



# Efficient mixture model for clustering of sparse high dimensional binary data

Marek Śmieja<sup>1</sup> · Krzysztof Hajto<sup>1</sup> · Jacek Tabor<sup>1</sup>

Received: 25 January 2018 / Accepted: 20 May 2019 / Published online: 1 June 2019  
© The Author(s) 2019

## Abstract

Clustering is one of the fundamental tools for preliminary analysis of data. While most of the clustering methods are designed for continuous data, sparse high-dimensional binary representations became very popular in various domains such as text mining or cheminformatics. The application of classical clustering tools to this type of data usually proves to be very inefficient, both in terms of computational complexity as well as in terms of the utility of the results. In this paper we propose a mixture model, SPARSEMIX, for clustering of sparse high dimensional binary data, which connects model-based with centroid-based clustering. Every group is described by a representative and a probability distribution modeling dispersion from this representative. In contrast to classical mixture models based on the EM algorithm, SPARSEMIX: is specially designed for the processing of sparse data; can be efficiently realized by an on-line Hartigan optimization algorithm; describes every cluster by the most representative vector. We have performed extensive experimental studies on various types of data, which confirmed that SPARSEMIX builds partitions with a higher compatibility with reference grouping than related methods. Moreover, constructed representatives often better reveal the internal structure of data.

**Keywords** Bernoulli mixture model · Entropy clustering · Data compression · High-dimensional binary data

---

Responsible editor: Aristides Gionis.

---

✉ Krzysztof Hajto  
krzysztof.hajto@ii.uj.edu.pl

Marek Śmieja  
marek.smieja@ii.uj.edu.pl

Jacek Tabor  
jacek.tabor@ii.uj.edu.pl

<sup>1</sup> Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland

## 1 Introduction

Clustering, one of the fundamental tasks of machine learning, relies on grouping similar data points together while keeping dissimilar ones separate. It is hard to overestimate the role of clustering in present data analysis. Cluster analysis has been widely used in text mining to collect similar documents together (Baker and McCallum 1998); in biology to build groups of genes with related expression patterns (Fränti et al. 2003); in social networks to detect clusters of communities (Papadopoulos et al. 2012) and many other branches of science.

While most of the clustering methods are designed for continuous data, binary attributes become very common in various domains. In text mining, sentences (or documents) are often represented with the use of the set-of-words representation (Ying Zhao 2005; Guansong Pang 2015; Juan and Vidal 2002). In this case, a document is identified with a binary vector, where bit 1 at the  $i$ -th coordinate indicates that the  $i$ -th word from a given dictionary appears in this document.<sup>1</sup> In cheminformatics, a chemical compound is also described by a binary vector, where every bit denotes the presence or absence of a predefined chemical pattern (Ewing et al. 2006; Klekota and Roth 2008). Since the number of possible patterns (words) is large, the resulting representation is high dimensional. Another characteristic of a typical binary representation is its sparsity. In text processing the number of words contained in a sentence is relatively low compared to the total number of words in a dictionary. In consequence, most coordinates are occupied by zeros, while only selected ones are nonzero.

On one hand, the aforementioned sparse high dimensional binary representation is very intuitive for the user, which is also important for interpretable machine learning (Ribeiro et al. 2016). On the other hand, it is extremely hard to handle by classical clustering algorithms. One reason behind that is commonly known as the curse of dimensionality. Since in high dimensional spaces any randomly chosen pairs of points have roughly similar Euclidean distances, then it is not a proper measure of dissimilarity (Steinbach et al. 2004; Mali and Mitra 2003). Moreover, direct application of standard clustering tools to this type of data may lead to a substantial increase of computational cost, which disqualifies a method from a practical use. Subspace clustering, which aims at approximating a data set by a union of low-dimensional subspaces, is a typical approach for handling high dimensional data (Li et al. 2018; Lu et al. 2012; You et al. 2016; Struski et al. 2017; Rahmani and Atia 2017). Nevertheless, these techniques have problems with binary data, because they also use Euclidean distances in their criterion functions. Moreover, they are unable to process extremely high dimensional data due to the non-linear computational complexity with respect to data dimension.<sup>2</sup> In consequence, clustering algorithms have to be designed directly for this type of data to obtain satisfactory results in reasonable time.

In this paper we introduce a version of model-based clustering, SPARSEMIX, which efficiently processes high dimensional sparse binary data.<sup>3</sup> Our model splits the data

<sup>1</sup> Another possibility is to use bag-of-words or tf-idf transform.

<sup>2</sup> They often apply linear regression or principal component analysis to match an affine subspace to data points, which result in a quadratic or cubic computational complexity with respect to data dimension.

<sup>3</sup> An implementation of SPARSEMIX algorithm, together with some example data sets, is available on GitHub: <https://github.com/hajtos/SparseMIX>.



**Fig. 1** Representatives of handwritten digits from MNIST database produced by SPARSEMIX. Observe that SPARSEMIX created two clusters for the digit 1 (written vertically and diagonally), while examples of the digit 5 were split into other clusters (see Fig. 7 for details)

into groups and creates probabilistic distributions for those groups (see Sect. 3). The points are assigned in a way that minimizes the entropy of the distributions. In contrast to classical mixture models using Bernoulli variables or latent trait models, our approach is designed for sparse data and can be efficiently optimized by an on-line Hartigan algorithm, which converges faster and finds better solutions than batch procedures like EM (Sect. 4).

SPARSEMIX builds a bridge between mixture models and centroid-based clustering, and describes every cluster by its representative (a single vector characterizing the most popular cluster patterns) and a probability distribution modeling dispersion from this representative. The relationship between the form of the representative and the associated cluster distribution is controlled by an additional parameter of the model. By placing a parameter selection problem on solid mathematical ground, we show that we can move from a model providing the best compression rate of data to the one obtaining high speed performance (Sect. 4.3 and Theorem 1). Our method is additionally regularized by the cost of clusters' identification. It allows to reduce clusters on-line and fulfills an idea similar to the maximum a posteriori clustering and other approaches that taking a model's complexity into account (Barbará et al. 2002; Plumbley 2002; Silvestre et al. 2014; Bontemps et al. 2013). We present a theoretical and experimental analysis how the number of clusters depends on the main characteristics of the data set (Example 1 and Sect. 5.2).

The paper contains extensive experiments performed on various types of data, including text corpora, chemical and biological data sets, as well as the MNIST image database, see Sect. 5. The results show that SPARSEMIX provides a higher compatibility with reference partition than existing methods based on mixture models and similarity measures. Visual inspection of clusters representatives obtained by SPARSEMIX on the MNIST data set suggests high quality of its results, see Fig. 1. Moreover, its running time is significantly lower than that of related model-based algorithms and comparable to methods implemented in the Cluto package, which are optimized for processing large data sets, see Sect. 5.3.

This paper is an extension to our conference paper Śmieja et al. (2016) with the following new content:

1. We introduce a cluster representative, which provides a sparser representation of a cluster's instances and describes the most common patterns of the cluster.
2. We analyze the impact of the cluster representative on fitting a Bernoulli model to the cluster distribution (Theorem 1) and illustrate exemplary representatives for the MNIST data set.
3. An additional trade-off parameter  $\beta$  is introduced to the cost function, which weights the importance of model complexity. Its selection allows us to decide whether the model should aim to reduce clusters.

4. We extend a theoretical analysis of the model by inspecting various conditions influencing clusters' reduction, see Example 1. Moreover, we verify the clustering reduction problem in the experimental study, see Sect. 5.2.
5. An efficient implementation of the algorithm is discussed in detail, see Sect. 4.
6. We consider larger data sets (e.g. MNIST and Reuters), compare our method with a wider spectrum of clustering algorithms (maximum likelihood approaches, subspace clustering and dimensionality reduction techniques) and verify the stability of the algorithms.

## 2 Related work

In this section we refer to typical approaches for clustering binary data including distance-based and model-based techniques. Moreover, we discuss regularization methods, which allow to select optimal number of clusters.

### 2.1 Distance-based clustering

A lot of clustering methods are expressed in a geometric framework, where the similarity between objects is defined with the use of the Euclidean metric, e.g.  $k$ -means (MacQueen et al. 1967). Although a geometric description of clustering can be insightful for continuous data, it becomes less informative in the case of high dimensional binary (or discrete) vectors, where the Euclidean distance is not natural.

To adapt these approaches to binary data sets, the authors consider, for instance,  $k$ -medoids or  $k$ -modes (Huang 1998; Chan et al. 2004) with a dissimilarity measure designed for this special type of data, such as Hamming, Jaccard or Tanimoto measures (Li 2005). Evaluation of possible dissimilarity metrics for categorical data can be found in dos Santos and Zárate (2015), Bai et al. (2011). To obtain a more flexible structure of clusters, one can also use hierarchical methods (Zhao and Karypis 2002), density-based clustering (Wen et al. 2002) or model-based techniques (Spurek 2017; Spurek et al. 2017). One of important publicly available tools for efficient clustering of high dimensional binary data is the Cluto package (Karypis 2002). It is built on a sophisticated multi-level graph partitioning engine and offers many different criteria that can be used to derive both partitional, hierarchical and spectral clustering algorithms.

Another straightforward way for clustering high dimensional data relies on reducing the initial data's dimension through a preprocessing step. This allows to transform the data into a continuous low dimensional space where typical clustering methods are applied efficiently. Such dimensionality reduction can be performed with the use of linear methods, such as PCA (principal component analysis) (Indhumathi and Sathiyabama 2010), or non-linear techniques, such as deep autoencoders (Jang et al. 2017; Serrà and Karatzoglou 2017). Although this approach allows for using any clustering algorithm on the reduced data space, it leads to a loss of information about the original data after performing the dimensionality reduction.

Subspace clustering is a class of techniques designed to deal with high dimensional data by approximating them using multiple low-dimensional subspaces (Li et al. 2017). This area received considerable attention in the last years and many techniques have been proposed including iterative methods, statistical approaches and spectral clustering based methods (You et al. 2016; Tsakiris and Vidal 2017; Struski et al. 2017; Rahmani and Atia 2017). Although these methods give very good results on continuous data such as images (Lu et al. 2012), they are rarely applied to binary data, because they also use the Euclidean distance in their cost functions. Moreover, they are unable to process extremely high dimensional data due to the quadratic or cubic computational complexity with respect to the number of dimensions in the data (Struski et al. 2017). In consequence, a lot of algorithms apply dimensionality reduction as a preprocessing step (Lu et al. 2012).

## 2.2 Model-based techniques

Model-based clustering (McLachlan and Peel 2004), where data is modeled as a sample from a parametric mixture of probability distributions, is commonly used for grouping continuous data using Gaussian models, but has also been adapted for processing binary data. In the simplest case, the probability model of each cluster is composed of a sequence of independent Bernoulli variables (or multinomial distributions), describing the probabilities on subsequent attributes (Celeux and Govaert 1991; Juan and Vidal 2002). Since many attributes are usually statistically irrelevant and independent of true categories, they may be removed or associated with small weights (Graham and Miller 2006; Bouguila 2010). This partially links mixture models with subspace clustering of discrete data (Yamamoto and Hayashi 2015; Chen et al. 2016). Since the use of multinomial distributions formally requires an independence of attributes, different smoothing techniques were proposed, such as applying Dirichlet distributions as a prior to the multinomial (Bouguila and ElGuebaly 2009). Another version of using mixture models for binary variables tries to maximize the probability that the data points are generated around cluster centers with the smallest possible dispersion (Celeux and Govaert 1991). This technique is closely related to our approach. However, our model allows for using any cluster representatives (not only cluster centers) and is significantly faster due to the use of sparse coders.

A mixture of latent trait analyzers is a specialized type of mixture model for categorical data, where a continuous univariate of a multivariate latent variable is used to describe the dependence in categorical attributes (Vermunt 2007; Gollini and Murphy 2014). Although this technique recently received high interest in the literature (Langseth and Nielsen 2009; Cagnone and Viroli 2012), it is potentially difficult to fit the model, because the likelihood function involves an integral that cannot be evaluated analytically. Moreover, its use is computationally expensive for large high dimensional data sets (Tang et al. 2015).

Information-theoretic clustering relies on minimizing the entropy of a partition or maximizing the mutual information between the data and its discretized form Li et al. (2004), Tishby et al. (1999), Dhillon and Guan (2003). Although both approaches are similar and can be explained as a minimization of the coding cost, the first creates

“hard clusters”, where an instance is classified to exactly one category, while the second allows for soft assignments (Strouse and Schwab 2016). Celeux and Govaert (1991) established a close connection between entropy-based techniques for discrete data and model-based clustering using Bernoulli variables. In particular, the entropy criterion can be formally derived using the maximum likelihood of the classification task.

To the best of the authors’ knowledge, neither model-based clustering nor information-theoretic methods have been optimized for processing sparse high-dimensional binary data. Our method can be seen as a combination of  $k$ -medoids with model-based clustering (in the sense that it describes a cluster by a single representative and a multivariate probability model), which is efficiently realized for sparse high-dimensional binary data.

### 2.3 Model selection criteria

Clustering methods usually assume that the number of clusters is known. However, most model-based techniques can be combined with additional tools, which, to some extent, allow to select an optimal number of groups (Grantham 2014).

A standard approach for detecting the number of clusters is to conduct a series of statistical tests. The idea relies on comparing likelihood ratios between models obtained for different numbers of groups (McLachlan 1987; Ghosh et al. 2006). Another method is to use a penalized likelihood, which combines the obtained likelihood with a measure of model complexity. A straightforward procedure fits models with different numbers of clusters and selects the one with the optimal penalized likelihood criterion (McLachlan and Peel 2004). Typical measures for model complexity are the Akaike Information Criterion (AIC) (Bozdogan and Sclove 1984) and the Bayesian Information Criterion (BIC) (Schwarz et al. 1978). Although these criteria can be directly applied to Gaussian Mixture Models, an additional parameter has to be estimated for the mixture of Bernoulli models to balance the importance between a model’s quality and its complexity (Bontemps et al. 2013). Alternatively, one can use coding based criteria, such as the Minimum Message Length (MML) (Baxter and Oliver 2000; Bouguila and Ziou 2007) or the Minimum Description Length (MDL) (Barbará et al. 2002) principles. The idea is to find the number of coders (clusters), which minimizes the amount of information needed to transmit the data efficiently from sender to receiver. In addition to detecting the number of clusters, these criteria also allow for eliminating redundant attributes. Coding criteria usually are used for comparing two models (like AIC or BIC criteria). Silvestre et al. (2015) showed how to apply the MML criterion simultaneously with a clustering method. This is similar to our algorithm, which reduces redundant clusters on-line.

