Apprentissage multi-vue de co-similarités pour la classification

Gilles Bisson[1] et Clément Grimal[1]

Université Joseph Fourier / Grenoble 1 / CNRS
Laboratoire LIG - Bâtiment CE4
38610 Gières FRANCE
clement.grimal@imag.fr, gilles.bisson@imag.fr

Abstract : En classification, les données se présentent souvent sous la forme d'une matrice de données dont les lignes représentent les instances d'un type d'objets, et les colonnes leurs caractéristiques. Cependant, dans de nombreuses applications, plusieurs types d'objets liés par des relations peuvent exister, ce qui conduit à avoir plusieurs matrices représentant chacune une vue particulière sur les données. C'est ainsi le cas dans l'étude des réseaux sociaux ou les différents nœuds d'un graphe d'interaction font intervenir des utilisateurs, des documents, des termes, etc. Dans le contexte de la co-classification (*co-clustering*), l'algorithme χ-Sim (Grimal & Bisson, 2011), qui permet de calculer conjointement les similarités entre les lignes et les colonnes d'une matrice de données, s'est avéré dépasser significativement les résultats de l'état de l'art sur plusieurs jeux de tests standards. Dans ce papier, nous introduisons une architecture générale permettant d'étendre les capacités de cet algorithme de calcul de co-similarité afin de le rendre apte à travailler sur des collections de matrices décrivant les relations entre plusieurs paires d'objets différents (*multi-view clustering*). Nous montrons que cette architecture offre non seulement un cadre formel intéressant mais quelle permet en outre de délivrer souvent des résultats supérieurs ou égaux aux approches classiques mono-relation tout en permettant, grâce à une parallélisation possible des calculs, de réduire la complexité en temps et en espace des problèmes traités.

Mots-clés : Classification multi-vue, Co-clustering, Co-similarité, Parallélisation.

1. Introduction

The clustering task consists in organizing instances of objects into homogeneous and contrasted groups, so that instances in a given cluster are more similar than instances in different clusters. Most of the clustering methods in the literature focus on datasets described by a unique data matrix, which can either be a feature matrix (objects described by their characteristics), or a relation matrix (intensity of the relation between instances of two types of objects[1]). In the latter case, both types of objects can be clustered, methods dealing with this task are referred as co-clustering approaches and have been extensively studied during the last decade.

However, in many applications, datasets involving more than two types of interacting objects, or simply related, are frequent. For instance, in a social network we can have simultaneously some relations between pairs of users, users and documents, and documents and terms. A simple way to represent such datasets is to use as many matrices as there are relations between the objects. Then, one could use classical (co-)clustering methods to separately cluster the objects occurring in the different matrices but in this way, interactions between objects would not been taken into account leading to a huge loss of information. In the following, we refer to this problem as multi-view clustering, each view being described by a relation matrix. Therefore, multi-view learning represents a great challenge for learning approaches.

The present work is an extension of an existing algorithm, named $\chi\text{-Sim}_p^k$ (Bisson & Hussain, 2008; Hussain *et al.*, 2010), which obtained good results on the co-clustering task. We selected this approach for two reasons. First, it simultaneously builds similarity matrices between the objects described in a data matrix; this is useful in the multi-view context since it allows us to combine easily the set of similarities measures, which have been computed from the different relation matrices of the dataset. Second, in this algorithm, the similarity matrices between objects can be externally initialized, allowing us to easily inject some *a priori* knowledge on the data; thus, it becomes possible to imagine a way to convey the similarities computed in one view to the others through an iterative process.

The rest of this paper is structured as follows. In Sect. 2 we define more formally the multi-view clustering problem, and we present some related work. In Sect. 3 we provide some insight about the $\chi\text{-Sim}_p^k$ method and then, in Sect. 4, we present and analyze a general framework allowing to adapt this algorithm to multi-view's needs. In Sect. 5, experimental results on some classical datasets are presented allowing to see the improvements provided by our approach with respect to the classical single view clustering. Finally, in Sect. 6, we present conclusions and future work.

[1]Such as [documents/terms] in NLP or [genes/expression] in genomics, for instance.

2. Definition and related work

As we said in the introduction, the co-clustering task has been intensively explored in many research domains during the last decade. In information retrieval, to deal with the co-clustering of documents according to their words and topics (Dhillon, 2001; Ingaramo *et al.*, 2010; Liu *et al.*, 2004; Long *et al.*, 2005; Rege *et al.*, 2008; Yen *et al.*, 2009; Zanghi *et al.*, 2010); in bioinformatics, to analyze gene expression data (Cheng & Church, 2000; Iam-on *et al.*, 2010; Madeira & Oliveira, 2004; Speer *et al.*, 2004); and in social networks, to detect community of users (Barber, 2007; Cruz Gomez *et al.*, 2011; Du *et al.*, 2007; Duch & Arenas, 2005; Fortunato, 2010). As emphasized by Long *et al.* (2005) and Bisson & Hussain (2008), co-clustering comes along with the double advantage of improving the cluster quality when dealing with large dimensional sparse matrices, and of highlighting similarities between objects described by *a priori* different set of features.

Following the case of co-clustering, in the multi-view context the idea is to take into account all the types of objects and their multiple relations described in the dataset, in order to improve the quality of the resulting clustering. An example of a simple multi-view dataset can be found when considering a movies database, in which a movie is described both by the actors appearing in it, and by the keywords used to describe it.

