A Hash-based Co-Clustering Algorithm for Categorical Data

Fabrício Olivetti de França[a]

[a]*Center of Mathematics, Computing and Cognition (CMCC), Universidade Federal do ABC (UFABC) – Santo André, SP, Brazil. E-mail: folivetti@ufabc.edu.br*

Abstract

Cluster analysis, or clustering, refers to the analysis of the structural organization of a data set. This analysis is performed by grouping together objects of the data that are more similar among themselves than to objects of different groups. The sampled data may be described by numerical features or by a symbolic representation, known as categorical features. These features often require a transformation into numerical data in order to be properly handled by clustering algorithms. The transformation usually assigns a weight for each feature calculated by a measure of importance (i.e., frequency, mutual information). A problem with the weight assignment is that the values are calculated with respect to the whole set of objects and features. This may pose as a problem when a subset of the features have a higher degree of importance to a subset of objects but a lower degree with another subset. One way to deal with such problem is to measure the importance of each subset of features only with respect to a subset of objects. This is known as co-clustering that, similarly to clustering, is the task of finding a subset of objects and features that presents a higher similarity among themselves than to other subsets of objects and features. As one might notice, this task has a higher complexity than the traditional clustering and, if not properly dealt with, may present an scalability issue. In this paper we propose a novel co-clustering technique, called HBLCoClust, with the objective of extracting a set of co-clusters from a categorical data set, without the guarantees of an enumerative algorithm, but with the compromise of scalability. This is done by using a probabilistic clustering algorithm, named Locality Sensitive Hashing, together with the enumerative algorithm named InClose. The experimental results are competitive when applied to labeled categorical data sets and text corpora. Additionally, it is shown that the extracted co-clusters can be of practical use to expert systems such as Recommender Systems and Topic Extraction.

Keywords:
co-clustering, categorical data, data mining, text mining, biclustering

1. Introduction

The abundance of data being collected nowadays demands a set of tools to automatically extract useful information from them. If the data are partially labeled, a possible information to be extract is in the form of a mathematical model that can deduce the label from a set of measured variables, this characterizes the supervised learning. On the other hand, if the data are unlabeled, the information can be extracted by modeling a group structure of the objects that may describe the generating process of the data or may give a summarization of the information contained on it. This is referred as unsupervised learning and it is commonly studied by means of clustering algorithms.

Data clustering can refer to the task of dividing a data set into subset of objects that are more similar to each other than to the remaining elements of the set. There is a wide range of applications such as segmenting a surveyed population (Morgan and Sonquist, 1963) for market purposes, image quantization (Feng et al., 2007), frequent patterns of gene expressions (de França and Von Zuben, 2010) and many more.

In order to accomplish such task, the objects of the data set are described by a set of features measured during the data collection. Each feature can be represented by a numerical quantity, such as height of a person, amount of gas measured on a car tank, or descriptive characteristic or category, such as the gender of a person or the research topics they are interested.

The objects described by numerical features can be conveniently represented as numerical vector and the objects can be naturally compared to each other by using distance metrics. It makes sense to say that one person has twice the height of another.

On the other hand, categorical features lacks these properties and should be transformed into numerical features in order to be compared among themselves. For example, describing one person as male and another as female does not imply that one is *more* than the other.

Some similarity metrics were proposed to quantify the difference between objects of categorical features (Boriah et al., 2008), mostly based on the matching features between two objects. One

example is the Jaccard metric that, given the sets of features for two objects, it calculates the ratio between the cardinality of the intersection between the two sets and the cardinality of their union.

One problem that must be dealt with when using categorical data is the high dimensionality. Since most clustering algorithms requires a vectorial representation of the objects, the categorical features are usually represented as a binary vector, with every position representing whether the object has a given feature or not. For example, if one measured feature is whether a person is male or female, it would be represented by a 2-dimensional vector.

But, some categorical features may span into tens, hundreds or even thousands of vector dimensions. When describing a song by its genre, each object would be represented as a vector with more than $1,500$ dimensions. This high dimension exponentially increases the search space and it can cause a loss of precision on the similarity metrics. This is called the Curse of Dimensionality (Har-Peled et al., 2012).

This problem can be dealt with by reducing the dimensionality of the objects while preserving the similarity relationship between them. Probabilistic Dimension Reduction (Har-Peled et al., 2012) is the family of algorithms that exploits the probability that two similar objects will be considered to be equal when a subset of features is randomly sampled and used for comparison.

One of these algorithms, called *Minhashing* (Broder, 1997; Zamora et al., 2016), approximates the Jaccard Index between two objects. This algorithm relies on the fact that the probability of the first non-zero position of any random permutation of the feature vector is the same for two objects is equal to the Jaccard Index between them. The algorithm generates a smaller dense representation of the data with this information.

The reduction of dimensionality has two drawbacks when applied prior the clustering procedure: i) the compact representation may hide some seemingly unimportant features that could be used to describe a smaller group and, ii) the new set of features will lose its interpretability, since there is no clear relationship between the original set and the reduced set.

A more direct approach is to perform the data clustering in a two-way manner by finding the clusters of objects conditioned to a subset of features. This defines the family of algorithms known as co-clustering.

Data Co-Clustering (de França, 2012; Dhillon et al., 2003; Labiod and Nadif, 2011; Gao and

Akoglu, 2014), also known as biclustering, tries to find subsets of objects and features that maximizes the similarity of the selected objects when considering the chosen features. It exploits the fact that a given object may belong to different categories when viewed by different aspects of its description. For example, a given news text may report a story about the economies of a football team. This document will have terms that are related to sports and other terms that relates to economy. So, this object can be assigned to the group of sports related documents and the group of economy related document, depending of the selected set of words.

This technique allows for many relaxations of the constraints imposed by traditional clustering [1]. For example, each object may belong to more than one group, given a different subset of features. Additionally, one feature may be used to define different groups, associated with different subset of features.

Moreover, since each group is explicitly defined by a subset of features, the reason for grouping together a subset of objects can be easily explained, thus improving the interpretability of the model.

In many situations these relaxations can benefit the cluster analysis. When a cluster analysis is performed, it is expected that the natural grouping of the data is related to the intended labeling of the objective of the study. But, this expectation may not hold true. For example, when trying to classify a set of animals, the clustering algorithm may correctly identify that lions, deers and horses belong to the same class, but may incorrectly classify sea lions, octopus and tuna as belonging to the same class if their common traits are prevalent. The co-clustering of this data set would still find these groups but, additionally, would find other groups relating sea lions with mammals, tuna with other fishes and octopus with invertebrates.

Another possibility regards the topics extraction of a textual data set. In this task it is sought to infer the set of words that describes the topic of each document, based on the analysis of the whole data set. The usual techniques applied to such task requires a prior knowledge of how many different topics there are in the corpus and then tries to find the features responsible for the generative process of each cluster of documents. Again, the restriction of a fixed generative

[1] Note: these relaxations are not present in every co-clustering algorithm.

process for each document (i.e., the generative process of the only cluster it belongs to) may fail to acknowledge a second or third topic inside the text. With the co-clustering algorithm, the document can be grouped with different sets of documents regarding different topics and, as this procedure also highlights the features used to create each cluster, it makes the topic of each cluster explicit.

Finally, in Recommender Systems a clustering algorithm would group together users with similar tastes. But then again, only the prevalent common taste of each set of users will be taken into account. If, for example, a given user rates positively many comedy movies and just a few action movies, they would be likely grouped together with other users that share a taste for comedy. On the other hand, the co-clustering algorithm could also assign them to a group of users that like action movie. Besides, the taste of each user could be described by the combined set of features of every group they belong to.

But, this flexibility comes with a price, the number of groups that can be found is usually large, given all the possible combinations of subset of objects and features. Some of these groups may be irrelevant to the subject of analysis, rendering a burden to the post-analysis procedure.

Some co-clustering algorithms coped with this problem by reintroducing some of the constraints of the classical clustering, such as the search for a pre-specified number of clusters and assigning each object to only one group (Dhillon et al., 2003; Labiod and Nadif, 2011). These algorithms retain only the explicit description of which features were used as part of the clustering process.