Finally, it is possible to use a fully Bayesian approach, which treats the number of clusters as a random variable. By defining a prior distribution on the number of clusters  $K$ , we can estimate its number using the maximum a posteriori method (MAP) (Nobile et al. 2004). Although this approach is theoretically justified, it may be difficult to optimize in practice (Richardson and Green 1997; Nobile and Fearnside 2007). To

our best knowledge, currently there are no available implementations achieving this technique for the Bernoulli mixture model.

### 3 Clustering model

The goal of clustering is to split data into groups that contain elements characterized by similar patterns. In our approach, the elements are similar if they can be efficiently compressed by the same algorithm. We begin this section by presenting a model for compressing elements within a single cluster. Next, we combine these partial encoders and define a final clustering objective function.

#### 3.1 Compression model for a single cluster

Let us assume that  $X \subset \{0, 1\}^D$  is a data set (cluster) containing  $D$ -dimensional binary vectors. We implicitly assume that  $X$  represents sparse data, i.e. the number of positions occupied by non-zero bits is relatively low.

A typical way for encoding such data is to directly remember the values at each coordinate (Barbará et al. 2002; Li et al. 2004). Since, in practice,  $D$  is often large, this straightforward technique might be computationally inefficient. Moreover, due to the sparsity of data, positions occupied by zeros are not very informative while the less frequent non-zero bits carry substantial knowledge. Therefore, instead of remembering all the bits of every vector, it might be more convenient to encode positions occupied by non-zero values. It occurs that this strategy can be efficiently implemented by on-line algorithms.

To realize the aforementioned goal, we first select a representative (prototype)  $m \in \{0, 1\}^D$  of a cluster  $X$ . Next, for each data point  $x = (x_1, \dots, x_D) \in X$  we construct a corresponding vector

$$\text{xor}(x, m) = (|x_1 - m_1|, \dots, |x_D - m_D|) \in \{0, 1\}^D$$

of differences with  $m$ . If a representative is chosen as the most probable point of a cluster (the centroid of a cluster), then the data set of differences will be, on average, at least as sparse as the original data set  $X$ . An efficient way for storing such sparse data relies on encoding the numbers of coordinates with non-zero bits. Concluding, the original data  $X$  is compressed by remembering a representative and encoding resulting vectors of differences in an efficient manner, see Fig. 2.

We now precisely follow the above idea and calculate the cost of coding in this scheme, which will be the basis of our clustering criterion function. Let the distribution at the  $i$ -th coordinate of  $x \in X$  be described by a Bernoulli random variable taking value 1 with a probability  $p_i \in [0, 1]$  and 0 with a probability  $(1 - p_i)$ , i.e.  $p_i = P(x_i = 1)$ . For a fixed  $T \in [0, 1]$ , we consider a representative  $m = m(T) = (m_1, \dots, m_D)$  defined by

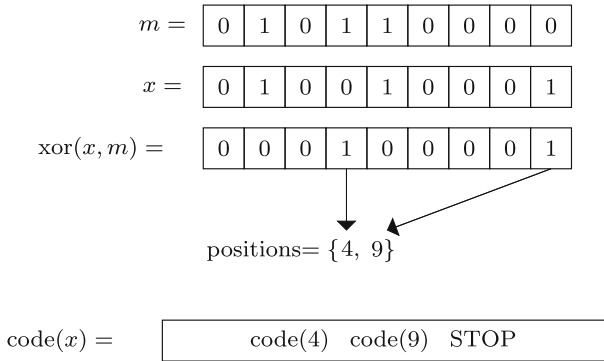


Fig. 2 Sparse data coding

$$m_i = \begin{cases} 0, & p_i \leq T, \\ 1, & p_i > T, \end{cases}$$

Although a representative  $m(T)$  depends on  $T$ , we usually discard this parameter and simply write  $m$ , when  $T$  is known from the context. The  $i$ -th coordinate of  $X$  is more likely to attain a value of 1, if  $p_i > \frac{1}{2}$  and, in consequence, for  $T = \frac{1}{2}$  the representative  $m$  coincides with the most probable point of  $X$ .

Given a representative  $m$ , we consider the differences  $\text{xor}(x, m)$ , for  $x \in X$ , and denote such a data set by

$$D_m(X) = \{\text{xor}(x, m) : x \in X\}.$$

The probability  $q_i = q_i(T)$  of bit 1 at the  $i$ -th position in  $D_m(X)$  equals

$$q_i = \begin{cases} p_i, & p_i \leq T, \\ 1 - p_i, & p_i > T. \end{cases}$$

Let us notice that  $q_i \leq p_i$ , for  $T \geq \frac{1}{2}$ , which makes  $D_m(X)$  at least as sparse as  $X$ , see Fig. 3.

To design an efficient coder for  $D_m(X)$ , it is sufficient to remember the positions of  $d \in D_m(X)$  with non-zero bits. Thus, we transform a random vector  $q$  into a random variable  $Q$  by:

$$Q_i = \frac{q_i}{Z},$$

where  $Z = Z(T) = \sum_{j=1}^D q_j$  is a normalization factor. A distribution  $\mathcal{Q} = \mathcal{Q}(T) = \{Q_i : q_i > 0\}$  describes the probabilities over variables that are 1 at least once.

The Shannon entropy theory states that the code-lengths in an optimal prefix-free coding depend strictly on the associated probability distribution (Cover and Thomas 2012). Given a distribution  $\mathcal{Q}$  of non-zero positions it is possible to construct  $|\mathcal{Q}| \leq D$



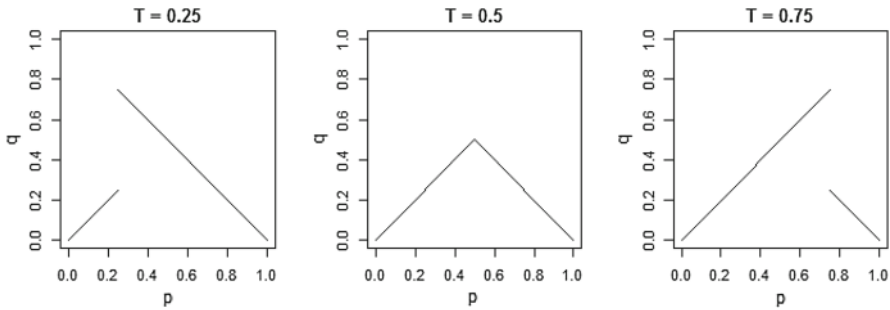


Fig. 3 Relation between probabilities  $p_i$  and  $q_i$

codes, each with the length<sup>4</sup>  $-\log Q_i$ . The short codes correspond to the most frequent symbols, while the longest ones are related to rare objects. Given an arbitrary element  $d = (d_1, \dots, d_D) \in D_m(X)$  we encode its non-zero coordinates and obtain that its compression requires

$$\sum_{i:d_i=1} -\log Q_i$$

bits. The above codes are optimal if every  $d_i$  is generated independently from the others. Otherwise, the data could be compressed further.

This leads to the SPARSEMIX objective function for a single cluster, which gives the average cost of compressing a single element of  $X$  by our sparse coder:

**Definition 1** (one cluster cost function) Let  $X \subset \{0, 1\}^D$  be a data set and let  $T \in [0, 1]$  be fixed. The SPARSEMIX objective function for a single cluster is given by:<sup>5</sup>

$$C_T(X) = C(D_m(X)) = \sum_{i=1}^D q_i (-\log Q_i). \tag{1}$$

Observe that, given probabilities  $p_1, \dots, p_D$ , the selection of  $T$  determines the form of  $m$  and  $D_m(X)$ . The above cost coincides with the optimal compression under the assumption of independence imposed on the attributes of  $d \in D_m(X)$ . If this assumption does not hold, this coding scheme gives a suboptimal compression, but such codes can still be constructed.

**Remark 1** To be able to decode the initial data set, we would also need to remember the probabilities  $p_1, \dots, p_D$  determining the form of the representative  $m$  and the corresponding probability  $Q$  used for constructing the codewords. These are the model parameters, which, in a practical coding scheme, are stored once in the header. Since they do not affect the asymptotic value of data compression, we do not include them in the final cost function.<sup>6</sup>

<sup>4</sup> in the limiting case.

<sup>5</sup> We put:  $0 \cdot \log 0 = 0$ .

<sup>6</sup> Nevertheless, these probabilities should be accounted in model selection criteria as AIC or BIC.

Moreover, to reconstruct the original data we should distinguish between the encoded representations of subsequent vectors. It could be realized by reserving an additional symbol for separating two encoded vectors or by remembering the number of non-zero positions in every vector. Although this is necessary for the coding task, it is less important for clustering and therefore we decided not to include it in the cost function.

The following theorem demonstrates that  $T = \frac{1}{2}$  provides the best compression rate of a single cluster. Making use of the analogy between the Shannon compression theory and data modeling, it shows that with  $T_1 \geq \frac{1}{2}$  our model allows for a better fitting of a Bernoulli model to the cluster distribution than using  $T_2$  greater than  $T_1$ .

**Theorem 1** *Let  $X \subset \{0, 1\}^D$  be a data set and let  $\frac{1}{2} \leq T_1 \leq T_2 \leq 1$  be fixed. If  $m(T_1), m(T_2)$  are two representatives and the mean number of non-zero bits in  $D_{m_1}(X)$  is not lower than 1, i.e.  $Z(T_1) \geq 1$ , then:*

$$C_{T_1}(X) \leq C_{T_2}(X).$$

**Proof** See Appendix A. □

### 3.2 Clustering criterion function

A single encoder allows us to compress simple data. To efficiently encode real data sets, which usually origin from several sources, it is profitable to construct multiple coding algorithms, each designed for one homogeneous part of the data. Finding an optimal set of algorithms leads to a natural division of the data, which is the basic idea behind our model. Below, we describe the construction of our clustering objective function, which combines the partial cost functions of the clusters.

Let us assume that we are given a partition of  $X$  containing  $k$  groups  $X_1, \dots, X_k$  (pairwise disjoint subsets of  $X$  such that  $X = X_1 \cup \dots \cup X_k$ ), where every subset  $X_i$  is described by its own coding algorithm. Observe that to encode an instance  $x \in X_i$  using such a multiple coding model one needs to remember its group identifier and the code of  $x$  defined by the  $i$ -th encoder, i.e.,

$$\text{code}(x) = [\text{code}(i), \text{code}_i(x)]. \quad (2)$$

Such a strategy enables unique decoding, because the retrieved coding algorithm subsequently allows us to discover the exact instance (see Fig. 4). The compression procedure should find a division of  $X$  and design  $k$  coding algorithms, which minimize the expected length of code given by (2).

The coding algorithms for each cluster are designed as described in the previous subsection. More precisely, let  $p_i = (p_1^i, \dots, p_D^i)$  be a vector, where  $p_j^i$  is the probability that the  $j$ -th coordinate in the  $i$ -th cluster is non-zero, for  $i = 1, \dots, k$ . Next, given a fixed  $T$ , for each cluster  $X_i$  we construct a representative  $m_i = (m_1^i, \dots, m_D^i)$  and calculate the associated probability distributions  $q_i = (q_1^i, \dots, q_D^i)$  and  $\mathcal{Q}_i =$

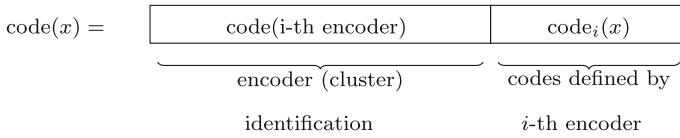


Fig. 4 Multi-encoder model

$\{Q_1^i, \dots, Q_D^i\}$  on the set of differences  $D_{m_i}(X_i)$ . The average code-length for compressing a single vector in the  $i$ -th cluster is given by (see (1)):

$$C_T(X_i) = C(D_{m_i}(X_i)) = \sum_{j=1}^D q_j^i (-\log Q_j^i). \tag{3}$$

To remember clusters' identifiers, we again follow Shannon's theory of coding (Tabor and Spurek 2014; Smieja and Tabor 2012). Given a probability  $P_i = P(x \in X_i)$  of generating an instance from a cluster  $X_i$  (the prior probability), the optimal code-length of the  $i$ -th identifier is given by

$$C(i) = -\log P_i. \tag{4}$$

Since the introduction of any new cluster increases the cost of clusters' identification, it might occur that maintaining a smaller number of groups is more profitable. Therefore, this model will have a tendency to adjust the sizes of clusters and, in consequence, some groups might finally disappear (can be reduced).

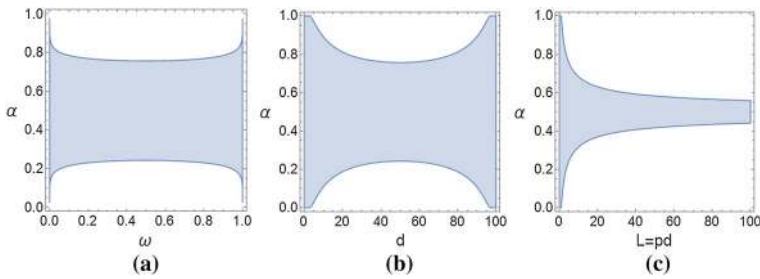
The SPARSEMIX cost function combines the cost of clusters' identification with the cost of encoding their elements. To add higher flexibility to the model, we introduce an additional parameter  $\beta$ , which allows to weight the cost of clusters' identification. Specifically, if the number of clusters is known a priori, we should put  $\beta = 0$  to prevent from reducing any groups. On the other hand, to encourage the model to remove clusters we can increase the value of  $\beta$ . By default  $\beta = 1$ , which gives a typical coding model:

**Definition 2** (*clustering cost function*) Let  $X = \{0, 1\}^D$  be a data set of  $D$ -dimensional binary vectors and let  $X_1, \dots, X_k$  be a partition of  $X$  into pairwise disjoint subsets. For a fixed  $T \in [0, 1]$  and  $\beta \geq 0$  the SPARSEMIX clustering objective function equals:

$$C_{\beta,T}(X_1, \dots, X_k) = \sum_{i=1}^k P_i \cdot (C_T(X_i) + \beta \cdot C(i)), \tag{5}$$

where  $P_i$  is the probability of a cluster  $X_i$ ,  $C(i)$  is the cost of encoding its identifier (4) and  $C_T(X_i)$  is the average code-length of compressing elements of  $X_i$  (3).

