Proc. ISMB '00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis

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

Abstract

Novel DNA microarray technologies enable the monitoring of expression levels of thousands of genes simultaneously. This allows a global view on the transcription levels of many (or all) genes when the cell undergoes specific conditions or processes. Analyzing gene expression data requires the clustering of genes into groups with similar expression patterns. We have developed a novel clustering algorithm, called CLICK, which is applicable to gene expression analysis as well as to other biological applications. No prior assumptions are made on the structure or the number of the clusters. The algorithm utilizes graph-theoretic and statistical techniques to identify tight groups of highly similar elements (kernels), which are likely to belong to the same true cluster. Several heuristic procedures are then used to expand the kernels into the full clustering.

CLICK has been implemented and tested on a variety of biological datasets, ranging from gene expression, cDNA oligo-fingerprinting to protein sequence similarity. In all those applications it outperformed extant algorithms according to several common figures of merit. CLICK is also very fast, allowing clustering of thousands of elements in minutes, and over 100,000 elements in a couple of hours on a regular workstation.

Keywords: Clustering, Gene Expression, Graph Algorithms, Minimum Cut.

Introduction

Novel DNA microarray technologies (Eisen & Brown 1999) 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 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-condition gene expression patterns. The grouping of genes with similar expression patterns into clusters helps in unraveling relations between genes, deducing the function of genes and revealing the underlying gene regulatory network.

A *clustering problem* consists of n elements and a characteristic vector for each element. In gene expression data, elements are genes, and the vector of each gene contains its expression levels under some conditions. These levels are obtained by measuring the intensity of hybridization of gene-specific oligonucleotides (or cDNA molecules), which are immobilized to a surface, to a labeled target RNA mixture (cf. (Eisen & Brown 1999)). A measure of pairwise *similarity* is then defined between such vectors. For example, similarity can be measured by the 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 in the same cluster are highly similar to each other; and *separation* - elements from different clusters have low similarity to each other. The problem has numerous applications in biology as well as in other disciplines.

There is a very rich literature on cluster analysis going back over three decades (cf. (Hartigan 1975; Everitt 1993; Mirkin 1996; Hansen & Jaumard 1997)). Several algorithmic techniques were previously used in clustering gene expression data, including hierarchical clustering (Eisen *et al.* 1998), self organizing maps (Tamayo *et al.* 1999), simulated annealing (Alon *et al.* 1999), and graph theoretic approaches (Hartuv *et al.* 1999; Ben-Dor, Shamir, & Yakhini 1999).

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 (Hartuv *et al.* 1999). The algorithm does not make any prior assumptions on the number or the structure of the clusters. At the heart of the algorithm is a process of recursively partitioning a weighted graph into components using minimum cut computations. The edge weights and the stopping criterion of the recursion are assigned probabilistic meaning, which gives the algorithm high accuracy. The speed of

Proc. ISMB '00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

the algorithm is achieved by a variety of experimentally tested heuristic procedures that shortcut, prepend and append the main process.

CLICK was implemented and tested on a variety of biological datasets. On two large-scale gene expression datasets, the algorithm outperformed previously published results, which utilized hierarchical clustering and self organizing maps. Substantial improvements over two extant algorithms were also demonstrated in clustering of cDNA oligo-fingerprinting data. As an additional test, we applied CLICK to two large protein similarity datasets, and obtained similar or better results than leading ad-hoc algorithms for this application. CLICK is also very fast, allowing clustering of thousands of elements in minutes, and over 100,000 elements in a couple of hours on a regular workstation.

Preliminaries

Let $G = (V, E)$ be a connected weighted graph. We denote the vertex set of G also by $V(G)$. For a vertex $v \in V$, define the *weight* of v to be the sum of weights of the edges incident on v. A *cut* C in G is a subset of its edges, whose removal disconnects G. The *weight* of C is the sum of the weights of its edges. A *minimum weight cut* is a cut in G with minimum weight. In case of non-negative edge weights, a minimum weight cut C partitions the vertices of G into two disjoint non-empty subsets $A, B \subset V$, $A \cup B = V$, such that $E \cap \{(u, v) : u \in A, v \in B\} = C$. An *s-t cut*, for $s, t \in V$, is a cut whose removal disconnects s from t. For a pair of vertices $u, v \in V$, the *distance* between u and v is the length of the shortest path which connects them. The *diameter* of G is the maximum distance between a pair of vertices in G.

Let $N = \{e_1, \ldots, e_n\}$ be a set of n elements, and let $\mathcal{C} = (C_1, \ldots, C_l)$ be a partition of N into subsets. Each of these subsets is called a *cluster*, and $\mathcal{C}$ is called a *clustering solution*, or simply a *clustering*. Two elements e_i and e_j are called *mates* with respect to $\mathcal{C}$ if they are both members of the same cluster in $\mathcal{C}$. When $\mathcal{C}$ is the true clustering of N (with respect to some predefined criterion) we simply call e_i and e_j mates.

The input data for a clustering problem is typically given in one of two forms: (1) *Fingerprint data* - each element is associated with a real-valued *fingerprint* vector, which contains p measurements on the element, e.g., expression levels of an mRNA at different conditions (cf. (Eisen & Brown 1999)), or hybridization intensities of a cDNA with different oligos (cf. (Lennon & Lehrach 1991)). (2) *Similarity data* - pairwise similarity values between elements, e.g., an E-value for the similarity between two protein sequences (cf. (Yona, Linial, & Linial 1999)).

Naturally, fingerprint data is more informative than similarity data: Given the data, one can compute any chosen pairwise similarity function between elements. Moreover, many other computations are possible, e.g., computing the mean vector for a group of elements.

The goal in a clustering problem is to partition the set of elements N into homogeneous and well-separated clusters. That is, we require that elements from the same cluster will be highly similar to each other, while elements from different clusters will have low similarity to each other.

Our algorithm can be applied to both fingerprint data and similarity data. In the next section we describe our algorithm for clustering fingerprint data. In the Applications Section we mention ad-hoc modifications of the algorithm to handle protein similarity data. A detailed description of our algorithm for handling similarity data will appear elsewhere, for lack of space.

