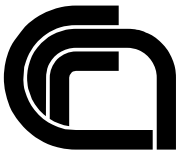

Consiglio Nazionale delle Ricerche

K-BOOST: A SCALABLE ALGORITHM FOR HIGH-QUALITY CLUSTERING OF MICROARRAY GENE EXPRESSION DATA

F. Geraci, M. Leoncini, M. Montangero, M. Pellegrini, M. E. Renda

IIT TR-15/2007

Technical report

Dicembre 2007

Istituto di Informatica e Telematica

K-Boost: a Scalable Algorithm for High-Quality Clustering of Microarray Gene Expression Data

Filippo Geraci [1], Mauro Leoncini [2], Manuela Montangero [2], Marco Pellegrini [1*] and M. Elena Renda [1]

[1]CNR, Istituto di Informatica e Telematica, via Moruzzi 1, 56124, Pisa, (Italy)
[2]Dipartimento di Ingegneria dell'Informazione, Università di Modena e Reggio Emilia, Via Vignolese 905, 41100 Modena (Italy)

ABSTRACT

Motivation: Microarray technology for profiling gene expression levels is a popular tool in modern biological research. Applications range from tissue classification to the detection of metabolic networks, from drug discovery to time-critical personalized medicine. Given the increase in size and complexity of the data sets produced, their analysis is becoming problematic in terms of time/quality trade-offs. Clustering genes with similar expression profiles is a key initial step for subsequent manipulations and the increasing volumes of data to be analyzed requires methods that are at the same time efficient (completing an analysis in minutes rather than hours) and effective (identifying significant clusters with high biological correlations).

Results: In this paper we propose *K-Boost*, a novel clustering algorithm based on a combination of the Furthest-Point-First (FPF) heuristic for solving the metric *k*-centers problem, a *stability-based* method for determining the number of clusters (i.e. the value of *k*), and a *k-means*-like cluster refinement. *K-Boost* is able to detect the optimal number of clusters to produce. It is scalable to large data-sets without sacrificing output quality as measured by several internal and external criteria.

Availability: http://bioalgo.iit.cnr.it/

Contact: marco.pellegrini@iit.cnr.it

1 INTRODUCTION

Several obstacles still lie on the path to exploiting the full potential of microarray technologies (Trent and Bexevanis, 2002). One issue is the scalability of the data processing software. In particular a critical initial phase is often the clustering of gene expression data into groups with homogeneous expression profile. In this paper we tackle this problem by proposing *K-Boost*, a clustering algorithm based on a combination of the Furthest-Point-First (FPF) heuristic for the *k*-centers problem (Gonzalez, 1985), a stability-based (SB) method for determining the optimal number of clusters *k* (Tibshirani *et al.*, 2005), and a *k-means*-like refinement. The experiments we report here demonstrate that our algorithm is scalable to large data-sets without sacrificing output quality.

The Furthest-point-first (FPF) heuristic is known to attain a result that is within a factor two of the optimum clustering according to the *k*-center criterion (i.e., minimizing the maximum diameter of any cluster). This theoretical guarantee, coupled with a small computational complexity and with a careful implementation, makes this algorithm an ideal candidate for attaining scalability. The FPF algorithm constructs the clusters incrementally (that is, the *k*-clustering is obtained by a refinement of the (*k*-1)-clustering), thus it works without any need for an initial guess for k.

For detecting the optimal number of clusters we use a stability-based technique for cluster validation by *prediction strength* (Tibshirani *et al.*, 2005) that guides the selection of the "best" number of clusters in which the data-set should be partitioned. The stability based criterion is well founded in an Information Theoretic framework. Moreover, since we can interleave and make incremental the computation of both FPF and SB, including the latter only adds a small cost to the computation time.

In the last phase we use the centers previously computed in an iterative loop that associates the other data-points to the closest center and iteratively updates center representation of each cluster.

The FPF heuristic has been applied also in the context of microarray clustering for time-series by (Ernst *et al.*, 2005). However our approach is different. We apply the FPF algorithm directly to real-world input data. In (Ernst *et al.*, 2005) it is applied to a set of artificially generated data points that are meant to uniformly cover the parametric space of possible experimental outcomes. This second approach suffers of scalability problems as the cardinality of the discrete search space grows exponentially in the parameters of the experiment.

