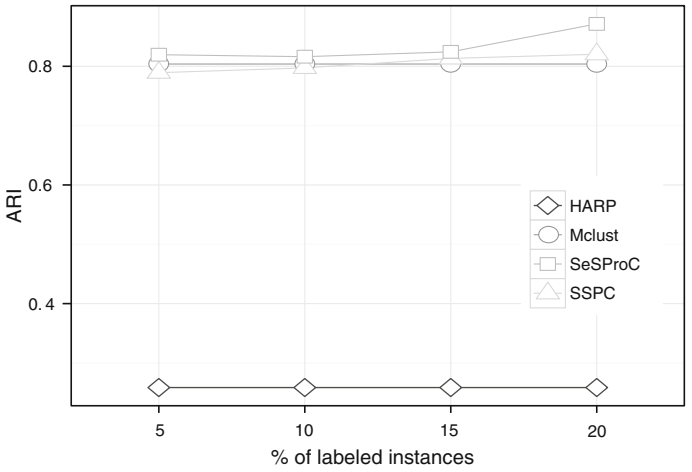

Fig. 6 ARI values depending on the algorithm and the percentage of labeled instances for the comparison study with a synthetic data set

Table 3 ARI results for SeSProC depending on the percentage of labeled instances with different numbers of hidden clusters in the algorithm input (H)

%	$H = 0$	$H = 1$	$H = 2$
5	0.82 ± 0.03	0.77 ± 0.11	0.74 ± 0.14
10	0.82 ± 0.02	0.81 ± 0.02	0.77 ± 0.09
15	0.83 ± 0.00	0.82 ± 0.02	0.81 ± 0.04
20	0.87 ± 0.02	0.84 ± 0.03	0.83 ± 0.03

the other algorithms, and differences between these values became even higher when the percentage of labeled instances reached 20 %, where the value for SeSProC was 0.87 ± 0.02. Values for SSPC were also higher values than for Mclust as of 10 % labeled instances, both semi-supervised algorithms (SeSProC and SSPCC) demonstrating that labeled instances correctly supported the clustering process. However, the results for Mclust were very competitive results even though this is an unsupervised algorithm that does not search for interesting subspaces. HARP was not able to correctly cluster the data set, obtaining very low ARI values.

The next scenario was set up to assess the performance of SeSProC when clusters are hidden in the algorithm input, i.e. when there are no labeled instances of some clusters. The aim is to check whether SeSProC is able to properly find the hidden clusters. Note that this scenario cannot be tested with many other related algorithms, such as SSPC, since the correct number of clusters has to be fixed beforehand with labeled instances from all final clusters.

Results are shown in Table 3. The number of clusters hidden in the input (H) ranged from 0 to 2. The clustering results were very accurate when $H = 0$ as high as 0.87 ± 0.02 with 20 % of labeled instances. These results were previously shown in Fig. 6 demonstrating the competitiveness of SeSProC in this scenario. Besides, the standard deviation values were low (only 0.03 with 5 % of labeled instances), indicating

that the algorithm was robust when dealing with this kind of data. The average ARI values were slightly lower with $H = 1$, whereas the standard deviations were higher, especially with 5 % of labeled instances (0.77 ± 0.11). However, when the percentage of labeled instances reached 15 and 20 %, the differences between results with $H = 0$ and $H = 1$ became reduced. Finally, this tendency was repeated with $H = 2$. This was a more difficult scenario for SeSProC, since the algorithm had to find two hidden clusters, as the values for 5 % of labeled instances show (0.74 ± 0.14). However, it was again demonstrated that when the percentage of labeled instances grew, the ARI values reached very competitive results (0.83 ± 0.03 with 20 % of labeled instances), and there were no significant differences regardless of the value of H.

4.3.2 Real data

Real data sets were collected from the UCI repository (Frank and Asuncion 2010) (see Table 4). These data sets are commonly used for evaluating different pattern recognition tasks. It is interesting to check the behaviour of the proposal with this kind of data where classes are not balanced and subspaces are not their main characteristics. Besides, we introduce some complexity into the data sets by adding five and 15 uniform-distributed features (between 0 and 1) for each data set, simulating some noise for the experiment. In all, we compare results with 12 data sets: four original data sets, four data sets with five noisy features, and four data sets with 15 noisy features.

Figure 7 shows the results for the *glass* data sets. SeSProC obtained the most accurate results with the original data set regardless of the percentage of labeled instances. Moreover, the behaviour of SeSProC was logical in two senses: firstly, the higher the percentage of labeled instances, the higher the resulting ARI value was; and secondly, the ARI value decreased when the number of injected noisy features grew. Regarding the other algorithms, the results for Mclust were very competitive even though the input data was fully unlabeled. This algorithm even obtained the highest ARI values with the *glass* data set with 15 added features. Mclust did not behave logically with respect to the injection of noisy features, since this algorithm obtained better ARI values when the noisy features were added. This algorithm does not take into account any feature subset selection process, and therefore this behaviour was merely motivated by the characteristics of the injected features. Results for HARP and SSPC were not competitive with these data sets. Besides, SSPC did not behave logically with respect to the number of labeled instances, since this algorithm did not always get better results with a higher percentage of labeled instances.