When considering such a dataset, two equivalent representation paradigms can be used: the first one is the collection of matrices, and the second one the *k*-partite graph (Long *et al.*, 2006). Within the k-partite graph[2] paradigm, a given subset of nodes contains the instances of one type of objects, and a link between two nodes of different subsets represents the relation between these two nodes. Alternatively, when working with a collection of matrices, each matrix describes a view on the data, i.e. a relation between two types of objects, one type of objects represented by the rows and the other one by the columns of the matrix. In our previous movie database example, we would have two matrices: a [movies/actors] matrix and a [movies/keywords] one.

Let's notice that in the rest of this paper, we will mostly use the later paradigm to represent the datasets, as it is better suited to explain our algorithm. Moreover, we will not consider relations linking more than two types of objects. Such relations can exist and may be described, either by an hyper-graph, or by a tensor (multidimensional generalization of the matrices)

[2]A graph is said to be *k*-partite when the nodes are partitioned into *k* subsets with the condition than no two nodes of the same subset are adjacent.

as in Acar & Yener (2009); Banerjee *et al.* (2007).

Compared with classical clustering and co-clustering, the field of multi-view clustering has not been as thoroughly explored so far. Multi-view setting became highly popular with the seminal work of Blum & Mitchell (1998), in which the authors trained two algorithms on two different views, introducing semi-supervised learning. Since then, several extensions of classical clustering methods have been proposed to deal with multi-view data. For example, Bickel & Scheffer (2004); Drost *et al.* (2006) describe extensions of the classical k-means and EM algorithms for the multi-view setting. Some approaches, such as the Canonical Correlation Analysis (Chaudhuri *et al.*, 2009), first extract relevant features from the multiple views, and then apply classical clustering algorithms to it.

In addition, the framework of spectral clustering has also been investigated (de Sa, 2005; Kumar & Daume III, 2011; Long *et al.*, 2006; Zhou & Burges, 2007), where most approaches consider a multi-partite graph describing the relations between objects and aim at finding the optimal cut of this graph. In Kumar & Daume III (2011), the similarities computed in one view are used to constrain the similarities computed in the other views. We can also cite Bekkerman *et al.* (2005) where the authors separately perform clustering on each view, but simultaneously maximize a unique objective function based on the mutual information between clusters.

Alternatively, closer to our approach, some works aim at combining multiple similarity matrices to perform a given learning task (de Carvalho *et al.*, 2012; Frigui *et al.*, 2007; Tang *et al.*, 2009). In Tang *et al.* (2009) information coming from different views (or sources) and describing different relations between the same instances are merged using Linked Matrix Factorization. Similarly de Carvalho *et al.* (2012) is building clusters from multiple similarity matrices computed from different views of a dataset. The algorithm learns a relevance weights matrix between classes and views, taking into account that some views may be better at describing some classes. Finally, in Pedrycz (2002), fuzzy clustering is used in the multi-view setting, by first computing membership matrices for every objects in the different views, and modifying these matrices through collaboration between the views. The end goal is quite different from other approaches though, as this method produce as many clusterings as views, without trying to obtain a sole clustering.

3. The χ-Sim$_p^k$ Algorithm

Throughout this paper, we will use the classical notations: matrices (in capital letters) and vectors (in small letters) are in bold; variables are in italic.

Type of objects: let N be the number of different types of objects considered in the dataset. $\forall i \in 1..N$, T_i is the type of objects i (i.e. users, documents, movies, words, etc.) For the sake of simplicity, we consider that each T_i has always the same number of n_i instances across the collection of matrices.

Relation matrices: let M be the number of relations between objects in the dataset, and thus the number of matrices in the collection. Then $\mathbf{R}_{ij}$ is the relation matrix describing the relation between objects T_i and T_j, of size $n_i \times n_j$. The element $(\mathbf{R}_{ij})_{ab}$ of a matrix expresses the link 'intensity' between the a^{th} instance of T_i and the b^{th} instance of T_j. For instance, in a document/term matrix it can be the frequency of the b^{th} term in the a^{th} document.

Similarity matrices: we can thus consider N similarity matrices $\mathbf{S}_1 \ldots \mathbf{S}_N$. Then $\mathbf{S}_i$ (of size $n_i \times n_i$) is the square and symmetrical matrix that contains the similarities between all the pairs of instances of T_i. The values of the similarity measure must be in $[0,1]$.

3.1. Presentation of χ-Sim$_p^k$

In this section, we present the main aspects of the χ-Sim$_p^k$ co-similarity measure which is a basic component of our architecture to deal with multi-view datasets. As this algorithm processes one matrix at a time, we can simplify the previous notations and we consider two types of objects T_1 and T_2, linked together by the relation matrix $\mathbf{R}_{12}$. Moreover, we will assume that $\mathbf{R}_{12}$ is a [documents/words] matrix, and that the task is to compute the similarities between every pair of documents and every pair of words.

The main idea of the χ-Sim$_p^k$ method is to make use of the duality between documents and words (each one being a descriptor of the other) by simultaneously generating the similarity matrices $\mathbf{S}_1$ (documents) and $\mathbf{S}_2$ (words). This is achieved by calculating similarities between documents on the basis of the similarities between their words, and similarities between words on the basis of the similarities between the documents in which they appear. Similar ideas have also been used for supervised leaning in Liu *et al.* (2004) or for image retrieval in Wang *et al.* (2004). With such approaches, documents (respectively words) can be seen as similar even if they do not explicitly share words (respectively documents), which is a great feature to deal with language difficulties such as terminological variability [3]. Once the similarity matrices have been generated with χ-Sim$_p^k$, they can be used by any classical clustering tool (*k*-means, etc.) to organize documents and/or words; however, due to the way these similarities have been built, the resulting clusters are comparable to those obtained with a genuine co-clustering algorithm.