The scalability of our algorithm can find applications in a number of different settings. One parameter is the sheer dimension of a single data-set: the technology of Tiling Arrays is capable of producing a complete profile of the transcript index of an individual genome (up to 50,000 transcript sequences and 60 tissues and cell lines can be mapped in a single chip experiment (Schadt *et al.*, 2004)). The second parameter is the trade-off between the number of experiments and the response time. Microarray technology, adapted towards the needs of personalized medicine, might be used to screen a vast range of different pathological conditions over large populations. In some applications there is the need to repeat the experiments many times and to have a prompt result. For example, data taken at different times from the same patient in a healthy state and in a pathological state could be clustered to highlight differences in the metabolism due to the pathology, filtering out the background effects of healthy individual metabolic profile. *K-Boost* might be useful in this context, where the great amount of data to be managed is one of the main bottlenecks for the existing techniques.

Scalability should by no means be paid for by a decrease in output quality. Ideally one would like new algorithms to be both faster and more accurate at the same time. Clustering is an inherently approximate activity and the issue of validating the quality of clusterings is an open area of research. Broadly speaking there are "internal" quality criteria (based on an inter-point metric that is assumed to be

*marco.pellegrini@iit.cnr.it

significant), or "external" quality criteria based on cross-referencing the produced clustering with a "golden standard" such as that derived by Gene Ontology annotations. We will use both methodologies to measure and compare clustering quality.

1.1 State of the art.

The papers by Eisen et al. (Eisen *et al.*, 1998), Alon et al. (Alon *et al.*, 1999) and Wen et al. (Wen *et al.*, 1988) have shown the rich potential of microarray gene expression data clustering as an exploratory tool for finding relevant biological relationships amidst large gene expression datasets. Since the late nineties a growing body of knowledge has been built up on several algorithmic aspects of the clustering task (see, *e.g.*, general surveys in (Jiang *et al.*, 2004; Shamir and Sharan, 2002)).

Among the most popular approaches we can broadly find those "distance based" such as *k*-means (Tavazoie *et al.*, 1999), Self Organized Maps (SOM) (Tamayo *et al.*, 1999), Hierarchical Agglomerative Clustering (HAC) (Eisen *et al.*, 1998), and several variants thereof. A second broad class of algorithms is graph-based (CLICK (Sharan *et al.*, 2003), CAST (Ben-Dor *et al.*, 1999) and CLIFF (Xing and Karp, 2001) all use weighted graph min cuts (with variants among them)). Excavator (Xu *et al.*, 2002) is based instead on Minimum Spanning Tree clustering. Other large families are those models-based (Ramoni *et al.*, 2002), fuzzy-logic-based (Belacel *et al.*, 2004), or based on principal component analysis (PCA) (Hastie *et al.*, 2000).

Among the main issues related to clustering there are 1) the problem of guessing the optimal number of clusters (Tibshirani *et al.*, 2001; Giurcaneanu *et al.*, 2003; Tibshirani *et al.*, 2005) and 2) cluster validation (Gibbons and Roth., 2000; Yeung *et al.*, 2001; Gat-Viks *et al.*, 2003). In biological data analysis a further issue is to provide metrics supported by ad hoc external biological knowledge (Huang, 2006; Hanisch *et al.*, 2002). A large and promising are of research is that of feature selection (Dugas *et al.*, 2004; J Taylor, 2006) leading to the more general concept of bi-clustering (Tanay *et al.*, 2006; Madeira and Oliveira, 2004). Special attention has been recently paid to particular kinds of micro-array experiments, notably time-series, in which there is a natural ordering and correlation for the conditions tested (Ernst *et al.*, 2005; Bar-Joseph, 2004).

At the present state of the art there is no clear overall winner in the area of clustering algorithms for gene expression data as any method has strong and weak points. However, one can safely say that the drive for higher quality results is always paid for by higher computational costs; therefore all these methods exhibit poor scalability or need educated guesses as to the setting of some critical parameter. In our approach we show that the two goals are not in contrast: scalability need not entail lower quality.

In a previous paper (Geraci *et al.*, 2007) we have proposed a scalable method for clustering gene expression data called *FPF-SB* that combines stability based computation of the optimal cluster number with the FPF heuristic for the k-center problem. Although FPF-SB was very effective in detecting the optimal number of clusters (a feature inherited by *K-Boost*) the final cluster decomposition was not tight enough to compete with traditional methods like *k-means* and SOM. *K-Boost* represents a non-trivial evolution of FPF-SB and attains much better performance.

In general we can split the algorithms for clustering field into two large groups: methods that take a suggested number of clusters as input and methods that are able to determine an optimal number of clusters (according to some internal criterion). Our proposal *K-Boost* is in the second category thus we will compare its performance with methods like CLICK and FPF-SB that are in the same class. We also do comparisons with methods like k-means and SOM, where we feed them a plausible number of clusters (as suggested by *K-Boost* or CLICK), thus giving them an advantage they do not have when applied in a stand-alone fashion.