Table 4 Number of instances (N), features (F), and classes (K) of original real data sets used in the experiments

Name	N	F	K
glass	214	9	6
iris	150	4	3
shape	160	17	9
wine	178	13	3

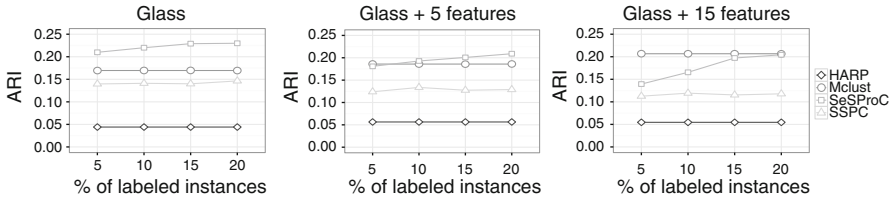

Fig. 7 ARI values for *glass* data sets depending on the number of injected features, algorithm, and percentage of labeled instances

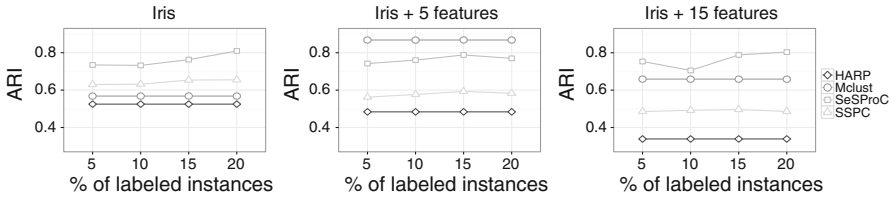

Fig. 8 ARI values for *iris* data sets depending on the number of injected features, algorithm, and percentage of labeled instances

Figure 8 shows the results with the *iris* data sets. The results for these data sets varied enormously depending on whether noisy features were injected. Using the scenario set up with the original *iris* data set SeSProC yielded the highest ARI results and behaved predictably regarding the percentage of labeled instances. SSPC achieved better results than Mclust and HARP. Therefore, the labeled instances correctly supported the semi-supervised algorithms to achieve these results. However, this scenario was totally different when noisy features were added, mainly for the *iris* data set with five added features. Mclust's performance was the best in this comparison with the indicated data set. The added features definitely helped this algorithm to achieve this result. SeSProC was also competitive, whereas SSPC and HARP obtained lower ARI values. Finally, SeSProC again achieved the best results with the *iris* data set with 15 added features, and Mclust obtained better results than for the original *iris* data set, but worse than for the previous scenario.

Figure 9 shows the results for the *shape* data sets. SSPC achieved better results than the other algorithms for these data sets with 5 and 10 % of labeled instances. However, the behaviour of SeSProC improved when the percentage of labeled instances grew and it achieved the best results with 20 % of labeled instances. Note that the *shape* data set contains nine real clusters and that they all have to be represented by some labeled instances in the SSPC input. This is not a necessary condition for SeSProC, and it is the reason for the significant improvement when the percentage of labeled instances grew. Mclust again achieved competitive results, whereas HARP was yet again the worse algorithm.

Figure 10 shows the results with the *wine* data sets. SeSProC achieved very accurate results in all these scenarios, again obtaining the best results compared with the other algorithms. However, there was no improvement in its behaviour with 15 and 20 % over 10% of labeled instances for the *wine* data set with 15 added features. Mclust's behaviour was again illogical regarding the added features, since this algorithm improved

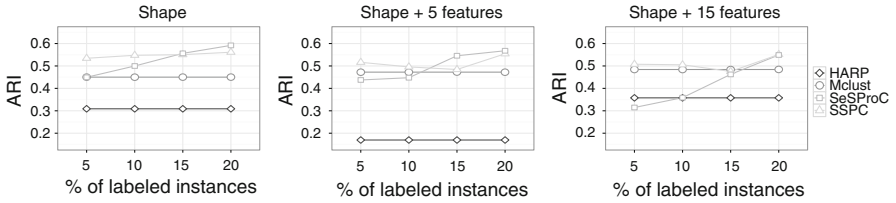

Fig. 9 ARI values for *shape* data sets depending on the number of injected features, algorithm, and percentage of labeled instances

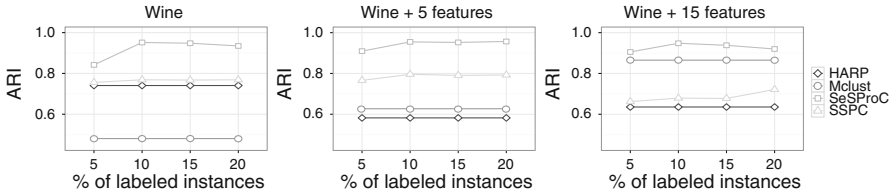

Fig. 10 ARI values for *wine* data sets depending on the number of injected features, algorithm, and percentage of labeled instances

with the injections and even achieved competitive results for the *wine* data set with 15 added features.

In sum, SeSProC was competitive in all scenarios, and mostly achieved better results than the other algorithms. At the other end of the scale, HARP obtained the worst results in most scenarios, even when the correct number of clusters was fixed beforehand. This is also a necessary condition for the execution of SSPC, and this algorithm did not achieve competitive results with the *glass* and *iris* data sets either, although it did yield better results with the *shape* and *wine* data sets. Finally, results for Mclust were very competitive in most scenarios even though this approach is completely unsupervised and does not take into account any feature subset selection process. The odd thing about this algorithm was its illogical behaviour with many data sets with added features.

5 Conclusion

We have proposed a semi-supervised method, called SeSProC, capable of discovering unknown clusters, based on EM algorithm, and including a LFSS. This algorithm includes available information in the search for subspaces and clusters. Besides, SeSProC has two major advantages over related algorithms. The first one is that our proposal has only one, easily adjustable input parameter. Whereas other algorithms are unable to find a final solution without proper parameter tuning, SeSProC always obtains a clustering solution regardless of the value of the input parameter. The second advantage is related to the known labels. SeSProC is able to find hidden clusters that are not represented by the labeled instances. It uses a novel greedy process to find these clusters, assuming that instances that fit the known clusters worst are candidates for initializing new clusters.

The performance of SeSProC has been assessed using synthetic and real data sets under very different scenarios, such as data generated by different distributions, data with different overlap levels, and different data sizes. SeSProC has also outperformed related and baseline algorithms, demonstrating that available data labels are a useful guide for the clustering process.

