Diagonal Co-clustering Algorithm for Document-Word Partitioning

Charlotte Laclau(✉) and Mohamed Nadif(✉)

LIPADE, Université Paris Descartes, 45 Rue des Saint-Pères, 75006 Paris, France
{charlotte.laclau,mohamed.nadif}@parisdescartes.fr

Abstract. We propose a novel diagonal co-clustering algorithm built upon the double Kmeans to address the problem of document-word co-clustering. At each iteration, the proposed algorithm seeks for a diagonal block structure of the data by minimizing a criterion based on the variance within and the centroid effect. In addition to be easy-to-interpret and efficient on sparse binary and continuous data, Diagonal Double Kmeans (DDKM) is also faster than other state-of-the art clustering algorithms. We illustrate our contribution using real datasets commonly used in document clustering.

1 Introduction

Co-clustering also known as biclustering or block-clustering involves simultaneous clustering of the set of observations and the set of features in a data matrix. By creating permutations of rows and columns, the co-clustering algorithms aim to reorganize the initial data matrix into homogeneous blocks. These blocks also called co-clusters can therefore be defined as subsets of the data matrix characterized by a set of observations and a set of features whose elements are similar. Other types of co-clustering approaches can be found in [11] and [12]. Co-clustering algorithms present several advantages: they reduce the initial matrix into a simpler form with the same basic structure and require far less computation when compared with separate processing of the same two sets; see for instance [8]. As a result, these methods are of interest to data mining.

In this work, we focus on co-clustering methods that seek a block diagonal structure, *i.e.* methods in which the number of clusters of rows is equal to the number of clusters of columns. An illustration is given in Fig. 1 where (a) represents an original binary matrix, (b) represents the same matrix after a proper permutation of rows whilst (c) adds a permutation of columns resulting in a clear block diagonal structure. These methods have proven efficient in dealing with the problem of document-word co-clustering. The objective is to group the documents based on the words within them and to group the words based on the documents in which they appear. The dataset is typically represented by a *document* × *words* matrix. In [10], the author proposed a block diagonal algorithm to deal with binary data. This algorithm alternates the clustering of observations and features minimizing the error between the original

E. Fromont et al. (Eds.): IDA 2015, LNCS 9385, pp. 170–180, 2015.
DOI: 10.1007/978-3-319-24465-5_15

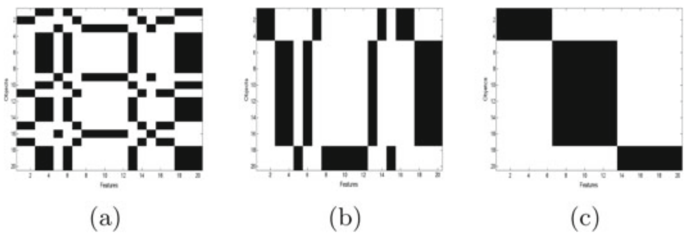


Fig. 1. Original binary data (a), (b) data reorganized according to rows, (c) reorganized according to rows and columns.

data matrix and the reconstructed matrix based on the cluster structure. In [3], the author proposed a spectral based solution. He built a bipartite graph from the *document* $\times$ *words* matrix which is partitioned in a way that minimize the cut objective function.

In this paper we propose a new diagonal co-clustering algorithm based on the minimization of an heterogeneity measure of blocks. This measure takes into account both the variance within blocks and a measure named the centroid effect [5] defined as the squared deviation from the mean entry in each block and the maximum entry in the input matrix. The proposed algorithm, in addition to be very efficient in terms of co-clustering on sparse data, are also faster than most of state-of-the art algorithms and therefore can deal with high dimensional data.

The remaining of this paper is organized as follows. Section 2 provides the needed background on the Double Kmeans (DKM) algorithm and presents the challenge of diagonal co-clustering. Section 3 presents the Diagonal Double Kmeans (DDKM) algorithm that we propose. Section 4 is devoted to numerical experiments on real datasets showing the appropriateness of our contribution for binary and continuous data. The final section sums up the study and gives recommendations for further research.