The Clustering Algorithm

Let $N = \{e_1, \ldots, e_n\}$ be a set of elements. Let M be an input real-valued matrix of order $n \times p$, where M_{ij} is the j-th attribute of e_i. The i-th row-vector in M is the fingerprint of e_i. For a set of elements $K \subseteq N$, we define the *fingerprint* of K to be the mean vector of the fingerprints of the members of K.

The analysis of the raw data involves three main steps: (1) Preprocessing - normalization of the data and calculation of pairwise similarity values between elements; (2) Clustering; and (3) Assessment of the results. We describe each of the three steps below.

Preprocessing of Fingerprint Data

Our goal in the preprocessing step is to normalize the data, define a similarity measure between elements and characterize mates and non-mates in terms of their pairwise similarity values.

Common procedures for normalizing fingerprint data include transforming each fingerprint to have mean zero and variance one, a fixed norm, a fixed maximum entry, etc. Choosing an appropriate procedure depends on the kind of data dealt with, and on the biological context of the study. Examples for different data normalization procedures are given in the Applications Section.

Often, each fingerprint in the normalized data has the same norm. If fixed-norm fingerprints are viewed as points in the Euclidean space, then these points lie on a p-dimensional sphere, and the dot-product of two vectors is proportional to the cosine of the angle between them. We therefore use the vector *dot-product* as the default similarity measure. In case all fingerprints have mean zero and variance one, the dot-product of two vectors equals their correlation coefficient.

A key probabilistic assumption in developing CLICK is that pairwise similarity values between elements are normally distributed: Similarity values between mates are normally distributed with mean μ_T and variance σ_T^2, and similarity values between non-mates are normally distributed with mean μ_F and variance σ_F^2, where $\mu_T > \mu_F$. This situation was observed on real data (for example, see Figure 5 in the Applications Section), and can be theoretically justified as follows: Suppose that for each fingerprint, its attribute values are independent and drawn from some random distribution.

Let $u = (u_1, \ldots, u_p)$ and $v = (v_1, \ldots, v_p)$ be two fingerprints, and let $S_{uv} = \sum_{i=1}^{p} u_i v_i$ be their similarity value. The distribution of $u_i v_i$ depends on whether u and v are mates. Suppose that u and v are mates, and let the mean and the variance of this distribution be μ_T/p and σ_T^2/p, respectively. Then by the Central Limit Theorem, provided that p is large enough, $S_{uv} \sim N(\mu_T, \sigma_T^2)$. We denote by $f(x|\mu_T, \sigma_T)$ the *mates probability density function*. Similarly, if u and v are non-mates, and the distribution of $u_i v_i$ has mean μ_F/p and variance σ_F^2/p, then $S_{uv} \sim N(\mu_F, \sigma_F^2)$. We denote by $f(x|\mu_F, \sigma_F)$ the *non-mates probability density function*.

We remark, that when similarity values are not normally distributed, then their distribution can be approximated, and the same ideas presented below can be applied. In practice, the normality assumption often holds, as demonstrated by the results in the Applications Section.

An initial step of the algorithm is estimating the distribution parameters μ_T, μ_F, σ_T and σ_F, and the probability p_{mates} that two randomly chosen elements are mates. There are two possible methods to compute these parameters: (1) In many cases the true partition for a subset of the elements is known. This is the case, for example, if the clustering of some of the genes in a cDNA chip experiment is found experimentally (cf. (Hartuv *et al.* 1999)), or more generally, if a subset of the elements has been analyzed using prior biological knowledge. Based on this partition one can compute the sample mean and sample variance for similarity values between mates and between non-mates, and use these as maximum likelihood estimates for the distribution parameters. The proportion of mates among all pairs can serve as an estimate for p_{mates}, if the subset was randomly chosen. (2) In case no additional information is given, these parameters can be estimated using the EM algorithm (see, e.g., the probabilistic clustering section in (Mirkin 1996)).

The Basic CLICK Algorithm

Let S be a pairwise similarity matrix for M, where S_{ij} is the dot-product between the fingerprint vectors of e_i and e_j, i.e., $S_{ij} = \sum_{k=1}^{p} M_{ik} M_{jk}$. The algorithm represents the input data as a weighted *similarity graph* $G = (V, E)$. In this graph vertices correspond to elements and edge weights are derived from the similarity values. The weight w_{ij} of an edge (i, j) reflects the probability that i and j are mates, and is set to be

$$w_{ij} = \log \frac{p_{mates} f(S_{ij}|i, j \text{ are mates})}{(1 - p_{mates}) f(S_{ij}|i, j \text{ are non-mates})}$$

Here $f(S_{ij}|i, j \text{ are mates}) = f(S_{ij}|\mu_T, \sigma_T)$ is the value of the mates probability density function at S_{ij}:

$$f(S_{ij}|i, j \text{ are mates}) = \frac{1}{\sqrt{2\pi}\sigma_T} e^{-\frac{(S_{ij}-\mu_T)^2}{2\sigma_T^2}}$$

Similarly, $f(S_{ij}|i, j \text{ are non-mates})$ is the value of the non-mates probability density function at S_{ij}:

$$f(S_{ij}|i, j \text{ are non-mates}) = \frac{1}{\sqrt{2\pi}\sigma_F} e^{-\frac{(S_{ij}-\mu_F)^2}{2\sigma_F^2}}$$

Hence,

$$w_{ij} = \log \frac{p_{mates}\sigma_F}{(1 - p_{mates})\sigma_T} + \frac{(S_{ij} - \mu_F)^2}{2\sigma_F^2} - \frac{(S_{ij} - \mu_T)^2}{2\sigma_T^2}$$

(All logarithms in this paper are natural-base logarithms.)

For efficiency, low weight edges are omitted from the graph, so that there is an edge between two elements iff their pairwise similarity value is above some predefined non-negative threshold t_S.