According to several measures of quality both internal and external for a variety of data sets, *K-Boost* shows significantly better performance than CLICK and FPF-SB. The time complexity of *K-Boost*, while being higher than that of FPF-SB, it is still orders of magnitude faster than CLICK even on relatively small data sets.

2 PRELIMINARIES

2.1 Clustering

Let $N = \{e_1, \ldots, e_n\}$ be a set of n vectors in R^m, a partition $\tilde{N} = \{N_1, \ldots, N_k\}$ of N is a *clustering*, where each N_t, for $t \in \{1, 2, \ldots, k\}$, is called a *cluster*. Given a clustering $\tilde{N}$, two elements $e_i, e_j \in N$ are *mates* according to $\tilde{N}$ if they belong to the same cluster $N_t \subseteq \tilde{N}$, "non-mates" otherwise.

2.2 Distance function

We want mates to be "similar" in some measurable sense that captures their common biological behavior (co-expression). It turns out that, by duality, one can define a "distance" function among points. Given two vectors $e_i, e_j \in N$ (with components $e_{s,t}$, $s \in \{i, j\}$ and $1 \leq t \leq m$), we denote with $d_{i,j}$ their distance and we say that they are *similar* (respectively, *different*) if $d_{i,j}$ is small (respectively, large). Our definition of $d_{i,j}$ is based on the *Pearson Coefficient*, $P(e_i, e_j)$, given by

$$P(e_i, e_j) = \frac{\sum_{t=1}^{m} (e_{i,t} - \mu_i)(e_{j,t} - \mu_j)}{\sqrt{\left(\sum_{t=1}^{m} (e_{i,t} - \mu_i)^2\right)\left(\sum_{t=1}^{m} (e_{j,t} - \mu_j)^2\right)}}$$

where μ_i and μ_j are the means of e_i and e_j, respectively.

The Pearson Coefficient is a very popular measure of similarity in the context of gene expression micro-array data clustering but it is not a distance. To come up with a measure suitable for the metric space method, we first define $\delta_{i,j} = 1 - P(e_i, e_j)$, with $0 \leq \delta_{i,j} \leq 2$ (since $-1 \leq P(e_i, e_j) \leq 1$). This quantity, which is in turn a widely accepted valid dissimilarity measure in gene expression analysis, violates the triangle inequality constraint, and thus is not a metric in a strict sense. However, the square root of $\delta_{i,j}$ is proportional to the Euclidean distance between e_i and e_j (see (Clarkson, 2006)), and hence can be adopted within algorithms (such as FPF) designed for metric spaces. Our definition of $d_{i,j}$ is precisely this, i.e.,

$$d_{i,j} = \sqrt{\delta_{i,j}}.$$

2.3 The k-center problem

We approach the problem of clustering microarray data as the one of finding a solution to the k-center problem, defined as follows:

Given a set of points N on a metric space M, a distance function $d(p_1, p_2) \geq 0$ satisfying the triangle inequality, and an integer k, a k-center set is a subset $C \subseteq N$ such that $|C| = k$. The k-center problem is to find a k-center set that minimizes the maximum distance of each point $p \in N$ to its nearest center in C,i.e., minimizes the quantity $\max_{p \in N} \min_{c \in C} d(p, c)$.

The problem is known to be NP-hard. The optimum value is approximable within a factor 2 by FPF (Gonzalez, 1985), but not within a factor 2-ϵ, for any $\epsilon > 0$, unless $P = NP$ (Feder and Greene, 1988).

In our problem, N is represented as a $n \times m$ matrix, where n is the number of gene probes in the dataset and m is the number of conditions tested on each probe, the metric space M is $\mathcal{R}^m$ with the distance function $d_{i,j}$ defined above. Once the centers are defined, the natural induced cluster decomposition is obtained by associating each point to its closest center.

3 K-BOOST ALGORITHM

In this section, we present the K-boost algorithm which is essentially structured in two main steps. In the first phase it attempts to guess the hidden number of classes in which elements are divided in. After which, for each class, k-boost initializes a cluster containing its most representative element. In the second phase K-boost incrementally updates clusters by adding each of the remaining elements to the closest cluster, each time updating its centroid accordingly.