Notation. Let $\mathbf{X} := \{x_{ij}; i \in I; j \in J\}$ be a data matrix of size $n \times p$ where $I = \{1, \ldots, n\}$ and $J = \{1, \ldots, p\}$. The set I corresponds to the set of n objects and the set J to the set of p attributes. In the sequel, our aim consists in obtaining co-clustering of $\mathbf{X}$. Let $\mathbf{Z} = \{z_1, \ldots, z_n\}$ be a label vector, where $z_i \in \{1, \ldots, K\}$, denotes the partition of I into K clusters and $\mathbf{W} = \{w_1, \ldots, w_p\}$ where $w_j \in \{1, \ldots, H\}$ denotes the partition of J into H clusters. The partition of I (respectively J) can be represented by a matrix of elements in $\{0,1\}^K$ (respectively $\{0,1\}^H$) satisfying $\sum_{k=1}^{K} z_{ik} = 1$ (respectively $\sum_{h=1}^{H} w_{jh} = 1$). Finally, to simplify the notation, the sums relating to rows, columns, row and column clusters will be subscripted respectively by the letters ($i = 1, \ldots, n, j = 1, \ldots, p$ and, $k = 1, \ldots, K$, without indicating the implicit limits of variation. For example, the sum $\sum_{i,k}$ stands for $\sum_{i=1}^{n} \sum_{k=1}^{K}$.

2 Co-clustering and Diagonal Block Structure

The co-clustering can be formulated as the search for a good matrix approximation of the original data matrix **X**. The quality is determined by the approximation error which can be measured by a large class of loss functions like the square Euclidean distances. This approximation is generally achieved through an alternate least square minimization process (see for instance [1,7]). The Double Kmeans algorithm [16] is based on this principle.

2.1 Double Kmeans Algorithm

Formally, the aim is to minimize an objective function $J(\mathbf{Z}, \mathbf{W}, \mathbf{G})$ where **Z** and **W** are the partitions and $\mathbf{G} := \{g_{kh}; k \in \{1, \ldots, K\}, h \in \{1, \ldots, H\}$ is a $K \times H$ matrix which can be viewed as a summary of the data matrix X (see Fig. 2).

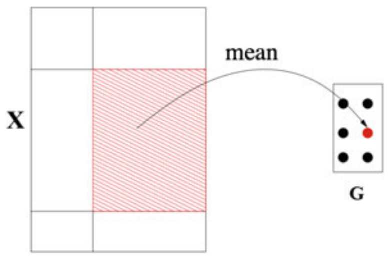


Fig. 2. Original data matrix **X** and its summary after co-clustering into 6 co-clusters.

Each element g_{kh} of **G** is called a prototype of co-cluster $\mathbf{X}_{kh} := \{x_{ij}; z_{ik}w_{jh} = 1\}$. Double Kmeans (DK) adopts the squared Euclidean distance to measure the dissimilarity between the matrix **X** and the structure described in **Z**,**W** and **G**. Therefore, $J(\mathbf{Z}, \mathbf{W}, \mathbf{G})$ is given by

$$\mathcal{J}(\mathbf{Z}, \mathbf{W}, \mathbf{G}) = \sum_{i,k,j,h} z_{ik} \times w_{jh}(x_{ij} - g_{kh})^2 = ||\mathbf{X} - \mathbf{ZGW}^T||^2, \qquad (1)$$

where $||.||$ denotes the Frobenius norm. It is easy to see that for a fixed $(\mathbf{Z}; \mathbf{W})$ the optimal values of **G** are the means of $\mathbf{X}_{kh}$'s. The optimal partitions **Z** and **W** are found using an iterative algorithm. A version of double Kmeans is presented in Algorithm 1 where $z_{.k}$ (resp. $w_{.h}$) represents the cardinality of the kth cluster (resp. hth cluster).

2.2 Block Diagonal Structure

The DKM algorithm appears to be not efficient when looking for a one-to-one correspondence between two partitions **Z** and **W**. In order to deal with this

Algorithm 1. Double Kmeans (DKM)

```
input: $\mathbf{X}$, $K$, $H$
initialization: $\mathbf{Z}$ and $\mathbf{W}$
repeat
    (1) Compute $g_{kh} = \sum_{i,j} \frac{z_{ik} w_{jh} x_{ij}}{z_{.k} w_{.h}}$, $\forall k, h$
    (2) Update $z_i = \arg\min_k \sum_{j,h} w_{jh}(x_{ij} - g_{kh})^2$, $\forall i$
    (3) Update $w_j = \arg\min_h \sum_{i,k} z_{ik}(x_{ij} - g_{kh})^2$, $\forall j$
until the $J$ value change is small or there is no change.
output: $\mathbf{G}$, $\mathbf{Z}$ and $\mathbf{W}$
```

