A Memetic Heuristic for the Co-clustering Problem

Mohammad Khoshneshin[1], Mahtab Ghazizadeh[2], W. Nick Street[1], and Jeffrey W. Ohlmann[1]

[1] The University of Iowa, Iowa City IA 52242, USA
{mohammad-khoshneshin,nick-street,jeffrey-ohlmann}@uiowa.edu
[2] University of Wisconsin-Madison, Madison WI 53706, USA
ghazizadeh@wisc.edu

Abstract. Co-clustering partitions two different kinds of objects simultaneously. Bregman co-clustering is a well-studied fast iterative algorithm to perform co-clustering. However, this method is very prone to local optima. We propose a memetic algorithm to solve the co-clustering problem. Experimental results show that this method outperforms the multi-start Bregman co-clustering in both accuracy and time.

Key words: co-clustering, Bregman co-clustering, memetic algorithm.

1 Introduction

Clustering refers to partitioning similar objects into groups [7]. Co-clustering partitions two different kinds of objects simultaneously and is a data mining approach with applications in many areas such as recommender systems [8, 6] and microarray analysis [11, 5, 3, 4]. If one views the clustering problem as grouping rows of a matrix together, then co-clustering is the simultaneous grouping of rows and columns. The intersection of a row cluster and a column cluster is called a block. A popular goal in co-clustering is minimizing the sum of block variances:

$$\sum_{i,j} w_{ij}(\bar{x}_{\rho(i)\gamma(j)} - x_{ij})^2 \tag{1}$$

where x_{ij} is the value of the element in i^{th} row, j^{th} column of the matrix, $\rho(i)$ denotes the cluster of row i, $\gamma(j)$ denotes the cluster of column j, $\bar{x}_{\rho(i)\gamma(j)}$ is the average value of the intersection block between row cluster $\rho(i)$ and column cluster $\gamma(j)$, and w_{ij} is the weight of the value of the element in i^{th} row, j^{th} column of the matrix.

As will be discussed, there is a limited literature regarding the optimization problem mentioned above. The most popular co-clustering algorithm is a fast iterative algorithm known as Bregman co-clustering [2] which is very prone to local optima. Banerjee et al. [2] show that if the objective function of a co-clustering problem can be expressed as a Bregman divergence [1] such as squared

Euclidean distance or KL-divergence, then the Bregman co-clustering algorithm can be used.

Bregman co-clustering moves from a co-clustering solution to a better solution by alternately fixing row (column) clusters and optimizing column (row) clusters. When updating row clusters, we assume all co-cluster means are constant and all columns are assigned to column clusters, then we assign each row so that the sum of squared errors is minimized. Column cluster updating follows the same approach. The updating step iterates until convergence; when given column (row) clusters, no improvement by row (column) clusters is possible. Banerjee et al. prove their algorithm will converge. However, the result may be very poor since Bregman co-clustering algorithm is a greedy algorithm which finds local optima.

To search the solution space more efficiently and effectively, we apply the memetic algorithm [10] as a meta-heuristic algorithm. The memetic algorithm outperformed multi-start Bregman co-clustering algorithm in the experiments.

There is limited literature regarding applying heuristic algorithms to the co-clustering problem. Examples include memetic algorithms [11], genetic algorithms [5], and evolutionary algorithms [3, 4, 8]. These works focus primarily on co-clustering gene expression data. The goal of the co-clustering problem in the above papers was minimizing entropy in diagonal blocks, by forcing the same number of row clusters and column clusters. Specifically, if the number of row and column clusters is N, only N blocks are considered and not N^2. In our work, the number of row and column clusters can be different, and all possible blocks are included in the objective function. This is a more general case of the problem described in the literature, to which none of the above approaches are applicable. In the most relevant work [8], an evolutionary algorithm was used to solve the general co-clustering problem. However, this work focuses on optimizing a group of co-clustering solutions while our algorithm searches for individual solutions.