As can be seen, every cluster is described by a single representative and a probability distribution modeling dispersion from a representative. Therefore, our model can be interpreted as a combination of  $k$ -medoids with model-based clustering. It is worth



**Fig. 5** Optimal number of clusters for data generated by the mixture of sources given by (7). Blue regions show the combinations of mixture parameters which lead to one cluster, while white areas correspond to two clusters. 5a presents the case when every source is characterized by the same number of bits  $d = \frac{1}{2}D$ , 5b corresponds to the situation when each source produces the same number of instances, while 5c is the combination of both previous cases (Color figure online)

mentioning that for  $T = 1$ , we always get a representative  $m = 0$ . In consequence,  $D_0(X) = X$  and a distribution in every cluster is directly fitted to the original data.

The cost of clusters' identification allows us to reduce unnecessary clusters. To get more insight into this mechanism, we present the following example. For simplicity, we use  $T = \beta = 1$ .

**Example 1** By  $P(p, \alpha, d)$ , for  $p, \alpha \in [0, 1]$  and  $d \in \{0, \dots, D\}$ , we denote a  $D$ -dimensional probability distribution, which generates bit 1 at the  $i$ -th position with probability:

$$p_i = \begin{cases} \alpha p, & i = 1, \dots, d, \\ (1 - \alpha)p, & i = d + 1, \dots, D. \end{cases} \tag{6}$$

Let us consider a data set generated by the mixture of two sources:

$$\omega P(p, \alpha, d) + (1 - \omega)P(p, 1 - \alpha, d), \tag{7}$$

for  $\omega \in [0, 1]$ .

To visualize the situation, we can arrange a data set in a matrix, where rows correspond to instances generated by the mixture components, while the columns are related to their attributes:

$$1 - \omega \left\{ \underbrace{\begin{pmatrix} \alpha p & (1 - \alpha)p \\ (1 - \alpha)p & \alpha p \end{pmatrix}}_d \quad \underbrace{\hspace{2cm}}_{D-d} \right\}$$

The matrix entries show the probability of generating bit 1 at a given coordinate belonging to one of the four matrix regions. The parameter  $\alpha$  determines the similarity between the instances generated from the underlying distributions. For  $\alpha = \frac{1}{2}$ , both components are identical, while for  $\alpha \in \{0, 1\}$  we get their perfect distinction.

We compare the cost of using a single cluster for all instances with the cost of splitting the data into two optimal groups (clusters are perfectly fitted to the sources generating the data). For the reader's convenience, we put the details of the calculations in B. The analysis of the results is presented below. We consider three cases:

1. *Sources are characterized by the same number of bits.* The influence of  $\omega$  and  $\alpha$  on the number of clusters, for a fixed  $d = \frac{1}{2}D$ , is presented in Fig. 5a. Generally, if sources are well-separated, i.e.  $\alpha \notin (0.2, 0.8)$ , then SPARSEMIX will always create two clusters regardless of the mixing proportion. This confirms that SPARSEMIX is not sensitive to unbalanced sources generating the data if only they are distinct.
2. *Sources contain the same number of instances.* The Fig. 5b shows the relation between  $d$  and  $\alpha$  when the mixing parameter  $\omega = \frac{1}{2}$ . If one source is identified by a significantly lower number of attributes than the other ( $d \ll D$ ), then SPARSEMIX will merge both sources into a single cluster. Since one source is characterized by a small number of features, it might be more costly to encode the cluster identifier than its attributes. In other words, the clusters are merged together, because the cost of cluster identification outweighs the cost of encoding the source elements.
3. *Both proportions of dimensions and instances for the mixture sources are balanced.* If we set equal proportions for the source and dimension coefficients, then the number of clusters depends on the average number of non-zero bits in the data  $L = pd$ , see Fig. 5c. For high density of data, we can easily distinguish the sources and, in consequence, SPARSEMIX will end up with two clusters. On the other hand, in the case of sparse data, we use less memory for remembering its elements and the cost of clusters' identification grows compared to the cost of encoding the elements within the groups.

## 4 Fast optimization algorithm

In this section, we present an on-line algorithm for optimizing the SPARSEMIX cost function and discuss its computational complexity. Before that, let us first show how to estimate the probabilities involved in the formula (5).

### 4.1 Estimation of the cost function

We assume that a data set  $X \subset \{0, 1\}^D$  is split into  $k$  groups  $X_1, \dots, X_k$ , where  $n = |X|$  and  $n_i = |X_i|$ . Let us denote by

$$n_j^i = \sum_{x \in X_i} x_j$$

the number of objects in  $X_i$  with the  $j$ -th position occupied by value 1. This allows us to estimate the probability  $p_j^i$  of bit 1 at the  $j$ -th coordinate in  $X_i$  as

$$p_j^i = \frac{n_j^i}{n_i}$$

and, consequently, rewrite the representative  $m_i = (m_1^i, \dots, m_D^i)$  of the  $i$ -th cluster as

$$m_j^i = \begin{cases} 0, & \frac{n_j^i}{n_i} \leq T, \\ 1, & \frac{n_j^i}{n_i} > T. \end{cases}$$

To calculate the formula for  $C_T(X_i)$ , we first estimate the probability  $q_j^i$  of bit 1 at the  $j$ -th coordinate in  $D_{m_i}(X_i)$ ,

$$q_j^i = \begin{cases} \frac{n_j^i}{n_i}, & \frac{n_j^i}{n_i} \leq T, \\ \frac{n_i - n_j^i}{n_i}, & \frac{n_j^i}{n_i} > T. \end{cases}$$

If we denote by

$$N_j^i = \begin{cases} n_j^i, & \frac{n_j^i}{n_i} \leq T, \\ n_i - n_j^i, & \frac{n_j^i}{n_i} > T \end{cases} \quad (8)$$

the number of vectors in  $D_{m_i}(X_i)$  with the  $j$ -th coordinate occupied by bit 1 and by

$$S_i = \sum_{j=1}^D N_j^i$$

the total number of non-zero entries in  $D_{m_i}(X_i)$ , then we can estimate the probability  $Q_j^i$  as:

$$Q_j^i = \frac{N_j^i}{S_i}.$$

This allows us to rewrite the cost function for a cluster  $X_i$  as

$$\begin{aligned} C_T(X_i) &= \sum_{j=1}^D q_j^i \left( -\log Q_j^i \right) \\ &= \sum_{j:p_j^i \leq T} \frac{n_j^i}{n_i} \left( -\log \frac{N_j^i}{S_i} \right) + \sum_{j:p_j^i > T} \left( 1 - \frac{n_j^i}{n_i} \right) \left( -\log \frac{N_j^i}{S_i} \right) \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{n_i} \sum_{j=1}^D N_j^i (-\log N_j^i + \log S_i) \\
&= \frac{1}{n_i} \left( S_i \log S_i + \sum_{j=1}^D N_j^i (-\log N_j^i) \right).
\end{aligned}$$

Finally, since the probability  $P_i$  of the  $i$ -th cluster can be estimated as  $P_i = \frac{n_i}{n}$ , then the optimal code-length of a cluster identifier equals

$$C(i) = -\log \frac{n_i}{n}.$$

In consequence, the overall cost function is computed as:

$$\begin{aligned}
C_{\beta, T}(X) &= \sum_{i=1}^k \frac{n_i}{n} (\beta \cdot C(i) + C_T(X_i)) \\
&= \sum_{i=1}^k \frac{n_i}{n} \left( \beta \cdot \left( -\log \frac{n_i}{n} \right) + \frac{1}{n_i} \left[ S_i \log S_i + \sum_{j=1}^D N_j^i (-\log N_j^i) \right] \right) \\
&= \beta \log n + \frac{1}{n} \sum_{i=1}^k \left( \beta n_i (-\log n_i) + S_i \log S_i + \sum_{j=1}^D N_j^i (-\log N_j^i) \right).
\end{aligned}$$

## 4.2 Optimization algorithm

To obtain an optimal partition of  $X$ , the SPARSEMIX cost function has to be minimized. For this purpose, we adapt a modified version of the Hartigan procedure, which is commonly applied in an on-line version of  $k$ -means (Hartigan and Wong 1979). Although the complexity of a single iteration of the Hartigan algorithm is often higher than in batch procedures such as EM, the model converges in a significantly lower number of iterations and usually finds better minima [see Śmieja and Geiger (2017) for the experimental comparison in the case of cross-entropy clustering].

The minimization procedure consists of two parts: initialization and iteration. In the initialization stage,  $k \geq 2$  nonempty groups are formed in an arbitrary manner. In the simplest case, it could be a random initialization, but to obtain better results one can also apply a kind of  $k$ -means++ seeding. In the iteration step the elements are reassigned between clusters in order to minimize the value of the criterion function. Additionally, due to the cost of clusters' identification, some groups may lose their elements and finally disappear. In practice, a cluster is reduced if its size falls below a given threshold  $\varepsilon \cdot |X|$ , for a fixed  $\varepsilon > 0$ .

A detailed algorithm is presented below ( $\beta$  and  $T$  are fixed):

```

1: INPUT:
2:  $X \subset \{0, 1\}^D$  – data set
3:  $k$  – initial number of clusters
4:  $\varepsilon > 0$  – cluster reduction parameter
5: OUTPUT:
6: Partition  $\mathcal{X}$  of  $X$ 
7: INITIALIZATION:
8:  $\mathcal{Y} = \{Y_1, \dots, Y_k\}$  – random partition of  $X$  into  $k$  groups
9: ITERATION:
10: repeat
11:   for all  $x \in X$  do
12:      $Y_x \leftarrow$  get cluster of  $x$ 
13:      $Y \leftarrow \arg \max_{Y \in \mathcal{Y}} \{C_T(Y_x) + C_T(Y) - C_T(Y_x \setminus \{x\}) - C_T(Y \cup \{x\})\}$ 
14:     if  $Y \neq Y_x$  then
15:       switch  $x$  from  $Y_x$  to  $Y$ 
16:       update probability models of  $Y_x$  and  $Y$ 
17:       if  $|Y_x| < \varepsilon \cdot |X|$  then
18:         delete cluster  $Y_x$  and assign its elements to these clusters which minimize the SPARSEMIX
           cost function
19:       end if
20:     end if
21:   end for
22: until no switch for all subsequent elements of  $X$ 

```

The outlined algorithm is not deterministic, i.e. its result depends on the initial partition. Therefore, the algorithm should be restarted multiple times to find a reasonable solution.

The proposed algorithm has to traverse all data points and all clusters to find the best assignment. Its computational complexity depends on the procedure for updating cluster models and recalculating the SPARSEMIX cost function after switching elements between clusters (see lines 13 and 16). Below, we discuss the details of an efficient recalculation of this cost.

We start by showing how to update  $C_T(X_i)$  when we add  $x$  to the cluster  $X_i$ , i.e. how to compute  $C_T(X_i \cup \{x\})$  given  $C_T(X_i)$ . The situation when we remove  $x$  from a cluster is analogous. The updating of  $n_j^i$  and  $n_i$  is immediate (by a symbol with a hat  $\hat{y}$  we denote the updated value of a variable  $y$ ):

$$\hat{n}_j^i = n_j^i + x_j \text{ and } \hat{n}_i = n_i + 1.$$

In particular,  $n_j^i$  only changes its value on these positions  $j$  where  $x_j$  is non-zero.

Recalculation of  $N_j^i$  is more complex, since it is calculated by using one of the two formulas involved in (8), depending on the relation between  $\frac{n_j^i}{n_i}$  and  $T$ . We consider four cases:

1. If  $n_j^i \leq (n_i + 1)T - 1$ , then before and after the update we use the first formula of (8):



$$\hat{N}_j^i = N_j^i + x_j.$$

Moreover, this value changes only when  $x_j = 1$ .

2. If  $n_j^i > (n_i + 1)T$ , then before and after the update we use the second formula:

$$\hat{N}_j^i = N_j^i + (1 - x_j).$$

It is changed only when  $x_j = 0$ .

3. If  $x_j = 0$  and  $n_j^i \in (n_i T, (n_i + 1)T]$  then we switch from the second to the first formula and

$$\hat{N}_j^i = n_j^i.$$

Otherwise, it remains unchanged.

4. If  $x_j = 1$  and  $n_j^i \in ((n_i + 1)T - 1, n_i T]$  then we switch from the first to the second formula and

$$\hat{N}_j^i = n_i - n_j^i.$$

Otherwise, it remains unchanged.

Due to the sparsity of  $X$  there are only a few coordinates of  $x$  satisfying  $x_j = 1$ . In consequence, the complexity of updates in the cases 1 and 4 depends only on the number of non-zero bits in  $X$ . On the other hand, although  $x_j = 0$  happens often, the situation when  $n_j^i > n_i T$  is rare (for  $T \geq \frac{1}{2}$ ), because  $X$  is sparse. Clearly,  $S_i$  changes only if  $N_j^i$  is changed as well.

Finally, to get the new cost of a cluster, we need to recalculate  $\sum_{j=1}^D N_j^i (-\log N_j^i)$ . If we remember its old value  $h(N_1^i, \dots, N_D^i) = \sum_{j=1}^D N_j^i (-\log N_j^i)$ , then it is sufficient to update it on coordinates  $j$  such that  $N_j^i \neq \hat{N}_j^i$  by:

$$h(\hat{N}_1^i, \dots, \hat{N}_D^i) = h(N_1^i, \dots, N_D^i) - \sum_{j: N_j^i \neq \hat{N}_j^i} \left( \hat{N}_j^i (-\log \hat{N}_j^i) - N_j^i (-\log N_j^i) \right).$$

### 4.3 Computational complexity

We analyze the computational complexity of the whole algorithm.