specific case, we have to consider w_{jk} instead of a w_{jh}; in other words, we assume that $H = K$. Secondly, the diagonal structure involve to impose some constraints on $\mathbf{G}$; for instance by taking $g_{kk} = \delta \ \forall k$. This leads us to consider the following criterion:

$$J(\mathbf{X}, \mathbf{Z}, \mathbf{W}) = \sum_{i,j,k} z_{ik} w_{jk} (x_{ij} - \delta)^2, \tag{2}$$

where δ is assumed to be known. The choice of this parameter will be discussed in the next section. The couple of partitions $(\mathbf{Z}, \mathbf{W})$ optimizing the criterion given in Eq. 2 are found using the following iterative algorithm

1. Update $\mathbf{Z}$ the partition of objects, with $\mathbf{W}$ fixed. This leads to the following formula
$$z_i = \arg\min_k \sum_j w_{jk}(x_{ij} - \delta)^2,$$
2. Update $\mathbf{W}$, the partition of features, with $\mathbf{Z}$ fixed. This leads to the following formula
$$w_j = \arg\min_k \sum_i z_{ik}(x_{ij} - \delta)^2.$$

From these formulae, one observes that seeking the diagonal structure of blocks indirectly introduces a strong dependency between object assignments (respectively features assignments) to a block and the number of features that belong to this block (respectively the number of objects). If we consider object assignments, we have $(x_{ij} - \delta)^2 \geq 0, \forall i, j$; therefore a higher number of features in a block will decrease the chance that an object will be assigned to this particular block. The same phenomenon occurs in the feature assignments. This leads to take into account the size of each co-cluster in order to avoid empty blocks.

3 Diagonal Double Kmeans

3.1 Criterion and Proposed Algorithm

In order to correct the bias introduced by the diagonal structure and to avoid certain blocks from vanishing, we propose a modified criterion that takes into account the number of elements in a block. This criterion takes the following form:

$$J(\mathbf{X},\mathbf{Z},\mathbf{W}) = \sum_{k} \frac{1}{z_{.k}w_{.k}} \sum_{i,j} z_{ik}w_{jk}(x_{ij}-\delta)^2. \tag{3}$$

where $z_{.k}$ and $w_{.k}$ denote respectively the number of objects and the number of features in the k-th block. Furthermore, it is interesting to note that the criterion given in Eq. 3 may be expressed depending on the variance of a given block $(\mathbf{Z}_k, \mathbf{W}_k)$ and the squared deviation of its mean from the maximum input of the data:

$$\begin{aligned} J(\mathbf{X},\mathbf{Z},\mathbf{W}) &= \sum_{i,j,k} \frac{z_{ik}w_{jk}}{z_{.k}w_{.k}}(x_{ij}-\overline{x}_k)^2 + \sum_{i,j,k} \frac{z_{ik}w_{jk}}{z_{.k}w_{.k}}(\overline{x}_k-\delta)^2 \\ &= \sum_k \overline{s}_k^2 + \sum_k (\overline{x}_k-\delta)^2. \end{aligned} \tag{4}$$

where

$$\overline{x}_k = \frac{1}{z_{.k}w_{.k}} \sum_{i,j} z_{ik}w_{jk}x_{ij} \text{ and } \overline{s}_k^2 = \frac{1}{z_{.k}w_{.k}} \sum_{i,j} z_{ik}w_{jk}(x_{ij}-\overline{x}_k)^2$$

denote the mean and the variance within the k-th block respectively. The first term of Eq. 4 ensures the homogeneity of each block while the second one provides the homogeneity between centers of the blocks and δ. This objective function (Eq. 3) is optimized by an alternating optimization of two conditional criteria given $\mathbf{W}$ and $\mathbf{Z}$ respectively.

$$\tilde{J}_1(\mathbf{X},\mathbf{Z}|\mathbf{W}) = \sum_k \frac{1}{z_{.k}} \sum_i z_{ik} \frac{1}{w_{.k}} \sum_j w_{jk}(x_{ij}-\delta)^2$$

and

$$\tilde{J}_2(\mathbf{X},\mathbf{W}|\mathbf{Z}) = \sum_k \frac{1}{w_{.k}} \sum_j w_{jk} \frac{1}{z_{.k}} \sum_i z_{ik}(x_{ij}-\delta)^2.$$