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 contains a single vertex, then this vertex is considered a *singleton* and is handled separately. Otherwise, a stopping criterion (which will be described later) is checked. If the component satisfies the criterion, it is declared a *kernel*. Otherwise, the component is split according to a minimum weight cut. The algorithm outputs a list of kernels which serve as a basis for the eventual clusters. It is detailed in Figure 1. We assume that procedure MinWeightCut(G) computes a minimum weight cut of G and returns a partition of G into two subgraphs H and $\bar{H}$ according to this cut.

```
Basic-CLICK(G):
If V(G) = {v} then move v to the singleton set R.
Else if G is a kernel then
    Output V(G).
Else
    (H, H̄) ←MinWeightCut(G).
    Basic-CLICK(H).
    Basic-CLICK(H̄).
```

Figure 1: The basic CLICK algorithm.

The idea behind the algorithm is the following. Given a connected graph G, we would like to decide whether $V(G)$ is a subset of some true cluster, or $V(G)$ contains elements from at least two true clusters. In the first case we say that G is *pure*. In order to make this decision we test for each cut C in G the following two hypotheses:

- H_0^C: C contains only edges between non-mates.
- H_1^C: C contains only edges between mates.

If G is pure then H_1^C is true for every cut C of G. If on the other hand G is not pure, then there exists at least one cut C for which H_0^C holds. We therefore determine that G is pure iff H_1^C is accepted for each cut C in G. In case we decide that G is pure, we declare it to be a kernel. Otherwise, we partition $V(G)$ into two

Proc. ISMB '00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

disjoint subsets, according to a cut C in G for which the posterior probability ratio $Pr(H_1^C|C)/Pr(H_0^C|C)$ is minimum. Here, $Pr(H_i^C|C)$ denotes the posterior probability for H_i^C, $i = 0, 1$, given a cut C (along with its edge weights). We call such a partition a *weakest bipartition* of G.

We first show how to find a weakest bipartition of G. To this end, we make a simplifying probabilistic assumption that for a cut C in G the random variables $\{S_{ij}\}_{(i,j)\in C}$ are mutually independent. We denote the weight of a cut C by $W(C)$. We denote by $f(C|H_0^C)$ the likelihood that the edges of C connect only non-mates, and by $f(C|H_1^C)$ the likelihood that the edges of C connect only mates. We let $Pr(H_i^C)$ denote the prior probability for H_i^C, $i = 0, 1$.

Lemma 1 *Let G be a complete graph. Then for any cut C in G*

$$W(C) = \log \frac{Pr(H_1^C|C)}{Pr(H_0^C|C)}$$

Proof: Using Bayes Theorem (cf. (DeGroot 1989)) we find that

$$\frac{Pr(H_1^C|C)}{Pr(H_0^C|C)} = \frac{Pr(H_1^C)f(C|H_1^C)}{Pr(H_0^C)f(C|H_0^C)}$$

The joint probability density function of the weights of the edges in C, given that they are normally distributed with mean μ_T and variance σ_T^2, is

$$f(C|H_1^C) = \prod_{(i,j)\in C} \frac{1}{\sqrt{2\pi}\sigma_T} e^{-\frac{(S_{ij}-\mu_T)^2}{2\sigma_T^2}}$$

Similarly,

$$f(C|H_0^C) = \prod_{(i,j)\in C} \frac{1}{\sqrt{2\pi}\sigma_F} e^{-\frac{(S_{ij}-\mu_F)^2}{2\sigma_F^2}}$$

The prior probability for H_1^C is $p_{mates}^{|C|}$ and for H_0^C is $(1 - p_{mates})^{|C|}$.

Therefore,

$$\begin{aligned}
\log \frac{Pr(H_1^C|C)}{Pr(H_0^C|C)} &= \log \frac{Pr(H_1^C)f(C|H_1^C)}{Pr(H_0^C)f(C|H_0^C)} \\
&= |C| \log \frac{p_{mates}\sigma_F}{(1-p_{mates})\sigma_T} \\
&\quad + \sum_{(i,j)\in C} \frac{(S_{ij}-\mu_F)^2}{2\sigma_F^2} \\
&\quad - \sum_{(i,j)\in C} \frac{(S_{ij}-\mu_T)^2}{2\sigma_T^2} \\
&= W(C)
\end{aligned}$$

■

Suppose at first that G is a complete graph and no threshold was used to filter edges. From Lemma 1 it follows that a minimum weight cut of G induces a weakest bipartition of G.

It remains to show how to decide if G is pure, or equivalently, which stopping criterion to use. For a cut C, we accept H_1^C iff $Pr(H_1^C|C) > Pr(H_0^C|C)$. That is, we accept the hypothesis with higher posterior probability.

Let C be a minimum weight cut of G, which partitions it into two subgraphs H and $\bar{H}$. By Lemma 1, for every other cut C' of G

$$\log \frac{Pr(H_1^C|C)}{Pr(H_0^C|C)} = W(C) \leq W(C') = \log \frac{Pr(H_1^{C'}|C')}{Pr(H_0^{C'}|C')}$$

Therefore, H_1^C will be accepted for C iff $H_1^{C'}$ will be accepted for every cut C' in G. Thus, we accept H_1^C and declare that G is a kernel iff $W(C) > 0$.

We now extend these ideas to the case that G is incomplete. Consider first the decision whether G is pure or not. It is now possible that H_1^C will be accepted for C but rejected for some other cut of G. Nevertheless, a test based on $W(C)$ approximates the desired test. In order to apply our test criterion, we have to approximate the contribution of the edges missing from C to the posterior probability ratio $Pr(H_1^C|C)/Pr(H_0^C|C)$. This is done as follows: Let $F = (H \times \bar{H}) \setminus C$ and let $r = |H||\bar{H}|$. Denote by $\Phi(\cdot)$ the cumulative standard normal distribution function. Define

$$\begin{aligned}
W^*(C) &\equiv \log \frac{\prod_{(i,j)\in F} p_{mates} Pr(S_{ij} \leq t_S|H_1^C)}{\prod_{(i,j)\in F}(1-p_{mates}) Pr(S_{ij} \leq t_S|H_0^C)} \\
&= (r - |C|) \log \frac{p_{mates}\Phi((t_S-\mu_T)/\sigma_T)}{(1-p_{mates})\Phi((t_S-\mu_F)/\sigma_F)}
\end{aligned}$$

We decide that G is pure and declare it to be a kernel iff $W(C) + W^*(C) > 0$.