We start with calculating the cost of switching an element from one cluster to another. As discussed above, given  $x \in X$  the recalculation of  $N_j^i$ , for  $j = 1, \dots, D$ , dominates the cost of updating any other quantity. Namely, we need to make updates on  $c_i(x)$  coordinates, where:

$$\begin{aligned}
 c_i(x) = c_{i,T}(x) &= |\{j : n_j^i \in ((n_i + 1)T - 1, n_i T] \text{ and } x_j = 1\}| \\
 &\quad + |\{j : n_j^i \in (n_i T, (n_i + 1)T] \text{ and } x_j = 0\}| \\
 &\quad + |\{j : n_j^i \leq (n_i + 1)T - 1 \text{ and } x_j = 1\}| \\
 &\quad + |\{j : n_j^i > (n_i + 1)T \text{ and } x_j = 0\}| \\
 &\leq |\{j : x_j = 1\}| + |\{j : n_j^i > (n_i + 1)T - 1\}| \\
 &\leq |\{j : x_j = 1\}| + |\{j : p_j^i > T - \frac{1-T}{n_i}\}|.
 \end{aligned}
 \tag{9}$$

Therefore,  $c_i(x)$  is bounded from above by the number of non-zero bits in  $x$  and the number of coordinates where the probability  $p_j^i$  of bit 1 exceeds the threshold  $T - \frac{1-T}{n_i}$ . For  $T = \frac{1}{2}$ , this threshold equals  $\frac{n_i-1}{2n_i} \approx \frac{1}{2}$ , while for  $T = 1$  it attains a value of 1 and, in consequence,  $c_i(x)$  is exactly the number of coordinates with non-zero bits in  $x$ . It is also easy to see that  $c_{i,T_1}(x) \geq c_{i,T_2}(x)$  if  $\frac{1}{2} \leq T_1 < T_2$ , i.e. the updates are faster for higher  $T$ .

In a single iteration, we need to visit every point and consider its possible switch to every other cluster. The required number of operations is, thus, bounded from above by:

$$\begin{aligned}
 \sum_{x \in X} \sum_{i=1}^k c_i(x) &\leq \sum_{x \in X} \sum_{i=1}^k |\{j : x_j = 1\}| + \sum_{x \in X} \sum_{i=1}^k \left| \left\{ j : p_j^i > T - \frac{1-T}{n_i} \right\} \right| \\
 &= k \sum_{x \in X} |\{j : x_j = 1\}| + n \sum_{i=1}^k \left| \left\{ j : p_j^i > T - \frac{1-T}{n_i} \right\} \right|
 \end{aligned}
 \tag{10}$$

The first term of (10) equals:

$$k \sum_{x \in X} |\{j : x_j = 1\}| = k \cdot N,$$

where  $N = \sum_{x \in X} |\{j : x_j = 1\}|$  is the total number of non-zero bits in the whole data set  $X$ . For  $T = 1$ , the second term vanishes and, in consequence, the complexity of the algorithm is linear with respect to the number of non-zero bits in a data set  $X$  and the number of clusters.

To get the complexity for  $T \in [\frac{1}{2}, 1)$ , we calculate the second term of (10):

$$\begin{aligned}
 \sum_{i=1}^k |\{j : p_j^i > T - \frac{1-T}{n_i}\}| &= \sum_{i=1}^k \left| \left\{ j : \frac{n_j^i}{n_i} > T - \frac{1-T}{n_i} \right\} \right| \\
 &= \sum_{i=1}^k |\{j : n_j^i > T(n_i + 1) - 1\}| \\
 &= \sum_{i=1}^k \sum_{j=1}^D \mathbb{1}_{\{n_j^i > T(n_i+1)-1\}} \\
 &= \sum_{i=1}^k \sum_{j=1}^D \mathbb{1}_{\left\{ \sum_{x \in X_i} x_j > T(n_i+1)-1 \right\}},
 \end{aligned}$$

where  $\mathbb{1}_A$  is the indicator function, i.e.  $\mathbb{1}_A = 1$ , if a condition  $A$  is true and 0 otherwise.

The condition inside the characteristic function is satisfied if the  $i$ -th cluster has more than  $T(n_i + 1) - 1$  non-zero entries at the  $j$ -th attribute. To consider the worst case, we want to satisfy this condition the maximal number of times. Thus, we calculate how many groups of  $T(n_i + 1) - 1$  non-zero bits we can create from their total number  $N$ . We have:

$$\sum_{i=1}^k \sum_{j=1}^D \mathbb{1}_{\left\{ \sum_{x \in X_i} x_j > T(n_i+1)-1 \right\}} \leq \sum_{i=1}^k \frac{N}{T(n_i + 1) - 1}.$$

This expression depends on the clusters sizes. In the worst case, we have one big cluster and  $k - 1$  tiny groups, which may results in an almost linear complexity with respect to the data dimension. This is, however, unusual. In most cases, all clusters have approximately the same number of instances, i.e.  $n_i = \frac{n}{k}$ . If we assume  $T = \frac{1}{2}$  (the worst case for complexity), then

$$\sum_{i=1}^k \frac{N}{T(n_i + 1) - 1} = k \frac{N}{\frac{1}{2} \left( \frac{n}{k} + 1 \right) - 1} = \frac{Nk}{\frac{n-2k}{2k}} = \frac{2Nk^2}{n - 2k}$$

Taking together the above considerations, for equally sized-clusters the second term of (10) is bounded by:

$$n \frac{2Nk^2}{n - 2k} = 2Nk^2 \frac{1}{1 - \frac{2k}{n}}.$$

The number of instances  $n$  is usually significantly greater than the number of clusters  $k$ . If we assume that  $n = rk$ , for  $r > 1$ , then

$$2Nk^2 \frac{1}{1 - \frac{2k}{n}} = 2Nk^2 \frac{1}{1 - \frac{2k}{rk}} = 2Nk^2 \frac{r}{r - 2} \approx 2Nk^2,$$

for large  $r$ . Thus, for equally-sized clusters and any  $T \in [\frac{1}{2}, 1)$ , the computational complexity is, at most, still linear with respect to the number of non-zero bits in  $X$  and quadratic with respect to the number of clusters.

### 5 Experiments

In this section we evaluate the performance of our algorithm and analyze its behavior in various clustering tasks. We compare its results with related methods. To denote our method we write SPARSEMIX( $\beta, T$ ), where  $\beta$  and  $T$  are the parameters of its cost function 2.

**Table 1** Summary of the data sets used in experiments

Dataset	Size	Dimensions	Avg. no. of non-zero bits	Classes
Reuters	291 127	47 236	55.58	18
20newsgroups	18 846	187 397	151.13	20
7newsgroups	6 997	26 411	99.49	7
Farm-ads	4 143	54 877	197.23	2
Questions	5 452	3 029	4.04	6
Sentiment	1 000	2 750	7.50	2
SMS	5 574	7 259	13.51	2
Chemical data	3 374	4 860	65.45	5
Mushroom	8 124	119	21	2
Splice	3 190	287	60	2
mnist	70 000	784	150.08	10

## 5.1 Quality of clusters

In this experiment we evaluated our method over various binary data sets, summarized in Table 1, and compared its results with related methods listed at the beginning of this section. Since we considered data sets designed for classification, we compared the obtained clusterings with a reference partition.

Their agreement was measured by Adjusted Rand Index (ARI) (Hubert and Arabie 1985), which is an extension of Rand Index (RI). RI counts the number of point pairs for which their relation with regards to a partition (i.e. are they in the same cluster or not) is the same for both partitions, divided by the number of all possible pairs. Although RI takes values between 0 and 1, it is usually significantly greater than 0 for random groupings. Thus, RI is additionally renormalized as:

$$ARI = \frac{RI - E(RI)}{\max(RI) - E(RI)},$$

where  $E(RI)$  and  $\max(RI)$  are the mean and maximal values of RI, respectively. ARI attains a maximal value of 1 for identical partitions, while for a random clustering<sup>7</sup> the expected ARI is 0. We also used two additional measures for verifying the quality of the results, Normalized Mutual Information and Clustering Accuracy. For the reader's convenience, we include these results in Appendix C.

We used seven text data sets: Reuters (a subset of documents from the Reuters Corpus Volume 1 Version 2 categorized into 18 topics) (Lewis et al. 2004), Questions (Li and Roth 2002), 20newsgroups, 7newsgroups (a subset containing only 7 classes of 20newsgroups), Farm-ads, SMS Spam Collection and Sentiment Labeled Sentences retrieved from UCI repository (Asuncion 2007). Each data set was encoded in a binary form with the use of the set-of-words representation: given a dictionary of words, a

<sup>7</sup> ARI might take negative values, when the produced partition is less compatible with the reference grouping than a random assignment.

document (or sentence) is represented as a binary vector, where coordinates indicate the presence or absence of words from a dictionary.

We considered a real data set containing chemical compounds acting on 5-HT<sub>1A</sub> receptor ligands (Warszycki et al. 2013). This is one of the proteins responsible for the regulation of the central nervous system. This data set was manually labeled by an expert in a hierarchical form. We narrowed that classification tree down to 5 classes: tetralines, alkylamines, piperidines, aimides and other piperazines. Each compound was represented by its Klekota–Roth fingerprint, which encodes 4860 chemical patterns in a binary vector (Yap 2011).

We also took a molecular biology data set (splice), which describes primate splice-junction gene sequences. Moreover, we used a data set containing mushrooms described in terms of physical characteristics, where the goal is to predict whether a mushroom is poisonous or edible. Both data sets were selected from the UCI repository.

Finally, we evaluated all methods on the MNIST data set (LeCun et al. 1998), which is a collection of handwritten digits made into gray scale. To produce binary images, pixels with intensity greater than 0 were turned into bit 1.

We considered two types of Bernoulli mixture models. The first one is a classical mixture model, which relies on the maximum likelihood principle (ML) (Elmore et al. 2004). We used the R package “mixtools” (Benaglia et al. 2009) for its implementation. The second method is based on classification maximum likelihood (CML) (Li et al. 2004). While ML models every data point as a sample from a mixture of probability distributions, CML assigns every example to a single component. CML coincides with applying entropy as a clustering criterion (Celeux and Govaert 1991).

We also used two distance-based algorithms. The first one is *k*-medoids (Huang 1998), which focuses on minimizing the average distance between data points and the corresponding clusters’ medoids (a generalization of a mean). We used the implementation of *k*-medoids from the cluster R package<sup>8</sup> with a Jaccard similarity measure.<sup>9</sup> We also considered the Cluto software (Karypis 2002), which is an optimized package for clustering large data sets. We ran the algorithm “direct” with a cosine distance function,<sup>10</sup> which means that the package will calculate the final clustering directly, rather than bisecting the data multiple times.

The aforementioned methods are designed to deal with binary data. To apply a typical clustering technique to such data, one could use a method of dimensionality reduction as a preprocessing step. In the experiments, we used PCA to reduce the data dimension to 100 principal components. Next, we used the *k*-means algorithm on the reduced space.

Finally, we used two subspace clustering methods: ORGEN (You et al. 2016) and LSR (Lu et al. 2012), which are often applied to continuous data. In the case of high dimensional data, LSR reduces their dimension with the use of PCA. In our experiments, we reduced the data to 100 principal components.

<sup>8</sup> <https://cran.r-project.org/web/packages/cluster>.

<sup>9</sup> We also considered the Hamming and cosine distances, but the Jaccard distance provided the best results.

<sup>10</sup> <http://glaros.dtc.umn.edu/gkhome/fetch/sw/cluto/manual.pdf>.

Since all of the aforementioned methods are non-deterministic (their results depends on the initialization), each was run 50 times with different initial partitions and the best result was selected according to the method's inner metric.

All methods were run with the correct number of groups. Since the expected number of groups was given, SPARSEMIX was run with  $\beta = 0$  to prevent the clusters' reduction. We examined its two parametrizations: (a)  $T = \frac{1}{2}$ , where a cluster representative is taken as the most probable point; (b)  $T = 1$ , where a representative is a zero vector.

The results presented in Table 2 show significant disproportions between the two best performing methods (SPARSEMIX and Cluto) and the other examined algorithms. The highest differences can be observed in the case of 7newsgroups and farm-adds data sets. Due to a large number of examples or attributes, most algorithms, except SPARSEMIX, Cluto and PCA-k-means, failed to run on the 20newsgroups and Reuters data, where only SPARSEMIX produced reasonable results. In the case of the questions and sentiment data sets, neither method showed results significantly better than a random partitioning. Let us observe that these sets are extremely sparse, which could make the appropriate grouping of their examples very difficult. In the mushroom example, on the other hand, all methods seemed to perform equally good. Slightly higher differences can be observed on MNIST and the chemical data sets, where ML and CML obtained good results. Finally, SPARSEMIX with  $T = \frac{1}{2}$  significantly outperformed other methods for splice. Let us observe that both subspace clustering techniques were inappropriate for clustering this type of data. Moreover, reducing the data dimension by PCA led to the loss of information and, in consequence, recovering the true clustering structure was impossible.

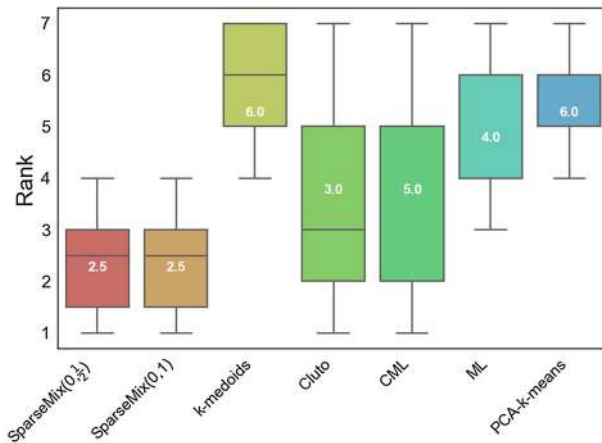
Although SPARSEMIX, ML and CML focus on optimizing similar cost functions, they use different algorithms, which could be the main reason for the differences in their results. SPARSEMIX applies an on-line Hartigan procedure, which updates clusters parameters at every switch, while ML and CML are based on the EM algorithm and perform updates after an entire iteration. On-line updates allow for better model fitting and, in consequence, lead to finding better local minima. This partially explains the more accurate clustering results of SPARSEMIX compared to related mixture models.