The optimization of $\tilde{J}_1$ and $\tilde{J}_2$ lead to the following update rules:

$$z_i = \arg\min_k \frac{1}{w_{.k}} \sum_j w_{jk}(x_{ij}-\delta)^2, \tag{5}$$

$$w_j = \arg\min_k \frac{1}{z_{.k}} \sum_i z_{ik}(x_{ij}-\delta)^2. \tag{6}$$

The proposed algorithm Diagonal Double Kmeans (DDKM) (Algorithm 2) is computationally efficient and its complexity can be shown to be $O(\tau \times npK)$ where τ denotes the number of iterations required to obtain the convergence, n, p and K are the number of objects (*i.e.* rows), features (*i.e.* columns) and clusters respectively.

Algorithm 2. Diagonal Double Kmeans (DDKM)

```
input: X and K
initialization: Z, W and δ
repeat
   (2) Update Z according to Eq. 5
   (3) Update W according to Eq. 6
until the J value change is small or there is no change.
output: Z and W
```

3.2 Choice of δ

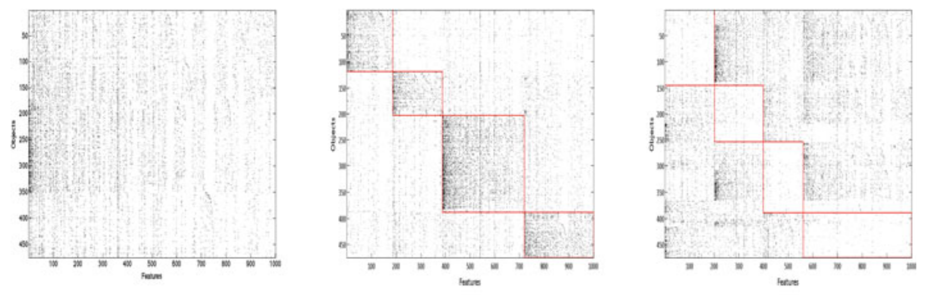
Herein, we discuss the choice of δ. Specifically, we compare between the optimal value of δ and the maximum entry of the matrix.

1. If we consider δ as an unknown parameter, its optimal value for the criterion minimized is equal to the average of blocks means. Indeed, with **Z** and **W** fixed and by setting the derivative of J (Eq. 3) to zero we obtain $\delta = \frac{1}{K}\sum_k \overline{x}_k$ where $\overline{x}_k$ denotes the mean of the k-th block. Although this value of δ is the optimal, we can observe that in the context of sparse data *i.e.* when the data matrix contains a high percentage of 0, its value will tend to 0 leading to a diagonal structure of blocks of 0. An illustration of the resulting co-clustering obtained with this value on the CSTR dataset (described in the numerical experiments section) is given in Fig. 3(c).
2. Another way to proceed is to set the value of δ at the initialisation step. DDKM aims at grouping objects and features with the strongest association possible. For instance, in the case of a binary data matrix **X**, the strongest association between an object i and a feature j is expressed in $x_{ij} = 1$ which is the maximum value of the entry of **X**. As a matter of fact, choosing the maximum allows to guarantee the homogeneity of diagonal blocks while ensuring blocks of 0 outside. In [5,13], the authors proposed hierarchical algorithms based on this idea. We use the same example as for the optimal value to show the result on Fig. 3(b). It is important to stress that this approach requires for values of a data matrix to be comparable. This is the case for binary data or normalized data as we will see in the next section devoted to the document-word partitioning.

4 Numerical Experiments

4.1 Performance Evaluation

In order to assess and to compare the performance of the proposed algorithm, we use commonly adopted metrics: the Accuracy, the Normalized Mutual Information [15] and the Adjusted Rand Index [9]. We focus only on the quality of



(a) (b) (c)

Fig. 3. (a) CSTR the original dataset, (b) CSTR reorganised according to the partitions when $\delta = \max_{i,j} x_{ij}$, (c) CSTR reorganised according to the partitions when δ estimated by $\frac{1}{K}\sum_k \overline{x}_k$.