SeSProC is open to future work related to methodological improvements. The iterative process for introducing new components could be improved. Currently, we consider only one subspace as a new component candidate. We could generate more than one subspace, selecting the best (one or more) to be introduced as new components. This would also be very useful for improving the efficiency of SeSProC in terms of computational cost, since SeSProC is a very demanding algorithm mainly due to its iterative approach. Other changes related to computational cost could be made to the code in order to directly improve efficiency. We also want to improve the search for subspaces in very low dimensional spaces. It can be considered a very difficult search, due to the vast search space when the number of features increases, and the very few relevant features that must be identified. A better initialization regarding the relevance of features can be considered by using, for instance, a statistical test that helps to a priori identify the most relevant features. Other distributions could also be used, for modeling either both relevant and irrelevant features sets or only one of them. This may introduce more flexibility to the model, relaxing some assumptions. SeSProC could be also extended in this direction to discover arbitrarily oriented subspaces.

Regarding the data labels, we have considered them to be fully reliable. Nevertheless, some uncertainty could be introduced to the point that labeled instances would have freedom either to change or maintain their labels. Another characteristic to be introduced is the search for outliers. Some instances may not fit any subspace, and an outlier detection method could improve the results of SeSProC. Finally,

Acknowledgments This research is partially supported by the Spanish Ministry of Economy and Competitiveness TIN2010-20900-C04-04 and TIN2010-21289-C02-02 projects, the Cajal Blue Brain project and Consolider Ingenio 2010-CSD2007-00018. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Centro de Supercomputación y Visualización de Madrid (CeSViMa). The authors are also very grateful for the useful comments and suggestions proposed by the anonymous reviewers, which have contributed definitely to the improvement of the manuscript.

Appendices

Appendix 1: Basic EM theory

The density function of an instance $\mathbf{x}_i$ is

$$p(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m).$$

We can define a binary random variable $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$, with $z_{im} = 1$ if instance $\mathbf{x}_i$ belongs to component m and with all other elements $z_{im'} = 0, \forall\, m' \neq m$. Besides, $p(z_{im} = 1) = \pi_m$. Therefore, we can write

$$p(\mathbf{z}_i) = \prod_{m=1}^{K} \pi_m^{z_{im}}. \tag{10}$$

Also, $p(\mathbf{x}_i \mid z_{im} = 1) = p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)$, which, extended, is

$$p(\mathbf{x}_i \mid \mathbf{z}_i, \Theta) = \prod_{m=1}^{K} p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)^{z_{im}}. \tag{11}$$

Using Eqs. (10) and (11), Eq. (1) is obtained by summing over all possible states of $\mathbf{z}_i$

$$\begin{aligned} p(\mathbf{x}_i \mid \Theta) &= \sum_{\mathbf{z}_i} p(\mathbf{x}_i, \mathbf{z}_i \mid \Theta) = \sum_{\mathbf{z}_i} p(\mathbf{z}_i)\, p(\mathbf{x}_i \mid \mathbf{z}_i, \Theta) \\ &= \sum_{\mathbf{z}_i} \left(\prod_{m=1}^{K} \pi_m^{z_{im}} \prod_{m=1}^{K} p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)^{z_{im}} \right) \\ &= \sum_{m=1}^{K} \pi_m\, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m). \end{aligned}$$

This mixture of distributions has unknown parameters in Θ that must be estimated. These parameters can be obtained using the maximum likelihood estimation method. Therefore, assuming that each instance is independent and identically distributed (i.i.d.), and building the log-likelihood function ($\log L$) from Eq. (1) and extending it to all the instances, we obtain

$$\begin{aligned} \log L(\Theta \mid \mathcal{X}) &= \log p(\mathcal{X}|\Theta) \\ &= \log \prod_{i=1}^{N} p(\mathbf{x}_i \mid \Theta) \\ &= \sum_{i=1}^{N} \log \left(\sum_{m=1}^{K} \pi_m\, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m) \right). \end{aligned}$$

This log-likelihood function is difficult to maximize because the summation over the components is inside the logarithm function. The log-likelihood function would change if both the latent variables ($\mathcal{Z}$) and the observable data ($\mathcal{X}$) were known. Then, based on Eqs. (10) and (11), we can define the complete-data log-likelihood function as

$$\begin{aligned}\log L(\Theta \mid \mathcal{X}, \mathcal{Z}) &= \log p(\mathcal{X}, \mathcal{Z}|\Theta) \\ &= \log \prod_{i=1}^{N} \prod_{m=1}^{K} \pi_m^{z_{im}} p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)^{z_{im}} \\ &= \sum_{i=1}^{N} \sum_{m=1}^{K} z_{im} \left(\log \pi_m + \log p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)\right). \end{aligned} \tag{12}$$

The maximization of this complete-data log-likelihood function is straightforward because the summation is outside the logarithm. Since the latent variables are unknown we cannot use this function directly. However, we can obtain the expectation of this log-likelihood function with respect to the posterior distribution of the latent variables. This expectation is calculated in iteration t, having fixed the parameters from the previous iteration $t-1$, in the E-step of the EM algorithm. After this, the parameters of the distributions are recalculated to maximize this expectation (M-step). These two steps are repeated until a convergence criterion is reached. Hence, the expectation of the complete-data log-likelihood function is given by

$$\begin{aligned}\mathcal{Q}(\Theta, \Theta^{t-1}) &= \mathbb{E}_{\mathcal{Z}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z})] \\ &= \sum_{\mathcal{Z}} p(\mathcal{Z} \mid \mathcal{X}, \Theta^{t-1}) \log p(\mathcal{X}, \mathcal{Z} \mid \Theta), \end{aligned} \tag{13}$$

where the posterior distribution of the latent variables given the data and the parameters of the previous iteration $t-1$ using Eq. (12), is

$$p(\mathcal{Z} \mid \mathcal{X}, \Theta^{t-1}) \propto \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\pi_m \, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)\right)^{z_{im}}. \tag{14}$$