In case we decide that G is not pure, we use C in order to partition G into two components. This yields an approximation of a weakest bipartition of G.

We remark, that since we are interested in testing H_0^C and H_1^C on a specific minimum weight cut C, we can accurately calculate the contribution of the missing edges to the posterior probability ratio by computing their real weights from the raw data. This of course increases the running time of the algorithm.

The Full Algorithm

The Basic-CLICK algorithm produces kernels of clusters, which should be expanded to yield the full clusters. The expansion is done by considering the singletons which were found during the execution of Basic-CLICK. We denote by $\mathcal{L}$ and R the current lists of kernels and singletons, respectively. An *adoption step* repeatedly searches for a singleton v and a kernel K whose pairwise fingerprint similarity is maximum among all pairs of singletons and kernels (in practice we consider only kernels with at least five members). If the value of this similarity exceeds some predefined threshold, then v is *adopted* to K, that is, v is added to K and removed from R, and the fingerprint of K is updated. Otherwise, the iterative process ends. We note, that the adoption step

Proc. ISMB '00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

has a theoretical justification (see (Ben-Dor, Shamir, & Yakhini 1999)).

The main advantage of this approach is that adoption is decided based on the full raw data M, in contrast to other approaches in which adoption is decided based on the similarity graph (see, e.g., (Hartuv *et al.* 1999)).

After the adoption step takes place, we start a recursive clustering process on the set R of remaining singletons. This is done by discarding all other vertices from the initial graph. We iterate that way until no change occurs.

At the end of the algorithm a *merging step* merges clusters whose fingerprints are similar. The merging is done iteratively, each time merging two kernels whose fingerprint similarity is the highest, provided that this similarity exceeds a predefined threshold. When two kernels are merged, they are removed from $\mathcal{L}$, the newly merged kernel is added to $\mathcal{L}$, and its fingerprint is calculated. Finally, a last singleton adoption step is performed.

The full algorithm is detailed in Figure 2. G_R is the subgraph of G induced by the vertex set R. Procedure Adoption($\mathcal{L}$, R) performs the singleton adoption step. Procedure Merge($\mathcal{L}$) performs the merging step.

```
R ← N.
While some change occurs do:
        Execute Basic-CLICK(G_R).
        Let L be the list of kernels produced.
        Let R be the set of singletons produced.
        Adoption(L, R).
Merge(L).
Adoption(L, R).
```

Figure 2: The CLICK algorithm.

Speedup Refinements

Several ad-hoc refinements were developed in order to reduce the running time of CLICK on very big instances. We describe those heuristics below.

Screening: When handling very large connected components (say, of size 100,000), computing a minimum weight cut is very costly. Moreover, in the graphs that we encountered, large connected components are rather sparse (see Table 7) and hence contain low weight vertices. Removing a minimum cut from such a component will generally separate a low weight vertex, or a few such vertices, from the rest of the graph. This is computationally very expensive and not informative at an early stage of the clustering.

To overcome this problem, we screen low weight vertices from large components, prior to their clustering. The screening is done as follows: We first compute the average vertex weight W in the component, and multiply it by a factor which is proportional to the logarithm of the size of the component. We denote the resulting threshold by W^*. We then remove vertices whose weight is below W^*, and continue to do so updating the weight of the remaining vertices, until the updated weight of every (remaining) vertex is greater than W^*. The removed vertices are marked as singletons and handled at a later stage.

Minimum s-t Cuts: CLICK uses our implementation of Hao-Orlin algorithm for computing a minimum weight cut (Hao & Orlin 1994). This algorithm was shown to outperform other minimum cut algorithms in practice (cf. (Chekuri *et al.* 1997)). Its running time using highest label selection (cf. (Chekuri *et al.* 1997)) is $O(n^2\sqrt{m})$. For large components this is computationally quite expensive. Thus, for components of size greater than 1,000 we apply a different strategy, which we found to work in practice almost as good as computing a minimum cut.

The idea is to compute a minimum s-t cut in the underlying unweighted graph (where the weight of each edge is one), instead of a minimum weight cut. The complexity of this computation using Dinic's algorithm (cf. (Even 1979)) is only $O(nm^{2/3})$ time. In order to use this approach we have to show how to properly choose the vertices that should be separated. s and t are chosen so that their distance equals the diameter of the graph. More accurately, we first compute the diameter d of the graph, using breadth first search. If $d \geq 4$ we choose two vertices s and t whose distance is d, and partition the graph according to a minimum s-t cut.

Assessing the Results

We present in this section figures of merit for measuring the quality of a clustering solution. Different measures are applicable in different situations, depending on whether a partial true solution is known or not, and whether the input is fingerprint or similarity data. Those measures will be used for assessing the results in the Applications Section.

Suppose at first that the true solution is known, and we wish to compare it to a suggested solution. Any clustering solution can be represented by a binary $n \times n$ matrix C, in which $C_{ij} = 1$ iff i and j belong to the same cluster in that solution. Let T and C be the matrices for the true solution and the suggested solution, respectively. Let n_{kl}, $k, l = 0, 1$, denote the number of pairs (i, j) $(i \neq j)$ for which $T_{ij} = k$ and $C_{ij} = l$. Thus, n_{11} is the number of mates which are detected by the suggested solution, n_{00} is the number of non-mates identified, while n_{01} and n_{10} count the disagreements between the true solution and the suggested one.

The *Minkowski measure* (see, e.g., (Sokal 1977)) is defined as

$$\sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}}$$

Hence, it measures the proportion of disagreements to the total number of mates in the true solution. A perfect solution has score zero, and the lower the score - the better the solution. The *Jaccard coefficient* (see, e.g., (Everitt 1993)) is the ratio

$$\frac{n_{11}}{n_{11}+n_{10}+n_{01}}$$

It is the proportion of correctly identified mates to the sum of the correctly identified mates plus the total number of disagreements. Hence, a perfect solution has score one, and the higher the score - the better the solution. Note, that both measures do not (directly) involve the term n_{00}, since solution matrices tend to be sparse and this term would dominate the other three in good and bad solutions alike. When the true solution is known only for a subset $N^* \subset N$, the Minkowski and Jaccard measures can be computed on the submatrices corresponding to N^*. In some cases, e.g., for cDNA oligo-fingerprint data (see the Applications Section), we have the additional information that no element of N^* has a mate in $N \setminus N^*$. In such a case, the Minkowski and Jaccard measures are evaluated using all the pairs $\{(i,j) : i \in N^*, j \in N \cup N^*, i \neq j\}$.