row clustering. Clustering accuracy noted Acc is one of the most widely used evaluation criterion and is defined as:

$$\text{Acc} = \frac{1}{n} max \left[\sum_{\mathcal{C}_k, \mathcal{L}_\ell} T(\mathcal{C}_k, \mathcal{L}_\ell) \right]$$

where $\mathcal{C}_k$ is the kth cluster in the final results, and $\mathcal{L}_\ell$ is the true ℓ^{th} class. $T(\mathcal{C}_k, \mathcal{L}_\ell)$ is the proportion of objects that were correctly recovered by the clustering algorithm *i.e.* $T(\mathcal{C}_k, \mathcal{L}_\ell) = \mathcal{C}_k \cap \mathcal{L}_\ell$. Accuracy computes the maximum sum of $T(\mathcal{C}_k, \mathcal{L}_\ell)$ for all pairs of clusters and classes, and these pairs have no overlaps. The second measure employed is the Normalized Mutual Information (NMI) and is calculated as follows:

$$\text{NMI} = \frac{\sum_{k,\ell} \frac{n_{k\ell}}{n} \log \frac{n_{k\ell}}{n_k \hat{n}_\ell}}{\sqrt{(\sum_k \frac{n_k}{n} \log \frac{n_k}{n})(\sum_\ell \frac{\hat{n}_k}{n} \log \frac{\hat{n}_\ell}{n})}}$$

where n_k denotes the number of data contained in the cluster $\mathcal{C}_k (1 \leq k \leq K)$, $\hat{n}_\ell$, the number of data belonging to the class $\mathcal{C'}_k (1 \leq \ell \leq K)$, and $n_{k\ell}$, the number of data that are in the intersection between the cluster $\mathcal{C}_k$ and the class $\mathcal{C'}_k$. The last measure *Adjusted Rand* noted ARI measures the similarity between two clustering partitions. From a mathematical standpoint, the Rand index is related to the accuracy. The adjusted form of the Rand Index is defined as:

$$\text{ARI} = \frac{\sum_{k,\ell} \binom{n_{k\ell}}{2} - \left[\sum_k \binom{n_k}{2} \sum_\ell \binom{\hat{n}_\ell}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_k \binom{n_k}{2} + \sum_\ell \binom{\hat{n}_\ell}{2}\right] - \left[\sum_k \binom{n_k}{2} \sum_\ell \binom{\hat{n}_\ell}{2}\right] / \binom{n}{2}}.$$

The value for these three metrics are between 0 and 1, a value close to 1 means a good result in terms of clustering.

4.2 Compared Algorithms

We compare against state-of-the-art (co)-clustering methods including Spherical Kmeans (SpKM)[4], Double Kmeans (DKM) and Spectral Co-Clustering (SpCo) [3]. We also report the clustering results by Kmeans and the Nonnegative Matrix Factorization (NMF) [2] as baseline. The Spherical Kmeans algorithm is basically a Kmeans algorithm that use the cosine dissimilarity instead of the Euclidean distance. It is known to be very efficient on sparse dataset and to converge quickly. We use the matlab implementation for kmeans and NMF. For SpCo algorithm we use the implementation proposed by Assaf Gottlieb[1]. We use the SpKM implementation given in [14]. Finally, we implement CROEUC [6], a fast version of Double Kmeans (DKM); its advantage is due to the use of intermediate matrices of reduced sizes rather than the original data.

4.3 Datasets and Results

We study the effectiveness of our algorithm for some well-known text datasets with different sizes and balances[2]: CSTR, Classic3, WebKB4 and 2 subsets of the 20 Newsgroups dataset. The 20 Newsgroups dataset is organized into 20 topics. Some of the topics are closely related while other are highly unrelated. We describe the topics included in the two subsets in Table 1. The NG2 dataset includes two topics not related (rec.motorcycles and sci.crypt,sci.space) while NG5 includes topics closely related involving a situation with overlapping clusters (rec.sport.baseball, sci.crypt, sci.med, talk.religion.misc, comp.windows.x, soc.religion.christian, talk.politics.mideast). A detailed description of all datasets can be found in Table 1.

Table 1. Description of the datasets in terms of size ($n \times p$), number of clusters (K), sparsity (%0) and size of the cluster. A partition is assumed balanced if the balance coefficient is close to 1 and unbalanced otherwise.

Dataset	$n \times p$	K	%0	Balance
CSTR	475×1000	4	96.60	0.399
Classic3	3891×4303	3	98.95	0.71
WebKB4	4199×1000	4	91.83	0.307
NG2	500×2000	2	96.90	1
NG5	500×2000	5	97.19	1

Originally each cell of these datasets denotes the number of occurrences of a word in a document. As we are interested in evaluating our algorithm on both binary and continuous data, we use one version of the datasets on which

[1] http://adios.tau.ac.il/.