The literature on applying heuristic search to the clustering problem in not trivially applicable to the co-clustering problem. Most related works encode a solution via centroid of clusters (for example see [9]) assuming each object is represented by a vector. In the co-clustering problem, two different types of object are related via one scalar and the goal is minimizing the divergence of blocks from objects. Therefore, the centroid- encoding approach is not applicable to the co-clustering problem.

2 Memetic algorithm

A memetic algorithm is a population-based hybrid of a genetic algorithm and local search [10]. Memetic algorithm is inspired by cultural evolution while genetic is by biological evolution. In this work, we use the Bregman co-clustering algorithm as a local search method in conjunction with genetic algorithm operations (crossover and mutation). In our context, let P co-clustering solutions exist. The goal is to find better solutions by combining the current solutions. Every genetic algorithm has three main steps: 1) selection, in which two or more

individuals are chosen to create offspring; 2) crossover, in which the selected items are combined to create new solutions; and 3) replacement, in which the new solutions replace the existing solutions if they satisfy some criteria.

```
Input {data matrix, number of row and column clusters K & L, Population size P}

Randomly initialize P co-clustering solutions {(ρ_p, γ_p)}_{p=1}^P
Bregman co-clustering {(ρ_p, γ_p)}_{p=1}^P
Repeat
     Randomly choose 2 unforbidden couples{(ρ_u^parent, γ_u^parent)}_{u=1,2}
     Add {(ρ_u^parent, γ_u^parent)}_{u=1,2} to forbidden couples
     ρ_1^child, ρ_2^child = crossover(ρ_1^parent, ρ_2^parent)
     γ_1^child, γ_2^child = crossover(γ_1^parent, γ_2^parent)
     Mutate {(ρ_u^child, γ_v^child)}_{u=1,2,v=1,2}
     Bregman co-clustering {(ρ_u^child, γ_v^child)}_{u=1,2,v=1,2}
     Update population ({(ρ_p, γ_p)}_{p=1}^P, {(ρ_u^child, γ_v^child)}_{u=1,2,v=1,2})
Until no unforbidden couple or maximum iteration
```

Fig. 1. Memetic heuristic outline

Figure 1 illustrates our Memetic co-clustering algorithm. The data matrix, number of row and column clusters, and population size are the inputs to the algorithm. Note that the number of clusters or population does not change during the algorithm execution. A group of co-clustering solutions is randomly generated and locally optimized via iterative Bregman co-clustering. In the evolutionary iteration phase, two co-clustering solutions are randomly selected for crossover. To speed up the convergence of the algorithm, we use the concept of *forbidden couples*. This means that a couple used for performing crossover in a given iteration will not be allowed for crossover in the following iterations. Without the forbidden couple strategy, the same two parents may be chosen and create very similar offspring, which is both inefficient and damaging to the population diversity. The algorithm terminates when no unforbidden couple is left for crossover or after reaching the preset maximum number of iterations.

After selecting two co-clusterings, a new solution is generated via the crossover function (see Figure 2). We explain crossover for two different clustering solutions which can be row or column clusterings. Let N objects be clustered in K clusters. There are $K!$ different ways to represent the same clustering (because of the indifference to the order of clusters). For effective clustering crossover, corresponding clusters must be found. We propose a way to find correspondence between two clusterings. Let C be an $N \times K$ matrix and let the $(i,k)^{th}$ element of C be 1 if object i is in cluster k and 0 otherwise. $O = C_1^T C_2$, the intersection matrix, shows the number of objects that are in two clusters of different clusterings where C_1 is the C matrix of the first clustering and C_2 is the C matrix of the second clustering. In other words, O shows the amount of overlap between two clusters of different clusterings. Correspondence can be computed relative to either parent clustering. Based on the first clustering, the first cluster of the first clustering will be associated with a cluster from the second clustering with which it has maximum overlap. Then, the second cluster of the first clustering

```
Input: parent clusters φ1 and φ2 ,cluster size K
Output: offspring clusters ψ1 and ψ2

∀x, r, c : ζrc ← x|φ1(x) = r & φ2(x) = c
Urow = {}
Ucol = {}
FOR k = 1 to K − 1
    c* = argmax |ζkc|
         c∉Ucol
    ∀x ∈ ζkc* : ψ1(x) ← k
    Ucol ← c*
    r* = argmax |ζrk|
         r∉Urow
    ∀x ∈ ζr*k : ψ2(x) ← k
    Urow ← r*
ENDFOR
∀x|ψ1(x) = 0 : ψ1(x) ← K
∀x|ψ2(x) = 0 : ψ2(x) ← K
```