To summarize the results, we ranked the methods on each data set (the best performing method got rank 1, second best got rank 2, etc.). Figure 6 presents a box plot of the ranks averaged over all data sets. The vertical lines show the range of the ranks, while the horizontal line in the middle denotes the median. It can be seen that both variants of SPARSEMIX were equally good and outperformed the other methods. Although the median rank of cluto was only slightly worse, its variance was significantly higher. This means that this model was not well suited for many data sets.

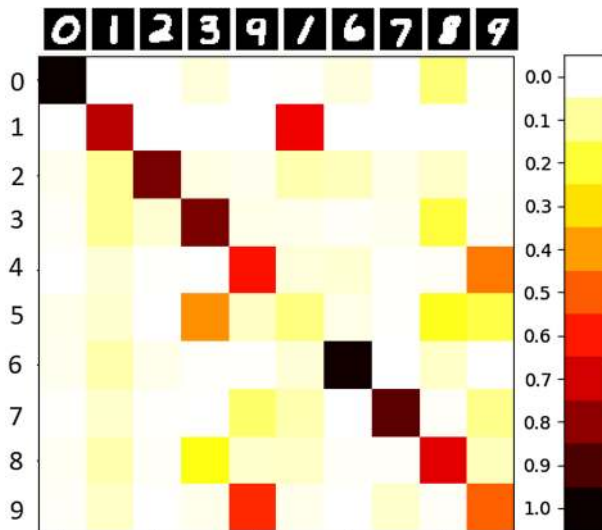
To further illustrate the effects of SPARSEMIX we present its detailed results obtained on the MNIST data set. Figure 7 shows a confusion matrix and clusters' representatives (first row) produced by SPARSEMIX with  $T = \frac{1}{2}$ . It is clear that most of the clusters' representatives resemble actual hand-drawn digits. It can be seen that SPARSEMIX had trouble distinguishing between the digits 4 and 9, mixing them up a bit in their respective clusters. The digit 5 also could not be properly separated, resulting in its scatter among other clusters. The digit 1 occupied two separate clusters, once for being written vertically and once for being written diagonally. Nevertheless, this example

**Table 2** Adjusted Rand Index of the considered methods. Parameter  $\beta$  was fixed at 0 to avoid deleting clusters

Data set	SPARSEMIX( $\frac{1}{2}$ )	SPARSEMIX(1)	ML	CML	k-medoids	Cluto	L <sub>SR</sub>	ORGEN	PCA-k-means
Reuters	<b>0.1399</b>	0.1316	-	-	-	-0.0454	-	-	0.0752
20newsgroup	0.1632	<b>0.4916</b>	-	-	-	0.057	-	-	0.045
7newsgroups	0.6337	<b>0.8293</b>	0.0948	0.0991	0.0012	0.7509	0.177	-	0.1029
Farm-ads	0.1957	<b>0.2664</b>	0.0552	0.0468	0.0192	0.2564	0.0095	-	-0.0039
Questions	0.0622	0.0622	0.0274	0.0087	0.03629	<b>0.0925</b>	0.0475	0.001	0.0299
Sentiment	<b>0.0667</b>	<b>0.0667</b>	0.0198	0.0064	0.0153	0.0571	0.0167	0	0.0083
SMS	0.5433	0.5433	0.3133	0.3063	0.1081	<b>0.5748</b>	-	0	0.127
Chemical	0.3856	0.4281	0.4041	<b>0.4472</b>	0.3724	0.3841	0.1745	0.128	0.3068
Mushroom	<b>0.6354</b>	0.6275	0.6120	<b>0.6354</b>	0.626	0.6228	-	0.2124	0.6205
Splice	<b>0.7216</b>	0.2607	0.2442	0.4885	0.1071	0.1592	0.0005	0.0005	0.2414
mnist	<b>0.4501</b>	0.395	0.4171	0.4277	0.3612	0.283	-	0.3763	0.3792



**Fig. 6** Ranking of the examined methods. Reuters and 20newsgroups were not taken into account, because most of the methods were unable to process this data. LSR and ORGEN were removed from this analysis since they failed to run on some of the data sets



**Fig. 7** Heat map of confusion matrix and clusters' representatives (first row) returned by applying SPARSEMIX to the MNIST data set. Rows correspond to reference digits, while columns correspond to clusters produced by SPARSEMIX

showed that SPARSEMIX is able to find reasonable clusters representatives that reflect their content in a strictly unsupervised way.



## 5.2 Detecting the number of clusters

In practice, the number of clusters may be unknown and a clustering method has to detect this number. In this section, we compare the performance of our method with a typical model-based approach when the number of clusters is unknown.

Our method uses the cost of clusters identification, which is a regularization term and allows for reducing redundant clusters on-line. In consequence, SPARSEMIX starts with a number of clusters defined by the user and removes unnecessary groups in the clustering process. Final effects depend on the selection of the parameter  $\beta$ , which controls the trade-off between the quality of the model and its complexity. For high  $\beta$ , the cost of maintaining clusters could outweigh the cost of coding elements within clusters and SPARSEMIX collapses them to a single cluster. On the other hand, for small  $\beta$  the cost of coding clusters identifiers could be irrelevant. Since the number of clusters is usually significantly lower than the number of non-zero attributes in a vector, we should put  $\beta$  greater than 1. Intuitively, both terms will be balanced for  $\beta$  close to the average number of non-zero bits in a data set. We will verify this hypothesis in the following experiment.

To analyze the impact of  $\beta$  on the clustering results, we ran SPARSEMIX with 10 groups for  $\beta = 0, 1, 2, \dots$ . We reported the final number of clusters and ARI for the returned partitions. The results presented in Fig. 8 indicate that the process of clusters' reduction does not have a negative effect on clusters quality. Indeed, ARI values obtained when SPARSEMIX ended with the correct number of groups were close to the ones obtained for  $\beta = 0$  when clusters reduction was not used<sup>11</sup> (see Table 2). The Figure also demonstrates that to enable cluster reduction,  $\beta$  should be lower than the average number of non-zero entries in a data set. Nevertheless, we were unable to provide a strict rule for selecting its optimal value.

In model-based clustering the number of clusters is usually chosen using information measures, such as Bayesian Information Criterion (BIC), or using related approaches, such as maximum a posteriori estimation (MAP), minimum message length (MML) or minimum description length principle (MDL). A typical approach relies on evaluating a clustering method on a data set with various numbers of clusters  $k$  and calculating the likelihood function combined with the complexity of the model  $\mathcal{M}$ :

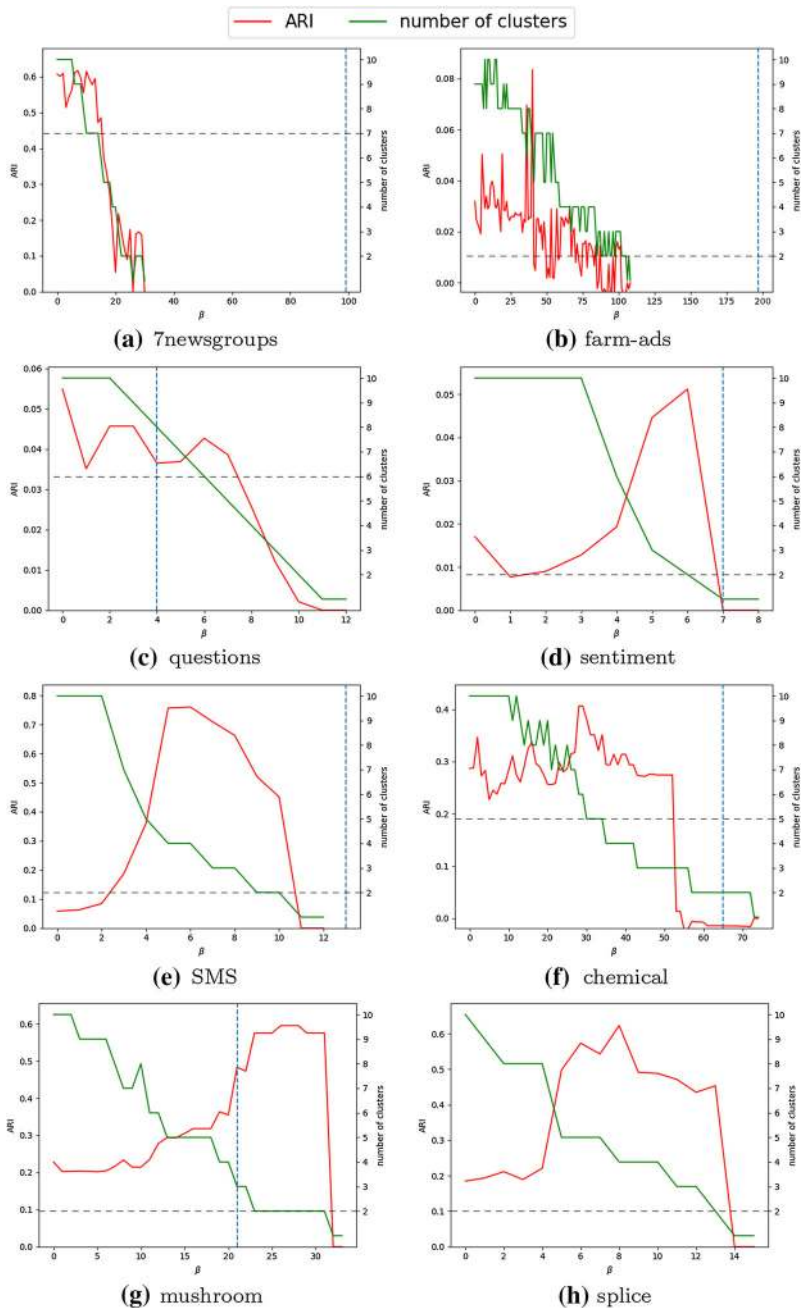
$$likelihood(\mathcal{M}) + \lambda penalty(\mathcal{M}),$$

where  $\lambda$  is a user-defined parameter. The optimal partition is the one with a minimal value of the above criterion function. Such an approach was analyzed by Bontemps et al. (2013) and implemented in the R package ClustMMDD<sup>12</sup> for the mixture of Bernoulli models. We use this method with a penalty function given by:

$$penalty(\mathcal{M}) = \frac{1}{2} \dim(\mathcal{M}) \log(n)$$

<sup>11</sup> exceptions are farm-ads and splice data.

<sup>12</sup> <https://cran.r-project.org/web/packages/ClustMMDD/index.html>.



**Fig. 8** Influence of  $\beta$  on clustering results. SPARSEMIX was initialized with 10 groups. We reported the final number of groups (green) and the corresponding ARI score (red). The vertical blue line denotes the average number of non-zero bits in a data set while the horizontal dotted line indicates the ground-truth number of clusters (Color figure online)

where  $\dim(\mathcal{M})$  is the number of free parameters of the model  $\mathcal{M}$ . This criterion corresponds to BIC for  $\lambda = 1$ .

The results illustrated in the Fig. 9 show a high sensitivity of this approach to the change of trade-off parameter  $\lambda$ . Although an additional penalty term allows to select the best model, there is no unified methodology for choosing the parameter<sup>13</sup>  $\lambda$ . In the case of a typical Gaussian mixture model, an analogical BIC formula does not contain additional parameters, which makes it easy to use in practice. However, for the Bernoulli mixture model, the analogical approach needs to specify the values of parameter  $\lambda$ , which could limit its practical usefulness.

In conclusion, SPARSEMIX allows for an effective reduction of the number of clusters by selecting a  $\beta$  close to the average number of non-zero bits in a data set. In contrast, selecting a reasonable value of the trade-off parameter in penalty criteria for model-based clustering may be more difficult. Nevertheless, the number of clusters in a ground truth partition does not have to determine the optimal partition for a given clustering task. One could also find a “natural” clustering structure, which is different from the reference partition, but reveals interesting patterns in the data set.