This factorizes over i so that the $\{\mathbf{z}_i\}$ in this distribution are independent. Using this posterior distribution and Bayes' theorem, we can calculate the expected value of each z_{im} (responsibility) as

$$\begin{aligned}\mathbb{E}_{z_{im}|\mathbf{x}_i,\boldsymbol{\theta}_m}[z_{im}] &= \gamma(z_{im}) \\ &= \frac{\sum_{z_{im}} z_{im} (\pi_m \, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m))^{z_{im}}}{\sum_{z_{im'}} (\pi_{m'} \, p(\mathbf{x}_i \mid \boldsymbol{\theta}_{m'}))^{z_{im'}}} \\ &= \frac{\pi_m \, p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)}{\sum_{m'=1}^{K} \pi_{m'} \, p(\mathbf{x}_i \mid \boldsymbol{\theta}_{m'})} \\ &= p(z_{im} = 1 \mid \mathbf{x}_i, \boldsymbol{\theta}_m), \end{aligned}$$

which we can use to calculate the expectation of the complete-data log-likelihood, as

$$\mathbb{E}_{\mathcal{Z}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z})] = \sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_{im}) \left(\log \pi_m + \log p(\mathbf{x}_i \mid \boldsymbol{\theta}_m)\right).$$

Appendix 2: Including subspaces

Defining, for each component and feature, $\rho_{mj} = p(v_{mj} = 1)$, the probability of feature j being relevant to component m, the new density function, including the search for subspaces, is

$$p(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj})\, p(x_{ij} \mid \lambda_{mj}) \Big). \tag{15}$$

To prove this new density function, we can obtain for a component m and an instance i,

$$p(\mathbf{v}_m \mid z_{im} = 1) = \prod_{j=1}^{F} (\rho_{mj})^{v_{mj}} (1 - \rho_{mj})^{1 - v_{mj}}.$$

This can be extended for all components as

$$p(\mathcal{V} \mid \mathbf{z}_i) = \prod_{m=1}^{K} \left(\prod_{j=1}^{F} (\rho_{mj})^{v_{mj}} (1 - \rho_{mj})^{1 - v_{mj}} \right)^{z_{im}}. \tag{16}$$

Besides, we can extend Eq. (11) introducing $\mathcal{V}$, as

$$p(\mathbf{x}_i \mid \mathcal{V}, \mathbf{z}_i, \Theta) = \prod_{m=1}^{K} \left(\prod_{j=1}^{F} p(x_{ij} \mid \theta_{mj})^{v_{mj}}\, p(x_{ij} \mid \lambda_{mj})^{1 - v_{mj}} \right)^{z_{im}}. \tag{17}$$

The new density function, based on Eq. (1), and using Eqs. (10), (16), and (17), is,

$$\begin{aligned} p(\mathbf{x}_i \mid \Theta) &= \sum_{\mathbf{z}_i} \sum_{\mathcal{V}} p(\mathbf{x}_i, \mathcal{V}, \mathbf{z}_i \mid \Theta) \\ &= \sum_{\mathbf{z}_i} \sum_{\mathcal{V}} p(\mathbf{x}_i \mid \mathcal{V}, \mathbf{z}_i, \Theta)\, p(\mathcal{V} \mid \mathbf{z}_i)\, p(\mathbf{z}_i). \end{aligned}$$

The summation over $\mathbf{z}_i$ is solved as in Eq. (1), obtaining,

$$p(\mathbf{x}_i \mid \Theta) = \sum_{\mathcal{V}} \left(\sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big([\rho_{mj}\, p(x_{ij} \mid \theta_{mj})]^{v_{mj}} \times [(1 - \rho_{mj})\, p(x_{ij} \mid \lambda_{mj})]^{1 - v_{mj}} \Big) \right).$$

And we can solve the summation over $\mathcal{V}$, summing over all the possible states of each v_{mj}, as,

$$p(\mathbf{x}_i \mid \Theta) = \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \sum_{v_{mj}=0}^{1} \Big([\rho_{mj}\, p(x_{ij} \mid \theta_{mj})]^{v_{mj}} \times [(1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj})]^{1-v_{mj}} \Big)$$
$$= \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj}) \Big).$$

Taking into account that each component can be described in a different feature subspace, this is the new density function of an instance, as shown in Eq. (15).

The new log-likelihood function that should be maximized, by extending Eq. (15) to all the instances, is

$$\log L(\Theta \mid \mathcal{X}) = \log p(\mathcal{X}|\Theta) = \log \prod_{i=1}^{N} p(\mathbf{x}_i \mid \Theta)$$
$$= \sum_{i=1}^{N} \Big(\log \sum_{m=1}^{K} \pi_m \prod_{j=1}^{F} \Big(\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj}) \Big) \Big).$$

This is again difficult to compute since the summation over the components is inside the logarithm function. This equation would change if we knew the sets of latent variables, $\mathcal{Z}$ and $\mathcal{V}$. Again by extending Eqs. (10), (16), and (17) to all the data, we can write

$$p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta) = \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\prod_{j=1}^{F} p(x_{ij} \mid \theta_{mj})^{v_{mj}}\, p(x_{ij} \mid \lambda_{mj})^{1-v_{mj}} \right)^{z_{im}}$$
$$\times \prod_{i=1}^{N} \prod_{m=1}^{K} \left(\prod_{j=1}^{F} (\rho_{mj})^{v_{mj}} (1-\rho_{mj})^{1-v_{mj}} \right)^{z_{im}}$$
$$\times \prod_{i=1}^{N} \prod_{m=1}^{K} \pi_m^{z_{im}},$$

which can be simplified to

$$p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta) = \prod_{i=1}^{N} \prod_{m=1}^{K} \Big(\pi_m^{z_{im}} \prod_{j=1}^{F} \big([\rho_{mj}\, p(x_{ij} \mid \theta_{mj})]^{v_{mj}}$$
$$\times [(1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj})]^{1-v_{mj}} \big)^{z_{im}} \Big). \tag{18}$$