Fig. 2. Crossover algorithm. ζ_{rc} is the intersection set between cluster r in clustering ϕ_1 and cluster c in clustering ϕ_2. U_{row} and U_{col} are respectively the used clusters of the first and the second clustering.

will be associated with a remaining cluster of the second clustering with maximum overlap and this continues until the last cluster. A similar approach can be used based on the second clustering. Note that this is a greedy algorithm to find the overlap between two clusters and will not necessarily result in an optimal overlap solution.

After finding all correspondences, objects that are located in a pair of correspondent clusters are assigned to a cluster. The remaining objects are assigned to a leftover cluster. After assigning all remaining objects to the leftover cluster, it may have more objects than the rest of the clusters. However, the rest of the clusters must have good quality since two different clusterings agree on them. Therefore, the Bregman co-clustering algorithm, which starts with computing block averages, is expected to find a good solution because the majority of blocks are of high quality.

As a result of crossover, there are two offspring for each row clustering and two for each column clustering. Two offspring for the row (column) clustering are the result of computing correspondence relative to the row (column) parent clusterings. Therefore, four child co-clustering solutions are possible. Note that the number of clusters in the offspring is exactly equal to the number of clusters in the parent. So the number of clusters does not change.

After crossover, mutation is performed on all offspring to encourage diversity. To perform mutation, a number of objects are chosen based on some probability, and then each is assigned to a random cluster. In all experiments, the algorithm selects objects with probability $1/K$ (where K is the number of clusters) and then it assigns them to equally probable clusters.

In the next step, the offspring from crossover will be locally optimized via the fast iterative Bregman co-clustering algorithm [2]. This is an important step since most objects might have been assigned to the last cluster in the crossover step. However, since the iterative co-clustering first estimates averages and then

assigns rows and columns, we hope that most of the blocks will have accurate averages via fewer but carefully selected rows and columns in crossover.

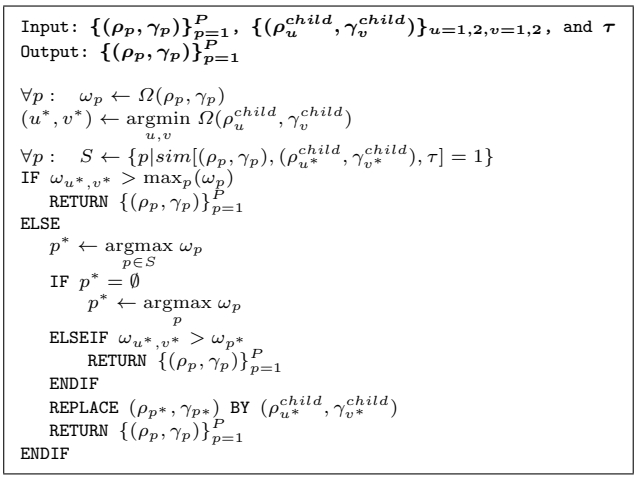

Fig. 3. Update algorithm. Function $\Omega()$ is the objective function. $sim[(\rho_p, \gamma_p), (\rho_{u^*}^{child}, \gamma_{v^*}^{child}), \tau]$ is the similarity function (τ is the threshold). S is the set of similar clusterings to the best offspring clustering.