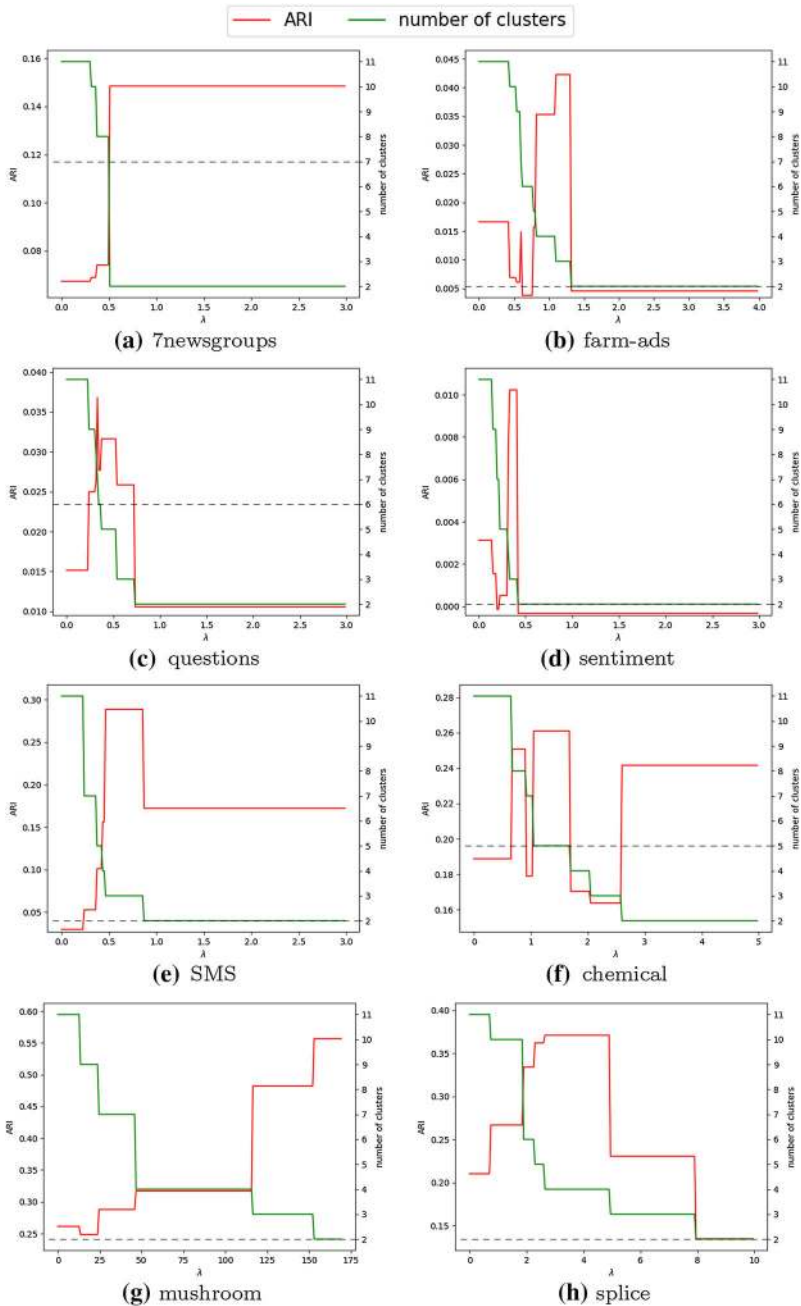
### 5.3 Time comparison

In real-world applications, the clustering algorithm has to process large portions of data in a limited amount of time. In consequence, high computational complexity may disqualify a method from practical usage. In this experiment we focus on comparing the evaluation time of our algorithm with other methods. We tested the dependence on the number of data points as well as on the number of attributes. For this illustration, we used the chemical data set from the previous subsection. From now on, we consider only algorithms which work on the original data directly, without additional preprocessing.

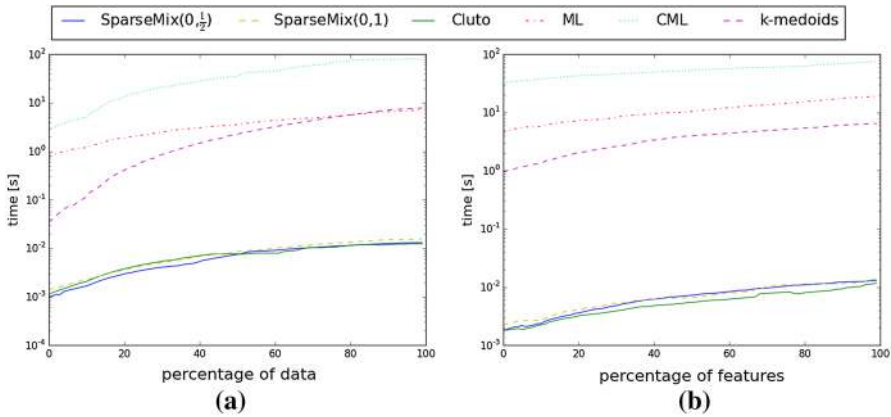
In the first scenario, we randomly selected a subset of the data containing a given percentage of instances, while in the second simulation, we chose a given percentage of attributes. The clustering algorithms were run on such prepared subsets of data. The results presented in Fig. 10 show that both versions of SPARSEMIX were as fast as the Cluto package, which is an optimized software for processing large data sets. The other algorithms were significantly slower. It might be caused both by a specific clustering procedure as well as by an inefficient programming language used for their implementations.

The interesting thing is that SPARSEMIX with  $T = \frac{1}{2}$  was often slightly faster than SPARSEMIX with  $T = 1$ , which at first glance contradicts the theoretical analysis of our algorithm. To investigate this observation, we counted the number of iterations needed for convergence of both methods. It is clear from Fig. 11 that SPARSEMIX with  $T = \frac{1}{2}$  needed less iterations to find a local minimum than with  $T = 1$ , which fully explains the relation between their running times. SPARSEMIX with  $T = \frac{1}{2}$  needed less than 20 iterations to converge. Since the scale of the graph is logarithmic, the difference in its cost decreased exponentially. Such a fast convergence follows from the fact that the SPARSEMIX cost function can be optimized by applying an on-line

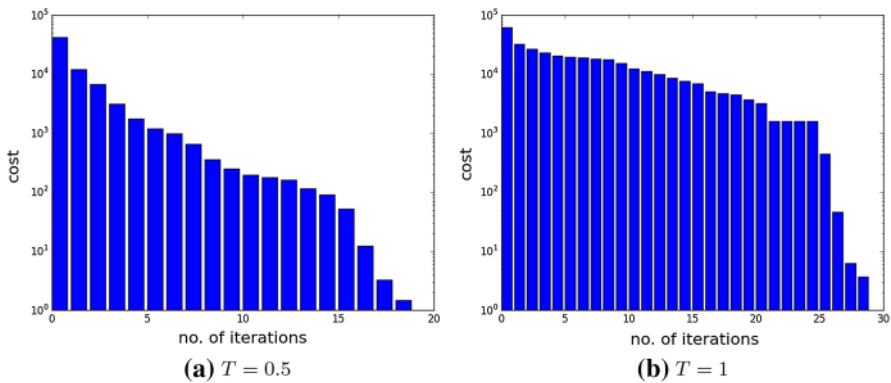
<sup>13</sup> see Bontemps et al. (2013) for specific heuristics.



**Fig. 9** Detecting the number of clusters by ML approach. We reported the optimal number of groups (green) and the corresponding ARI score (red) for various weights  $\lambda$  attached to a penalty term (see Sect. 5.2 for details). The horizontal dotted line indicates the ground-truth number of clusters (Color figure online)



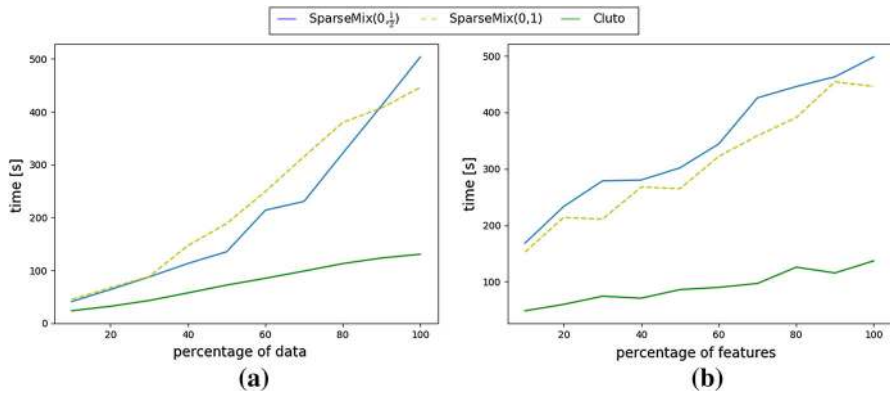
**Fig. 10** The running times on chemical data set with respect to the number of points and attributes given in logarithmic scale. Parameter  $\beta$  was fixed at 0 to match the previous experiments on this data set



**Fig. 11** The difference between the cost in each iteration and the cost of the final clustering of SPARSEMIX with  $T = 0.5$  11a and  $T = 1$  11a, given in logarithmic scale

Hartigan algorithm (it is computationally impossible to use an on-line strategy for CML or ML models).

To further analyze the efficiency of our algorithm, we performed an analogical experiment on the Reuters data. We ran only SPARSEMIX and Cluto because other algorithms were unable to process such a large data set. The results presented in Fig. 12 show some disproportion between the considered methods. One can observe that our algorithms were 2–5 times slower than Cluto. Nevertheless, Fig. 12 demonstrates that the complexity of SPARSEMIX is close to linear with respect to the number of data points and the number of attributes. In consequence, more efficient implementation should increase the speed of our method.



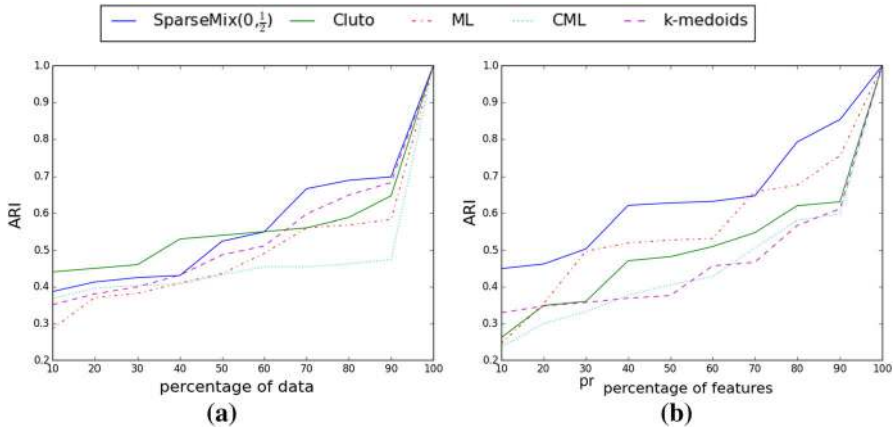
**Fig. 12** The running times of SPARSEMIX and Cluto on Reuters data set with respect to the number of points 10a and to the number attributes 10b.  $\beta$  was fixed at 0 to match the previous experiments on this dataset

## 5.4 Clustering stability

In this experiment, we examined the stability of the considered clustering algorithms in the presence of changes in the data. More precisely, we tested whether a method was able to preserve clustering results when some data instances or attributes were removed. In practical applications, high stability of an algorithm can be used to speed up the clustering procedure. If a method does not change its result when using a lower number of instances or attributes, we can safely perform clustering on a reduced data set and assign the remaining instances to the nearest clusters. We again used the chemical data set for this experiment. In this simulation we only ran SPARSEMIX with  $T = \frac{1}{2}$  (our preliminary studies showed that parameter  $T$  does not visibly influence overall results).

First, we investigated the influence of the number of instances on the clustering results. For this purpose, we performed the clustering of the whole data set  $X$  and randomly selected  $p$  percent of its instances  $X^p$  (we considered  $p = 0.1, 0.2, \dots, 0.9$ ). Stability was measured by calculating the ARI between the clusters  $X_1^p, \dots, X_k^p$  created from the selected fraction of data  $X^p$  and from the whole data set (restricted to the same instances), i.e.  $(X_1 \cap X^p), \dots, (X_k \cap X^p)$ . To reduce the effect of randomness, this procedure was repeated 5 times and the final results were averaged. The results presented in Fig. 13a show that for a small number of data points Cluto gave the highest stability, but as the number of instances grew, SPARSEMIX performed better.

In the second part of the experiment, we examined how the clustering results changed when a smaller number of attributes were taken into account. The procedure was analogical to the previous one: we compared the clustering results obtained on the whole data set with the ones produced on data set with randomly selected  $p$  percent of attributes (as before we considered  $p = 0.1, 0.2, \dots, 0.9$ ). One can observe in Fig. 13b that SPARSEMIX obtained the highest stability on all subsets of data. The performance of Cluto was significantly worse than previously – in particular, ML showed a higher stability.



**Fig. 13** Compatibility between clustering results obtained on the whole data set and its fixed percentage. Changes in  $\beta$  and  $T$  provided similar results

### 5.5 Sensitivity to imbalanced data

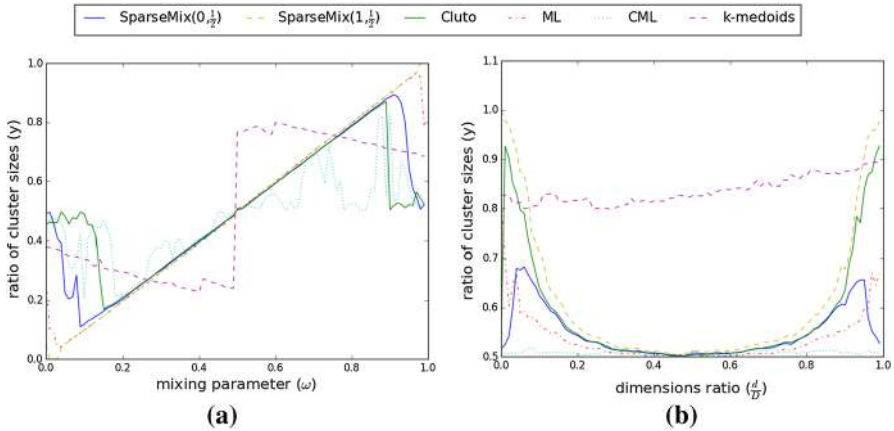
In the following section, we examined the sensitivity of the clustering algorithms to data imbalance, which very often proves to be a crucial trait of an algorithm (Chawla 2009). This extends the theoretical analysis presented in Example 1.

First, we examined whether the algorithm is able to detect clusters of different sizes. For this purpose, we considered a data set  $X \subset \{0, 1\}^D$ , for  $D = 100$  and  $|X| = 1000$ , generated from a distribution

$$\omega P(p, \alpha, d) + (1 - \omega)P(p, 1 - \alpha, d),$$

where  $p = 0.1$ ,  $\alpha = 0.05$  and  $d = \frac{D}{2}$  were fixed and  $\omega$  changed from 0 to 1. We refer the reader to Example 1 for the definition of the distribution  $P$  and its interpretation. The mixing parameter  $\omega$  induces the fraction of examples produced by these two sources. We would expect that a clustering method will be able to discover true distributions, so the resulting sizes of the clusters will be, roughly,  $\omega|X|$  and  $(1 - \omega)|X|$ . However, as  $\omega$  approaches either to 0 or 1, the data becomes very imbalanced, which makes the task of separating them more difficult. We considered SPARSEMIX with  $\beta = 0$  and  $\beta = 1$  to account for different costs of maintaining clusters (our preliminary studies showed that parameter  $T$  does not visibly influence overall results and thus we used  $T = \frac{1}{2}$ ).