Despite theses difficulties, there are some co-clustering algorithms capable of dealing with such flexibility, the most recent being the *HBLCoClust* (de França, 2012) and *CoClusLSH* (Gao and Akoglu, 2014). They both have in common the use of a probabilistic dimension reduction technique, named Locality Sensitive Hashing (LSH), used to find promising regions with high probability of containing co-clusters.

In the original *HBLCoClust* algorithm, after the search for the promising regions, a graph partitioning technique, called *METIS* (Karypis and Kumar, 2012), was used in order to generate

meaningful results, but constraining the algorithm to a pre-defined number of clusters [2].

This paper proposes a reformulation of *HBLCoClust* algorithm that does not depend of external algorithms while maintaining the scalability and automatically determining the number of groups to return. The general idea is to pre-process the set of features, eliminating those possessed by the minority of the objects, then the LSH algorithm is applied to the pre-processed data set in order to identify promising regions to be explored. These regions are then explored by an enumerative algorithm, called *InClose*, returning every co-cluster contained inside the given regions. Finally, the co-clusters sharing a percentage of their elements are merged in order to reduce the number of groups to be analyzed.

The experimental results will evaluate the groups generated by *HBLCoClust* through the consistency of the grouped objects according to their labels and the information conveyed by the subset of features of each group when considering each subset of objects, and scalability with respect to the number of elements in the data set.

Additionally, some practical implications of the interpretability of the groups will be illustrated in order to show the usefulness of Data Co-Clustering to different applications.

In Section 2 the co-clustering definition will be described in more details as well as some of its practical applications. Section 3 explains the proposed algorithm with all of its key aspects. Next, in Section 4, a complete set of experiments will be performed in order to assess how well the proposed algorithm performs on real-world scenarios against other similar co-clustering algorithms. Finally, Section 5 will give final comments on this work along with some future perspectives.

2. Co-Clustering

Co-Clustering refers to the task of finding subsets of rows and columns from a data matrix such as the values of the extracted submatrix presents a desired relationship (de França, 2012; Dhillon et al., 2003; Labiod and Nadif, 2011; Gao and Akoglu, 2014; Hartigan, 1972; Cheng and Church, 2000; Mirkin, 1996; de França and Von Zuben, 2010). Usually, the rows and columns are

[2] It is also worth mentioning that the newest versions of *METIS* did not compile correctly in some systems, thus making the HBLCoClust inaccessible to many users

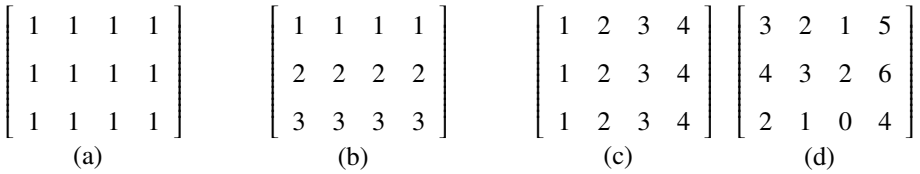


Figure 1: Example of different quality measures: (a) constant bicluster, (b) constant rows, (c) constant columns, (d) coherent values (rows or columns should exhibit high correlation).

described as objects and features, respectively, from a Data Mining perspective. The relationship sought by this algorithm will depend of the nature of the data set. Some possible relationships are the submatrices with constant values, with constant values along the rows or along the columns, correlated values, values expressing any consistent ordering or dense submatrices from sparse data (see Fig. 1).

This technique was already applied to a diverse set of applications such as gene expression analysis (de França and Von Zuben, 2010; de França et al., 2008; Mitra and Banka, 2006; Coelho et al., 2009), text mining (Dhillon et al., 2003; de Castro et al., 2010), recommendation systems (Symeonidis et al., 2008; de Castro et al., 2007) and data imputation (de França et al., 2013). The most popular application of the co-clustering algorithms is the search for additive coherence (de França and Von Zuben, 2011) from a large gene expression data.

Recently, there has been an increase of interest on the extraction of information of categorical data sets, such as text document data. These data sets are composed by a set of documents described by the tokenized terms. A co-cluster of this data set would represent a subset of documents that exclusively share a subset of these terms. Hopefully, this subset of terms describes the topic of the set of documents.

Formally, given the set O of objects and the set F of observed features, a Co-Cluster C of a subset $O' \subset O$, a subset $F' \subset F$ and a relation $R \subset O \times F$ can be described as:

$$C(O', F') = \{O' \times F' \mid o \in O', f \in F' \wedge (o, f) \in R\}, \tag{1}$$

where O' and F' can also be denoted as the objects and features clusters, respectively.

The relation R contains all the tuples (o, f) inside the data set. Notice, though, that the con-

straint that every pair of object and feature of the co-cluster must exist in R may lead to a large set of very small groups.

So, in order to minimize the number of groups while maximizing the size of such groups, the previous equation can be reformulated as:

$$C(O', F') = \{O' \times F' \mid |R'| \geq \rho \cdot |O' \times F'|\}, \tag{2}$$

where $|.|$ is the cardinality of a set and

$$R' = \{(o, f) \in R \mid o \in O', f \in F'\}. \tag{3}$$

Notice that there is no restriction whether an object or a feature must belong to only one group. There are other possible variations of this problem such as the $k, l-$Co-Clustering, in which it seeks to partition the data into k objects clusters and l features clusters by maximizing the number of non-null elements from the submatrices induced by each combination of k and l. Specifically, in this work, the focus will be on the formulation given in Eq. 2.

3. Hash-based Linear Co-Clustering

This section will describe some fundamental definitions and algorithms in order to better understand the proposed algorithm. First, the basics of probabilistic data clustering through random hash functions will be introduced. Following, the enumerative algorithm, called *InClose*, will be explained and, finally, the proposed algorithm will be detailed.

3.1. Locality Sensitive Hashing

A popular algorithm when dealing with high dimensional and high volume sets of data is a probabilistic algorithm called Locality Sensitive Hashing (LSH). This algorithm exploits the fact that two very similar samples will likely collide when mapped by a weak hash function. In fact, depending on how this hash function is created, the probability of this collision is known to be proportional to their similarity.

One of such hash functions is the Minwise Independent Permutation (Minhash) (Carter and Wegman, 1979; Har-Peled et al., 2012) that states that the probability of collision of two hashed

objects is proportional to their Jaccard similarity. Jaccard Index or Jaccard Similarity is used when the data set is described through sets of categorical features. This similarity can be calculated as:

$$J = \frac{|o_1 \cap o_2|}{|o_1 \cup o_2|}, \tag{4}$$

where o_1 and o_2 are the two objects being compared. The Jaccard Index varies from 0 to 1, with the later meaning that the two objects are equal.

Given a predefined order for the set of features, the Minhash algorithm generates a random permutation π of this set and take the first feature describing each object as its representative feature. The probability that two objects have the same representative feature is given by:

$$P(o_1^\pi = o_2^\pi) = \frac{|o_1 \cap o_2|}{|o_1 \cup o_2|}, \tag{5}$$

that is equal to the Jaccard Index.

So, the expected value of the Jaccard Index is estimated by averaging the number of collisions of two objects over k independent random permutations. Since generating random permutations can be computationally expensive, the permutation is approximated by an universal hash function such as the one proposed in (Carter and Wegman, 1979):

$$h(x) = a \cdot x + b \mod P, \tag{6}$$

where a and b are randomly chosen numbers from an uniform distribution, and P is a large prime number. The variable x is the value to be hashed, i.e., a number associated with the original index of the ordered set of features. Notice that, this prime number should be at least as large as the number of features. This hash function will map each original index to an index in the range $[0, P[$.

The Minhash of an object j for permutation i is simply the feature index x that minimizes the hash function $h_i(x)$:

$$mh_i(O_j) = \arg\min_x \left\{ h_i(x) \mid \forall x \in O_j \right\}. \tag{7}$$

Given a data set of n objects, each object containing an average of $\bar{m}$ observed features, out of m possible features, and k hash functions. The complexity of this algorithm is $O(n \cdot \bar{m} \cdot k)$, with $\bar{m} << m$ on sparse data sets.