Finally, we should either discard a current solution or the offspring to preserve the total number of solutions. We conduct replacement to maintain diversity as well as improve the solution quality of the population. In favor of diversification, only the best offspring is considered for replacement since all offspring are expected to be very similar. The best offspring (with respect to the objective function) replaces a member of the current population that is deemed similar to the offspring and has a worse quality than the best offspring and the worst quality among similar individuals. If there is no similar individual to the best offspring, then the best offspring replaces the worst individual in the population. Otherwise, no replacement occurs. The algorithm is illustrated in Figure 3.

Determining similarity between two clusterings is similar to finding correspondence between them (function $sim()$ in Figure 3). We propose a method to measure similarity based on the intersection matrix O. Let J be a matrix with the same size as O. Also, let $(u, v)^{th}$ element of J be 1 if $(u, v)^{th}$ of O is greater than a threshold τ, and 0 otherwise. If J is an assignment matrix, then two clusterings are similar ($sim() = 1$). In an assignment matrix, each row is assigned to only one column (shown by 1), and each column is assigned to only one row. Mathematically, the formula $(\prod_u \sum_v j_{uv}) \cdot (\prod_v \sum_u j_{uv})$ (where j_{uv} is the $(u, v)^{th}$ element of the matrix J) gives the value 1 if the two clusterings are similar and a value other than one, otherwise.

3 Experiments

The proposed memetic algorithm for co-clustering is compared to the multi-start Bregman co-clustering algorithm with new random initializations. We used root mean squared error (RMSE) as the objective function. Note that minimizing RMSE is equivalent to minimizing the proposed objective function in (1).

3.1 Dataset

Both artificial and real data were used for evaluating the proposed algorithm. For the artificial dataset, we generated a 1000×1000 data matrix. The number of row and column clusters was set to 20 in advance. Then each row and column was assigned to a row and column cluster randomly. The average of each block was chosen from 1 to 10, with equal probability. Then, each element of the data matrix was set to its block average. That is, all data in one block have the same value. Therefore, for this dataset, there is a co-clustering solution with RMSE=0. We name this dataset AF (full artificial dataset). To simulate realistic problems, we introduced noise to AF. First, each element of AF was chosen with probability of 0.1. Then one of the values {-2,-1,1,2} (selected with equal probability) was added to the chosen data. Values above 10 were set to 10 and those below 1 were set to 1. This data is called AFN (full artificial dataset with noise). Using the original generative co-clustering, RMSE is 0.4591 for AFN. Finally, to establish similarity with real data, we added missing values to the data. To this end, each element of the data matrix was chosen with probability of 0.1 and added to the sparse data matrix from AF and AFN. This results in datasets AS (artificial sparse data) and ASN (artificial sparse data with noise). Note that the density (number of known elements over all possible elements) of these datasets is around 0.1. Using the original generative co-clusterings, RMSE is 0 for AS and 0.4592 for ASN.

To examine the algorithm on a real-life problem, a subset of the Netflix dataset[1] was used. In this dataset, rows are users, columns are movies, and values are viewer ratings on a 1 to 5 scale. Our subset included 12,326 users, 9,730 movies, and 1,638,799 ratings. The number of row clusters was set to 30 and the number of column clusters was set to 20. We refer to this dataset as NS (Netflix dataset subset).

3.2 Results

We implemented the multi-start (MS) of the Bregman co-clustering algorithm with random initialization and the proposed memetic algorithm (MA) on the datasets mentioned above. For the MS, we set the number of iterations to 4000. For the MA, the maximum number of iterations was set to 1000. Therefore, these two methods are equivalent in terms of the opportunity to generate individual co-clusterings since MA generates 4 co-clustering solutions in each iteration.

[1] http://www.netflixprize.com

However, in many cases MA terminates before 1000 iterations while MS cannot terminate before reaching to 4000 iterations. The threshold to measure similarity between the two clusterings was set to 30. This value was set arbitrarily and no fine-tuning was done.

