Apprentissage multi-vue de co-similarités pour la classification

Gilles Bisson et Clément Grimal

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

Résumé : En classification, les données se présentent souvent sous la forme d'une matrice de données unique [objets/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 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 ce papier, nous introduisons une architecture permettant d'étendre les capacités de l'algorithme de calcul de co-similarité χ-SIM (Hussain et al., 2010) 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 un cadre formel intéressant et permet 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

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, such as [documents/terms] matrix in natural language processing). 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 also frequent. For instance, in a social network

we can have simultaneously relations between pairs of users, users and documents, documents and terms, ... each relation providing a different *view* on the data. 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 are not taken into account, thus leading to a loss of information (under the hypothesis that the views are not self-contradictory). Therefore, handling the views together, referenced as the *multi-view clustering* task, is an interesting challenge in the learning domain. The present work is an extension to the multi view problem of an existing algorithm, named χ-SIM Hussain et al. (2010), which obtained good results on the co-clustering task. We selected this approach for two complementary reasons. First, it builds similarity matrices rather than clusters between rows and columns of a data matrix ; this is useful in the multi-view context since it allows us to combine easily the set of similarity measures 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 about the data ; thereby, it becomes possible to iteratively transfer the similarities computed from one view to the others. The rest of the paper is structured as follows. In Sect. 2 we present some related work about multi-view clustering. In Sect. 3 we provide a rapid insight about the χ-SIM method and then, in Sect. 4, we present and analyze the MVSIM architecture allowing to adapt this algorithm to the multi-view context. In Sect. 5, experimental results are reported to quantify the improvements achieved by our approach with respect to single and multi-view approaches. Finally, in Sect. 6, we present conclusions and future work.

2. Definitions and related work

The co-clustering task has been intensively explored in many domains, such as *information retrieval*, to deal with documents Dhillon (2001); Long et al. (2005) ; *bioinformatics*, to analyze gene expression data and in *social networks*, to detect community of users. As emphasized by Long et al. (2005), co-clustering comes along with the advantages of improving the cluster quality when dealing with large dimensional sparse data, and of highlighting similarities between objects described by sets of different features.

Following the case of co-clustering, the idea behind multi-view approaches is to take into account all the objects and relations of the dataset, to improve

the quality of the clusters. An example of a simple multi-view dataset can be found when considering a movies database, in which movies are described both by their actors and by their keywords. When considering such a dataset, two equivalent representation paradigms can be used : a collection of matrices, and a k-partite graph Long et al. (2006). In the latter, each subset of nodes contains the instances of one type of objects, and a link between two nodes of different subsets represents the relation between them. Alternatively, when working with a collection of matrices, each matrix describes a relation between two types of objects corresponding to the rows and the columns of the matrix. In the previous example, we would have two matrices : a [movies/actors] matrix and a [movies/keywords] one. Note that in the remaining of this paper, we will mostly use the collection of matrices paradigm to represent the datasets, as it is better suited to explain our algorithm.

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, Drost et al. (2006) and Bickel & Scheffer (2004) describe an extension of the classical k-means (MVKM) and of EM algorithms for the multi-view setting. Some approaches such as Chaudhuri et al. (2009), are built upon the canonical correlation analysis, a statistics method to find linear combinations of variables from different highly correlated sets, to extract relevant features from multiple views, and then to 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 multipartite 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 through the eigenvectors of the Laplacian matrix (MVSC). Finally, information theory can also be used to separately cluster the objects on each view, but by simultaneously maximizing a unique objective function based on the mutual information between clusters, as in Bekkerman et al. (2005).

Multi-view clustering can also be tackle by *consensus clustering* methods which aims at combining the results of multiple clusterings (coming from different views or different methods) into one Azimi & Fern (2009); Fred & Jain (2002); Li & Ding (2008); Strehl & Ghosh (2003). In Strehl & Ghosh (2003), one present three different approaches to combine clusters based on graph-