This idea was extended in (Har-Peled et al., 2012) as an scalable algorithm for the Nearest-Neighbor problem named Locality Sensitive Hashing (LSH). This algorithm creates p hash signatures for each object by grouping together a sequence of k Minhashes. Each signature represents a *bucket* and the probability that two objects o_1 and o_2 will collide into the same bucket is given by:

$$P_{collision}(o_1, o_2) = 1 - (1 - J^k)^p) \tag{8}$$

where J is their Jaccard Index, k is the number of minhashes used to build one hash signature and p is the number of hash signatures.

The values of k and p can be adjusted to maximize the probability of grouping together the objects with a desired similarity of J.

It is interesting to notice that each bucket groups together a subset of objects and is described as a subset of k common features. So, the LSH can be used to find a variable number of co-clusters with k features. We will show in the next sections that these co-clusters can be used to limit the search space prior to the application of an enumerative algorithm.

3.2. Enumerative Algorithm for Co-Clustering

In mathematics, there is a very similar field of study called Formal Concept Analysis (Andrews, 2011) (FCA). In this field, the data set is described as a Formal Context composed of a triple $C = (O, F, R)$ where O stands for the set of objects, F is the set of features and $R \subseteq O \times F$ is the binary relationship of the objects with respect to the set of features. If a given object $o \in O$ contains the feature $f \in F$, then $(o, f) \in R$. In FCA, the objective is to find pairs (O', F'), called formal concept of context C such as:

- every object in O' has every feature in F',
- for every $o \in O, o \notin O'$, there is one $f' \in F'$ such that $(o, f') \notin R$,
- for every $f \in F, f \notin F'$, there is one $o' \in O'$ such that $(o', f) \notin R$.

In other words, this technique seeks the maximal subset of objects O' and the subset of features F' that satisfies Eq. 1.

In (Andrews, 2009) the enumerative algorithm *InClose* was proposed and further improved in (Andrews, 2011). This algorithm guarantees the exact set of formal concepts (i.e., co-clusters) without generating the same solution more than once (i.e., avoiding redundant search).

For the sake of brevity and clarity, the *InClose* algorithm will be explained following the notations used in the previous sections (i.e., formal concepts will be called co-clusters from now on) and with a small adaptation that will be explained later in this section.

The general idea of the algorithm is to recursively enumerate the Co-Clusters by following the lexicographical order of the feature set F, expanding the current co-cluster until it becomes maximal and branching the algorithm to explore new candidate solutions.

Initially, the algorithm starts with the initial co-cluster (O', F') where $O' = O$, the entire set of objects and $F' = \{f\}$, a subset containing the first feature of F when following the lexicographical order.

By following the lexicographical order of the features in F, the algorithm sequentially creates a new set $F'' = F' \cup f'$ for every feature $f' \in F$ that is positioned after f. For each F'' it generates a new set O'' that satisfies Eq. 1.

After this step, the subset F' is merged with every generated set F'' whenever the corresponding $O'' = O'$, in other words, F' will contain every feature that maintains the integrity of the set O', maximizing the co-cluster.

The remaining pairs (O'', F'') will be passed to a recursive call of the algorithm together with $f' = F'' \cap F'$ as long as (O'', F'') is cannonical with respect to f'.

A Co-Cluster (O', F') is considered cannonical with respect to a feature f if and only if there is no feature $f' < f$ that can be inserted into F' such as Eq. 1 is still satisfied. This verification is required to avoid redundancy.

The *InClose* algorithm is summarized in Alg. 1. Notice that two other parameters are included in this algorithm: *minObjs* and *minFeats*, that are used to discard co-clusters smaller than these thresholds.

Algorithm 1: InClose

input : (global) data set of objects, features and relations (O, F, R), minimum cardinality for the cluster of objects *minObjs* and the cluster of features *minFeats*.

input : (local) initial subset of objects O' and features F', and current feature f.

output: set of Co-Clusters C

`/* Insert new features in lexicographical order. */`

$$\text{Insertions} \leftarrow \Big\{(f', O'') \mid f' \in F \wedge f' < f \wedge |O''| \geq \text{minObjs} \text{ with } O'' = O' \cap \{o'' \in O \mid (o'', f') \in R\}\Big\};$$

`/* Generate the maximal co-cluster with the fixed set` O'`. */`

$$F' \leftarrow F' \cup \Big\{f' \mid (f', O'') \in \text{Insertions} \wedge O'' = O'\Big\};$$

`/* List of candidates to be recursively expanded. */`

$$\text{Candidates} \leftarrow \Big\{(O'', F' \cup \{f'\}, f') \mid (O'', f') \in \text{Insertions} \wedge \text{Cannonical}(O'', F')\Big\};$$

if $|F'| \geq$ *minFeats* **then**

`/* The operator ++ means concatenation. */`

return $\Big\{(O', F')\Big\} + +\Big\{\text{InClose}(O'', F'', f') \mid (O'', F'', f') \in \text{Candidates}\Big\}$;

else

return $\Big\{\text{InClose}(O'', F'', f') \mid (O'', F'', f') \in \text{Candidates}\Big\}$;

3.3. Hash-Based Linear Co-Clustering Algorithm

The Hash-based Linear Co-Clustering algorithm was initially proposed in (de França, 2012) as an scalable algorithm to find k Co-Clusters from a categorical data set. This algorithm can be divided in three simple steps:

1. Find a co-cluster set C composed of a set of tuples (O', F') by applying the LSH algorithm. The set O' is composed of the objects inside a bucket and the set F' is the set of k features used as a hash signature.
2. Maximize the co-clusters in C by inserting features in F' while satisfying Eq. 1 and then inserting objects in O' obeying the same constraint.
3. Induce a graph from the set C where the vertices are the objects and the edges connect two

objects that belongs to the same co-cluster. After that, perform the community detection algorithm *METIS* (Karypis and Kumar, 2012) to find a set of k co-clusters.

The third step is needed because the second step generates a high number of small co-clusters when dealing with a sparse data set. But, this steps introduced theoretical and technical drawbacks.

For the theoretical drawbacks, the complexity of the first two steps is $O(n)$ with n representing the cardinality of the set of relations R. The third step introduces a complexity proportional to $O(m^3)$ Kernighan and Lin (1970), with m representing the number of vertices of the induced graph or, in this case, the cardinality of the set of objects O. Also, this algorithms introduce a number of different parameters that should be adjusted, together with the inherent parameters of *HBLCoClust*. Besides, it constrains the co-clusters to a $k-$co-clustering algorithm, with a pre-specified k.

About the technical drawbacks, there were some issues for the compilation process under some Operational Systems that prevented the use of the *HBLCoClust* and the reproduction of the original experiments was compromised.

In order to eliminate the dependency of an external implementation and the constraint of defining the number of clusters, we now propose a new version of *HBLCoClust* by following these six steps:

1. **Remove rare features:** remove any feature $f \in F$ such as $\left|\{o \mid (o, f) \in R\}\right| \leq \tau \cdot \left|O\right|$, for an specified $0 \leq \tau \leq 1$ [optional].
2. **Find the promising regions with LSH:** apply the LSH algorithm to find a set of $2 \cdot p$ candidates tuples (O', F') with k features by *bucketing* the objects with hash signatures from the features set and, also, *bucketing* a set of features with hash signatures from the objects set.
3. **Create subsets of the original data set:** expand each candidate (O', F') creating a set of promising regions (O'', F'') such as $O'' = \{o \in O \mid (o, f) \in R \wedge f \in F'\}$ and $F'' = \{f \in F \mid (o, f) \in R \wedge o \in O'\}$.
4. **Enumerate inside the promising regions:** apply the *InClose* algorithm to every promising region thus creating the co-clusters set $C = \{(O''', F''')\}$. The duplicated regions can be avoided by storing the already explored regions on a hash table.

5. **Expand the co-clusters by allowing sparseness:** sequentially insert objects and features into every co-cluster such as $\left|\{(o, f) \notin R \mid o \in O''' \wedge f \in F'''\}\right| \leq \tau_{sparse} \cdot \left|O'''\right| \cdot \left|F'''\right|$, for an specified $0 \leq \tau_{sparse} \leq 1$ [optional].
6. **Merge related co-clusters:** merge any two co-clusters that shares the same subset of features.