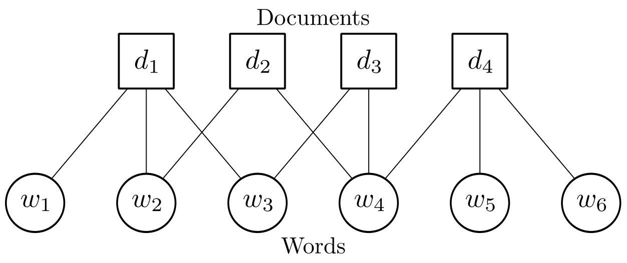

Figure 1: Example of a documents-words bi-partite graph.

To illustrate in an intuitive way the method, let us consider the toy dataset in Fig. 1 extracted from Hussain *et al.* (2010). Documents d_1 and d_4 do not share any words and thus their similarity would be equal to 0 with a classical similarity measure. However, as words w_3 and w_4 are both appearing in document d_3, they have a non-null similarity value in $\mathbf{S}_2$; therefore, as document d_1 contains word w_3 and document d_4 contains word w_4, the resulting similarity between d_1 and d_4 in $\mathbf{S}_1$ will not be null.

In practice, the similarity matrix $\mathbf{S}_1$ between documents is defined in two steps (see Hussain *et al.* (2010) for complete details and justifications):

$$\mathbf{S}_1 = \mathbf{R}_{12}^{\circ k} \times \mathbf{S}_2 \times \left(\mathbf{R}_{12}^{\circ k}\right)^{\mathrm{T}} \tag{1}$$

$$\forall a,b \; (\mathbf{S}_1)_{ab} \leftarrow \left(\frac{(\mathbf{S}_1)_{ab}}{\sqrt{(\mathbf{S}_1)_{aa} \times (\mathbf{S}_1)_{bb}}} \right)^{1/k} \tag{2}$$

First, Eq. (1) defines the similarity matrix $\mathbf{S}_1$ according both to the data matrix $\mathbf{R}_{12}$ and to the similarity matrix between words $\mathbf{S}_2$, with $\left(\mathbf{R}_{12}^{\circ k}\right)_{ab} = (\mathbf{R}_{12})_{ab}^k$ being the element-wise exponentiation of $\mathbf{M}$ to the power of k. Second, Eq. (2) allows to normalize the elements of $\mathbf{S}_1$ in $[0,1]$. The k parameter

[3]Two documents written by two different authors dealing with the same topic, may contain different words if their respective author has used a different vocabulary. In such case, classical similarity measures, such as Cosine, lead to underestimate the real similarity between documents and/or between words.

is analogous to the one used in the L_k-norm (Minkowski distance), the idea being to adjust this parameter as suggested in Aggarwal *et al.* (2001) to deal with high dimensional spaces. Symmetrically, the similarity matrix $\mathbf{S}_2$ between words is defined as follows:

$$\mathbf{S}_2 = \left(\mathbf{R}_{12}^{\circ k}\right)^{\mathrm{T}} \times \mathbf{S}_1 \times \mathbf{R}_{12}^{\circ k} \tag{3}$$

$$\forall a,b \; (\mathbf{S}_2)_{ab} \leftarrow \left(\frac{(\mathbf{S}_2)_{ab}}{\sqrt{(\mathbf{S}_2)_{aa} \times (\mathbf{S}_2)_{bb}}} \right)^{1/k} \tag{4}$$

3.2. Computation algorithm

Clearly, equations (1)-(2) and (3)-(4) define a system of linear equations, whose solutions correspond to the co-similarities of each pair of documents and each pair of words. Thus, the χ-Sim$_p^k$ algorithm proposed in Hussain *et al.* (2010) is based on an iterative approach, in which each iteration t consists in evaluating the similarities brought by the order-t paths of the documents/words bipartite graph (see Fig. 1). The algorithm is as follows:

```
Algorithm 1 The $\chi$-Sim$_p^k$ algorithm
Input: $\mathbf{R}_{12}$, $It$, $k$, $p$
Output: $\mathbf{S}_1$, $\mathbf{S}_2$
  $\mathbf{S}_1^{(0)} \leftarrow \mathbf{I}$
  $\mathbf{S}_2^{(0)} \leftarrow \mathbf{I}$
  for $t = 1 \rightarrow It$ do
    Compute $\mathbf{S}_1^{(t)}$ with $\mathbf{S}_2^{(t-1)}$ using Eq. (1) and Eq. (2)
    Compute $\mathbf{S}_2^{(t)}$ with $\mathbf{S}_1^{(t-1)}$ using Eq. (3) and Eq. (4)
    Pruning step on $\mathbf{S}_1^{(t)}$ and $\mathbf{S}_2^{(t)}$ consisting of zeroing $p\%$ of the matrices
  end for
```

The inputs of this algorithm is the relation matrix $\mathbf{R}_{12}$, the number It of iterations to compute the final values of $\mathbf{S}_1$ and $\mathbf{S}_2$ and two numerical parameters k and p. We discussed the meaning of k in the previous section. Concerning p, it indicates the percentage of the smallest similarity values in the matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ to set to zero at the end of each iteration; this allows to deal with noise in the data. A discussion about this step being out of the scope of this paper, the interested reader can find further information in Hussain *et al.* (2010).

We need to emphasize that $\mathbf{S}_1^{(0)}$ and $\mathbf{S}_2^{(0)}$ are both initialized to the identity matrix, nevertheless these matrices could be initialized to other values and, in practice, can be seen also as two input parameters. This is used by Hussain & Bisson (2010) in order to adapt this algorithm for text categorization. In this case, input matrices indicate some *a priori* knowledge about the similarity values between documents and between words. In the next section we are going to use this feature for multi-view clustering.