To determine the number k of clusters, k-boost extracts a small representative sample from the input set and obtains two partitions: the sample set and the target set. After that, both sets are clustered with an enhanced version of the *furthest-point-first* (FPF) algorithm (Geraci *et al.*, 2006). At each iteration FPF generates a new cluster for the sample and a new one for the target, and a stability-based technique ((Tibshirani *et al.*, 2005)) is used to compute the ability of the sample clustering to predict the target clustering (*prediction strength*). K is the dimension of the clustering correspondent to the first local maximum of the prediction strength values.

Before to enter in the details of K-boost we describe the enhanced FPF algorithm and our implementation of the stability-based method for determining k.

3.1 Furthest-Point-First clustering algorithm

FPF is based on a greedy approach: it increasingly computes the set of centers $C_1 \subset C_2 \subset \ldots \subset C_k$, where C_k is the solution to the problem. The first set C_1 contains only one randomly chosen point $c_1 \in N$. Each iteration i, with $1 \leq i \leq k-1$, has the set of centers C_i at its disposal and works as follows:

1. for each point $p \in N \backslash C_i$ compute its closest center c_p, i.e., the point c_p satisfying:
$$d(p, c_p) = \min_{c \in C_i} d(p, c);$$
2. determine the point $p \in N \backslash C_i$ that is farthest form its closest center c_p and let it be c_{i+1}; clearly, c_{i+1} satisfies:
$$d(c_{i+1}, c_{c_{i+1}}) = \max_{p \in C_i} d(p, c_p);$$
3. $C_{i+1} = C_i \cup C_{i+1}$.

At each iteration a new center is added to the set of centers that is being computed. The algorithm stops after $k-1$ iterations giving as result the set C_k. Observe that, at step $i+1$, there is no need to recalculate the distance of p to all centers, but just the distance $d(p, c_i)$, the distance to the unique center c_i added during the previous iteration. Then, just compare this distance with $d(p, c_p)$, the minimum distance to centers of C_i. According to the result of this comparison, c_p can be updated or not. Hence, if for each p the value $d(p, c_p)$ is stored, then each iteration can be executed in $O(n)$ space and a k-center set can be computed in $O(kn)$ distance computations.

To actually compute a clustering associated to such a k-center set, N is simply partitioned into k subsets $N_1, \ldots, N_k$, each corresponding to a center in C_k and such that

$$N_j = \{p \in N | c_p = c_j\} \quad (1)$$

In other words, the cluster N_j is composed of all points for which c_j is the closest center, for each $j = 1, \ldots, k$. Here we use the FPF algorithm with the technique improvement described in (Tibshirani *et al.*, 2005). Taking advantage of the triangle inequality, the modified algorithm avoids considering points that cannot change their closest center. To this aim, at each iteration i we maintain, for each center $c_j \in C_i$, the set N_j of points for which c_j is the closest center, defined as in (1), for $j = 1, \ldots, i$ (i.e., we build the clusters associated to intermediate solutions). We store the points in N_j in order of decreasing distance from c_j. When we scan the set of points to find the closest center, we follow the order given by the N_j's: given $p \in N_j$, with $1 \leq j < i$, if $d(p, c_i) \leq \frac{1}{2} d(c_j, c_i)$ then we stop scanning N_j, as there cannot be any other point closer to c_i than to c_j. The distances between centers must be stored, requiring additional $O(k^2)$ space. As a consequence, storage consumption is linear in n only provided that $k \leq \sqrt{n}$.

3.2 Stability-based technique

The FPF algorithm must be fed with the number k of clusters into which N has to be partitioned. To appropriately estimate this number, here we use an efficient variant of the prediction strength method developed by Tibshirani et al. (Tibshirani *et al.*, 2005). Here we briefly describe the original prediction strength method, and in next section we give details of how we embed it in k-boost. To obtain the estimate, the method proceeds as follows. Given the set N of n elements, randomly choose a sample N_r of cardinality η. Then, for increasing values of t ($t = 1, 2, \ldots$) repeat the following steps: (i) using the clustering algorithm, cluster both $N_{ds} = N \backslash N_r$ and N_r into t clusters, obtaining the partitions $C_t(ds)$ and $C_t(r)$, respectively; (ii) measure how well the t-clustering of N_r predicts co-memberships of mates in N_{ds} (i.e., count how many pairs of elements that are mates in $C_t(ds)$ are also mates according to the centers of $C_t(r)$).