Table 1 summarizes the results. For artificial data without noise, MA was able to find the best known solution. In the case of artificial data with noise (both full and sparse), the solution is better than the result obtained from the original generative co-clustering solution; however, it is not guaranteed to be optimal. MS could find the best solution (only for the artificial sparse data (AS)). The performance of MS is very poor on the full data matrices. The reason for poor performance of the pure Bregman co-clustering on full matrices is not obvious; however, it might be due to the greediness of this algorithm. For the Netflix subset (NS), the memetic algorithm outperforms MS as well. Note that increasing the population in MA adds to the power of the algorithm; however, it delays convergence. MA outperforms MS in running time as well. This is because crossover is a quick way to finding good solutions and thus, a few iterations might be enough to reach promising results. However, random starting does not benefit from such a feature.

Data	Method	Population	RMSE	Time (sec)
AS	MS	4000	0	1455
	MA	10	0	199
ASN	MS	4000	0.6208	1454
	MA	10	0.4570	190
AF	MS	4000	1.5808	4884
	MA	10	0	504
AFN	MS	4000	1.6755	4885
	MA	10	0.4581	504
NS	MS	4000	0.8893	114231
	MA	10	0.8825	5472
	MA	30	0.8818	30069

Table 1. Result of experiments on different datasets for MS (multi-start) and MA (memetic algorithm). Population shows the population size used in memetic search and is simply the total number of runs in multi-start.

4 Conclusion

In this paper, we proposed a memetic algorithm for solving the co-clustering problem with the goal of minimizing Euclidean distance between data and block averages. The best known algorithm for this problem is the Bregman co-clustering algorithm which is very fast but prone to local optima and can lead to very poor results. To address this shortcoming, we proposed a memetic algorithm which employs the Bregman co-clustering algorithm as a local search.

Experimental results indicate that the proposed memetic algorithm outperforms the multi-start Bregman co-clustering algorithm in both accuracy and time.

There are several direction to improve the proposed memetic algorithm. Two drawbacks can be mentioned for the similarity measure. The similarity measure only determines whether two clusterings are similar. However, similarity is a continuous concept. Incorporating a continuous metric as a similarity measure in the algorithm may help diversify the population and therefore improve the result. Another drawback is related to the size of clusters. If two clusterings are exactly the same but some of the clusters are very small, our similarity measure cannot detect it. In the update function of the memetic algorithm, we only consider the best offspring. However, it is possible to measure the similarity among offspring and discard the worse ones if they are similar. In another direction, instead of giving the number of clusters to the memetic algorithm, it can search for proper numbers in an evolutionary manner.

References

1. K. Azoury and M. Warmuth. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.
2. A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *The Journal of Machine Learning Research*, 8:1919–1986, 2007.
3. H. Banka and S. Mitra. Evolutionary biclustering of gene expressions. *Ubiquity*, 7(42):1–12, 2006.
4. S. Bleuler, A. Prelic, and E. Zitzler. An ea framework for biclustering of gene expression data. In *Proceedings of the IEEE 2004 Congress on Evolutionary Computation (CEC 2004)*, volume 1, pages 166–173, 2004.
5. A. Chakraborty and H. Maka. Biclustering of gene expression data using genetic algorithm. In *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8, 2005.
6. T. George and S. Merugu. A scalable collaborative filtering framework based on co-clustering. In *IEEE Intl. Conf. on Data Mining*, pages 625–628, 2005.
7. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
8. M. Khoshneshin and W. N. Street. Incremental collaborative filtering via evolutionary co-clustering. In *Proc. ACM Conference on Recommender Systems*. ACM, 2010.
9. U. Maulik and S. Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 2000.
10. P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. *International Series in Operations Research and Management Science*, pages 105–144, 2003.
11. N. Speer, C. Spieth, and A. Zell. A memetic co-clustering algorithm for gene expression profiles and biological annotation. In *Proceedings of the IEEE 2004 Congress on Evolutionary Computation, CEC 2004*, volume 2, pages 1631–1638, 2004.