partitioning. The first method, similarly to Fred & Jain (2002), produces a meta-similarity matrix based on how many times objects appear in the same cluster, and then perform the final clustering from this matrix. In Azimi & Fern (2009); Li & Ding (2008), the authors deal with the *clustering ensemble* selection problem, which aims at choosing the best clustering among a large set of partitions. Li & Ding (2008) build a weighted consensus clustering method ; whereas Azimi & Fern (2009) adapt their selection strategy according to the stability of the clusterings. 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). So, in Tang et al. (2009), information coming from distinct sources, describing different relations between the same instances of objects, are merged using Linked Matrix Factorization. Similarly de Carvalho et al. (2012) builds clusters from multiple similarity matrices computed along different views ; the algorithm then learns a matrix of "relevance weights" between clusters and views, taking into account that some views are better at describing some clusters. Finally, in Pedrycz (2002), fuzzy clustering is used in the multi-view setting, by first computing membership matrices for all objects in the views, and next modifying these matrices through collaboration between views. The end goal is quite different from the other approaches though, as this method produces as many clusterings as views, and does not build a sole clustering.

Throughout this paper, we 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 types of objects in the dataset. $\forall i \in 1..N$, T_i is the type of object i (i.e. users, documents, words, etc.) For the sake of simplicity, we assume that each T_i has 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 connections 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 example, in a [documents/terms] matrix it can be expressed as the frequency of the b^{th} term in the a^{th} document.

Similarity matrices : thus, we consider N square and symmetrical similarity matrices $\mathbf{S}_1 \dots \mathbf{S}_N$, each $\mathbf{S}_i$ containing the similarities between all the pairs of instances of T_i. The values of the similarity measure must be in $[0, 1]$.

3. The χ-SIM Algorithm

We present the main aspects of the χ-SIM co-similarity measure Hussain et al. (2010) which is a basic component of our MVSIM architecture (Sect. 4.). As this algorithm processes one matrix at a time, we simplify the notations by just considering 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 similarity matrices $\mathbf{S}_1$ (documents) and $\mathbf{S}_2$ (words). The main idea behind χ-SIM is to make use of the duality between documents and words : each one being a descriptor of the other. This is achieved by simultaneously 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. A similar idea has also been used for supervised leaning in Liu et al. (2004). With such approaches, documents (resp. words) can be seen as similar even if they do not explicitly share words (resp. documents). This is an interesting feature to deal with some language difficulties like terminological variability : two authors writing about the same topic may used different vocabulary, leading classical measure to underestimate similarity between documents of this topic. Once the similarity matrices have been generated with χ-SIM, they can be used by any clustering algorithm (for example k-means) to organize documents and/or words into clusters. However, due to the interleaved way these similarities have been computed, the resulting clusters are similar to those obtained with a genuine co-clustering algorithm. In practice, the similarity matrix $\mathbf{S}_1$ between documents is evaluated in two steps :

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

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

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 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. The similarity matrix $\mathbf{S}_2$ is defined in a similar way, by

just "toggling" the role of matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ in Eq. (1) and (2). Finally, these equations lead to a set of equations solved with this iterative approach :

Algorithm 1 χ-SIM algorithm

Input: $\mathbf{R}_{12}$, I_t, k, p
  Let $\mathbf{S}_1$ and $\mathbf{S}_2$ the similarity matrices
  Initialize $\mathbf{S}_1^{[0]}$ and $\mathbf{S}_2^{[0]}$ with $\mathbf{I}$
  for $t = 1 \rightarrow I_t$ **do**
    Compute $\mathbf{S}_1^{[t]}$ with $\mathbf{S}_2^{[t-1]}$ using Eq. (1) and (2)
    Compute $\mathbf{S}_2^{[t]}$ with $\mathbf{S}_1^{[t-1]}$
    Pruning of $\mathbf{S}_1^{[t]}$ and $\mathbf{S}_2^{[t]}$
  end for

Thus, inputs of this algorithm are the matrix $\mathbf{R}_{12}$, the number I_t of iterations to compute the values of $\mathbf{S}_1$ and $\mathbf{S}_2$ and two numerical parameters k and p. We already discussed the meaning of k. Variable p 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. One can notice that with parameters $I_t = 1$, $k = 1$ and $p = 0$, χ-SIM is strictly equivalent to the Cosine similarity. Here, $\mathbf{S}_1^{[0]}$ and $\mathbf{S}_2^{[0]}$ are both initialized to the identity matrix, expressing the fact that, without any prior knowledge, an object is similar to itself. However, other initializations can be used (and thus these matrices can be seen as input parameters as well) to provide some information about the similarity values between documents and between words. This possibility will be used in the next section to deal with multi-view similarity learning.