We can obtain the complete-data log-likelihood function by taking the logarithm of the previous function as,

$$\begin{aligned}\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V}) &= \log p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta) \\ &= \log \prod_{i=1}^{N} \prod_{m=1}^{K} \Bigg(\pi_m^{z_{im}} \prod_{j=1}^{F} \left([\rho_{mj}\, p(x_{ij} \mid \theta_{mj})]^{v_{mj}} \right. \\ &\quad \left. \times [(1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj})]^{1-v_{mj}} \right)^{z_{im}} \Bigg),\end{aligned}$$

and operating again,

$$\begin{aligned}&\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V}) \\ &= \sum_{i=1}^{N} \sum_{m=1}^{K} \Bigg(z_{im} \log \pi_m + \sum_{j=1}^{F} \Big(z_{im} \Big[v_{mj} (\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\ &\quad + (1 - v_{mj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})) \Big] \Big) \Bigg). \qquad (19)\end{aligned}$$

Appendix 3: Expectation of the complete-data log-likelihood function

Similarly to Eq. (13), the expectation of the complete-data log-likelihood function can be written as

$$\mathbb{E}_{\mathcal{Z},\mathcal{V}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V})] = \sum_{\mathcal{Z}} \sum_{\mathcal{V}} p(\mathcal{Z}, \mathcal{V} \mid \mathcal{X}, \Theta^{t-1}) \log p(\mathcal{X}, \mathcal{Z}, \mathcal{V} \mid \Theta).$$

As in Eq. (14), the posterior distribution of the latent variables given the data, having fixed the parameters of the previous iteration $t-1$, and using Eq. (18), can be written as

$$\begin{aligned}p(\mathcal{Z}, \mathcal{V} \mid \mathcal{X}, \Theta^{t-1}) \propto \prod_{i=1}^{N} \prod_{m=1}^{K} \Bigg(\pi_m^{z_{im}} \prod_{j=1}^{F} \left([\rho_{mj}\, p(x_{ij} \mid \theta_{mj})]^{v_{mj}} \right. \\ \left. \times [(1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj})]^{1-v_{mj}} \right)^{z_{im}} \Bigg).\end{aligned}$$

Before computing the expected values of each v_{mj} and each z_{im}, we need to define some other necessary probabilities:

$$p(x_{ij}, v_{mj} = 1 \mid \theta_{mj}) = \rho_{mj}\, p(x_{ij} \mid \theta_{mj}),$$

and, similarly

$$p(x_{ij}, v_{mj} = 0 \mid \theta_{mj}) = (1-\rho_{mj})\, p(x_{ij} \mid \lambda_{mj}).$$

Taking both expressions into account, we have

$$\begin{aligned} p(x_{ij} \mid \theta_{mj}) &= p(x_{ij}, v_{mj} = 1 \mid \theta_{mj}) + p(x_{ij}, v_{mj} = 0 \mid \theta_{mj}) \\ &= \rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj}) p(x_{ij} \mid \lambda_{mj}). \end{aligned}$$

Now, as detailed after Eq. (14), we can calculate the expected value of each v_{mj}, as

$$\begin{aligned} &\mathbb{E}_{v_{mj}, \mid \mathbf{x}_{ij}, \boldsymbol{\theta}_{mj}}[v_{mj}] = \gamma(v_{mj}) \\ &= \frac{\rho_{mj}\, p(x_{ij} \mid \theta_{mj})}{\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj}) p(x_{ij} \mid \lambda_{mj})} \\ &= p(v_{mj} = 1 \mid x_{ij}, \theta_{mj}). \end{aligned}$$

Using this, we calculate the expected value of each z_{im}

$$\begin{aligned} &\mathbb{E}_{z_{im} \mid \mathbf{v}_m, \mathbf{x}_i, \boldsymbol{\theta}_m}[z_{im}] = \gamma(z_{im}) \\ &= \frac{\pi_m \prod_{j=1}^{F} [\rho_{mj}\, p(x_{ij} \mid \theta_{mj}) + (1 - \rho_{mj}) p(x_{ij} \mid \lambda_{mj})]}{\sum_{m'=1}^{K} \pi_{m'} \prod_{j=1}^{F} [\rho_{m'j}\, p(x_{ij} \mid \theta_{m'j}) + (1 - \rho_{m'j}) p(x_{ij} \mid \lambda_{m'j})]} \\ &= p(z_{im} = 1 \mid \mathbf{v}_m, \mathbf{x}_i, \boldsymbol{\theta}_m). \end{aligned}$$

Thus the expectation of the complete-data log-likelihood, as in Eq. (3) and using Eq. (19), is

$$\begin{aligned} &\mathbb{E}_{\mathcal{Z}, \mathcal{V} \mid \mathcal{X}, \Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V})] \\ &= \sum_{i=1}^{N} \sum_{m=1}^{K} \gamma(z_{im}) \\ &\quad \times \Bigg(\log \pi_m + \sum_{j=1}^{F} \Big(\gamma(v_{mj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\ &\quad + (1 - \gamma(v_{mj}))(\log(1 - \rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})) \Big) \Bigg). \end{aligned}$$

Then, for simplicity's sake, we define

$$\begin{aligned} \gamma(u_{imj}) &= \gamma(z_{im}) \gamma(v_{mj}), \\ \gamma(w_{imj}) &= \gamma(z_{im})(1 - \gamma(v_{mj})). \end{aligned}$$

Now we can obtain the expectation of the complete-data log-likelihood as

$$\begin{aligned}
&\mathbb{E}_{\mathcal{Z},\mathcal{V}|\mathcal{X},\Theta^{t-1}}[\log L(\Theta \mid \mathcal{X}, \mathcal{Z}, \mathcal{V})] \\
&= \sum_{i=1}^{N}\sum_{m=1}^{K} \gamma(z_{im}) \log \pi_m \\
&\quad + \sum_{i=1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(u_{imj})(\log \rho_{mj} + \log p(x_{ij} \mid \theta_{mj})) \\
&\quad + \sum_{i=1}^{N}\sum_{m=1}^{K}\sum_{j=1}^{F} \gamma(w_{imj})(\log(1-\rho_{mj}) + \log p(x_{ij} \mid \lambda_{mj})).
\end{aligned}$$