[2] The balance coefficient is defined as the ratio of the number of documents in the smallest class to the number of documents in the largest class.

Table 2. Accuracy, Normalized Mutual Information and Adjusted Rand Index obtained on tf-idf datasets.

Dataset	Metric	Algorithms					
		NMF	Kmeans	SpKM	DKM	SpCo	DDKM
CSTR	Acc	81.47	85.05	88.63	62.95	79.79	**90.95**
	NMI	69.91	64.74	74.07	27.93	66.67	**76.39**
	ARI	70.26	68.14	76.57	46.72	70.20	**82.99**
Classic3	Acc	96.32	90.47	97.33	94.17	70.60	**98.69**
	NMI	84.53	73.81	91.41	80.90	59.64	**96.10**
	ARI	89.04	75.34	94.38	82.82	40.20	**93.32**
WebKB4	Acc	**80.38**	49.46	62.85	38.75	64.32	78.90
	NMI	**52.08**	25.15	37.00	04.65	41.04	51.15
	ARI	**56.48**	18.03	31.55	04.54	38.09	54.93
NG2	Acc	62.60	-	60.92	50.70	88.98	**94.40**
	NMI	4.85	-	11.92	0.15	53.16	**68.88**
	ARI	6.17	-	4.68	0.00	60.69	**78.81**
NG5	Acc	41.48	23.25	56.31	31.06	53.91	**70.54**
	NMI	27.42	6.21	30.69	9.47	45.59	**41.53**
	ARI	13.66	0.34	19.85	4.45	30.03	**41.40**

the data were converted into binary *i.e.* each cell having a value higher to 0 is considered equal to 1 and 0 otherwise, and a second version where a TF-IDF (Term Frequency - Inverse Document Frequency) transformation is applied. The TF-IDF normalization is one of the most used in text mining, and is defined as $x'_{ij} = tf_{ij} \times \log \frac{n}{df_j}$ where tf_{ij} denotes the number of occurrence of the j-th word in the i-th document and df_j denotes the number of documents containing the j-th word.

We set the number of clusters as the true number of classes on all datasets. For each method, given the number of clusters, no parameter selection is needed. We run the algorithms 100 times and report the best result (in percentage), *i.e.* the one that corresponds to a local minimum of the criterion of all trials in Tables 2 and 3. Several observations can be made based on these results: the proposed algorithm outperforms the other method on each dataset whether it is the TF-IDF version or the binary one, except on the TF-IDF version of WebKB4. On NG2 and NG5 whose classes are not well separated, the performance difference is all the more important. We can also note that on the TF-IDF version of NG2, Kmeans is unable to find a partition into two clusters as required.

4.4 Computational Complexity

We study the computational complexity of the compared clustering and co-clustering algorithms. We repeat clustering 100 times for each algorithm on

Table 3. Accuracy, Normalized Mutual Information and Adjusted Rand Index obtained on binary datasets.

Dataset	Metric	Algorithms					
		NMF	Kmeans	SpKM	DKM	SpCo	DDKM
CSTR	Acc	85.68	85.05	88.63	62.95	79.79	**91.37**
	NMI	67.08	64.74	74.07	27.93	66.67	**79.06**
	ARI	70.65	68.14	76.57	46.72	70.20	**83.00**
Classic3	Acc	97.66	90.47	97.33	94.17	70.60	**98.20**
	NMI	88.78	73.81	**91.41**	80.90	59.64	91.31
	ARI	93.06	75.34	94.38	82.82	40.20	**94.61**
WebKB4	Acc	**73.26**	49.46	62.85	38.75	64.32	72.80
	NMI	41.05	25.15	37.00	04.65	41.04	**44.14**
	ARI	43.60	18.03	31.55	04.54	38.09	**44.29**
NG2	Acc	60.60	57.60	60.80	54.60	90.20	**94.60**
	NMI	12.85	12.93	13.09	2.38	55.35	**70.55**
	ARI	4.41	2.26	4.58	0.77	64.57	**79.53**
NG5	Acc	50.10	33.07	52.10	29.66	60.32	**76.75**
	NMI	32.25	16.61	28.31	8.15	50.75	**51.49**
	ARI	20.68	4.08	24.84	2.82	37.31	**51.37**