When the true solution is not known, we evaluate the quality of the solution by computing two figures of merit to measure the homogeneity and separation of the produced clusters. For fingerprint data, homogeneity is evaluated by the average and minimum correlation coefficient between the fingerprint of an element and the fingerprint of its corresponding cluster. Precisely, if $cl(u)$ is the cluster of u, $F(X)$ and $F(u)$ are the fingerprints of a cluster X and an element u, respectively, and $S(x,y)$ is the correlation coefficient (or any other similarity measure) of fingerprints x and y, then

$$\begin{aligned} H_{Ave} &= \frac{1}{|N|}\sum_{u \in N} S(F(u), F(cl(u))) \\ H_{Min} &= \min_{u \in N} S(F(u), F(cl(u))) \end{aligned}$$

Separation is evaluated by the weighted average and the maximum correlation coefficient between cluster fingerprints. That is, if the clusters are $X_1, \ldots X_t$, then

$$\begin{aligned} S_{Ave} &= \frac{1}{\sum_{i \neq j} |X_i||X_j|} \sum_{i \neq j} |X_i||X_j| S(F(X_i), F(X_j)) \\ S_{Max} &= \max_{i \neq j} S(F(X_i), F(X_j)) \end{aligned}$$

Hence, a solution improves if H_{Ave} increases and H_{Min} increases, and if S_{Ave} decreases and S_{Max} decreases.

For similarity data, we use a measure suggested by Z. Yakhini (private communication): Suppose that the input is a similarity graph $G = (V, E)$ with edges representing high similarity (exceeding some threshold). Homogeneity is evaluated by the fraction of edges inside clusters, and separation is evaluated by the percentage of edges between different clusters. That is,

$$\begin{aligned} H &= \frac{|\{(i,j)|cl(i) = cl(j) \text{ and } (i,j) \in E\}|}{|\{(i,j)|cl(i) = cl(j)\}|} \\ S &= \frac{|\{(i,j)|cl(i) \neq cl(j) \text{ and } (i,j) \in E\}|}{|\{(i,j)|cl(i) \neq cl(j)\}|} \end{aligned}$$

Applications

In the following we describe CLICK's results on several biological datasets. The architecture used for the execution of CLICK is described after the specific applications.

Gene Expression

CLICK was first tested on the yeast cell cycle dataset of (Cho *et al.* 1998). That study monitored the expression levels of 6,218 *S. cerevisiae* putative gene transcripts (identified as ORFs) measured at 10-minutes intervals over two cell cycles (160 minutes). We compared CLICK's results to those of the program GENECLUSTER (Tamayo *et al.* 1999) that uses Self-Organizing Maps. To this end, we applied the same filtering and data normalization procedures of (Tamayo *et al.* 1999). The filtering removes genes which do not change significantly across samples, resulting in a set of 826 genes. The data preprocessing includes the removal of the 90-minutes time-point and normalizing the expression levels of each gene to have mean zero and variance one within each of the two cell-cycles.

CLICK clustered the genes into 30 clusters in 14 seconds. These clusters are shown in Figure 3. A summary of the homogeneity and separation parameters for the solutions produced by CLICK and GENECLUSTER is shown in Table 1. CLICK obtained results superior in all the measured parameters. Two putative true clusters are the sets of late G1-peaking genes and M-peaking genes, reported in (Cho *et al.* 1998). Out of the late G1-peaking genes that passed the filtering, CLICK placed 91% in a single cluster (see Figure 3, cluster 3). In contrast, (Tamayo *et al.* 1999) report that in their solution 87% of these genes were contained in three clusters. Out of the M-peaking genes that passed the filtering, CLICK placed 95% in a single cluster (see Figure 3, cluster 1).

Program	#Clusters	Homogeneity		Separation	
		H_{Ave}	H_{Min}	S_{Ave}	S_{Max}
CLICK	30	0.8	-0.19	-0.07	0.65
GENECLUSTER	30	0.74	-0.88	-0.02	0.97

Table 1: A comparison between CLICK and GENECLUSTER on the yeast cell-cycle dataset (Cho *et al.* 1998).

As another test, we analyzed the dataset of (Iyer *et al.* 1999) which studied the response of human fibroblasts to serum. It contains expression levels of 8,613 human genes obtained as follows: Human fibroblasts were deprived of serum for 48 hours and then stimulated by addition of serum. Expression levels of genes were measured at 12 time-points after the stimulation. An additional data-point was obtained from a separate unsynchronized sample. A subset of 517 genes whose expression levels changed substantially across samples was