To illustrate this algorithm, let us consider the toy dataset in Fig. 1 extracted from Hussain et al. (2010). At the first iteration, documents d_1 and d_4 do not share any words and thus their similarity is equal to 0, as with any classical similarity measure. However, as words w_3 and w_4 are both appearing in d_3, they have a non-null similarity value in $\mathbf{S}_2^{[1]}$ (and the same for w_2 and

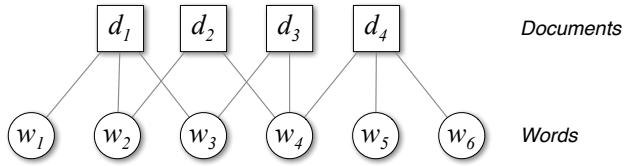

FIGURE 1: Example of a documents-words bi-partite graph.

w_4 through d_2) ; therefore, at the second iteration as d_1 contains word w_3 and d_4 contains word w_4, the similarity between documents d_1 and d_4 in $\mathbf{S}_1^{[2]}$ will become not null. Practically, each iteration t consists in evaluating the similarities brought by the order-t paths in the documents/words bipartite graph.

4. The MVSIM Architecture

In this section, we introduce the MVSIM architecture to deal with datasets having multiple relation matrices, that are seen as different views. This approach is both simple and scalable through the processes parallelization. As we saw in Section 3., from a functional point of view, χ-SIM can be generalized in the following way (Fig. 2) in which $\mathbf{S}_i$, $\mathbf{S}_j$ are the input similarity matrices and $\mathbf{S}_i^{(i,j)}$, $\mathbf{S}_j^{(i,j)}$ those learned using the dataset $\mathbf{R}_{ij}$. This diagram is the basic component we are using to deal with the multi-view setting.

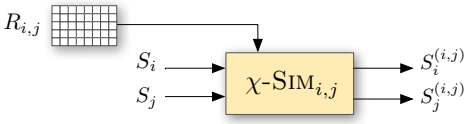

FIGURE 2: Functional diagram of χ-SIM.

4.1. General learning architecture

Now, we consider again a very general model in which the dataset is composed of M relation matrices $\mathbf{R}_{ij}$, describing the connections between N different kind of objects T_i. Our goal is then to compute a co-similarity matrix $\mathbf{S}_i$ for each of these kinds of objects taking into account all the information expressed in the relations. The idea behind our learning architecture is to create a learning network isomorphic to the relational structure of the dataset (Fig. 3). In this model, an instance of χ-SIM is associated to each relation matrix $\mathbf{R}_{ij}$ of the dataset. This instance is denoted χ-SIM(i,j) and it computes the similarity matrices $\mathbf{S}_i^{(i,j)}$ and $\mathbf{S}_j^{(i,j)}$. For the sake of simplicity, we consider that parameters k, p, and I_t are set to the same values for every χ-SIM(i,j) instances. For a given type of object T_i, as each χ-SIM$(i,.)$ instance produces a similarity matrix, we then obtain a set of similarity matrices : $\{\mathbf{S}_i^{(i,1)}, \mathbf{S}_j^{(i,2)} ...\}$. We thus need to introduce an *aggregation function*, denoted Σ_i, to compute a consensus similarity matrix merging the elements of this set with the current

matrix $\mathbf{S}_i$. These consensus matrices, in turn, are used as input of the χ-SIM (i, j) instances, thus *creating feedback loops*. The dynamic of the network and the computation complexity will be discussed in the remaining of this section.