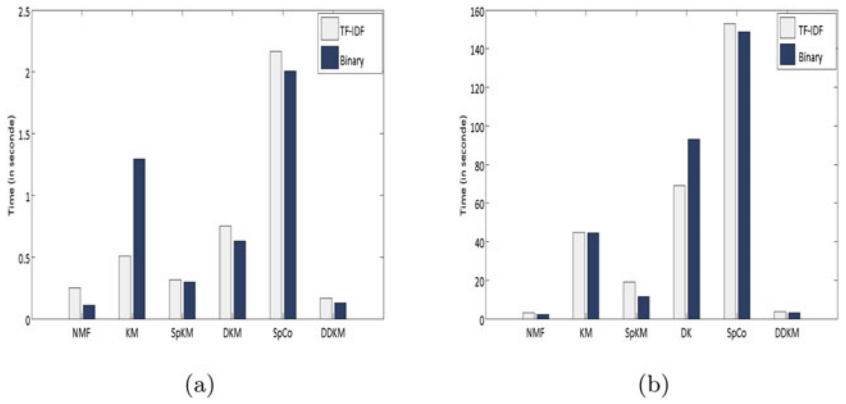


Fig. 4. Running time in seconds of the compared algorithms on the binary and the tf-idf versions of CSTR (a) and classic3 (b) datasets.

each dataset. We report the average convergence time in Fig. 4 for the CSTR (a) and Classic 3 (b) datasets. The obtained results show that the proposed algorithm DDKM is only very slightly slower than NMF method while it requires far less time to converge than all other state-of-the-art algorithms. The same observations were made on the other datasets presented in this article.

5 Conclusion

In this paper we presented DDKM, a fast co-clustering algorithm that looks for homogeneous diagonal blocks. Compared with other methods, we demonstrate that our proposed algorithm is more effective for document-word partitioning datasets and especially in presence of classes having a high degree of overlap. In addition, DDKM requires less time to converge; up to 20 times less time than DKM and 40 times less time than SpCo commonly used in the domain of document clustering. In real world application, the knowledge of the number of co-clusters is mostly required. For further research, it will be worthwhile to investigate an efficient way to assess this parameter.

References

1. Baier, D., Gaul, W., Schader, M.: Two-mode overlapping clustering with applications to simultaneous benefit segmentation and market structuring. In: Klar, R., Opitz, O. (eds.) Classification and knowledge organization. Springer, Heidelberg (1997)
2. Berry, M.W., Browne, M., Langville, A.N., Pauca, V.P., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. In: Computational Statistics and Data Analysis, pp. 155–173 (2006)
3. Dhillon, I.S.: Co-clustering documents and words using bipartite spectral graph partitioning. In: KDD 2001, pp. 269–274 (2001)
4. Dhillon, I.S., Modha, D.S.: Concept decompositions for large sparse text data using clustering. Mach. Learn. **42**(1–2), 143–175 (2001)
5. Eckes, T., Orlik, P.: An error variance approach to two-mode hierarchical clustering. J. Classif. **10**(1), 51–74 (1993)
6. Govaert, G.: Classification croisée. Ph.D. thesis, Université Paris 6, France (1983)
7. Govaert, G., Nadif, M.: Co-Clustering: Models, Algorithms and Applications. Wiley, New York (2013)
8. Govaert, G., Nadif, M.: Block clustering with bernoulli mixture models: comparison of different approaches. Comput. Stat. Data Anal. **52**(6), 3233–3245 (2008)
9. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**, 193–218 (1985)
10. Li, T.: A general model for clustering binary data. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD 2005, pp. 188–197 (2005)
11. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Trans. Comput. Biol. Bioinf. **1**, 24–45 (2004)
12. Mechelen, I.V., Bock, H.H., Boeck, P.D.: Two-mode clustering methods: a structured overview. Stat. Methods Med. Res. **13**(5), 363–394 (2004)
13. Mirkin, B., Arabie, P., Hubert, L.: Additive two-mode clustering: the error-variance approach revisited. J. Classif. **12**(2), 243–263 (1995)
14. Nguyen, X.V.: Gene clustering on the unit hypersphere with the spherical k-means algorithm: coping with extremely large number of local optima. In: International Conference on Bioinformatics & Computational Biology, BIOCOMP 2008, pp. 226–233 (2008)
15. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2003)
16. Vichi, M.: Double k-means clustering for simultaneous classification of objects and variables. In: Borra, S., Rocci, R., Vichi, M., Schader, M. (eds.) Advances in classification and data analysis, pp. 43–52. Springer, Heidelberg (2001)