Proc. ISMB '00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

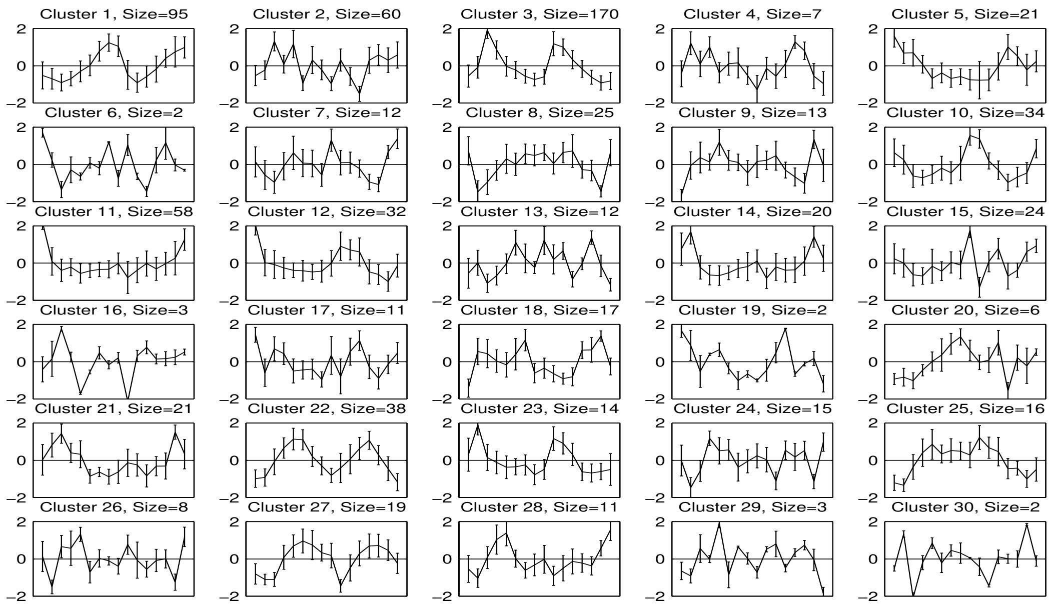

Figure 3: CLICK's clustering of the yeast cell cycle data (Cho *et al.* 1998). x axis: time points 0-80,100-160 at 10-minutes intervals. y axis: normalized expression levels. The solid line in each sub-figure plots the average pattern for that cluster. Error bars display the measured standard deviation. The cluster size is printed above each plot.

analyzed by the hierarchical clustering method of (Eisen *et al.* 1998). The data was normalized by dividing each entry by the expression level at time zero, and taking a logarithm of the result. For ease of manipulation, we also transformed each fingerprint to have a fixed norm. The similarity function used was dot-product, giving values identical to those used by (Eisen *et al.* 1998). CLICK clustered the genes into 10 clusters in 32 seconds. These clusters are shown in Figure 4. Table 2 presents a comparison between the clustering quality of CLICK and the hierarchical clustering of (Eisen *et al.* 1998) on this dataset. Again, CLICK performs better in all parameters.

Program	#Clusters	Homogeneity		Separation	
		H_{Ave}	H_{Min}	S_{Ave}	S_{Max}
CLICK	10	0.88	0.13	-0.34	0.65
Hierarchical	10	0.87	-0.75	-0.13	0.9

Table 2: A Comparison between CLICK and the hierarchical clustering of (Eisen *et al.* 1998) on the dataset of response of human fibroblasts to serum (Iyer *et al.* 1999).

cDNA oligo-fingerprints

We studied two datasets of oligonucleotide fingerprints of cDNAs obtained from Max Planck Institute of Molecular Genetics in Berlin. In oligo-fingerprinting (Lennon & Lehrach 1991), poly-dT primed cDNAs are extracted from a tissue, cloned and spotted on high density filters. The filter is then hybridized with a short oligonucleotide (oligo) and the hybridization levels for each spot are recorded. The experiment is repeated with 100-300 oligos, and the vector of hybridization levels of each spot with all oligos form its *oligo-fingerprint*. Clustering can be used to identify cDNAs that originated from the same gene, and consequently, pinpoint gene abundance levels and save on sequencing. The technique is used, for example, when sequences of all the genes of an organism are not known.

The first dataset we analyzed contains 2,329 cDNAs fingerprinted using 139 oligos. This dataset was part of a library of some 100,000 cDNAs prepared from purified peripheral blood monocytes by the Novartis Forschungsinstitut in Vienna, Austria (see (Hartuv *et al.* 1999)). The true clustering of these 2,329 cDNAs is known from back hybridization experiments performed with long, gene-specific oligonucleotides. It contains 18 gene clusters varying in size from 709 to 1. The second dataset contains 20,275 cDNAs originating from sea urchin egg, fingerprinted using 217 oligos (see (Poustka *et al.* 1999)). For this dataset the true solution is known on a subset of 1,811 cDNAs. Fingerprint normalization was done as explained in (Meier-Ewert *et al.* 1998).

Similarity values (dot-products) between pairs of cDNA fingerprints from the blood monocytes dataset are plotted in Figure 5. To test our hypotheses that the distributions of the similarity values between mates and between non-mates are normal, we applied a Kolmogorov-Smirnov test (see, e.g., (DeGroot 1989)) with a significance level of 0.05. The hypotheses were

Proc. ISMB '00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

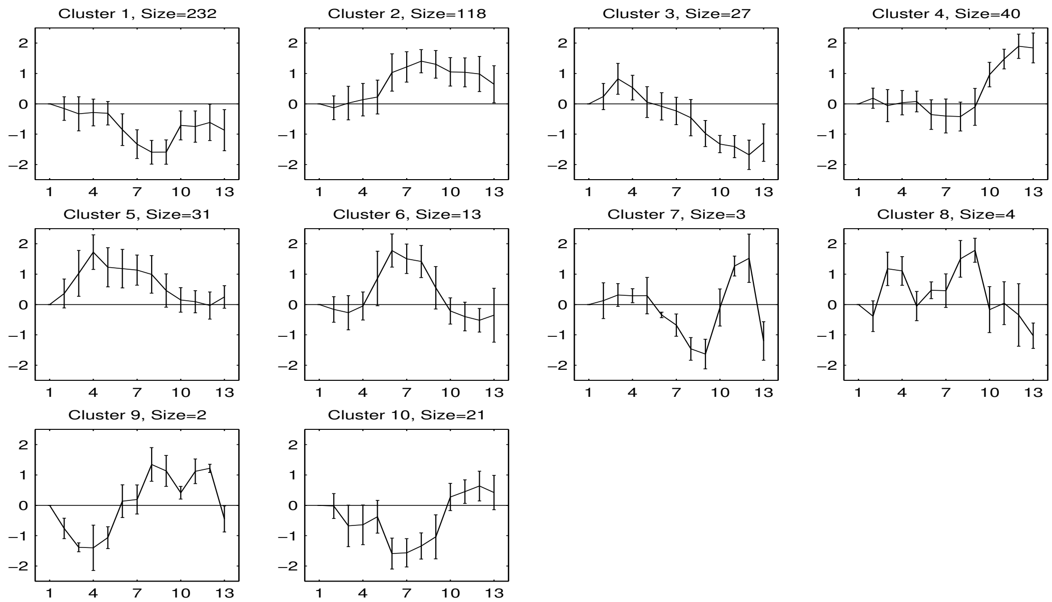

Figure 4: CLICK's clustering of the fibroblasts serum response data (Iyer *et al.* 1999). x axis: 1-12: synchronized time points. 13: unsynchronized point. y axis: normalized expression levels. The solid line in each sub-figure plots the average pattern for that cluster. Error bars display the measured standard deviation. The cluster size is printed above each plot.

accepted for both distributions, with the hypothesis regarding the non-mates distribution being accepted with higher confidence.