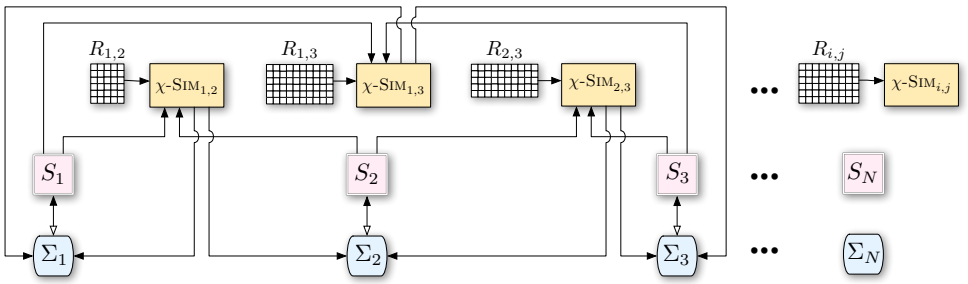

FIGURE 3: Functional diagram of MVSIM. In this figure, we have only three objects T_1, T_2 and T_3 and a complete linkage between them.

In the topology presented in Fig. 3, we assume a *complete linkage* between the objects T_i (i.e. T_1 and T_2 are linked through $\mathbf{R}_{1,2}$, T_1 and T_3 through $\mathbf{R}_{1,3}$ and T_2 and T_3 through $\mathbf{R}_{2,3}$) but, in practice, many other situations may arise.

First of all, in many cases, relationships among several objects are missing. For instance, taking back our movie database example (Sect. 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 simply be absent of the network. However, the outputs of χ-SIM (i, j) are nevertheless always connected to the co-similarity matrix $\mathbf{S}_i$ through the Σ_i and Σ_j functions ; this constraint will become obvious after detailing the network dynamic.

Secondly, in some datasets, several relation matrices exist between two objects T_i and T_j. For example, if you consider a dataset containing a collection of emails, one can easily defines 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 cases, there will be two instances of χ-SIM (i, j) in the network, one for each relationship but the overall structure will be the same.

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, two cases can be considered : first, if the link is asymmetrical, we would create two similarity matrices to differentiate the *sender* ($\mathbf{S}_{i\rightarrow}$) from the *receiver* ($\mathbf{S}_{i\leftarrow}$), second, when the link is seen as symmetric, only one occurrence of $\mathbf{S}_i$ would exist, connected to both inputs of the instance χ-SIM (i, i). To conclude, MVSIM is generic enough to deal with all these special cases without any modification.

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$(i,.)$, allowing the system to spread the information within the network. Various scheduling policies may be considered, about the order in which the instances of χ-SIM$\,(i,j)$ and Σ_i must be fired. One may consider mainly two opposite policies, both of them allowing to parallelize computation, since each instance could be executed on a different core.

- *Asynchronous* : χ-SIM(i,j) instances are run in a static or dynamic order and $\mathbf{S}_i$ are updated immediatly. 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 $\mathbf{R}_{ij}$ leading to increase the "weight" of this node. Thus, without any prior knowledge about the relative interest of the data matrices, this approach seems difficult to optimize. In the rest of the paper we do not use it.
- *Synchronous* : χ-SIM(i,j) instances are run in parallel, then the similarity matrices $\mathbf{S}_i$ are simultaneously updated with Σ_i aggregation function. This policy offers two benefits. First, all the instances of χ-SIM$\,(i,j)$ have the same influence (that could be adjusted by adding some weighting parameters in Σ_i but for sake of simplicity we will not consider this possibility). Second, it becomes possible to study the convergence criteria of the system according to the way Σ_i functions are defined.

It is worth to notice that the synchronous approach can be seen as a generalization of χ-SIM in the sense that the iteration loop used in Algorithm 1 to compute the $\mathbf{S}_i$ and $\mathbf{S}_j$ matrices can be done at the level of the network. In this way, it becomes possible to set the parameter I_t of each χ-SIM(i,j) to 1 and to introduce a more general parameter, denoted I_G, indicating the global number of iterations to perform within the network. The meaning of I_G remains the same as in χ-SIM : each iteration t allows us to take into account the *order-t* paths of the bipartite graph associated to each matrix $\mathbf{R}_{ij}$.