Appendix 4: M-step

The parameters are recalculated in the M-step to maximize the value of the expectation of the complete-data log-likelihood function. As already mentioned, these updates are obtained by computing the partial derivatives of this expectation and equaling to zero. The univariate Gaussian distribution for each feature and component is used for this explanation. Therefore $\theta_{mj} = (\mu_{\theta_{mj}}, \sigma^2_{\theta_{mj}})$, and

$$\log p(x_{ij} \mid \theta_{mj}) = \log(\sigma^{-1}_{\theta_{mj}}(2\pi)^{-\frac{1}{2}}) - \frac{1}{2}(x_{ij} - \mu_{\theta_{mj}})^2\sigma^{-2}_{\theta_{mj}}.$$

All the detailed steps of how to update each parameter, follow.

– π_m is updated[3] using a Lagrange multiplier to enforce constraint $\sum_{m=1}^{C+1} \pi_m = 1$:

$$\begin{aligned}
\frac{\partial}{\partial \pi_m}\Bigg(&\sum_{i=1}^{L}\sum_{m=1}^{C} z_{im} \log \pi_m \\
&+ \sum_{i=L+1}^{N}\sum_{m=1}^{C+1} \gamma(z_{im}) \log \pi_m \\
&+ \lambda\left(\sum_{m=1}^{C+1} \pi_m - 1\right)\Bigg) = 0, \quad \forall m = 1, \ldots, C+1,
\end{aligned}$$

whose derivative is

$$\sum_{i=1}^{L} z_{im}\frac{1}{\pi_m} + \sum_{i=L+1}^{N} \gamma(z_{im})\frac{1}{\pi_m} + \lambda = 0.$$

[3] Note that the classification term only iterates theoretically until $m = C$, but we can assume that this iteration finishes at $m = C + 1$ with $z_{i,C+1} = 0, \forall i = 1, \ldots, L$.

Multiplying both sides by π_m and summing over m, with $m = 1, \ldots, C+1$, we have $\lambda = -N$, as

$$-\lambda = \sum_{i=1}^{L}\sum_{m=1}^{C+1} z_{im} + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1} \gamma(z_{im}) = N,$$

and then we update each π_m by using

$$\pi_m = \frac{\sum_{i=1}^{L} z_{im} + \sum_{i=L+1}^{N} \gamma(z_{im})}{N}.$$

– $\mu_{\theta_{mj}}$ is updated solving the following partial derivative equation:

$$\frac{\partial}{\partial \mu_{\theta_{mj}}}\Bigg(\sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F}\gamma(u_{imj})\log p(x_{ij} \mid \theta_{mj}) + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\sum_{j=1}^{F}\gamma(u_{imj})\log p(x_{ij} \mid \theta_{mj})\Bigg) = 0.$$

Then the result is

$$\sum_{i=1}^{L}\Big(\gamma(u_{imj})\sigma_{\theta_{mj}}^{-2}x_{ij} - \gamma(u_{imj})\sigma_{\theta_{mj}}^{-2}\mu_{\theta_{mj}}\Big) + \sum_{i=L+1}^{N}\Big(\gamma(u_{imj})\sigma_{\theta_{mj}}^{-2}x_{ij} - \gamma(u_{imj})\sigma_{\theta_{mj}}^{-2}\mu_{\theta_{mj}}\Big) = 0,$$

and the value of the parameter can be found as

$$\begin{aligned}\mu_{\theta_{mj}} &= \frac{\sum_{i=1}^{L}\gamma(u_{imj})x_{ij} + \sum_{i=L+1}^{N}\gamma(u_{imj})x_{ij}}{\sum_{i=1}^{L}\gamma(u_{imj}) + \sum_{i=L+1}^{N}\gamma(u_{imj})} \\ &= \frac{\sum_{i=1}^{N}\gamma(u_{imj})x_{ij}}{\sum_{i=1}^{N}\gamma(u_{imj})}, \quad \forall m = 1, \ldots, C+1;\ j = 1, \ldots, F.\end{aligned}$$

– And for $\sigma_{\theta_{mj}}^2$,

$$\frac{\partial}{\partial \sigma_{\theta_{mj}}^2}\Bigg(\sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F}\gamma(u_{imj})\log p(x_{ij} \mid \theta_{mj}) + \sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\sum_{j=1}^{F}\gamma(u_{imj})\log p(x_{ij} \mid \theta_{mj})\Bigg) = 0.$$

The derivative is

$$\sum_{i=1}^{L}\left(\gamma(u_{imj})(x_{ij}-\mu_{mj})^2\sigma_{\theta_{mj}}^{-4}-\gamma(u_{imj})\sigma_{\theta_{mj}}^{-2}\right)$$
$$+\sum_{i=L+1}^{N}\left(\gamma(u_{imj})(x_{ij}-\mu_{mj})^2\sigma_{\theta_{mj}}^{-4}-\gamma(u_{imj})\sigma_{\theta_{mj}}^{-2}\right)=0,$$

and the parameter update is

$$\sigma^2_{\theta_{mj}}=\frac{\sum_{i=1}^{L}\gamma(u_{imj})(x_{ij}-\mu_{\theta_{mj}})^2+\sum_{i=L+1}^{N}\gamma(u_{imj})(x_{ij}-\mu_{\theta_{mj}})^2}{\sum_{i=1}^{L}\gamma(u_{imj})+\sum_{i=L+1}^{N}\gamma(u_{imj})}$$
$$=\frac{\sum_{i=1}^{N}\gamma(u_{imj})(x_{ij}-\mu_{\theta_{mj}})^2}{\sum_{i=1}^{N}\gamma(u_{imj})},\quad\forall m=1,\ldots,C+1;\ j=1,\ldots,F.$$

The same development is valid for $\lambda_{mj}=(\mu_{\lambda_{mj}},\sigma^2_{\lambda_{mj}})$ but using $\gamma(w_{imj})$ instead of $\gamma(u_{imj})$ to indicate that feature j is irrelevant for component m.