4. An architecture to compute multi-view co-similarities

As we saw in Section 3., from a functional point of view, the χ-Sim$_p^k$ method can be represented in the following way (Fig. 2) where $\mathbf{S}_1$ and $\mathbf{S}_2$ are the input similarity matrices and $\mathbf{S}'_1$ and $\mathbf{S}'_2$ the matrices learned by the algorithm (and previously denoted $\mathbf{S}_1$ and $\mathbf{S}_2$ in the section 3.). This diagram is the basic component we used to deal with multiple matrices.

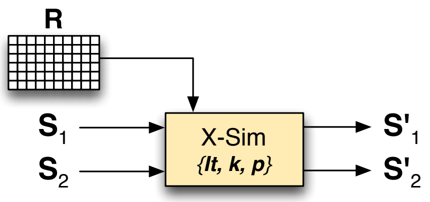

Figure 2: Functional diagram of χ-Sim$_p^k$.

4.1. A general learning architecture

In this section, we now consider a very general model in which the learning dataset is composed of M relation matrices $\mathbf{R}_{ij}$, describing the connection between N different types of objects T_i. Thus, the relational structure of this dataset is a clique since each pair of objects is connected through a data matrix. Our goal is then to compute a co-similarity matrix $\mathbf{S}_i$ for each of the N kinds of objects. The idea behind our learning architecture is to create a learning network isomorphic to the dataset (Fig. 3).

At first, an instance of χ-Sim$_p^k$ algorithm is associated to each relation matrix $\mathbf{R}_{ij}$. This instance is denoted χ-Sim$\,(i,j)$ and computes two similarity matrices $\mathbf{S}'_i$ and $\mathbf{S}'_j$. For sake of simplicity, we consider now that the parameters k, p, and It are set to the same values for every instances χ-Sim

(i, j). Next, for a given type of object T_i, as these instances produce a set of different similarity matrices $\{\mathbf{S}'_i, \mathbf{S}''_i, \ldots\}$, we also need to introduce an *aggregation function*, denoted Σ_i, to compute a consensus matrix merging the current matrix $\mathbf{S}_i$ and the set of matrices produce by the instances χ-Sim(i, j). The dynamic of this network will be discussed in the next section 4.2.

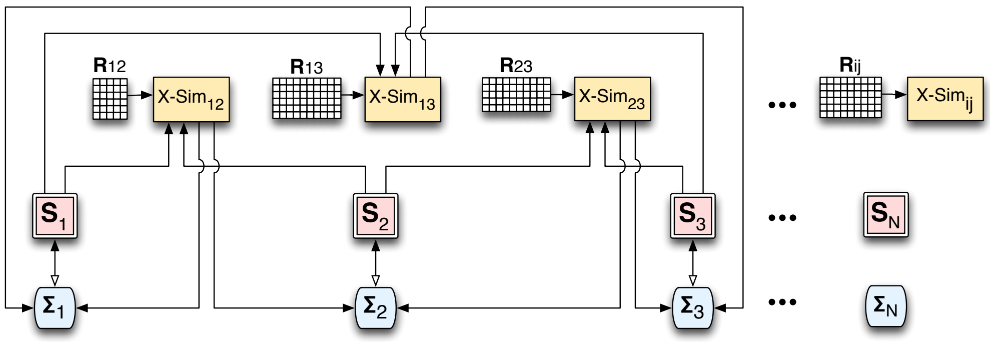

Figure 3: Diagram of the generic architecture for the multi-view clustering. In this figure, we have a complete linkage between the objects T_1, T_2 and T_3.

The topology presented in figure 3 assumes a complete linkage between the objects T_i. However, in practice, other situations may occur. First of all, in many cases, relationships among several objects are missing. For instance, taking back our movie database example (Section 2.): movies are described by two matrices, [movies/actors] and [movies/keywords], but there is no [actors/keywords] relation; in such case the corresponding χ-Sim(i, j) and Σ_i instances and their associated links, will be simply absent of the network. But, it is important to notice that the outputs of χ-Sim(i, j) are always connected to the co-similarity matrix $\mathbf{S}_i$ through the Σ_i and Σ_j functions.

Secondly, in some dataset several relation matrices exist between two objects: indeed, for a given collection of emails, one can define two matrices [users/words], the first one describing the word occurrences in the subjects of the email and the second in their bodies; in such case, there will be two instances of χ-Sim(i, j) in the network, one for each relationship.

Third, a relation can link an object T_i with itself as for a [users/users] matrix denoting the relation "*has sent an email to*"; here, if the relationship is asymmetrical there will be two similarity matrices if we want to differentiate the *sender* ($\mathbf{S}_{i\rightarrow}$) from the *receiver* ($\mathbf{S}_{i\leftarrow}$), else when the relationship is seen as symmetrical just one occurrence of $\mathbf{S}_i$ will exist linked to both inputs of the instance χ-Sim(i,i). To conclude, we must notice that all of these variants are totally supported by our architecture.

4.2. Dynamic of the network and algorithm

As we saw in Fig. 3, the similarity matrices $\mathbf{S}_i$ are connected to the inputs of each corresponding χ-Sim(i,j), allowing the system to spread the information in the network. Thus, with this architecture, various scheduling policies may be considered, about the order in which instances of both χ-Sim(i,j) and Σ_i are fired. One may consider mainly two opposite policies:

- Asynchronous : the χ-Sim(i,j) instances are sequentially run in a static or dynamic order and the similarity matrices $\mathbf{S}_i$ are progressively updated with Σ_i. The problem with this approach is that the order matters: the last instance χ-Sim(i,j) fired will tend to shift $\mathbf{S}_i$ and $\mathbf{S}_j$ toward the implicit similarities expressed by its relation matrix $\mathbf{R}_{ij}$. Thus, without any prior knowledge about the relative interest of the relation matrices this approach seems difficult to optimize.