4.3. Aggregation function and algorithm

The Σ_i functions have two roles : first, they aggregate the 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 along the iterations. Concerning the first aspect, in an unsupervised learning framework and without any prior knowledge, relatively few strategies exist to merge several similarity matrices. For a given set $\{\mathbf{S}_i^{(i,1)}, \mathbf{S}_j^{(i,2)} ...\}$ of simila-

rity matrices, we can mainly consider the element-wise operations *minimum*, *maximum* and *average*. Concerning the convergence problem, we can first define each Σ_i in such a way that it does not replace the current similarity matrix $\mathbf{S}_i$ by the new consensus one, but rather that it adds this one to the previously computed $\mathbf{S}_i$, this sum being weighted by a *damping factor*. To ensure the convergence of the computation, let $\lambda \in [0,1[$ be a damping parameter, let F be a merging function (minimum, maximum, average, etc.) combining the matrices $\{\mathbf{S}_i^{(i,1)}, \mathbf{S}_j^{(i,2)}...\}$ into a new matrix whose elements belong to $[0,1]$, and let $\mathbf{S}_i^{[t-1]}$ be the previously computed similarity matrix of instances of T_i. Here is the formula used to compute the aggregated matrix at iteration t :

$$\Sigma_i = \frac{1}{1+\lambda^t}\left(\mathbf{S}_i^{[t-1]} + \lambda^t \times F(\mathbf{S}_i^{(i,1)}, \mathbf{S}_i^{(i,2)}...)\right) \quad (3)$$

As the F function is bounded and the damping factor λ^t is exponentially decreasing with t, this formula ensures the convergence of the sequence composed of the successive similarity matrices computed by the Σ_i functions. This explains why a Σ_i function must always exist in the network even if there is just one instance of χ-SIM$(i,.)$. We can also notice that such damping mechanism does not exist in χ-SIM. The complete algorithm for MVSIM follows :

Algorithm 2 The MVSIM algorithm

Input: A collection of relational matrices $\{\mathbf{R}_{i,j}\}$
Input: Parameters I_G, λ, k, p
  Let $\{\mathbf{S}_i\}$ the similarity matrices
  foreach i : $\mathbf{S}_i \leftarrow \mathbf{I}$
  for $t = 1 \rightarrow I_G$ **do**
    Execute every χ-SIM(i,j) with I_t=1, k, p
    Update every $\mathbf{S}_i$ with Σ_i using Eq. (3).
  end for

4.4. Complexity and parallelization

The complexity of the MVSIM architecture is obviously closely related to the one of χ-SIM (see Algorithm 1). Let us consider a relational matrix $\mathbf{R}_{ij}$ of size n by m, as this algorithm consists in multiplying three matrices, the overall complexity to compute a similarity matrix of size n^2 (rows) is

$\mathcal{O}(nm^2 + n^2m)$, that is identical to the complexity to compute a similarity matrix of size m^2 (columns). In the MVSIM, as each instance of χ-SIM(i, j) can easily run on an independent core, this method can easily be parallelized, thus keeping the global complexity unchanged. In a single core implementation, we need to multiply the complexity by a factor M corresponding to the size of the collection (assuming the $\mathbf{R}_{ij}$ matrices have similar sizes). The complexity of the Σ_i functions being equal to $\mathcal{O}(m^2)$ or $\mathcal{O}(n^2)$, it can be ignored.

Until now, we considered multi-view clustering as a way to combine different sources of data. However, the MVSIM can be also interesting *to split a large problem into a collection of smaller ones*. For example, let us consider a problem with one [documents/words] matrix of size n by m in which we just want to cluster the documents. If the number of words is huge with respect to the number of documents, we can divide the problem into a collection of h matrices of size n by m/h. Thus, by using a distributed version of MVSIM on h cores, we will gain both in time and space complexity : indeed, the time complexity decreases from $\mathcal{O}(nm^2 + n^2m)$ to $\mathcal{O}(1/h^2(nm^2) + 1/h(n^2m))$ leading to an overall gain of $1/h^2$ when $n < m$. In the same way, the memory needed to store the similarity matrices between words will decrease by a $1/h$ factor (not $1/h^2$ since we have now h similarity matrices to compute).

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 4.4., 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* [1] 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.

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.

TABLE 1: Description of the 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 [2] describing the pages of four universities (Cornell, Texas, Washington and Wisconsin), classified in five classes (student, project, staff, course, fa-

1. http://www.imdb.com/interfaces/
2. http://www.cs.umd.edu/projects/linqs/projects/lbc/

culty). 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 **χ-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 *k*-means 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. 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. 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, MVSIM 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 4.4., 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). The results of our experiment are provided in Table 4, 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 MVSIM, 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 4: 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 (Hussain et al., 2010) 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.

