A Robust Biclustering Method Based on Crossing Minimization in Bipartite Graphs

Cesim Erten and Melih Sözdinler

Computer Science and Engineering, Işık University

Clustering refers to the process of organizing a set of input vectors into clusters based on similarity defined according to some preset distance measure. In many cases it is more desirable to simultaneously cluster the dimensions as well as the vectors themselves. This special instance of clustering, referred to as *biclustering*, was introduced by Hartigan [3]. It has many applications in areas including data mining, pattern recognition, and computational biology. Considerable attention has been devoted to it from the gene expression data analysis; see [5] for a nice survey. Input is represented in a data matrix, where the rows and columns of the matrix correspond to genes and conditions respectively. Each entry in the matrix reflects the expression level of a gene under a certain condition. From a graph-teoretical perspective the data matrix can be viewed as a weighted bipartite graph, where the vertex set of one partition is the set of genes and the vertex set of the other partition is the set of conditions. An existing weighted edge incident on a *gene-condition* pair reflects the expression level of the gene under that specific experimental condition. The biclustering problem may then be described in terms of the various versions of the biclique extraction problem in bipartite graphs. Many interesting versions that directly apply to the biclustering problem are NP-hard [4]. Various graph-theoretical approaches employing heuristics have been suggested [1,4,6,7].

One drawback of these approaches is the assumption that the corresponding bipartite graph is unweighted. Of these approaches the one following a direct graph-drawing approach is [1]. A crossing minimization procedure is applied on the unweighted bipartite graph resulting from preprocessing the original input data matrix. Our approach is similar in essence. However we do not have a discretization/normalization step to convert the weighted bipartite graph into an unweighted one as this would cause some data loss and produce erroneous output. Instead we apply crossing minimization directly on the original weighted graph. Various efficient crossing minimization heuristics have been shown to work well on weighted bipartite graphs and a 3-approximation algorithm has been suggested [2]. Our algorithm consists mainly of three steps. *Initial placement* phase applies a two-sided crossing minimization on the weighted graph until there is no change on the node orders. To do this we employ algorithm `3-WOLF` of [2] (one-sided crossing minimization procedure) repeatedly, each time alternating the fixed layer. We have verified that if the input data is noise-free then this initial placement is usually enough to identify bicliques and extract the biclusters. *Adaptive Noise Hiding* phase removes the weighted edges in the graph that correspond to noise in the original input data. Sliding a window around the primeter of each node pair, where $(i \pm 1, j), (i, j \pm 1), (i \pm 1, j \pm 1)$ constitutes the perimeter of a pair (i, j), we check whether the window satisfies a threshold density in terms of the number of nonzero

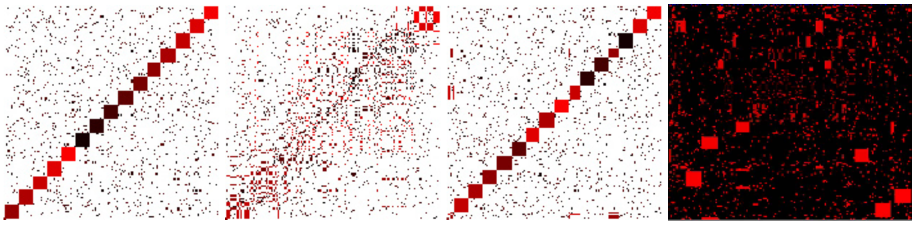

Fig. 1. Assumed noise is 0.05. (a) Initial artificial design with 15 biclusters of $K_{12,12}$; (b) Without noise removal; (c) Our complete algorithm; (d) Matlab Bioinformatics Box

weight edges. If it does not, the pairs on the perimeter are considered *suspicious*. Once sliding is finished we find the *most suspicious* weight and remove all the suspicious pairs with that weight. We adaptively apply our two-sided crossing minimization procedure on the new graph and continue noise hiding after incrementing the threshold density. The removal of the *suspicious* edges and the crossings couple each other in terms of noise removal. Each time the partitions of the graph are reordered to reduce crossings, new suspicious pairs are created. Once the noise removal phase is over, we finally gather the biclusters by applying a procedure similar to the one described in [1]. Details of the algorithm are left for the final paper. We note that different from previous approaches we directly apply weighted crossing minimization on the original input data, not to lose possibly important data that can not be considered noise. Secondly our application of the crossing minimization is two-folds. Besides providing a good initial placement, crossing minimization is also used to handle noise removal. Our preliminary experiments provided better results than the *clustergram* function of the Matlab Bioinformatics Box. Figure 1 provides a sample visualization from our initial tests.

References

1. Abdullah, A., Hussain, A.: A new biclustering technique based on crossing minimization. Neurocomputing 69(16-18), 1882–1896 (2006)
2. Çakiroglu, O., Erten, C., Karatas, Ö., Sözdinler, M.: Crossing minimization in weighted bipartite graphs. In: Demetrescu, C. (ed.) WEA 2007. LNCS, vol. 4525, pp. 122–135. Springer, Heidelberg (2007)
3. Hartigan, J.A.: Direct clustering of a data matrix. Journal of the American Statistical Association 67(337), 123–129 (1972)
4. Lonardi, S., Szpankowski, W., Yang, Q.: Finding biclusters by random projections. Theor. Comput. Sci. 368(3), 217–230 (2006)
5. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: A survey. IEEE/ACM Trans. on Comp. Biol. and Bioinf.(TCBB) 1(1), 24–45 (2004)
6. Mishra, N., Ron, D., Swaminathan, R.: On finding large conjunctive clusters. In: COLT, pp. 448–462 (2003)
7. Tanay, A., Sharan, R., Shamir, R.: Discovering statistically significant biclusters in gene expression data. Bioinformatics 18(supagesl. 1), 136–144 (2002)