In the first step, the algorithm optionally removes any feature that appears in less than a percentage of the objects O. This step avoids that, during the next stage, the *LSH* algorithm samples one of these rare features to compose the hash signature. This may create a bucket containing a single object, thus generating an uninteresting region.

The second step is similar to the original algorithm. It first generates $p{\cdot}k$ random hash functions and then creates p buckets for every object by grouping k keys generated by the hash functions (see Section 3.1). Additionally, it creates another set of $p \cdot k$ random hash functions and then creates p buckets for every feature by grouping the keys generated from the hash functions applied to the objects set. This step generates a candidate set of regions to be explored.

For every generated region, a new subset of the original data set is created with the subset of objects that contains a relationship with every feature of the specified region and, similarly, a subset of features that are related to every object of the region. This creates a sparse subset that contains at least one co-cluster (as defined by (O', F')).

These regions are then used by the enumerative algorithm *InClose* to find the complete set of co-clusters. Since this algorithm only enumerates co-clusters that satisfies Eq. 1, the next step optionally insert objects and features that do not violate Eq. 2. The order of insertion is defined by the number of missing values each element introduces to the co-cluster.

Finally, since every co-cluster was generated from a constrained region of the original data set and further expanded by the introduction of sparseness, some co-clusters may share the same set of features but with a different set of objects. For this purpose, those co-clusters sharing the same set of features are merged together.

The *HBLCoClust* algorithm requires a total of six parameters, some of them optional: the minimum relative frequency of features (τ, optional), the number of buckets (p) and number of hash

functions per bucket (k), the minimum cardinality for the objects and features subsets (*minObjs* and *minFeats*), required by *InClose*, and the maximum allowed percentage of missing values (τ_{sparse}, optional). The sensitivity of these parameters will be shown in the next Section.

The pseudo-algorithm for *HBLCoClust* is depicted in Alg. 2 and followed by some auxiliary routines in Algs. 4 and 3.

One important thing to notice regarding the proposed algorithm is that the first three steps have a linear complexity with respect to the cardinality of the relations set R, the fourth step is exponential proportional to the number of clusters contained inside the *Regions* created in step 3. The last two steps are quadratic with respect to the average co-cluster cardinality. We will show in the next Section empirical evidence that the whole algorithm scales linearly with respect to the cardinality of R for the tested data sets.

3.4. Literature review

In the literature, the most similar algorithm to *HBLCoClust* is the one proposed in (Gao and Akoglu, 2014) and called *CoClusLSH*. This algorithm iteratively applies the LSH algorithm using the set of objects and features alternately, and merging the groups defined by the buckets using an entropic metric. The authors tested *CoClusLSH* on a diverse set of real world data, presenting competitive numerical results measured by purity, mutual information and number of groups found (the closest to the number of labels, the better). The algorithm is linearly scalable regarding the number of objects and features on the data set but, it does require that the whole data set resides in memory during the processing stage.

Another co-clustering algorithm that recently presented good results is the *SpecCo* (Labiod and Nadif, 2011) which reformulates the co-clustering problem as a graph partitioning problem. It then optimizes the modularity criteria (Newman, 2006) in order to find a set of k, l-clusters. The obtained results were quantitatively better than some of the considered state-of-the-art co-clustering algorithms. But, unlike *HBLCoClust* and *CoClusLSH*, this algorithm requires a pre-specified number of clusters and it does not scale linearly.

In the next section, we will compare the results obtained by *HBLCoClust* with those obtained by *CoClusLSH* and *SpecCo*.

input : data set of objects, features and relations (O, F, R), minimum cardinality for the cluster of objects *minObjs* and the cluster of features *minFeats*, threshold of minimum frequency of features τ, the number of buckets p and hash functions per buckets k and the maximum ratio of missing data τ_{sparse}.

output: set of Co-Clusters C

`/* Step 1: Removal of rare features */`

$F \leftarrow \left\{ f \mid f \in F \wedge \text{count}(f) \geq \tau \cdot |O| \right\}$;

`/* Step 2: LSH */`

$B \leftarrow \text{RandomBucketFunctions}(p, k)$;

for $o \in O$ **do**

  for $b \in B$ **do**

    `/* Bucket is an associative array where the key is a set of features and the value is a set of objects */`

    $\text{Bucket}[b(o)] \leftarrow \text{Bucket}[b(o)] \cup \{o\}$;

`/* Step 3: Candidate Regions */`

$\text{Regions} \leftarrow \left\{ \text{Region}(O', F') \mid (O', F') \in \text{Bucket} \right\}$;

`/* Step 4: Enumeration */`

$C \leftarrow \left\{ \text{InClose}(O'', F'', f) \mid (O'', F'') \in \mathit{Regions} \wedge f = \text{head}(F'') \right\}$;

`/* Step 5: Expansion */`

$C \leftarrow \left\{ \text{Expand}(O''', F''') \mid (O''', F''') \in C \right\}$;

`/* Step 6: Merge */`

for $c_1, c_2 \in C \times C$ **do**

  if $F'''_1 = F'''_2$ **then**

    $c_1 \leftarrow c_1 \cup c_2$;

    $\text{Remove}(c_2)$;

return C

Algorithm 2: function Region

input : subset of objects O' and subset of features F'.

output: subsets O'', F'' defining a search space.

$O'' \leftarrow \left\{o \mid o \in O \text{ if } (o, f) \in R \text{ for } \forall f \in F'\right\}$;

$F'' \leftarrow \left\{f \mid f \in F \text{ if } (o, f) \in R \text{ for } \forall o \in O'\right\}$;

return (O'', F'')

Algorithm 3: function Expand

input : subset of objects O' and subset of features F'.

output: subsets O'', F'' respecting maximum missing values rate τ_{sparse}.

$O^* \leftarrow$ sort O by number of sparse features ;

$F^* \leftarrow$ sort F by number of sparse objects ;

$O'' \leftarrow \{o \mid o \in O^* \text{ if constraint is respected }\}$;

$F'' \leftarrow \{f \mid f \in F^* \text{ if constraint is respected }\}$;

return (O'', F'')

4. Experimental Results

In the Section we will present some experiments with the *HBLCoClust* algorithm on three different types of data sets: categorical, textual and rating data. The categorical data sets, as previously discussed, are sets comprised of objects described by one or more features from a set F. The textual data are acquired from text documents extracted from newsgroups, in these data sets, each document represents one object and every word is a feature from the set.

Finally, the rating data set was specifically created for this paper by combining a data set of ratings given by users on a set of movies and a set of descriptive features for the set of movies. The objects for this set are the users and the features are composed of the name and keyword of the movies concatenated with words describing whether the users liked or disliked the movie.

For the categorical and textual data sets, the obtained results will be quantitatively compared to the algorithms *InClose*, *CoClusLSH* and *SpecCo*, with some exceptions. The *InClose* algorithm results will be available only for the data sets which the total processing time did not take more

than 5 hours. For the *CoClusLSH*, we could only obtain the results for the categorical data sets, due to memory limitations. Finally, due to the unavailability of the *SpecCo* source code, we will use the reported values in (Labiod and Nadif, 2011).

The *HBLCoClust* and *InClose* algorithms were implemented in Python 2.7 and it is readily available at `https://github.com/folivetti/HBLCoClust`. The experiments with the algorithm *CoClusLSH* were performed using the Matlab implementation available at `http://www.cs.sunysb.edu/~leman/pubs.html`. These algorithms were run under a Linux Debian 7.6 system on a i5-2450 @ 2.5 GHz machine with 6GB of RAM.

4.1. Data sets and Experiments

In order to have a concise discussion of the experiments, they will be divided into five different subsections: Categorical Data Clustering, Text Clustering, Topic Modeling, Recommender Systems, Scalability and Sensitivity analysis.

The Categorical Data Clustering subsection will have the goal of finding groups of objects that share common features. For this purpose we have selected four well known categorical data sets named: Zoo, House Votes 84', Sybean Small and Soybean Large Bache and Lichman (2013). All four data sets are described by categorical features and a label that classifying each object. The features may be many-valued or binary, as described in (Labiod and Nadif, 2011), the many-valued features are converted to binary. The labels will be used to measure the quality of the co-clusters found by the algorithms. Notice that, even though it is expected a set of clusters grouping together objects with the same label, this does not imply the inexistence of groups with objects of different labels.