Références

Acar, E. and Yener, B. Unsupervised multiway data analysis : A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21 :6–20, 2009.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. On the surprising behavior of distance metrics in high dimensional space. In *LNCS*, pp. 420–434, 2001.

Azimi, J. and Fern, X. Adaptive cluster ensemble selection. In *IJCAI'2009*, pp. 992–997, 2009.

Banerjee, A., Basu, S., and Merugu, S. Multi-way clustering on relation graphs. In *SDM*. SIAM, 2007.

Bekkerman, R., El-Yaniv, R., and McCallum, A. Multi-way distributional clustering via pairwise interactions. In *ICML'05*, pp. 41–48, 2005.

Bickel, S. and Scheffer, T. Multi-view clustering. In *ICDM'2004*, 2004.

Bickel, S. and Scheffer, T. Estimation of mixture models using co-em. *ECML'2005*, pp. 35–46, 2005.

Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 92–100. ACM, 1998.

Chaudhuri, K., Kakade, S.M., Livescu, K., and Sridharan, K. Multi-view clustering via canonical correlation analysis. In *ICML'2009*, pp. 129–136, 2009.

de Carvalho, F. de A.T., Lechevallier, Y., and de Melo, F. M. Partitioning hard clustering algorithms based on multiple dissimilarity matrices. *Pattern Recognition*, 45 :447 – 464, 2012.

de Sa, V. R. Spectral clustering with two views. In *ICML workshop on learning with multiple views*, 2005.

Deerwester, Scott, Dumais, Susan T., Furnas, George W., Thomas, and Harshman, Richard. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41 :391–407, 1990.

Dhillon, I. S. Co-clustering documents and words using bipartite spectral graph partitioning. In *SIGKDD'2001*, pp. 269–274, 2001.

Dhillon, I. S., Mallela, S., and Modha, D. S. Information-theoretic co-clustering. In *SIGKDD*, pp. 89–98, 2003.

Drost, I., Bickel, S., and Scheffer, T. Discovering communities in linked data by multi-view clustering. *From Data and Information Analysis to Knowledge Engineering*, pp. 342–349, 2006.

Fred, A.L.N. and Jain, A.K. Data clustering using evidence accumulation. In *Pattern Recognition 2002*, volume 4, pp. 276–280, 2002.

Frigui, H., Hwang, C., and Rhee, F. Chung-Hoon. Clustering and aggregation of relational data with applications to image database categorization. *Pattern Recognition*, 40(11) : 3053 – 3068, 2007.

Hussain, F., Grimal, C., and Bisson, G. An improved co-similarity measure for document clustering. In *ICMLA'2010*, 2010.

Kumar, A. and Daume III, H. A co-training approach for multi-view spectral clustering. In *ICML*, pp. 393–400, 2011.

Li, T. and Ding, C. Weighted consensus clustering. *Mij*, 1 :2, 2008.

Liu, N., Zhang, B., Yan, J., Yang, Q., Yan, S., Z.Chen, Bai, F., and Ma, W. Learning similarity measures in non-orthogonal space. In *CIKM'2004*, pp. 334–341, 2004.

Long, B., Zhang, Z., and Yu, P. S. Co-clustering by block value decomposition. In *SIGKDD*, pp. 635–640, 2005.

Long, Bo, Zhang, Z., Wú, X., and Yu, P. S. Spectral clustering for multi-type relational data. In *ICML'06*, pp. 585–592, 2006.

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

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29(3) :93–106, 2008.

Strehl, A. and Ghosh, J. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *JMLR*, 3 :583–617, 2003.

Tang, W., Lu, Z., and Dhillon, I. S. Clustering with multiple graphs. In *ICDM'09*, pp. 1016–1021, 2009.

Yen, L., Fouss, F., Decaestecker, C., Francq, P., and Saerens, M. Graph nodes clustering with the sigmoid commute-time kernel : A comparative study. *Data Knowl. Eng.*, 68(3) : 338–361, 2009.

Zhou, D. and Burges, C.J.C. Spectral clustering and transductive learning with multiple views. In *ICML'2007*, pp. 1159–1166. ACM, 2007.