– Then, we update $\mu_{\lambda_{mj}}$ as

$$\mu_{\lambda_{mj}}=\frac{\sum_{i=1}^{N}\gamma(w_{imj})x_{ij}}{\sum_{i=1}^{N}\gamma(w_{imj})},\quad\forall m=1,\ldots,C+1;\ j=1,\ldots,F.$$

– And for $\sigma^2_{\lambda_{mj}}$

$$\sigma^2_{\lambda_{mj}}=\frac{\sum_{i=1}^{N}\gamma(w_{imj})(x_{ij}-\mu_{\lambda_{mj}})^2}{\sum_{i=1}^{N}\gamma(w_{imj})},\quad\forall m=1,\ldots,C+1;\ j=1,\ldots,F.$$

– Finally in $\mathcal{M}^1$, ρ_{mj} is updated by

$$\frac{\partial}{\partial\rho_{mj}}\Bigg(\sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F}\gamma(u_{imj})\log\rho_{mj}$$
$$+\sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\sum_{j=1}^{F}\gamma(u_{imj})\log\rho_{mj}$$
$$+\sum_{i=1}^{L}\sum_{m=1}^{C}\sum_{j=1}^{F}\gamma(w_{imj})\log(1-\rho_{mj})$$
$$+\sum_{i=L+1}^{N}\sum_{m=1}^{C+1}\sum_{j=1}^{F}\gamma(w_{imj})\log(1-\rho_{mj})\Bigg)=0,$$

whose partial derivative solution is,

$$\sum_{i=1}^{N} \gamma(u_{imj}) \frac{1}{\rho_{mj}} - \sum_{i=1}^{N} \gamma(w_{imj}) \frac{1}{1-\rho_{mj}} = 0.$$

This parameter is updated by

$$\rho_{mj} = \frac{\sum_{i=1}^{N} \gamma(u_{imj})}{\sum_{i=1}^{L} z_{im} + \sum_{i=L+1}^{N} \gamma(z_{im})}, \quad \forall m = 1, \ldots, C+1;\ j = 1, \ldots, F.$$

Note that $z_{i,C+1} = 0$ for $i = 1, \ldots, L$ for the three sets of parameters, θ_{mj}, λ_{mj} and ρ_{mj}.

References

Aggarwal C, Yu P (2000) Finding generalized projected clusters in high dimensional spaces. SIGMOD Rec 29(2):70–81

Aggarwal C, Han J, Wang J, Yu P (2004) A framework for projected clustering of high dimensional data streams. In: Proceedings of 30th international conference on very large data bases, pp 852–863

Aggarwal C, Procopiuc C, Wolf J, Yu P, Park J (1999) Fast algorithms for projected clustering. SIGMOD Rec 28(2):61–72

Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. SIGMOD Rec 27:94–105

Ahmed M, Khan L (2009) SISC: a text classification approach using semi supervised subspace clustering. In: IEEE international conference on data mining workshops, pp 1–6

Alexandridis R, Lin S, Irwin M (2004) Class discovery and classification of tumor samples using mixture modeling of gene expression data, a unified approach. Bioinformatics 20(16):2545–2552

Basu S, Banjeree A, Mooney E, Banerjee A, Mooney R (2004) Active semi-supervision for pairwise constrained clustering. In: Proceedings of the SIAM international conference on data mining, pp 333–344

Basu S, Davidson I, Wagstaff K (eds) (2009) Constrained clustering: advances in algorithms, theory and applications. Chapman and Hall/CRC, Boca Raton

Bishop C (2007) Pattern recognition and machine learning. Springer, New York

Boutemedjet S, Ziou D, Bouguila N (2010) Model based subspace clustering of non-Gaussian data. Neurocomputing 73(10–12):1730–1739

Chandel A, Tiwari A, Chaudhari N (2009) Constructive semi-supervised classification algorithm and its implement in data mining. In: Proceedings of the 3rd international conference on pattern recognition and machine intelligence. Springer, Berlin, pp 62–67

Chapelle O, Schölkopf B, Zien A (eds) (2006) Semi-supervised learning. MIT Press, Cambridge

Chawla N, Karakoulas G (2005) Learning from labeled and unlabeled data: an empirical study across techniques and domains. J Artif Intell Res 23:331–366

Chen L, Jiang Q, Wang S (2012) Model-based method for projective clustering. IEEE Trans Knowl Data Eng 24(7):1291–1305

Cheng C, Fu A, Zhang Y (1999) Entropy-based subspace clustering for mining numerical data. In: Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 84–93

Cheng H, Hua K, Vu K (2008) Constrained locally weighted clustering. In: Proceedings of the 34th international conference on very large data bases, vol 1, Auckland, pp 90–101

Cordeiro R, Traina A, Faloutsos C, Traina C (2010) Finding clusters in subspaces of very large, multi-dimensional datasets. In: International conference on data engineering, Long Beach, pp 625–636

Dempster A, Laird N, Rubin D (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc 39(1):1–38