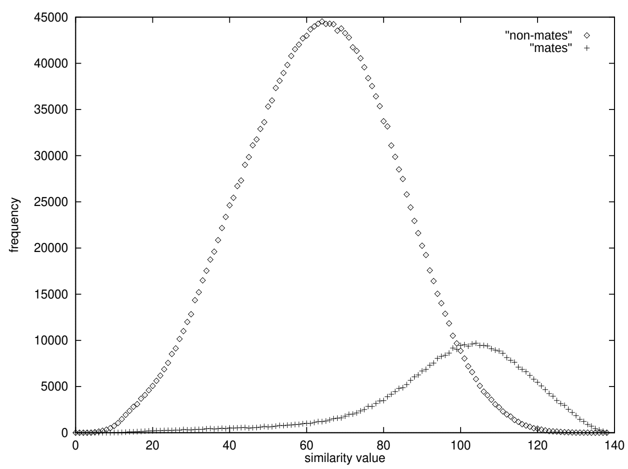

Figure 5: Similarity values between mates and between non-mates in the peripheral blood monocytes cDNA dataset of (Hartuv *et al.* 1999).

Table 3 shows a comparison of CLICK's results on the blood monocytes dataset with those of the HCS algorithm (Hartuv *et al.* 1999). Table 4 shows a comparison of CLICK's results on the sea urchin dataset with those of the K-means algorithm of (Herwig *et al.* 1999). CLICK outperforms the other algorithms in all figures of merit, and also obtains solutions with fewer unclustered singletons.

Protein classes

CLICK was also applied to two protein similarity datasets. The first dataset contains 72,623 proteins from the ProtoMap project (Yona, Linial, & Linial 1999). The second originated from the SYSTERS project (Krause, Stoye, & Vingron 2000) and contains 117,835 proteins. Both datasets contain for each pair of proteins an E-value of their similarity as computed by BLAST.

Protein classification is inherently hierarchical, so the assumption of normal distribution of mate similarity values does not seem to hold. In order to apply CLICK to the data, we made the following modifications:

1. The weight of an edge (i,j) was set to be $w_{ij} = \log \frac{p_{mates}(1-p_{ij})}{(1-p_{mates})p_{ij}}$, where p_{ij} is the E-value, and hence, also practically the p-value, of the similarity between i and j. We used a similarity threshold which corresponds to an E-value of 10^{-20}.
2. For a minimum weight cut C which partitions G into H and $\bar{H}$, we used $W^*(C) = (r - |C|)\log \frac{p_{mates}}{1-p_{mates}}$, where $r = |H||\bar{H}|$.
3. For the adoption step we used a variant of the approach of (Ben-Dor, Shamir, & Yakhini 1999): We calculated for each singleton r and each kernel K

Program	#Clusters	#Singletons	Minkowski	Jaccard	Time(min)
CLICK	31	46	0.57	0.7	0.8
HCS	16	206	0.71	0.55	43

Table 3: A comparison between CLICK and HCS on the blood monocytes cDNA dataset.

Program	#Clusters	#Singletons	Minkowski	Jaccard	Time(min)
CLICK	2,952	1,295	0.59	0.69	32.5
K-Means	3,486	2,473	0.79	0.4	–

Table 4: A comparison between CLICK and K-means on the sea urchin cDNA dataset.

(considering the set of singletons R as an additional kernel) the ratio

$$\frac{\sum_{k \in K} w_{rk}}{|K|}$$

We then chose the pair r, K with the highest ratio and r was adopted to K if this ratio exceeded some predefined threshold w^*.

4. For the merging step we calculated for each pair of kernels K_1 and K_2 the ratio

$$\frac{\sum_{k_1 \in K_1, k_2 \in K_2} w_{k_1 k_2}}{|K_1||K_2|}$$

We then chose the pair K_1, K_2 with the highest ratio and merged K_1 and K_2 if this ratio exceeded w^*.

For the evaluation of the ProtoMap dataset we used a Pfam classification for a subset of the data consisting of 17,244 single-domain proteins, which is assumed to be the true solution for this subset. We compared our results to the results of ProtoMap with a confidence level of 10^{-20} on this dataset. The comparison is shown in Table 5. The results are very similar, with a slight advantage to CLICK.

For the SYSTERS dataset, no "true solution" was assumed, so we evaluated the solutions of CLICK and SYSTERS using the figures of merit described in the previous section. Table 6 presents the results of the comparison. The results show a significant advantage to CLICK.

Implementation and Running Times

The CLICK program was written in C and executed on an SGI ORIGIN200 machine utilizing one IP27 processor. The implementation uses linear space and stores the similarity graph in a compact form by using linked adjacency lists. Table 7 summarizes the time performance of CLICK on the datasets described above.

#Elements	#Edges	Density	Time(min)
517	22,084	0.17	0.5
826	10,978	0.03	0.2
2,329	134,352	0.05	0.8
20,275	303,492	0.001	32.5
72,623	1,043,937	0.0004	53
117,835	4,614,038	0.0007	126.3

Table 7: A summary of the time performance of CLICK on the above mentioned datasets. The second column specifies the number of edges in the similarity graph for each instance. The third column specifies the fraction of edges with respect to the total number of element pairs.

Concluding Remarks

We presented in this paper a novel clustering algorithm which utilizes graph-theoretic and statistical techniques. The algorithm was tested on several biological datasets, originating from a variety of applications, and was shown to outperform extant clustering algorithms according to several common figures of merit. It is also very fast, allowing high-accuracy clustering of large datasets of size over 100,000 in a couple of hours. The speed of CLICK enables re-analyzing a dataset several times, with different thresholds. This allows using prior biological understanding to evaluate the clustering results and fine-tune the parameters accordingly.