The Text Clustering subsection will comprehend a similar experiment as described for the categorical data clustering. The only practical difference being the high dimensionality. For this set of experiments we chose the data sets Classic3 [3], containing documents from 3 different collections, and two subsets of the 20-newsgroups (Lang, 1995) data set, named Multi5 and Multi10.

The Multi5 data set contains documents from the groups: comp.graphics, rec.sport.baseball, rec.motorcycle, sci.space, and talk.politics.mideast. The Multi10 data set contains documents

[3] ftp://ftp.cs.cornell.edu/pub/smart

extracted from: alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.electronics, sci.crypt, sci.med, sci.space, talk.politics.guns.

Regarding the Topic Modeling subsection, we will just illustrate one possible practical application of co-clustering algorithms. As such, we will show some evidence that the subset of features found by *HBLCoClust* algorithm for each group can be used to describe the topic concerning the corresponding subset of documents. For this purpose we will use the results from the Text Clustering subsection.

Similarly, the Collaborative Filtering subsection will illustrate how a co-clustering algorithm can be used to find some explicit information regarding subset of users sharing the same taste. For this experiment we will merge the information of two well known data sets, named Movielens (Herlocker et al., 1999) and IMDB `ftp://ftp.fu-berlin.de/pub/misc/movies/database/` in order to generate a categorical data set of movies ratings.

The Movielens Data Set contains a set of $80,000$ tuples (user, movie, rating) with the users represented by their id, the movies represented by their titles and the ratings as a numbered scale between 1 to 5.

This data was complemented by using the keywords, genre, actors and actress, and directors describing each movie provided by the IMDB data set. These keywords were used to form the relations (user, feature, rating), where rating is the average rating given by the user to movies possessing this feature.

This data set was then converted to a set of tuples (user, feature_Y) or (user, feature_N), whether the rating was higher than 3 or not, respectively.

The *HBLCoClust* algorithm was applied separately to the *Y* and *N* relations in order to generate co-clusters of *likes* and *dislikes* for each subset of users.

Every user was then described by means of the features exclusively contained in the clusters of the *likes* data set they belonged to and the features in the *dislikes* data set as well.

To assess the recommendation error, a test set of $20,000$ tuples (user, movie, rating) was converted to a categorical data in the same way as the training data. For every (user, movie) in the test set, the Jaccard Index of the *likes* and *dislikes* profiles of this user is calculated against the movie features set. The profile with higher similarity is used to decide whether to recommend the movie

Table 1: Data set properties: number of objects, number of features, the number of relations (non-zero elements) and number of classes.

	|O|	**|F|**	**|R|**	**classes**
Zoo	101	16	738	7
Soybean Small	47	21	880	4
Soybean Large	307	35	4865	19
House Vote 84	435	16	6568	2
Classic 3	3891	15034	227355	3
Multi 5	5000	44323	539933	5
Multi 10	10000	64444	987443	10
Movielens	943	4233	154628	–

or not.

Besides exemplifying with the generated profiles, the results will be compared to two commonly used machine learning algorithms: regularized SVD (Funk, 2006) and Naive Bayes (Lewis, 1998).

In Table 1 the properties of each data set used on these experiments are summarized. The adopted parameters for the *HBLCoClust* are depicted in Table 2 for each data set. These values were found by performing a grid search in order to maximize the coverage of the objects in the data set, with the combination of the following values for each parameter: $minObjs = [2, 50]$, $minFeats = [2, 50]$, $p = \{1000, 2000, 3000, 4000, 5000\}$, $k = \{2, 3, 4, 5\}$, $\tau = [0, 1]$ with intervals of 0.05, $\tau_{sparse} = [0, 1]$ with intervals of 0.1.

For the *InClose* parameters, we have adopted the same values of $minObjs$ and $minFeats$ as chosen for *HBLCoClust*. The parameters of the *CoClusLSH* algorithm was fixed to $p = 100$ and $k = 3$ after a grid search with the combination of the values of $p = 100, 200, 300$ and $k = 2, 3, 4$. This limited set of values was due to memory limitations imposed by the algorithm.

Finally, the parameters for the *SpecCo* algorithm were the same as reported in (Labiod and Nadif, 2011) due to the unavailability of the source code.

Table 2: Parameters used for *HBLCoClust* on each data set.

data set	minObjs	minFeats	p	k	τ	τ_{sparse}
Zoo	4	6	1000	2	0.0	1.0
Soybean Small	4	8	1000	2	0.1	0.8
Soybean Large	4	10	1000	2	0.0	0.8
House Vote 84	10	10	1000	3	0.4	0.8
Classic 3	50	4	2000	3	0.2	0.5
Multi 5	5	5	1000	3	0.95	0.5
Multi 10	5	5	2000	3	0.95	0.5
Movielens	2	2	5000	4	0.0	0.8

4.2. Metrics

In order to quantify the quality of the Co-Clusters obtained by each algorithm, we have chosen three different metrics: Purity, Normalized Mutual Information and Pointwise Mutual Information.

Purity of a Co-Cluster measures the ratio between the number of the most frequent label inside the cluster by the number of objects in the cluster. It essentially quantifies, for a given cluster, if the majority of its objects agree with respect to their labels.

Normalized Mutual Information calculates how likely it is to find an object of a given label if a given co-cluster is selected at random. This metric is related to Purity, but it also verifies the compactness of the Co-Clusters set, i.e. if the set has a minimum number of Co-Clusters and is given by:

$$NMI = \frac{1}{H_C H_l} \sum_{c \in C, o \in O'} P(c, l(o)) \times \log \frac{P(c, l(o))}{P(c)P(l(o))}, \tag{9}$$

where H_C, H_l is the entropy of the Co-Clusters set and the labels, respectively, O' is the subset of objects of the co-cluster, and $l(o)$ is the label of object o.

Finally, the Pointwise Mutual Information measures the likelihood that the co-occurrence of any two features of a Co-Cluster was not by chance. This verifies if the subset of features selected by the Co-Cluster have significance regarding the corresponding objects:

$$PMI = -\sum_{Of \in C} \sum_{f1, f2 \in Of} \frac{\log P(f1, f2) - \log P(f1)P(f2)}{\log P(f1, f2)}. \quad (10)$$

4.3. Categorical Data Clustering

The results for the first set of experiments, categorical data, are depicted in Table 3 and 4. These Tables show the average results over 30 runs of *HBLCoClust* and *CoClusLSH*, a single run of *InClose* and the reported results for *SpecCo*. The first table reports the number of Co-Clusters found by each algorithm, the percentage of the covered objects and features and the average size of the Co-Clusters. The second table reports the average value of Purity, NMI and PMI.

From Table 3 we can notice some interesting properties of each algorithm. First of all, *InClose* algorithm obtained the largest set of co-clusters with the largest size, something that should be expected since it is an enumerative algorithm. Also, since this algorithm only finds dense co-clusters, it was not always capable of covering the entire set of objects. Comparing the *HBLCoClust* and *CoClusLSH* algorithms, the later found a smaller number of groups while covering the entire set of objects, except for the Zoo data set. This means that *CoClusLSH* was more competent when maximizing the compactness. Since the number of groups is a parameter of *SpecCo*, it always returned the exact number of expected clusters.

In Table 4 we can see that, except for the Zoo Data Set, the *HBLCoClust* obtained the best, or close to the best, results regarding Purity and PMI. Regarding the NMI metric, the *SpecCo* has again the advantage of generating a compact set of groups, thus maximizing this metric. In the Zoo Data Set, though the proposed algorithm has not obtained the best results, it was close to the best values. It is worth highlighting that, a consistent high PMI value means that the algorithm is capable of selecting the significant features.

4.4. Text Clustering

For the second set of experiments, regarding the textual data, neither *InClose* algorithm nor *CoClusLSH* could be used due to limitation in computational or memory complexity. Also, the comparison will be limited to the reported values in (Labiod and Nadif, 2011) for the *SpecCo* algorithm. It should be noticed that, the data sets used in (Labiod and Nadif, 2011) differs from

Table 3: Statistics of obtained Co-Clusters set for the Categorical data sets.