- Synchronous : the χ-Sim (i,j) instances are run in parallel then the similarity matrices $\mathbf{S}_i$ are simultaneously updated with Σ_i. This policy offers several benefits. First, all the instances of χ-Sim (ij) have the same influence (that could be adjusted by adding some weighting parameters in Σ_i). Second, it becomes possible to study the convergence criteria of the system according to the way Σ_i are defined. Third, this approach allows to do a strong parallelization of the processes, since each instance could be executed on a different core of the CPU.

The synchronous approach can be seen as a generalization of the χ-Sim$_p^k$ algorithm in the sense that it computes the different $\mathbf{S}_i$ matrices as the system just iterates several times equations (1)-(2) and (3)-(4) for each instance of χ-Sim (i,j). By the way, it becomes possible to set to 1 the parameter It of each χ-Sim (i,j) and to introduce a more general parameter, denoted I_G, indicating the global number of iterations to perform in the network. As with χ-Sim$_p^k$ this number of iterations can be generally set to the value 4, leading the system to explore the similarities brought by the order-4 paths of bipartite graphs associated to each matrix $\mathbf{R}_{ij}$.

In this schema, the Σ_i functions have an important role to play: first, they aggregate the multiple similarity matrices produced by the χ-Sim (i, j) instances into one unique similarity matrix; second, the way they are defined must ensure the convergence of the method. Concerning the first point, in an unsupervised learning framework and without any prior knowledge, few strategies can be considered, namely the minimum, the maximum and the average of the similarity values for each pair of objects. In the experimental part (section 5.) we will use the similarity averages. Concerning the convergence we can define each Σ_i in such a way it does not replace the current similarity matrix $\mathbf{S}_i$ by the new consensus matrix but rather it adds this one to the previously computed $\mathbf{S}_i$ weighted by a damping factor.

Thus, under some conditions, convergence of the system can be ensured. More formally, let assume $\lambda \in [0,1[$ to be a damping factor, F to be a merging function (minimum, maximum, average, etc.) of the matrices $\{\mathbf{S}'_i, \mathbf{S}''_i, ...\}$ returning a matrix which values are all in $[0,1]$, and $\mathbf{S}_i^{(t-1)}$ the previously computed similarity matrix of instances of T_i, the generic formula used to compute the aggregated matrix at iteration t is as follows:

$$\Sigma_i = \frac{1}{1+\lambda^t}\left(\mathbf{S}_i^{(t-1)} + \lambda^t \times \mathrm{F}(\mathbf{S}'_i, \mathbf{S}''_i, \ldots)\right) \qquad (5)$$

As the F function is bounded and the damping factor λ^t is exponentially decreasing, this formula ensure the convergence of the sequence composed of the successive similarity matrices computed by the Σ_i function. In the experiments, λ equals 0.5. The complete algorithm is as follows:

Algorithm 2 The multi-view algorithm

```
Input: A collection of relational matrices $\{\mathbf{R}_{i,j}\}$, $I_G$, $k$, $p$
Input: Parameters ($I_G$, $\lambda$, $k$, $p$) — Default values (4, 0.5, 0.8, 0.4)
Output: A collection of similarity matrices $\{\mathbf{S}_i\}$
  foreach i : $\mathbf{S}_i \leftarrow \mathbf{I}$
  for $t = 1 \rightarrow I_G$ do
    Execute every $\chi$-Sim$(i, j)$ with parameters $It$=1, $k$, $p$
    Update every $\mathbf{S}_i$ using Eq. (5).
  end for
```

4.3. Complexity and parallelization

The complexity of the multi-view architecture is obviously closely related to the one of the χ-Sim$_p^k$ algorithm (see algorithm 1). Let a relational matrix $\mathbf{R}_{i,j}$ of size n by p, as this algorithm consists in the multiplication of three matrices, the complexity to compute a similarity matrix of size n^2 between raws is $\mathcal{O}(np^2 + n^2p)$ and identical to the complexity to compute a similarity matrix of size p^2 between columns. In the multi-view architecture, as each instance of χ-Sim(i, j) can easily run on an independent core of a CPU, the method can be easily parallelized, the global similarity remaining unchanged. Let notice that the functions Σ_i have a quadratic complexity in $\mathcal{O}(p^2)$ or $\mathcal{O}(n^2)$ and thus can be ignored.

Until now, we considered the multi-view clustering as a way to combine knowledge coming from different sources of data. However, this approach can be also interesting to turn a large problem into a collection of simpler ones. For example, let us consider a problem with one relational matrix [documents/words] of size n by p in which we just want to cluster the documents. If the number of words is huge with respect to the number of documents, we could split the problem into a collection of h matrices of size n by p/h. By using the multi-view architecture we gain both in time and space complexity: indeed, the time complexity decreases from $\mathcal{O}(np^2 + n^2p)$ to $\mathcal{O}(1/h^2(np^2) + 1/h(n^2p))$ leading to an overall gain of $1/h$ when $n < p$. In the same way the memory needed to store the similarity matrices between words decreases in $1/h$.

5. Experiments

In this section, we present the experiments conducted to assess the performance of our multi-view architecture on real-world datasets. More precisely, we tried to answer two questions discussed in two separated sections:

1. Does the co-similarity based multi-view architecture allow to provide better results than the classical co-clustering methods working on only one relation matrix? (section 5.1.)