Acknowledgments

We thank Ralf Herwig for the results of the K-means algorithm on the sea urchin cDNA data, Antje Krause for the data and results from the SYSTERS project, Golan Yona for providing the data and results from the ProtoMap project and for the Pfam subset, and Pablo Tamayo for the GENECLUSTER results on the yeast cell cycle data. We would like to thank Erez Hartuv, Ralf Herwig, Rani Elkon and Zohar Yakhini for fruitful discussions.

R. Sharan was supported by an Eshkol fellowship from the Ministry of Science, Israel. R. Shamir was supported by a grant from the Ministry of Science, Israel, and by the Israel Science Foundation formed by the Israel Academy of Sciences and Humanities.

References

Alon, U.; Barkai, N.; Notterman, D. A.; Gish, G.; Ybarra, S.; Mack, D.; and Levine, A. J. 1999. Broad

Proc. ISMB ’00, pp. 307-316, AAAI Press, Menlo Park, CA (2000)

Program	#Clusters	#Singletons	Minkowski	Jaccard	Time(min)
CLICK	7,747	16,612	0.88	0.39	53
ProtoMap	7,445	16,408	0.89	0.39	–

Table 5: A comparison between CLICK and ProtoMap on a dataset of 72,623 proteins.

Program	#Clusters	#Singletons	Homogeneity	Separation	Time(min)
CLICK	9,429	17,119	0.24	0.03	126.3
SYSTERS	10,891	28,300	0.14	0.03	–

Table 6: A comparison between CLICK and SYSTERS on a dataset of 117,835 proteins.

patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS* 96:6745–6750.

Ben-Dor, A.; Shamir, R.; and Yakhini, Z. 1999. Clustering gene expression patterns. *Journal of Computational Biology* 6(3/4):281–297.

Chekuri, C.; Goldberg, A.; Karger, D.; Levine, M.; and Stein, C. 1997. Experimental study of minimum cut algorithms. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 324–333.

Cho, R.; Campbell, M.; Winzeler, E.; Steinmetz, L.; Conway, A.; Wodicka, L.; Wolfsberg, T.; Gabrielian, A.; Landsman, D.; Lockhart, D.; and Davis, R. 1998. A genome-wide transcirptional analysis of the mitotic cell cycle. *Mol Cell* 2:65–73.

DeGroot, M. 1989. *Probability and Statistics.* Addison-Wesley.

Eisen, M. B., and Brown, P. O. 1999. DNA arrays for analysis of gene expression. In *Methods in Enzymology, Vol. 303.* 179–205.

Eisen, M. B.; Spellman, P. T.; Brown, P. O.; and Botstein, D. 1998. Cluster analysis and display of genome-wide expression patterns. *PNAS* 95:14863–14868.

Even, S. 1979. *Graph Algorithms.* Rockville, Maryland: Computer Science Press.

Everitt, B. 1993. *Cluster analysis.* London: Edward Arnold, third edition.

Hansen, P., and Jaumard, B. 1997. Cluster analysis and mathematical programming. *Mathematical Programming* 79:191–215.

Hao, J., and Orlin, J. 1994. A faster algorithm for finding the minimum cut in a directed graph. *Journal of Algorithms* 17(3):424–446.

Hartigan, J. 1975. *Clustering Algorithms.* John Wiley and Sons.

Hartuv, E.; Schmitt, A.; Lange, J.; Meier-Ewert, S.; Lehrach, H.; and Shamir, R. 1999. An algorithm for clustering cDNAs for gene expression analysis using short oligonucleotide fingerprints. In *Proceedings Third International Symposium on Computational Molecular Biology (RECOMB 99)*, 188–197. ACM Press. To appear in *Genomics*.

Herwig, R.; Poustka, A.; Muller, C.; Bull, C.; Lehrach, H.; and O’Brien, J. 1999. Large-scale clustering of cDNA-fingerprinting data. *Genome Research* 9:1093–1105.

Iyer, V.; Eisen, M.; Ross, D.; Schuler, G.; Moore, T.; Lee, J.; Trent, J.; Staudt, L.; Jr., J. H.; Boguski, M.; Lashkari, D.; Shalon, D.; Botstein, D.; and Brown, P. 1999. The transcriptional program in the response of human fibroblasts to serum. *Science* 283 (1).

Krause, A.; Stoye, J.; and Vingron, M. 2000. The SYSTERS protein sequence cluster set. *Nucleic Acid Research* 28 (1).

Lennon, G., and Lehrach, H. 1991. Hybridization analysis of arrayed cDNA libraries. *Trends Genet* 7:60–75.

Meier-Ewert, S.; Lange, J.; Gerst, H.; Herwig, R.; Schmitt, A.; Freund, J.; Elge, T.; Mott, R.; Herrmann, B.; and Lehrach, H. 1998. Comparative gene expression profiling by oligonucleotide fingerprinting. *Nucleic Acids Research* 26(9):2216–2223.

Mirkin, B. 1996. *Mathematical Classification and Clustering.* Kluwer.

Poustka, A.; Herwig, R.; Krause, A.; Hennig, S.; Meier-Ewert, S.; and Lehrach, H. 1999. Toward the gene catalogue of sea urchin development: The construction and analysis of an unfertilized egg cDNA library highly normalized by oligonucleotide fingerprinting. *Genomics* 59:122–133.

Sokal, R. R. 1977. Clustering and classification: Background and current directions. In Van Ryzin, J., ed., *Classification and Clustering*, 1–15. Academic Press.

Tamayo, P.; Slonim, D.; Mesirov, J.; Zhu, Q.; Kitareewan, S.; Dmitrovsky, E.; Lander, E. S.; and Golub, T. 1999. Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *PNAS* 96:2907–2912.

Yona, G.; Linial, N.; and Linial, M. 1999. Protomap: Automatic classification of protein sequences, a hierarchy of protein families, and local maps of the protein space. *Proteins: Structure, Function, and Genetics* 37:360–378.