Zoo	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	27.07	1.00	0.80	83.07
CoClusLSH	37.00	1.00	1.00	52.00
InClose	67.00	0.81	0.75	114.00
SpecCo	7.00	1.00	−−	−−
Soybean S	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	13.80	1.00	57.78	83.13
CoClusLSH	10.00	1.00	1.00	225.00
InClose	225.00	1.00	70.83	121.00
SpecCo	4.00	1.00	−−	−−
Soybean L	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	42.40	1.00	0.74	205.73
CoClusLSH	20.00	1.00	1.00	1089.00
InClose	6470.00	0.98	0.93	109.00
SpecCo	19.00	1.00	−−	−−
House Votes	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	23.30	1.00	0.85	1173.80
CoClusLSH	18.00	1.00	1.00	734.00
InClose	124371	0.95	1.00	225.00
SpecCo	2.00	1.00	−−	−−

the ones used in our experiment. The Classic3 data set used only a setof 150 documents and $3,625$ features, and the Multi5 and Multi10 data sets used a selection of 500 documents and $2,000$ words. These selections were performed by means of a supervised algorithm in order to use only the most significant word-features regarding the document labels.

Notice that, for our experiment with *HBLCoClust* we have used the entire data. So, the focus of this experiment will be on assessing if our proposed algorithm is capable of automatically selecting

the correct set of features without the use of a supervised algorithm.

Table 4: Obtained results for the categorical data sets.

zoo	**Purity**	**NMI**	**PMI**
HBLCoClust	0.88	0.29	0.18
CoClusLSH	0.79	0.25	0.29
InClose	0.93	0.19	0.21
SpecCo	0.90	0.92	−−
Soybean S	**Purity**	**NMI**	**PMI**
HBLCoClust	0.89	0.40	0.25
CoClusLSH	0.56	0.30	0.08
InClose	0.59	0.09	−0.26
SpecCo	1.00	1.00	−−
Soybean L	**Purity**	**NMI**	**PMI**
HBLCoClust	0.73	0.26	0.15
CoClusLSH	0.28	0.09	0.06
InClose	0.50	0.07	0.13
SpecCo	0.67	0.78	−−
House Votes	**Purity**	**NMI**	**PMI**
HBLCoClust	0.91	0.29	0.43
CoClusLSH	0.85	0.24	0.45
InClose	0.93	0.10	0.26
SpecCo	0.87	0.47	−−

In Table 5 we can see that, as explained by the aforementioned situation, the *HBLCoClust* algorithm returned a much larger set of clusters when compared to *SpecCo*. Also, even though dealing with a larger set of documents, it still managed to cover the entirety of the objects.

Interesting, in Table 6 we can see that *HBLCoClust* obtained a purity value equal or much higher than the results obtained by *SpecCo*. This means that, even though it resulted in a larger

Table 5: Statistics of obtained Co-Clusters set for the Textual data sets.

Classic3	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	219.20	1.00	0.60	2807.23
SpecCo	3.00	1.00	−−	−−
Multi5	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	1054.97	1.00	0.18	1051.23
SpecCo	5.00	1.00	−−	−−
Multi10	**#**	**Objs.**	**Feats.**	**Size**
HBLCoClust	2819.33	1.00	0.16	661.00
SpecCo	10.00	1.00	−−	−−

set of clusters, those clusters are accurate regarding the labels. The lower value of Purity obtained by *SpecCo* is a side-effect of the pre-determined number of clusters, since there are clusters of documents on these data sets pertaining to the same topic that do not share any feature. Concerning the PMI, the *HBLCoClust* algorithm still managed to obtain a high positive value, thus assessing the quality of the select subsets of features for each cluster.

4.5. Topic Modeling

The PMI metric is a very popular metric for assessing the quality of Topic Modeling algorithms. These algorithms try to search for a set of words of a corpus that correctly describes the topic of each document. A value of PMI around 0.5 is considered good for the newsgroup data sets, as we can see in (Anandkumar et al., 2013), while the most well-known algorithm for Topic modeling, Latent Dirichlet Allocation (Blei et al., 2001), obtains a value around 0.2.

To illustrate the quality of the obtained subsets of features, in Table 7 we report the terms belonging to the set of features of 5 different clusters, one from each topic of Multi5 data set, chosen at random. It is easy to see that most of these terms are meaningful words revolving around the corresponding topic.

Table 6: Obtained results for the Textual data sets.

Classic3	**Purity**	**NMI**	**PMI**
HBLCoClust	0.86	0.14	0.20
SpecCo	0.86	0.73	−−
Multi5	**Purity**	**NMI**	**PMI**
HBLCoClust	0.91	0.18	0.37
SpecCo	0.59	0.53	−−
Multi10	**Purity**	**NMI**	**PMI**
HBLCoClust	0.82	0.14	0.33
SpecCo	0.57	0.55	−−

4.6. Recommender System

Regarding the Movielens data set, the *HBLCoClust* algorithm obtained a total of $1,320$ co-clusters covering 60% of the users and only 63% of the movies. As such, from the total of $20,000$ test ratings, we could generate a recommendation (or a *not recommended* classification) to $12,862$ ratings. From this total, the co-clustering based recommendation obtained an accuracy of 83.4% against 79.68% and 69.32% obtained by SVD and Naive Bayes, respectively.

Additionally, the information given by the profiles of each user may enrich the post-analysis of the recommender system by giving directions of what can be more relevant to a given user. In Table 8, the profiles of a randomly selected user is depicted.

4.7. Scalability and Sensitivity of the algorithm

As a final experiment, we have measured the time taken by *HBLCoClust* for each one of the experimental data sets with a fixed set of parameters: $p = 1000$, $k = 3$, $minObjs = minFeats = 4$, $\tau = 0.0$, $\tau_{sparse} = 1.0$.

In Fig. 2 we can see the correlation between the time, in seconds, with the number of relations inside each data set, fitted with a regression line. This experiment corroborates with the claim that the algorithm scales linearly proportional to the cardinality of the set of relations.

Table 7: Terms extracted from features clusters of Multi5 data set.

sport.baseball	reds, houston, standings, cincinnati, colorado, mets, scores, marlins, including, milwaukee, oakland, pirates, city, expos, los, west, indians, minnesota, ocf, rangers, joseph, white, angels, texas, giants, toronto, pittsburgh, phillies, cardinals, cubs, atlanta, mariners, orioles, mlb, lost, braves, louis, detroit, teams, athletics, streak, hernandez, san, boston, cleveland, dodgers, sox, seattle, astros, blue, diego, jays, jtchern, rockies, twins, brewers, tigers, red, francisco, philadelphia, kansas, yesterday, royals, california, padres, berkeley, league, chicago, florida, angeles, april, montreal, yankees, baltimore, york
motorcycles	handlebars, motorcycle, speed, countersteering, foward, handle, faq, awful, turns, debating, ummm, uiuc, happens, turning, pushing, fgc, convert, unb, explain, duke, unbvm, acpub, slack, zkcl, infante, cbr, eric, cso, csd, methinks, push
politics.mideast	later, muslims, ohanus, vol, turkish, involved, roads, argic, hand, muslim, armenian, document, russian, armenians, including, army, sahak, proceeded, serdar, soul, killed, among, children, published, blood, appressian, mountain, often, exists, turks, armenia, general, soviet, serve, escape, genocide, melkonian, ways, extermination, passes, closed
sci.space	six, rigel, aurora, wings, mary, dfrf, alaska, dryden, spin, military, facility, speak, unknown, digex, prb, flight, pilot, fly, kotfr, air, nsmca, pat, edwards, mig, wire, fighter, shafer
comp.graphics	plot, recommend, pascal, hidden, routines, object, basic, cost, address, bob, cad, mac, info, frame, short, offer, animation, price, sites, across, package, low, building, directory, removal, documentation, robert, built, recommendations, libraries, various, tasks, shading, fast, files, code, objects, tools, handle, demo, library, contact, book

Table 8: Profile generated for one of the user on Movielens data set.