2. Is the splitting approach, proposed in section 4.3., an efficient way to deal with large matrices in the same amount of runtime and memory space thanks to the parallelization of the algorithm? (section 5.2.)

As we are in the clustering context, our evaluation is classically based on the following method. First, we select some dataset in which labeled clusters already exist and then we evaluate the correlation degree between the learned and known clusters by the analysis of the confusion matrix using for instance the micro-averaged precision (Pr) from Dhillon *et al.* (2003).

5.1. Evaluation of the multi-view approaches

Test dataset. We used seven databases, the quantitative characteristics of them being described in Table 1. The first dataset is extracted from the *IMDb*[4] website. Some pre-processing steps has been done in order to remove the rarest actors and keywords. We have three types of objects: movies, actors and keywords; and two relation matrices: the [movies/actors] matrix and the [movies/keywords] matrix. In addition, we built a third matrix which is the concatenation of the two previous matrices, referred to as [Key+Act], in order to provide the single-view approach with a dataset containing all the data.

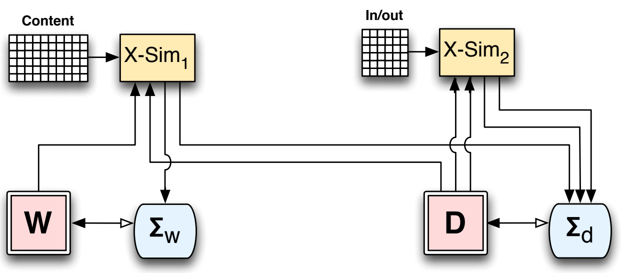

Figure 4: Architecture used to deal with the Web data.

The six other databases concern Web data and are all constructed on the same structure with two types of objects (Documents and Words) and four relations matrices. The [documents/words] matrix describes the content of the documents using a classical bag of words representation, and the three other [documents/documents] matrices corresponding to the inbound, outbound and citation links between documents. However, on the one hand the outbound is just a transposition of the inbound matrices and on the other hand, the citation matrix is just the sum of the two others. Therefore, in the multi-view architecture used here we just have two relation matrices (Figure 5.1.).

[4]`http://www.imdb.com/interfaces/`

Table 1: Description of the seven databases. The Links column gives the number of relations occurring in the [Documents/Documents] matrices.

Dataset	Movies	Keywords	Actors	Clusters
IMDb	617	1878	1398	17

Dataset	Documents	Words	Links	Clusters
Cora	2708	1433	5429	7
Citeseer	3312	3703	4732	6
Cornell	195	1703	569	5
Texas	187	1703	578	5
Washington	230	1703	783	5
Winconsin	265	1703	938	5

More precisely, we used the Cora and CiteSeer dataset (Bickel & Scheffer, 2005; Drost *et al.*, 2006; Sen *et al.*, 2008) and four datasets coming from the WebKB[5] describing the pages of four universities (Cornell, Texas, Washington and Wisconsin), classified in five classes (student, project, staff, course, faculty). On the basis of these seven benchmarks, we compared our multi-view architecture based on χ-Sim with:

- **Cosine**, **LSA** (Deerwester *et al.*, 1990), **SNOS** (Liu *et al.*, 2004), **CTK** (Yen *et al.*, 2009) and $\boldsymbol{\chi}$**-Sim**$_p^k$ (Hussain *et al.*, 2010) that are five classical similarity or co-similarity measures;
- **ITCC** (Dhillon *et al.*, 2003) a well-known co-clustering system;
- **MVKM** (Drost *et al.*, 2006) which is an adaptation of the *k*-means clustering algorithm to the multi-view context.

Test methodology. For the similarity measures : Cosine, LSA, SNOS, CTK and χ-Sim, the clusters has been generated by an *Agglomerative Hierarchical Clustering* (AHC) method on the similarity matrices along with Ward's linkage. Then, we cut the clustering tree at the level corresponding to the number of document clusters we are waiting for (17 for the dataset IMDb, etc). We used the classical *micro-averaged precision* (Pr) (Dhillon *et al.*, 2003) for comparing the accuracy of the document clustering for which the higher value, the better performance: value 100% representing a perfect correlation

[5]`http://www.cs.umd.edu/projects/linqs/projects/lbc/`

between the learned and known clusters. For MVKM, as we don't have a running implementation, we directly quote the best values for the CiteSeer dataset from Drost *et al.* (2006).

It is worth noticing that many of the these methods have some setting parameters. Thus, in order to keep a fair comparison, we sought for the better values for these parameters either by testing several values and reporting the best precision and/or by using the values recommended by the authors.

Table 2: Results of the experiments performed on the 7 dataset. The best results obtained for each dataset is written in bold

Dataset	Single-view algorithms					Multi-view approach
	View	Best	Pr	Second	Pr	
Movie	Key+Act	CTK	33.2%	χ-Sim	30.6%	**34.7%**
Cora	Citation	χ-Sim	63.4%	LSA	49.8%	**69.7%**
Citeseer	Content	χ-Sim	60.8%	ITCC	46.8%	**63.5%**
Cornell	Content	χ-Sim	63.1%	LSA	57.9%	**69.2%**
Texas	Content	χ-Sim	**72.2%**	LSA	66.3%	61.5%
Washington	Content	LSA	**65.2%**	χ-Sim	63.5%	61.7%
Wisconsin	Content	χ-Sim	**67.5%**	LSA	60.4%	**67.5%**

Table 2 reports the results obtained by the different clustering methods. We tested every mono-view algorithms on all the seven datasets, however, in order to easy the reading of the results we just report for each dataset: the name of the view providing the best result and the precision of the best and second methods on this view. As we can see, the multi-view architecture, described in Sect. 4., obtains the best micro-averaged precision in all the datasets but two : Texas (best: χ-Sim) and Washington (best: LSA). We are investing the reason why our algorithm partially fails on these two datasets. But, in many cases this architecture which can be seen as a generalization of the χ-Sim method, is better that this one. Moreover, we observe the multi-view architecture is far less sensible to the value of the pruning parameter, providing a more robust approach.

