CLICK: A Clustering Algorithm for Gene Expression Analysis

Ron Shamir **Roded Sharan**
shamir@math.tau.ac.il roded@math.tau.ac.il
Department of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel

Keywords: clustering, gene expression, graph algorithms, minimum cut, likelihood

1 Introduction

Novel DNA microarray technologies enable the monitoring of expression levels of thousands of genes simultaneously. This allows for the first time a global view on the transcription levels of many (or all) genes when the cell undergoes specific conditions or processes. The potential of such technologies for functional genomics is tremendous: Measuring gene expression levels in different developmental stages, different body tissues, different clinical conditions and different organisms is instrumental in understanding genes function, gene networks, biological processes and effects of medical treatments.

A first key step in the analysis of gene expression data is the identification of groups of genes that manifest similar expression patterns over several conditions. The corresponding algorithmic problem is to cluster multi-conditional gene expression patterns. A *clustering problem* consists of n elements and a *characteristic vector* for each element. A measure of *similarity* is defined between pairs of such vectors. (In gene expression, elements will be genes, the vector of each gene will contain its expression levels under each of the conditions, and similarity can be measured, for example, by correlation coefficient between vectors.) The goal is to partition the elements into subsets, which are called *clusters*, so that two criteria are satisfied: *Homogeneity* - elements inside a cluster are highly similar to each other; and *separation* - elements from different clusters have low similarity to each other.

There is a very rich literature on cluster analysis going back over three decades. Several algorithmic techniques were previously used in clustering gene expression data, including hierarchical clustering, self organizing maps, simulated annealing, and graph theoretic approaches.

We have developed a novel clustering algorithm that we call CLICK - CLuster Identification via Connectivity Kernels. Our work builds on a recent clustering approach of Hartuv and Shamir (cf. [1]). The algorithm does not make any prior assumption on the number or the structure of the clusters. The algorithm generates results with guaranteed properties, and is capable of handling large datasets very fast. It has been tested successfully on a variety of clustering problems in different areas of biology.

2 Method and Results

The Algorithm: We assume that the input data is an $n \times m$ matrix, whose rows correspond to elements, each row being a real vector of attributes of the corresponding element. The analysis of the data involves three main phases: (1) Preprocessing - normalization of the data and calculation of pairwise similarity between elements; (2) Clustering; and (3) Assessment and refinement of the results. We concentrate here on the second phase.

The algorithm represents the input data as a weighted graph G, where vertices correspond to elements and edge weights reflect pairwise similarity between the corresponding elements. Under some simplified probabilistic assumptions, the weight of an edge reflects the likelihood that its endpoints originated from the same cluster.

Suppose we remove from the graph all edges whose weight is below a conservative threshold. It is easy to see that clusters will tend to reside inside connected components of G. In the course of the algorithm low likelihood edges are removed from G yielding smaller connected components, thus refining the partition of the vertices of G. This iterative process continues until a stopping criterion is met. Upon termination, the set of elements in each component is called a *kernel*.

The basic CLICK algorithm can be described recursively as follows: In each step the algorithm handles some connected component of the subgraph induced by the yet unclustered elements. If the component satisfies the stopping criterion, it is declared a kernel. Otherwise, a minimum weight cut is computed, and the component is split into its two most loosely connected pieces according to this cut. After the above process terminates, an *adoption procedure* enriches kernels by adding to them singletons whose vectors are highly similar to the mean vector of the kernel. Finally, a *merging procedure* merges similar clusters. Several ad-hoc refinements were developed in order to reduce the running time of CLICK on very big instances.

Properties of the Clustering: The algorithm recognizes a connected component H as a kernel iff its diameter is two, and the likelihood that it should be further partitioned is below a threshold. H adopts a singleton iff its similarity to the mean of H is above some threshold. Two clusters are merged iff their means are highly similar. Hence, the resulting clusters are homogeneous.

If H is not recognized as a kernel, it is cut into two pieces by removing a minimum weight cut from H. Under some simplifying probabilistic assumptions, the two resulting pieces are the most loosely connected. Along with the merging process, this ensures separation of the eventual clusters.

Results and Performance: We have applied our algorithm to several biological datasets of various types, including gene expression data, cDNA oligo-fingerprint data, protein similarity data and DNA sequence data. We compared the quality of our results against those of the HCS algorithm [1] on a cDNA oligo-fingerprint dataset of size 2329×139, for which the true solution is known. Solution quality was assessed by its *Minkowski score* which is defined as follows: Let M^S be a binary matrix with $M^S_{ij} = 1$ iff i, j are in the same cluster in a solution S. Let M^T be the corresponding matrix for the true solution T. The Minkowski score of S is $\|M^T - M^S\|/\|M^T\|$. Thus, the smaller the score the better the solution. On this dataset CLICK reached a score of 0.6, while HCS got a score of 0.71.

Our algorithm is very fast and can cluster over 100,000 elements in several hours on a regular workstation. The performance of the algorithm on real data sets is illustrated in Table 1.

Table 1: The performance of CLICK on various datasets.

Data Type	#Elements	#Edges	Time (minutes)
oligo-fingerprint	2,329	134,352	0.5
oligo-fingerprint	22,118	20,915,098	2
protein similarity	72,623	1,796,067	11
protein similarity	117,835	7,277,067	149

References

[1] Hartuv, E., Schmitt, A., Lange, J., Meier-Ewert, S., Lehrach, H., and Shamir, R., An algorithm for clustering cDNAs for gene expression analysis using short oligonucleotide fingerprints, *Proc. Third International Symposium on Computational Molecular Biology*, 188–197, 1999.