Movies	Jaws, Back to the Future, Twelve Monkeys, Dumb & Dumber, ...
like	disaster, infidelity, horse, gunfight, USA, automobile, hospital, bathroom, jealousy, racism, elevator, fight, beer, male-nudity, helicopter, impalement, good-versus-evil, outer-space, murder, washington-d.c., fire, shot-to-death, los-angeles-california, independent-film, small-town, train, drunkenness, one-man-army, baby, teenage-boy, lifting-someone-into-the-air, redemption, f-word, photograph, tough-guy, gangster, main-character-dies
dislike	second-part, beaten-to-death, haunted-by-the-past, Washington,DistrictofColumbia,USA, cell-phone, vengeance, bulletproof-vest, obsession, book, die-hard-scenario

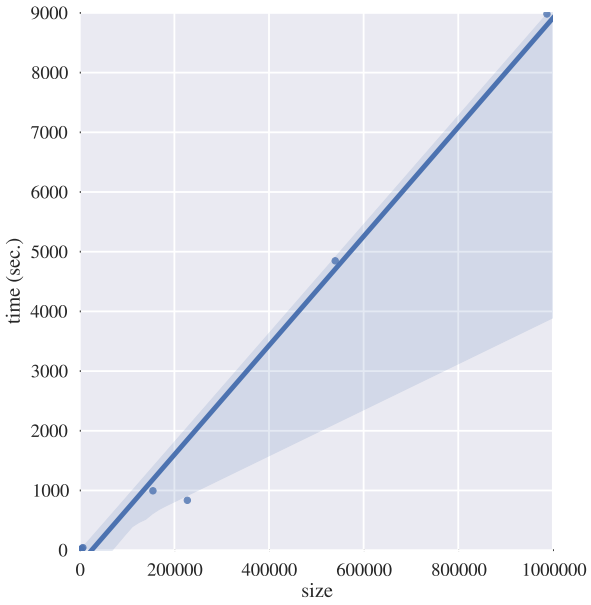
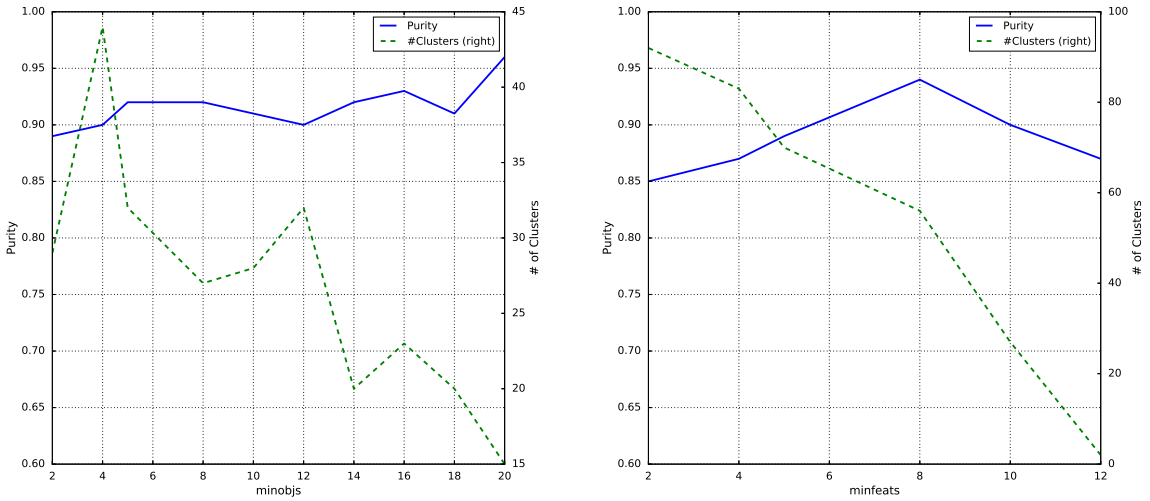


Figure 2: Time complexity estimation.



(a) (b)

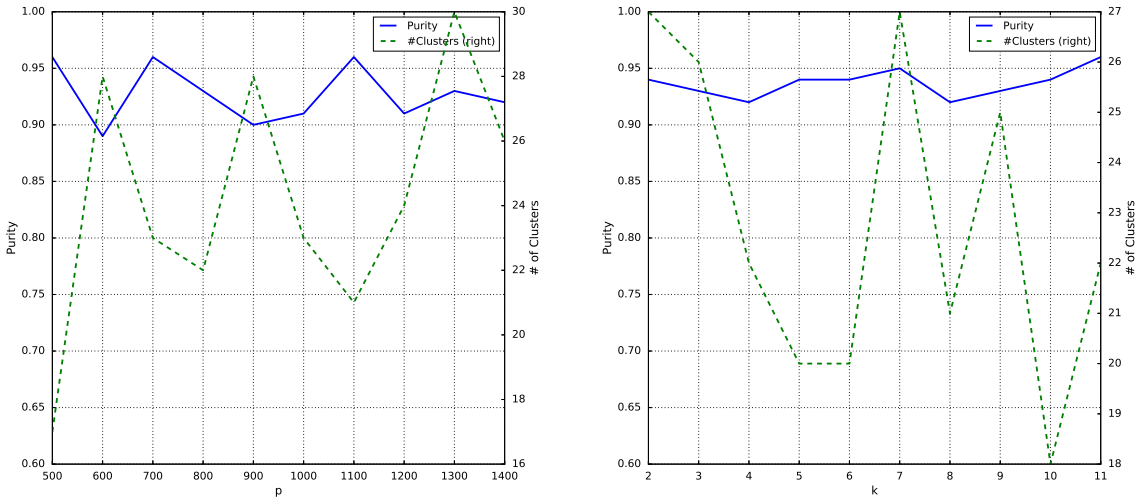
Figure 3: Parameter sensitivity for (a) *minObjs* and (b) *minFeats*.

Regarding the sensitivity of the parameters, we have devised a short experiment in order to assess how the values of each parameter affects the Purity and the number of clusters found by the algorithm.

For this purpose we have chosen the House Vote 84' data set, the largest of the categorical data sets used in this paper. In this experiment, we have tested 10 different values for every parameter around the values described in Sect. 4.1 while fixing the remaining parameters with the same values from this table.

The results will be reported through line plots where the left y-axis represents the Purity value associated to each value of the parameter, and the right y-axis is the number of clusters found by the algorithm. Notice that the y-axis of all the plots are fixed between $[0.6, 1.0]$ in order to highlight the significance of the variations, while the right axis varies according to the results, in order to verify the tendency of generating more or less clusters.

In Fig. 3 we can see that the parameters *minObjs* and *minFeats* do not seem to affect the Purity of the clusters found by the algorithm. On the other hand, the more restricted the number of objects and features, a smaller number of clusters are found. This means that the algorithm will maintain



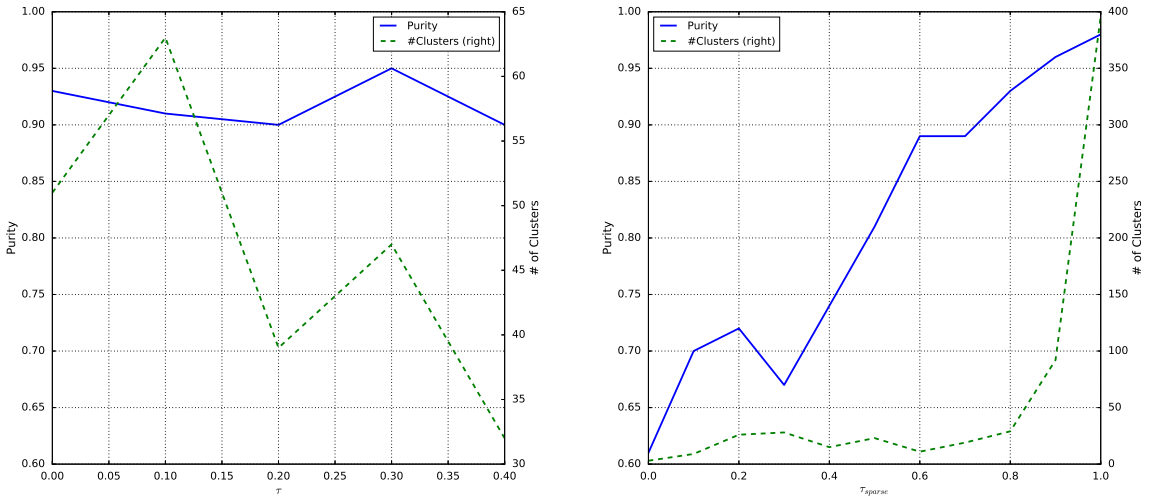
(a) (b)

Figure 4: Parameter sensitivity for (a) k and (b) p.

the quality of the clusters regardless of how many of them are found.