Figure 14a reports the fraction of the data that belongs to the first cluster. The optimal solution is  $y = \omega$ . We can see that the  $k$ -medoids method did not respond to the changes in  $\omega$ . Other algorithms seemed to perform well on the mid section, but gradually steered off the optimal line as  $\omega$  approached 0 or 1. The highest robustness to imbalanced data was obtained by ML and SPARSEMIX with  $\beta = 1$  (cost of clusters identification was taken into account). If the cost of maintaining clusters is not considered ( $\beta = 0$ ), then SPARSEMIX tends to create more balanced groups. These results are consistent with a discussion outlined before Definition 2 and in Example 1.



**Fig. 14** The ratio of cluster sizes for data sets generated from imbalanced sources: we varied the number of instances generated from each source **a** and the number of attributes characteristic for each source **b**

In the second experiment, we investigated the influence of attribute imbalance on the clustering results. For this purpose we sampled a data set from the mixture of distributions given by:

$$\frac{1}{2}P(p, \alpha, d) + \frac{1}{2}P(p, 1 - \alpha, d),$$

where  $p = 0.1$ ,  $\alpha = 0.05$ ,  $|X| = 1000$  and  $D = 100$  were constants, while  $d$  ranged from 0 to  $D$ . When  $d < D$ , then the second source is identified by a smaller number of bits than the first one. Therefore, by changing the value of the parameter  $d$  we scale the number of features characteristic for each component. This time, we expect that the clusters will remain equally-sized regardless of the parameter  $d$ .

Figure 14b presents the fraction of data that belongs to the first cluster (perfect solution is given by  $y = \frac{1}{2}$ ). It can be observed that SPARSEMIX with  $\beta = 1$  was very sensitive to attribute imbalance. According to the conclusion given in Example 1, the cost of encoding elements within a cluster is outweighed by the cost of clusters' identification, as  $\alpha \rightarrow 0$  (or 1), which results in the reduction of the lighter group. Since the data is sampled from an underlying distribution and SPARSEMIX flows to a local minimum, some attempts result in creating one group, while the others produce two clusters, which explains why the corresponding line is not equal to 1, for  $\alpha < 0.2$ . This effect was not apparent when SPARSEMIX used  $\beta = 0$ , because there was no cost of creating an additional cluster. Its results were comparable to ML and CML, which also do not use any cost of clusters' identification.

## 6 Conclusion

In this paper, we proposed SPARSEMIX, a new approach for the clustering of sparse high dimensional binary data. Our results showed that SPARSEMIX is not only more



accurate than related model-based clustering algorithms, but also significantly faster. Its evaluation time is comparable to algorithms implemented in the Cluto package, the software optimized for processing large data sets, but its clusters quality is better. SPARSEMIX provides a description of each cluster by its representative and the dispersion from this representative. Experimental results demonstrated that representatives obtained for the MNIST data set provide high resemblance to the original examples of handwritten digits. The model was theoretically analyzed.

**Acknowledgements** This research was partially supported by the National Science Centre (Poland) grant no. 2016/21/D/ST6/00980 and grant no. 2017/25/B/ST6/01271.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

### A Proof of Theorem 1

We will show that  $C_{T_2}(X) - C_{T_1}(X) \geq 0$ . We have:

$$\begin{aligned}
 C_{T_2}(X) - C_{T_1}(X) &= \sum_{i=1}^D (q_i(T_1) \log Q_i(T_1) - q_i(T_2) \log Q_i(T_2)) \\
 &= \sum_{i:p_i \leq T_1} \left( p_i \log \frac{p_i}{Z(T_1)} - p_i \log \frac{p_i}{Z(T_2)} \right) \\
 &\quad + \sum_{i:T_1 \leq p_i \leq T_2} \left( (1 - p_i) \log \frac{1 - p_i}{Z(T_1)} - p_i \log \frac{p_i}{Z(T_2)} \right) \\
 &\quad + \sum_{i:p_i \geq T_2} \left( (1 - p_i) \log \frac{1 - p_i}{Z(T_1)} - (1 - p_i) \log \frac{1 - p_i}{Z(T_2)} \right) \\
 &= \log \frac{Z(T_2)}{Z(T_1)} \left( \sum_{i:p_i \leq T_1} p_i + \sum_{i:p_i \geq T_2} (1 - p_i) \right) \\
 &\quad + \sum_{i:T_1 \leq p_i \leq T_2} (1 - p_i) \log(1 - p_i) - p_i \log p_i \\
 &\quad + \sum_{i:T_1 \leq p_i \leq T_2} p_i \log Z(T_2) - (1 - p_i) \log Z(T_1).
 \end{aligned}$$

Observe that  $Z(T_1) \leq Z(T_2)$  and thus  $\log \frac{Z(T_2)}{Z(T_1)} \geq 0$ . Consequently,

$$\begin{aligned}
 C_{T_2}(X) - C_{T_1}(X) &\geq \sum_{i:T_1 \leq p_i \leq T_2} ((1 - p_i) \log(1 - p_i) - p_i \log p_i \\
 &\quad + p_i \log Z(T_2) - (1 - p_i) \log Z(T_1)).
 \end{aligned}$$

The above expression is non-negative if the function:

$$f(p) = p(-\log p) - (1-p)(-\log(1-p)) + p \log Z(T_2) - (1-p) \log Z(T_1)$$

is non-negative for every  $T_1 \leq p \leq T_2$ .

A derivative of  $f$  equals:

$$\begin{aligned} f'(p) &= -\log p(1-p) + \log(Z(T_1)Z(T_2)) - 2 \\ &= -\log p(1-p) + \log \frac{Z(T_1)Z(T_2)}{4}. \end{aligned}$$

It is greater than zero when:

$$\log \frac{Z(T_1)Z(T_2)}{4} \geq \log p(1-p),$$

which simplifies to:

$$\frac{Z(T_1)Z(T_2)}{4} \geq p(1-p).$$

Since  $Z(T_2) \geq Z(T_1) \geq 1$ , then  $Z(T_1)Z(T_2) \geq 1$ . Moreover, due to the fact that  $p(1-p) \leq \frac{1}{4}$ , for  $p \in [0, 1]$ , we have

$$\frac{Z(T_1)Z(T_2)}{4} \geq \frac{1}{4} \geq p(1-p),$$

which means that for every  $p$  satisfying  $T_1 \leq p \leq T_2$  the function  $f$  is nondecreasing. Finally,

$$f\left(\frac{1}{2}\right) = \frac{1}{2} \log \frac{Z(T_2)}{Z(T_1)} \geq 0,$$

and consequently,  $f(p) \geq 0$ , for  $\frac{1}{2} \leq T_1 \leq p \leq T_2 \leq 1$ . This means that the best compression is achieved for  $T = \frac{1}{2}$ .

## B Clusters reduction—details of Example 1

We compare the cost of using a single cluster for all instances with the cost of splitting the data into two optimal groups (first  $\omega|X|$  examples are assigned to the first group while the remaining instances are assigned to the second cluster). For the convenience of calculations, we define the function:

$$D(x, d) := xd + (1-x)(D-d),$$

The conditional probability  $Q_i^1$  that the  $i$ -th position holds a non-zero value in the first cluster equals:

$$Q_i^1 = \begin{cases} \frac{\alpha}{D(\alpha, d)}, & j = 1, \dots, d, \\ \frac{1-\alpha}{D(\alpha, d)}, & j = d + 1, \dots, D. \end{cases}$$

while for the second group:

$$Q_i^2 = \begin{cases} \frac{1-\alpha}{D(1-\alpha, d)}, & j = 1, \dots, d, \\ \frac{\alpha}{D(1-\alpha, d)}, & j = d + 1, \dots, D. \end{cases}$$

Then, the cost of using two clusters equals:

$$\begin{aligned} C(X_1, X_2) &= \omega(-\log \omega) + (1 - \omega)(-\log(1 - \omega)) \\ &\quad - \omega \left( d\alpha p \log \frac{\alpha}{D(\alpha, d)} + (D - d)(1 - \alpha)p \log \frac{1 - \alpha}{D(\alpha, d)} \right) \\ &\quad - (1 - \omega) \left( d(1 - \alpha)p \log \frac{1 - \alpha}{D(1 - \alpha, d)} + (D - d)\alpha p \log \frac{\alpha}{D(1 - \alpha, d)} \right) \\ &= p(\omega D(\alpha, d) \log D(\alpha, d) + (1 - \omega)D(1 - \alpha, d) \log D(1 - \alpha, d) \\ &\quad - \alpha D(\omega, d) \log \alpha - (1 - \alpha)D(1 - \omega, d) \log(1 - \alpha)) + h(\omega, 1 - \omega). \end{aligned} \tag{11}$$

To calculate the cost of one cluster, let us put  $\beta = \omega\alpha + (1 - \omega)(1 - \alpha)$ . Then,  $(1 - \beta) = \omega(1 - \alpha) + (1 - \omega)\alpha$  and the conditional probability  $Q_i$  is given by

$$Q_i = \begin{cases} \frac{\beta}{D(\beta, d)}, & j = 1, \dots, d, \\ \frac{1-\beta}{D(\beta, d)}, & j = d + 1, \dots, D. \end{cases}$$

The cost of one cluster can be written as follows:

$$\begin{aligned} C(X) &= -dp\beta \log \frac{\beta}{D(\beta, d)} - (D - d)p(1 - \beta) \log \frac{1 - \beta}{D(\beta, d)} \\ &= p(D(\beta, d) \log D(\beta, d) - d\beta \log \beta - (D - d)(1 - \beta) \log(1 - \beta)). \end{aligned} \tag{12}$$

It is more profitable to use one cluster instead of two if (12) is lower than (11). Since it is difficult to analyze this relation in general, we consider three special cases:

1. *Dimensions are balanced.* We fix the dimension parameter  $d = \frac{1}{2}D$ . Then  $D(\alpha, d) = D(\omega, d) = D(\beta, d) = d$  and the formula (11) simplifies to:

$$C(X_1, X_2) = pd(\log d + h(\alpha, 1 - \alpha)) + h(\omega, 1 - \omega),$$

while (12) equals:

$$C(X) = pd(h(\beta, 1 - \beta) + \log d).$$

2. *Sources are balanced.* If we fix the mixing proportion  $\omega = \frac{1}{2}$  then the cost of two clusters is:

$$C(X_1, X_2) = -\frac{1}{2}p(h(D(\alpha, d), D(1 - \alpha, d)) + Dh(\alpha, 1 - \alpha)) + \log 2$$

and for one cluster we have

$$C(X) = \frac{1}{2}pD \log D.$$

3. *Both dimensions and sources are balanced.* For fixed  $d = \frac{1}{2}D$  and  $\omega = \frac{1}{2}$  the cost of two clusters is given by

$$C(X_1, X_2) = dp(h(\alpha, 1 - \alpha) + \log d) + \log 2,$$

while for one cluster we have

$$C(X) = pd \log D.$$

## C Quality of clusters—detailed results

Besides the values of ARI presented in the main part of the paper, we also used two additional measures to verify the quality of the results: Normalized Mutual Information (NMI) (Vinh et al. 2010) and Clustering Accuracy (He et al. 2006).

Mutual Information (MI) between two partitions  $\mathcal{X}$ ,  $\mathcal{Y}$  is defined by:

$$MI(\mathcal{X}, \mathcal{Y}) = \sum_{X_i \in \mathcal{X}} \sum_{Y_j \in \mathcal{Y}} p(X_i, Y_j) \log \frac{p(X_i, Y_j)}{p(X_i)p(Y_j)},$$

where  $p(X_i)$  is the probability that a randomly selected data point belongs to cluster  $X_i$  and  $p(X_i, Y_j)$  is the joint probability that an arbitrary data point belongs to the clusters  $X_i$  as well as  $Y_j$  at the same time. Since MI can take values greater than 1, NMI is its normalization:

$$NMI(\mathcal{X}, \mathcal{Y}) = \frac{MI(\mathcal{X}, \mathcal{Y})}{\sqrt{H(\mathcal{X}) \cdot H(\mathcal{Y})}},$$

where  $H(\mathcal{X}) = -\sum_{X_i \in \mathcal{X}} p(X_i) \log p(X_i)$  is the entropy of  $\mathcal{X}$ . The maximal value, 1, is attained for identical partitions, but random groupings might result in a value greater than 0.