Finally, we compare (table 3) our architecture with MVKM (Drost *et al.*, 2006). As the authors provided a measure of entropy of their resulting clustering, we simply quote their best result. The lower the entropy is, the better the clustering, as it measure disagreement between the known clusters, and the learnt one. Here too, our approach achieves better results that MVKM.

Table 3: Comparison between MVKM, χ-Sim$_p^k$ and our architecture.

CiteSeer	MVKM	χ-Sim$_p^k$	Multi-view arch.
Entropy	1.60	1.27	1.07

5.2. Evaluation of the splitting approach

Here, we analyze the performance of our multi-view architecture when a relational matrix is splitted into a collection of smaller matrices. As explained in Section 4.3., with such approach, we can process larger datasets with the same running time and a smaller memory footprint.

For this test we use the classical NG20 dataset consisting of approximately 20,000 newsgroup articles collected from 20 different Usenet groups. We create subsets of NG20 named M2, M5 and M10 (Dhillon *et al.*, 2003), as well as the subsets NG1, NG2, and NG3 (Long *et al.*, 2006). Details about the content of these subsets are given in Table 4.

Table 4: Description of the subsets of the NG20 dataset used.

Dataset	Newsgroups included	Clusters	Docs.
M2	talk.politics.mideast, talk.politics.misc	2	500
M5	comp.graphics, rec.motorcycles, rec.sport.baseball, sci.space, talk.politics.mideast	5	500
M10	alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.gun	10	500
NG1	rec.sports.baseball, rec.sports.hockey	2	400
NG2	comp.os.ms-windows.misc, comp.windows.x, rec.motorcycles, sci.crypt, sci.space	5	1000
NG3	comp.os.ms-windows.misc, comp.windows.x, misc.forsale, rec.motorcycles, sci.crypt, sci.space, talk.politics.mideast, talk.religion.misc	8	1600

The results of our experiment are presented in Table 5, two configurations being tested: in the first one, we generate three datasets of 1, 2 and 4 matrices, each matrix containing 500 different words; in the second one, the matrices contain 1000 different words. The words have been selected by running k-medoids to get the most representative ones. It is important to emphasize that each configuration needs the same time to run for a parallelized version of our multi-view architecture, independently of the number of matrices.

For the simpler dataset (M2, NG1), the better results are obtained when there is only one matrix. This result can be explained by the fact that smaller dataset contains less different words, thus the added words have a high probability to be not relevant. When the dataset becomes bigger (M5, NG2, NG3) the datasets containing several views achieve the best results.

Table 5: Results of the splitting approach

Dataset	M2	M5	M10	NG1	NG2	NG3
Multi-view (1×500)	**74.5**	73.8	**50.7**	78.2	64.2	50.9
Multi-view (2×500)	71.0	74.6	47.4	62.9	66.8	60.9
Multi-view (4×500)	65.3	75.6	46.3	54.6	68.1	59.5
Multi-view(1×1000)	**74.5**	73.8	**50.7**	**80.9**	68.0	58.2
Multi-view (2×1000)	71.9	76.9	49.9	64.9	**71.8**	**63.3**
Multi-view (4×1000)	64.3	**78.4**	49.1	57.3	68.9	63.1

6. Conclusion

In this article, we proposed a multi-view architecture to tackle the problem of learning co-similarities from a collection of matrices describing interrelated types of objects. Our approach is an extension of the χ-Sim$_p^k$ co-similarity (Bisson & Hussain, 2008) to the multi-view clustering problem.

This new architecture provides some interesting properties both in term of convergence and scalability and it allows an simple parallelization of the processes. The experiments shown this method outperform in several tests the classical approaches dealing with one matrix at a time.

Acknowledgment

This work is supported by the French ANR project FRAGRANCES under grant 2008-CORD 00801.

References

ACAR E. & YENER B. (2009). Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, **21**, 6–20.

BANERJEE A., BASU S. & MERUGU S. (2007). Multi-way clustering on relation graphs. In *SDM*: SIAM.

BICKEL S. & SCHEFFER T. (2004). Multi-view clustering. In *Proceedings of the IEEE international conference on data mining*: Citeseer.

PEDRYCZ W. (2002). Collaborative fuzzy clustering. *Pattern Recognition Letters*, **23**(14), 1675–1686.

TANG W., LU Z. & DHILLON I. S. (2009). Clustering with multiple graphs.

AGGARWAL C. C., HINNEBURG A. & KEIM D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Lecture Notes in Computer Science*, p. 420–434: Springer.

BARBER M. (2007). Modularity and community detection in bipartite networks. *Physical Review E*, **76**(6), 066102.

BEKKERMAN R., EL-YANIV R. & MCCALLUM A. (2005). Multi-way distributional clustering via pairwise interactions. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, p. 41–48, New York, NY, USA: ACM.

BICKEL S. & SCHEFFER T. (2005). Estimation of mixture models using co-em. *Machine Learning: ECML 2005*, p. 35–46.

BISSON G. & HUSSAIN F. (2008). Chi-sim: A new similarity measure for the co-clustering task. In *Proceedings of the Seventh ICMLA*, p. 211–217: IEEE Computer Society.

BLUM A. & MITCHELL T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, p. 92–100: ACM.