In Fig. 4, we can see the sensitivity analysis for the parameters pertaining to the LSH algorithm. We can still observe that the Purity is not much affected by these parameters as well. The number of clusters seems to be more affected by the number of buckets. More buckets mean that we will create more possibilities for finding different groups, thus leading to a higher number of clusters at the final stage.

Finally, in Fig. 5, we can see that the pre-processing stage, controlled by τ, affects only the number of clusters found, the higher the cutoff, less features will remain in the data set and less clusters can be found. The sparse parameter, on the other hand, does affect the Purity and the number of clusters. When this parameter is set to 1, it will only find dense clusters and, thus, the purity tends to be maximized, but it is less likely to find few larger clusters. When the parameter is set to 0, the whole data set can be assigned to the same cluster, thus minimizing the Purity and returning a single cluster.



(a) (b)

Figure 5: Parameter sensitivity for (a) τ and (b) τ_{sparse}

5. Conclusion

In this paper a new algorithm for the co-clustering of categorical data was presented. This algorithm consists of six sequential steps and scales linearly with the number of non-empty entries of the data set. This is achieved by using a probabilistic algorithm to approximate the partial similarity between objects, thus creating *seed* co-clusters. These seed co-clusters are used to define regions of the data set to be searched by an enumerative algorithm called *InClose*, finally merging the Co-Clusters found inside these different regions.

The experiments performed on categorical and textual data showed that this algorithm is at least *on par* with other co-clustering algorithms from the literature, and in many times significantly better. Most noticeably, the algorithm was capable of selecting subsets of features that maximized the point-wise mutual information, leading to a meaningful set of features describing each cluster.

Some examples were provided illustrating the practical implications of having an explicit subset of features. In one of these examples, the features of the text data clusters were used to describe the topic of a group of documents. In another example, the descriptive features were used to generate a profile of what a given user likes or not in a movie.

Despite the good results, the proposed algorithm has a tendency of dividing the data sets into a larger set of clusters, when compared to other algorithms, though this may imply a more natural segmentation of the data. Also, the number of parameters may require an additional effort to adjust, in some situations.

For future research, we intend to propose heuristic algorithms to automatically suggest the values of some of the required parameters by using entropic metrics and knowledge extracted from basic statistics of the data set. This will be performed together with a more detailed sensitivity analysis.

Additionally, we will explore in more details the idea of using co-clustering algorithms for Topic Modeling and Recommender Systems, as well as some other possible applications.

References

Anandkumar, A., Valluvan, R., et al., 2013. Learning loopy graphical models with latent variables: Efficient methods and guarantees. The Annals of Statistics 41, 401–435.

Andrews, S., 2009. In-close, a fast algorithm for computing formal concepts, in: International Conference on Conceptual Structures.

Andrews, S., 2011. In-close2, a high performance formal concept miner, in: Conceptual Structures for Discovering Knowledge. Springer, pp. 50–62.

Bache, K., Lichman, M., 2013. UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2001. Latent dirichlet allocation, in: Advances in neural information processing systems, pp. 601–608.

Boriah, S., Chandola, V., Kumar, V., 2008. Similarity measures for categorical data: A comparative evaluation. red 30, 3.

Broder, A.Z., 1997. On the resemblance and containment of documents, in: Compression and Complexity of Sequences 1997. Proceedings, IEEE. pp. 21–29.

Carter, J., Wegman, M.N., 1979. Universal classes of hash functions. Journal of Computer and System Sciences 18, 143 – 154.

de Castro, P.A.D., de França, F.O., Ferreira, H.M., Coelho, G.P., Von Zuben, F.J., 2010. Query expansion using an immune-inspired biclustering algorithm. Natural Computing , 1–24.

Cheng, Y., Church, G.M., 2000. Biclustering of expression data, in: Proc. of the 8th Int. Conf. on Intelligent Systems for Molecular Biology, pp. 93–103.

Coelho, G.P., de França, F.O., Von Zuben, F.J., 2009. Multi-objective biclustering: When non-dominated solutions are not enough. Journal of Mathematical Modelling and Algorithms .

de Castro, P.A.D., de França, F.O., Ferreira, H.M., Von Zuben, F.J., 2007. Applying Biclustering to Perform Collaborative Filtering, in: Proc. of the 7th International Conference on Intelligent Systems Design and Applications, Rio de Janeiro, Brazil. pp. 421–426.

Dhillon, I.S., Mallela, S., Modha, D.S., 2003. Information-theoretic co-clustering, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 89–98.

Feng, H.M., Chen, C.Y., Ye, F., 2007. Evolutionary fuzzy particle swarm optimization vector quantization learning scheme in image compression. Expert Systems with Applications 32, 213–222.

de França, F.O., Coelho, G.P., Von Zuben, F.J., 2008. bicaco: An ant colony inspired biclustering algorithm, in: Ant Colony Optimization and Swarm Intelligence. Springer, pp. 401–402.

de França, F.O., Coelho, G.P., Von Zuben, F.J., 2013. Predicting missing values with biclustering: A coherence-based approach. Pattern Recognition 46, 1255–1266.

de França, F.O., 2012. Scalable overlapping co-clustering of word-document data, in: Machine Learning and Applications (ICMLA), 2012 11th International Conference on, IEEE. pp. 464–467.

de França, F.O., Von Zuben, F.J., 2010. Finding a high coverage set of 5-biclusters with swarm intelligence, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE. pp. 1–8.

de França, F.O., Von Zuben, F.J., 2011. Extracting additive and multiplicative coherent biclusters with swarm intelligence, in: Evolutionary Computation (CEC), 2011 IEEE Congress on, IEEE. pp. 632–638.

Funk, S., 2006. Netflix update: Try this at home.

Gao, T., Akoglu, L., 2014. Fast information-theoretic agglomerative co-clustering, in: Wang, H., Sharaf, M. (Eds.), Databases Theory and Applications. Springer International Publishing. volume 8506 of *Lecture Notes in Computer Science*, pp. 147–159.

Har-Peled, S., Indyk, P., Motwani, R., 2012. Approximate nearest neighbor: Towards removing the curse of dimensionality. Theory OF Computing 8, 321–350.

Hartigan, J.A., 1972. Direct clustering of a data matrix. Journal of the American Statistical Association (JASA) 67, 123–129.

Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J., 1999. An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, ACM. pp. 230–237.

Karypis, G., Kumar, V., 2012. Metis-serial graph partitioning and fill-reducing matrix ordering.

Kernighan, B.W., Lin, S., 1970. An efficient heuristic procedure for partitioning graphs. Bell system technical journal 49, 291–307.

Labiod, L., Nadif, M., 2011. Co-clustering for binary and categorical data with maximum modularity., in: ICDM, pp.

1140–1145.

Lang, K., 1995. Newsweeder: Learning to filter netnews, in: Proceedings of the Twelfth International Conference on Machine Learning, pp. 331–339.

Lewis, D.D., 1998. Naive (bayes) at forty: The independence assumption in information retrieval, in: Machine learning: ECML-98. Springer, pp. 4–15.

Mirkin, B., 1996. Mathematical Classification and Clustering. Nonconvex Optimization and Its Applications, Springer.

Mitra, S., Banka, H., 2006. Multi-objective evolutionary biclustering of gene expression data. Pattern Recognition 39, 2464–2477.

Morgan, J.N., Sonquist, J.A., 1963. Problems in the analysis of survey data, and a proposal. Journal of the American statistical association 58, 415–434.

Newman, M.E., 2006. Modularity and community structure in networks. Proceedings of the national academy of sciences 103, 8577–8582.

Symeonidis, P., Nanopoulos, A., Manolopoulos, Y., 2008. Providing justifications in recommender systems. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 38, 1–1272.

Zamora, J., Mendoza, M., Allende, H., 2016. Hashing-based clustering in high dimensional data. Expert Systems with Applications 62, 202–211.