Fraley C, Raftery A (1998) How many clusters? Which clustering method? Answers via model-based cluster analysis. Comput J 41(8):578–588
Fraley C, Raftery A (2012) MCLUST version 4 for R: normal mixture modeling for model-based clustering, classication and density estimation. Technical report no. 597, Department of Statistics, University of Washington, Seatlle
Frank A, Asuncion A (2010) UCI machine learning repository. http://archive.ics.uci.edu/ml
Friedman J, Meulman J (2004) Clustering objects on subsets of attributes. J R Stat Soc 66:815–849
Fromont E, Prado A, Robardet C (2009) Constraint-based subspace clustering. In: Proceedings of the 9th SIAM international conference on data mining, pp 26–37
Goil S, Nagesh H, Choudhary A (1999) MAFIA: efficient and scalable subspace clustering for very large data sets. In: International conference on data engineering
Graham M, Miller D (2006) Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. IEEE Trans Signal Process 54(4):1289–1303
Günnemann S, Färber I, Müller E, Seidl T (2010) ASCLU: alternative subspace clustering. In: Multiclust: first international workshop on discovering, summarizing and using multiple clustering, held in conjunction with KDD 2010
Günnemann S, Färber I, Virochsiri K, Seidl T (2012) Subspace correlation clustering: finding locally correlated dimensions in subspace projections of the data. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, pp 352–360
Hoff P (2005) Subset clustering of binary sequences, with an application to genomic abnormality data. Biometrics 61(4):1027–1036
Hoff P (2006) Model based subspace clustering. Bayesian. Analysis 1(2):321–344
Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218
Kriegel H, Kröger P, Zimek A (2009) Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering and correlation clustering. ACM Trans Knowl Disc Data 3(1):1–58
Kriegel H, Kröger P, Ntoutsi I, Zimek A (2011) Density based subspace clustering over dynamic data. In: Proceedings of the 23rd international conference on scientific and statistical database management, pp 387–404
Kriegel H, Kröger P, Zimek A (2012) Subspace clustering. Wiley Interdiscip. Rev 2(4):351–364
Lange T, Law M, Jain A, Buhmann J (2005) Learning with constrained and unlabelled data. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, pp 731–738
Law M, Figueiredo M, Jain A (2004) Simultaneous feature selection and clustering using mixture models. IEEE Trans Pattern Anal Mach Intell 26(9):1154–1166
Li Y, Dong M, Hua J (2007) A Gaussian mixture model to detect clusters embedded in feature subspace. J Commun Inf Syst 7(4):337–352
Li Y, Dong M, Hua J (2009) Simultaneous localized feature selection and model detection for Gaussian mixtures. IEEE Trans Pattern Anal Mach Intell 31(5):953–960
Lu Z, Leen T (2005) Semi-supervised learning with penalized probabilistic clustering. Adv Neural Inf Process Syst 17:849–856
Maitra R, Melnykov V (2010) Simulating data to study performance of finite mixture modeling and clustering algorithms. J Computd Graph Stat 19(2):354–376
Markley S, Miller D (2010) Joint parsimonious modeling and model order selection for multivariate Gaussian mixtures. IEEE J Sel Top Signal Process 4(3):548–559
McLachlan G, Basford K (1988) Mixture models: inference and applications to clustering. Marcel Dekker, New York
McLachlan G, Peel D (2000) Finite mixture models. Wiley-Interscience, New York
Melnykov V, Maitra R (2010) Finite mixture models and model-based clustering. Stat Surv 4:80–116
Melnykov V, Chen W, Maitra R (2012) MixSim: an R package for simulating data to study performance of clustering algorithms. J Stat Softw 51(12):1–25
Miller D, Browning J (2003) A mixture model and EM-based algorithm for class discovery, robust classification, and outlier rejection in mixed labeled/unlabeled data sets. IEEE Trans Pattern Anal Mach Intell 25(11):1468–1483
Miller D, Chu-Fang L, Kesidis G, Collins C (2009) Semisupervised mixture modeling with fine-grained component-conditional class labeling and transductive inference. In: IEEE international workshop on machine learning for signal processing, pp 1–6
Moise G, Sander J, Ester M (2008) Robust projected clustering. Knowl Inf Syst 14(3):273–298

Müller E, Assent I, Seidl T (2009) HSM: heterogeneous subspace mining in high dimensional. In: Proceedings of the 21st international conference on scientific and statistical database management, pp 497–516
Parsons L, Haque E, Liu H (2004) Subspace clustering for high dimensional data: a review. ACM SIGKDD Explor Newsl 6(1):90–105
Procopiuc C, Jones M, Agarwal P, Murali T (2002) A Monte Carlo algorithm for fast projective clustering. In: Proceedings of the ACM international conference on management of data, pp 418–427
R Core Team (2012) R: a language and environment for statistical computing. R Foundation for Statisical Computing, Vienna
Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6(2):461–464
Shental N, Bar-Hillel A, Hertz T, Weinshall D (2003) Computing Gaussian mixture models with EM using equivalence constraints. Adv Neural Inf Process Syst 16:1–8
Sim K, Gopalkrishnan V, Zimek A, Cong G (2012) A survey on enhanced subspace clustering. Data Min Knowl Discov. doi:10.1007/s10618-012-0258-x
Wang F, Zhang C, Shen H, Wang J (2006) Semi-supervised classification using linear neighborhood propagation. In: IEEE Computer Society Conference on Computer Vision and. Pattern Recognition, vol 1:160–167
Watanabe M, Yamaguchi K (2003) The EM algorithm and related statistical models. CRC Press, Boca Raton
Witten I, Frank E, Hall M (2011) Data mining: practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann, Burlington
Woo K, Lee J, Kim M, Lee Y (2004) FINDIT: a fast and intelligent subspace clustering algorithm using dimension voting. Inf Softw Technol 46(4):255–271
Yip K, Cheung D, Ng M (2004) HARP: a practical projected clustering algorithm. IEEE Trans Knowl Data Eng 16:1387–1397
Yip K, Cheung D, Ng M (2005) On discovery of extremely low-dimensional clusters using semi-supervised projected clustering. In: International conference on data engineering, pp 329–340
Zhang X, Wu Y, Qiu Y (2010) Constraint based dimension correlation and distance divergence for clustering high-dimensional data. In: IEEE 10th International conference on data mining, pp 629–638
Zhang X, Qiu Y, Wu Y (2011) Exploiting constraint inconsistence for dimension selection in subspace clustering: a semi-supervised approach. Neurocomputing 74(17):3598–3608
Zhu X (2005) Semi-supervised learning literature survey. Tech. rep., Computer Sciences, University of Wisconsin-Madison
Zhu X, Goldberg A (2009) Introduction to semi-supervised learning. Morgan & Claypool Publishers, New York