**Table 3** Normalized Mutual Information of the considered methods. Parameter  $\beta$  was fixed at 0 to avoid deleting clusters

Data set	SPARSEMIX( $\frac{1}{2}$ )	SPARSEMIX(1)	ML	CML	k-medoids	Cluto	LSR	ORGEN	PCA-k-means
reuters	0.3485	<b>0.3545</b>	-	-	-	0.217	-	-	0.3301
20newsgroup	0.3667	<b>0.6761</b>	-	-	-	0.2525	-	-	0.147
7newsgroups	0.6836	<b>0.8165</b>	0.1697	0.1723	0.018	0.6919	0.3852	-	0.1771
Farm-ads	0.2208	0.2201	0.048	0.0821	0.0388	<b>0.2854</b>	0.0677	-	0.0611
Questions	0.088	0.088	0.082	0.0193	0.1385	<b>0.1581</b>	0.1549	0.0127	0.1333
Sentiment	<b>0.0499</b>	<b>0.0499</b>	0.026	0.0058	0.0221	0.0433	0.018	0.0027	0.014
SMS	0.4423	0.4423	0.2382	0.2149	0.081	<b>0.4561</b>	-	-	0.0675
Chemical	0.5462	0.5072	0.5221	<b>0.5652</b>	0.5142	0.5245	0.281	0.3038	0.3986
Mushroom	0.5867	<b>0.6055</b>	0.5971	0.5867	0.5841	0.5794	-	0.3119	0.573
Splice	<b>0.6151</b>	0.1985	0.2952	0.3523	0.084	0.1216	0.019	0.0179	0.1841
mnist	<b>0.6636</b>	0.6139	0.5812	0.6625	0.4821	0.3087	-	0.5694	0.5033

**Table 4** Clustering Accuracy of the considered methods. Parameter  $\beta$  was fixed at 0 to avoid deleting clusters

Data set	SPARSEMIX( $\frac{1}{2}$ )	SPARSEMIX(1)	ML	CML	k-medoids	Cluto	LSR	ORGEN	PCA-k-means
reuters	0.2973	0.2965	-	-	-	<b>0.3574</b>	-	-	0.2411
20newsgroup	0.2772	<b>0.5826</b>	-	-	-	0.1004	-	-	0.1559
7newsgroups	0.7553	<b>0.9183</b>	0.1577	0.1589	0.1447	0.8176	0.4893	-	0.3192
Farm-ads	0.6811	<b>0.7581</b>	0.5781	0.5623	0.5332	0.7327	0.558	-	0.5049
Questions	0.3073	0.3073	0.2619	0.2048	0.3043	<b>0.3785</b>	0.3617	0.2385	0.3039
Sentiment	<b>0.604</b>	<b>0.604</b>	0.571	0.539	0.563	0.573	0.566	0.503	0.547
SMS	0.8792	0.8792	0.8546	0.8248	0.7642	<b>0.9084</b>	-	0.864	0.7158
Chemical	0.5767	0.5527	0.5671	<b>0.5942</b>	0.5056	0.5789	0.4188	0.4529	0.5273
Mushroom	0.8938	<b>0.8995</b>	0.891	0.8938	0.8956	0.8948	-	0.7307	0.8939
Splice-1	<b>0.9247</b>	0.7483	0.739	0.8751	0.5724	0.6542	0.5219	0.5216	0.7458
mnist	0.6847	<b>0.6856</b>	0.5827	0.6732	0.5012	0.5356	-	0.5195	0.5548

Clustering Accuracy (AC) is an adaptation of the classical classification accuracy to the case of unlabeled clusters. Let  $r_i$  and  $s_i$  be cluster labels for a data point  $x_i \in X$  provided by two partitions  $\mathcal{X}, \mathcal{Y}$ . AC is defined by:

$$AC = \frac{\sum_{x_i \in X} \delta(s_i, \text{map}(r_i))}{|X|},$$

where  $\delta(x, y) = 1$ , for  $x = y$  and 0 otherwise, while  $\text{map}(r_i)$  is the permutation function that maps each cluster label  $r_i$  in  $\mathcal{X}$  to the equivalent label in  $\mathcal{Y}$  in such a way as to maximize the above criteria. The optimal permutation can be feasibly found using the Hungarian Algorithm.

The results presented in Tables 3 and 4 are highly correlated with the ones reported in Table 2.

## References

- Asuncion DNA (2007) UCI machine learning repository
- Bai L, Liang J, Dang C, Cao F (2011) A novel attribute weighting algorithm for clustering high-dimensional categorical data. *Pattern Recognit* 44(12):2843–2861
- Baker LD, McCallum AK (1998) Distributional clustering of words for text classification. In: *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*. ACM, pp 96–103
- Barbará D, Li Y, Couto J (2002) Coolcat: an entropy-based algorithm for categorical clustering. In: *Proceedings of the eleventh international conference on information and knowledge management*. ACM, pp 582–589
- Baxter RA, Oliver JJ (2000) Finding overlapping components with mml. *Stat Comput* 10(1):5–16
- Benaglia T, Chauveau D, Hunter D, Young D (2009) mixtools: an r package for analyzing finite mixture models. *J Stat Softw* 32(6):1–29
- Bontemps D, Toussile W et al (2013) Clustering and variable selection for categorical multivariate data. *Electron J Stat* 7:2344–2371
- Bouguila N (2010) On multivariate binary data clustering and feature weighting. *Comput Stat Data Anal* 54(1):120–134
- Bouguila N, ElGuebaly W (2009) Discrete data clustering using finite mixture models. *Pattern Recognit* 42(1):33–42
- Bouguila N, Ziou D (2007) High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length. *IEEE Trans Pattern Anal Mach Intell* 29(10):1716–1731
- Bozdogan H, Sclove SL (1984) Multi-sample cluster analysis using akaike's information criterion. *Ann Inst Stat Math* 36(1):163–180
- Cagnone S, Viroli C (2012) A factor mixture analysis model for multivariate binary data. *Stat Model* 12(3):257–277
- Celeux G, Govaert G (1991) Clustering criteria for discrete data and latent class models. *J Classif* 8(2):157–176
- Chan EY, Ching WK, Ng MK, Huang JZ (2004) An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognit* 37(5):943–952
- Chawla NV (2009) *Data mining for imbalanced datasets: An overview*. In: *Data mining and knowledge discovery handbook*. Springer, pp 875–886
- Chen L, Wang S, Wang K, Zhu J (2016) Soft subspace clustering of categorical data with probabilistic distance. *Pattern Recognit* 51:322–332
- Cover TM, Thomas JA (2012) *Elements of information theory*. Wiley, Hoboken
- Dhillon IS, Guan Y (2003) Information theoretic clustering of sparse cooccurrence data. In: *Third IEEE international conference on data mining, 2003 (ICDM 2003)*. IEEE, pp 517–520

- dos Santos TR, Zárate LE (2015) Categorical data clustering: what similarity measure to recommend? *Exp Syst Appl* 42(3):1247–1260
- Elmore RT, Hettmansperger TP, Thomas H (2004) Estimating component cumulative distribution functions in finite mixture models. *Commun Stat Theory Methods* 33(9):2075–2086
- Ewing T, Baber JC, Feher M (2006) Novel 2d fingerprints for ligand-based virtual screening. *J Chem Inf Model* 46(6):2423–2431
- Fränti P, Xu M, Kärkkäinen I (2003) Classification of binary vectors by using  $\delta$ sc distance to minimize stochastic complexity. *Pattern Recognit Lett* 24(1):65–73
- Ghosh JK, Delampady M, Samanta T (2006) Hypothesis testing and model selection. *Theory and methods: an introduction to Bayesian analysis*, pp 159–204
- Gollini I, Murphy TB (2014) Mixture of latent trait analyzers for model-based clustering of categorical data. *Stat Comput* 24(4):569–588
- Graham MW, Miller DJ (2006) Unsupervised learning of parsimonious mixtures on large spaces with integrated feature and component selection. *IEEE Trans Signal Process* 54(4):1289–1303
- Grantham NS (2014) Clustering binary data with bernoulli mixture models. Unpublished written preliminary exam. NC State University
- Guansong Pang SJ, Jin Huidong (2015) Cenkn: a scalable and effective text classifier. *Data Min Knowl Discov* 29(3):593–625
- Hartigan JA, Wong MA (1979) Algorithm as 136: A k-means clustering algorithm. *Appl Stat* 28:100–108
- He X, Cai D, Niyogi P (2006) Laplacian score for feature selection. In: *Advances in neural information processing systems*, pp 507–514
- Huang Z (1998) Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min Knowl Discov* 2(3):283–304
- Hubert L, Arabie P (1985) Comparing partitions. *J Classif* 2(1):193–218
- Indhumathi R, Sathiyabama S (2010) Reducing and clustering high dimensional data through principal component analysis. *Int J Comput Appl* 11(8):1–4
- Jang E, Gu S, Poole B (2017) Categorical reparametrization with gumble-softmax. In: *International conference on learning representations 2017*. <http://OpenReviews.net/>
- Juan A, Vidal E (2002) On the use of bernoulli mixture models for text classification. *Pattern Recognit* 35(12):2705–2710
- Karypis G (2002) Cluto—a clustering toolkit. Technical report, DTIC document
- Klekota J, Roth FP (2008) Chemical substructures that enrich for biological activity. *Bioinformatics* 24(21):2518–2525
- Langseth H, Nielsen TD (2009) Latent classification models for binary data. *Pattern Recognit* 42(11):2724–2736
- LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
- Lewis DD, Yang Y, Rose TG, Li F (2004) Rcv1: a new benchmark collection for text categorization research. *J Mach Learn Res* 5(Apr):361–397
- Li T (2005) A general model for clustering binary data. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, pp 188–197
- Li X, Roth D (2002) Learning question classifiers. In: *Proceedings of the 19th international conference on Computational linguistics-volume 1*. Association for Computational Linguistics, pp 1–7
- Li T, Ma S, Ogihara M (2004) Entropy-based criterion in categorical clustering. In: *Proceedings of the twenty-first international conference on machine learning*. ACM, p 68
- Li CG, You C, Vidal R (2017) Structured sparse subspace clustering: ajoint affinity learning and subspace clustering framework. *IEEE Trans Image Process* 26(6):2988–3001
- Li CG, You C, Vidal R (2018) On geometric analysis of affine sparse subspace clustering. *arXiv preprint arXiv:1808.05965*
- Lu CY, Min H, Zhao ZQ, Zhu L, Huang DS, Yan S (2012) Robust and efficient subspace segmentation via least squares regression. In: *European conference on computer vision*. Springer, pp 347–360
- MacQueen J et al (1967) Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol 1, pp 281–297
- Mali K, Mitra S (2003) Clustering and its validation in a symbolic framework. *Pattern Recognit Lett* 24(14):2367–2376



- McLachlan GJ (1987) On bootstrapping the likelihood ratio test statistic for the number of components in a normal mixture. *Appl Stati* 36:318–324
- McLachlan G, Peel D (2004) *Finite mixture models*. Wiley, Hoboken
- Nobile A et al (2004) On the posterior distribution of the number of components in a finite mixture. *Ann Stat* 32(5):2044–2073
- Nobile A, Fearnside AT (2007) Bayesian finite mixtures with an unknown number of components: the allocation sampler. *Stat Comput* 17(2):147–162
- Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P (2012) Community detection in social media. *Data Min Knowl Discov* 24(3):515–554
- Plumbley MD (2002) Clustering of sparse binary data using a minimum description length approach
- Rahmani M, Atia G (2017) Innovation pursuit: a new approach to the subspace clustering problem. In: *International conference on machine learning*, pp 2874–2882
- Ribeiro MT, Singh S, Guestrin C (2016) Why should i trust you? Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, pp 1135–1144
- Richardson S, Green PJ (1997) On bayesian analysis of mixtures with an unknown number of components (with discussion). *J R Stat Soc Ser B (Stat Methodol)* 59(4):731–792
- Schwarz G et al (1978) Estimating the dimension of a model. *Aann Stat* 6(2):461–464
- Serrà J, Karatzoglou A (2017) Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks. In: *Proceedings of the eleventh ACM conference on recommender systems*. ACM, pp 279–287
- Silvestre C, Cardoso MG, Figueiredo M (2015) Feature selection for clustering categorical data with an embedded modelling approach. *Exp Syst* 32(3):444–453
- Silvestre C, Cardoso MG, Figueiredo MA (2014) Identifying the number of clusters in discrete mixture models. *arXiv preprint arXiv:1409.7419*
- Smieja M, Tabor J (2012) Entropy of the mixture of sources and entropy dimension. *IEEE Trans Inf Theory* 58(5):2719–2728
- Śmieja M, Nakoneczny S, Tabor J (2016) Fast entropy clustering of sparse high dimensional binary data. In: *2016 International joint conference on neural networks (IJCNN)*. IEEE, pp 2397–2404
- Śmieja M, Geiger BC (2017) Semi-supervised cross-entropy clustering with information bottleneck constraint. *Inf Sci* 421:254–271
- Spurek P (2017) General split gaussian cross-entropy clustering. *Exp Syst Appl* 68:58–68
- Spurek P, Tabor J, Byrski K (2017) Active function cross-entropy clustering. *Exp Syst Appl* 72:49–66
- Steinbach M, Ertöz L, Kumar V (2004) The challenges of clustering high dimensional data. In: *New directions in statistical physics*. Springer, pp 273–309
- Strouse D, Schwab DJ (2016) The deterministic information bottleneck. In: *Proceedings conference on uncertainty in artificial intelligence (UAI)*, New York City, NY, pp 696–705
- Struski Ł, Tabor J, Spurek P (2017) Lossy compression approach to subspace clustering. *Inf Sci* 435:161–183
- Tabor J, Spurek P (2014) Cross-entropy clustering. *Pattern Recognit* 47(9):3046–3059
- Tang Y, Browne RP, McNicholas PD (2015) Model based clustering of high-dimensional binary data. *Comput Stat Data Anal* 87:84–101
- Tishby N, Pereira FC, Bialek W (1999) The information bottleneck method. In: *Proceedings Allerton conference on communication, control, and computing*, Monticello, IL, pp 368–377
- Tsakiris MC, Vidal R (2017) Hyperplane clustering via dual principal component pursuit. In: *International conference on machine learning*, pp 3472–3481
- Vermunt JK (2007) Multilevel mixture item response theory models: an application in education testing. In: *Proceedings of the 56th session of the International Statistical Institute Lisbon, Portugal* 2228
- Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J Mach Learn Res* 11(Oct):2837–2854
- Warszycki D, Mordalski S, Kristiansen K, Kafel R, Sylte I, Chilmonczyk Z, Bojarski AJ (2013) A linear combination of pharmacophore hypotheses as a new tool in search of new active compounds—an application for 5-HT<sub>1A</sub> receptor ligands. *PLoS ONE* 8(12):e84,510. <https://doi.org/10.1371/journal.pone.0084510>
- Wen JR, Nie JY, Zhang HJ (2002) Query clustering using user logs. *ACM Trans Inf Syst* 20(1):59–81
- Yamamoto M, Hayashi K (2015) Clustering of multivariate binary data with dimension reduction via l1-regularized likelihood maximization. *Pattern Recognit* 48(12):3959–3968

- Yap CW (2011) Padel-descriptor: an open source software to calculate molecular descriptors and fingerprints. *J Comput Chem* 32(7):1466–1474
- Ying Zhao UF, Karypis George (2005) Hierarchical clustering algorithms for document datasets. *Data Min Knowl Discov* 10(2):141–168
- You C, Li CG, Robinson DP, Vidal R (2016) Oracle based active set algorithm for scalable elastic net subspace clustering. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 3928–3937
- Zhao Y, Karypis G (2002) Evaluation of hierarchical clustering algorithms for document datasets. In: *Proceedings of the eleventh international conference on information and knowledge management*. ACM, pp 515–524

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.