CHAUDHURI K., KAKADE S., LIVESCU K. & SRIDHARAN K. (2009). Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th annual international conference on machine learning*, p. 129–136: ACM.

CHENG Y. & CHURCH G. M. (2000). Biclustering of expression data. In *Proceedings of the International Conference on Intelligent System for Molecular Biology*, p. 93–103, Boston.

CRUZ GOMEZ J. D., BOTHOREL C. & POULET F. (2011). Entropy based community detection in augmented social networks. In ., p. 163–168, Salamanca, Espagne. 10898.

DE CARVALHO F., LECHEVALLIER Y. & DE MELO F. M. (2012). Partitioning hard clustering algorithms based on multiple dissimilarity matrices. *Pattern Recognition*, **45**(1), 447 – 464.

DE SA V. R. (2005). Spectral clustering with two views. In *ICML workshop on learning with multiple views*.

DEERWESTER S., DUMAIS S. T., FURNAS G. W., THOMAS & HARSHMAN R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, **41**, 391–407.

DHILLON I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 269–274, New York, NY, USA: ACM.

DHILLON I. S., MALLELA S. & MODHA D. S. (2003). Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD*, p. 89–98.

DROST I., BICKEL S. & SCHEFFER T. (2006). Discovering communities in linked data by multi-view clustering. *From Data and Information Analysis to Knowledge Engineering*, p. 342–349.

DU N., WU B., PEI X., WANG B. & XU L. (2007). Community detection in large-scale social networks. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, p. 16–25, New York, NY, USA: ACM.

DUCH J. & ARENAS A. (2005). Community detection in complex networks using extremal optimization. *Physical Review E*, **72**(2), 027104.

FORTUNATO S. (2010). Community detection in graphs. *Physics Reports*, **486**(3-5), 75–174.

FRIGUI H., HWANG C. & RHEE F. C.-H. (2007). Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, **40**(11), 3053 – 3068.

GRIMAL C. & BISSON G. (2011). Amélioration de la co-similarité pour la classification de documents. In *Conférence Francophone sur l'Apprentissage Automatique*.

HUSSAIN S. F. & BISSON G. (2010). A supervised approach to text categorization using higher order co-occurrences. In *Society for Industrial and Applied Mathematics International Conference on Data Mining (SDM 2010)*, Columbus, Ohio.

HUSSAIN S. F., GRIMAL C. & BISSON G. (2010). An improved co-similarity measure for document clustering. In *ICMLA*.

IAM-ON N., BOONGOEN T. & GARRETT S. (2010). Lce: a link-based cluster ensemble method for improved gene expression data analysis. *Bioinformatics*, **26**(12), 1513–1519.

INGARAMO D., ERRECALDE M. & ROSSO P. (2010). A general bio-inspired method to improve the short-text clustering task. In A. GELBUKH, Ed., *Computational Linguistics and Intelligent Text Processing*, volume 6008 of *Lecture Notes in Computer Science*, p. 661–672. Springer Berlin / Heidelberg.

KONTOSTATHIS A. & POTTENGER W. M. (2004). A framework for understanding latent semantic indexing (lsi) performance.

KUMAR A. & DAUME III H. (2011). A co-training approach for multi-view spectral clustering. In L. GETOOR & T. SCHEFFER, Eds., *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, p. 393–400, New York, NY, USA: ACM.

LIU N., ZHANG B., YAN J., YANG Q., YAN S., CHEN Z., BAI F. & YING MA W. (2004). Learning similarity measures in non-orthogonal space. In *Proceedings of the 13th ACM CIKM*, p. 334–341: ACM Press.

LONG B., ZHANG Z. M., WÚ X. & YU P. S. (2006). Spectral clustering for multi-type relational data. In *Procceedings of ICML'06*, p. 585–592, New York, NY, USA: ACM.

LONG B., ZHANG Z. M. & YU P. S. (2005). Co-clustering by block value decomposition. In *Proceedings of the Eleventh ACM SIGKDD*, p. 635–640, New York, NY, USA: ACM.

MADEIRA S. C. & OLIVEIRA A. L. (2004). Biclustering algorithms for biological data analysis: A survey.

REGE M., DONG M. & FOTOUHI F. (2008). Bipartite isoperimetric graph partitioning for data co-clustering. *Data Min. Knowl. Discov.*, **16**(3), 276–312.

SEN P., NAMATA G. M., BILGIC M., GETOOR L., GALLAGHER B. & ELIASSI-RAD T. (2008). Collective classification in network data. *AI Magazine*, **29**(3), 93–106.

SPEER N., SPIETH C. & ZELL A. (2004). A memetic clustering algorithm for the functional partition of genes based on the gene ontology.

TANG W., LU Z. & DHILLON I. S. (2009). Clustering with multiple graphs.

WANG X.-J., MA W.-Y., XUE G.-R. & LI X. (2004). Multi-model similarity propagation and its application for web image retrieval. In *Proceedings of the 12th annual ACM MULTIMEDIA*, p. 944–951, New York, NY, USA: ACM.

YEN L., FOUSS F., DECAESTECKER C., FRANCQ P. & SAERENS M. (2009). Graph nodes clustering with the sigmoid commute-time kernel: A comparative study. *Data Knowl. Eng.*, **68**(3), 338–361.

ZANGHI H., VOLANT S. & AMBROISE C. (2010). Clustering based on random graph model embedding vertex features. *Pattern Recognition Letters*, **31**(9), 830 – 836.

ZHOU D. & BURGES C. (2007). Spectral clustering and transductive learning with multiple views. In *Proceedings of the 24th international conference on Machine learning*, p. 1159–1166: ACM.