Formally, the measure computed in step (ii) is obtained as follows. Given t, clusterings $C_t(ds)$ and $C_t(r)$, and elements e_i and e_j belonging to N_{ds}, let $D[i, j] = 1$ if e_i and e_j are mates according to both $C_t(ds)$ and $C_t(r)$, otherwise $D[i, j] = 0$. Let $C_t(ds) = C_{t,1}(ds), \ldots, C_{t,t}(ds)$, then the *prediction strength* $PS(t)$ of $C_t(ds)$ is defined as:

$$PS(t) = \min_{1 \leq \ell \leq t} \frac{1}{\#\text{pairs in } C_{t,\ell}(ds)} = \sum_{\substack{i,j \in C_{t,\ell}(ds) \\ i<j}} D[i, j] \quad (2)$$

where the number of pairs in $C_{t,\ell}(ds)$ is given by its binomial coefficient over 2. In other words, $PS(t)$ is the minimum fraction of pairs, among all clusters in $C_t(ds)$, that are mates according to both clusterings, hence $PS(t)$ is a worst case measure. The procedure outlined above terminates at the largest value of t such that $PS(t)$ is above a given threshold, setting k equal to such t.

3.3 k-boost clustering algorithm

In our implementation, we extract from N a random sample N_r by fixing the parameter η to $\sqrt{n}$. Then we run the FPF algorithm on N_r until $k = \eta$. This makes each obtained cluster contains just the center of the cluster. Note that, if one consider the order in which centers are extracted by FPF, this clustering operation is equivalent to rank points $N_r = \{N_{r,1}, \ldots, N_{r,\eta}\}$ such that $N_{r,i}$ is the center made at the i-th iteration of FPF. The computational

cost of this procedure is $O(\eta^2)$. For convenience from here we denote as N_r^t the set of the first t elements in the ranked N_r.

We then run FPF heuristic with input set $N_{ds} = N \backslash N_r$. Suppose at step t we have computed the clusters $C_{t,1}(ds), \ldots, C_{t,t}(ds)$, and suppose, for each $e \in N_{ds}$, we keep the index $i = i(e,t)$ of its closest point in N_r^t. Such index can be updated in constant time by comparing $d(e, N_{r,i(e,t-1)})$ with $d(e, N_{r,t})$, i.e., the distance of e from the "current" closest point in N_r^{t-1} and that to the point $N_{r,t}$.

Now, for each $C_{t,\ell}(ds)$, $\ell = 1, \ldots, t$, we can easily count in time $O(|C_{t,\ell}(ds)|)$ the number of elements that are closest to the same point $N_{r,j}$, $j = 1, \ldots, t$, and finally compute the summations in formula (2) in time $O(|N_{ds}|)$.

Differently from the original stability-based technique, we stop this procedure at the first value of t such that $PS(t) < PS(t-1)$ or when $t = \sqrt{n}$ and set $k = t - 1$.

After the last iteration, the simplest solution to obtain the clustering of N could be associate the points in N_r to their closest centers in $C_k(ds)$. This is what we did in (Geraci *et al.*, 2007). Despite $C_k(ds)$ give a good prediction of the value of k, the main disadvantage of this approach is that both the centers of N_{ds} and N_r give rise to a still not enough good clustering. This is due to the fact the use of centers for microarray data is not the best possible choice. We experimentally observed that *centroids* can be more representative than centers in this data domain. The concept of centroid is well known in the information retrieval community. In this content a centroid is a vector with as many components as the number of probes that genes have in the microarray. The i-th component of the centroid of cluster c_j is the average of the values of the i-th probe of the genes in c_j. Thus we modified the approach of (Geraci *et al.*, 2007) to take advantage from this observation. We create a k-clustering by initializing a new cluster with an element of N_r^k. Each element in $N \backslash N_r^k$ is assigned to the cluster with closest centroid one at time, then centroid is updated accordingly. Note that the update of the centroid can be made in constant time without recompute the mean over all the elements of the cluster, since, given a set $P = \{p_1, \ldots, p_n\}$ of n probes we have:

$$\frac{\sum_{i=1}^{n} p_i}{n} = \frac{\sum_{i=1}^{n-1} p_i}{n-1} \frac{n-1}{n} + \frac{p_n}{n} \tag{3}$$

Thus this last clustering operation can be done in time $(n - \sqrt{n})k = O(nk)$. The overall cost of k-boost procedure is $O(n + 2k(n - \sqrt{n})) = O(nk)$.

3.4 Experiments

3.4.1 Evaluation We performed experiments on data sets derived from yeast and other species. We compare *K-Boost* (and also FPF-SB or FPF when appropriate) with some of the most used and robust clustering algorithms for microarray gene expression data, namely CLICK, *k*-means, and SOM. The CLICK, k-means and SOM algorithms have been run under EXPANDER[1] (EXpression Analyzer and DisplayER) (Sharan *et al.*, 2003), a java-based tool for analysis of gene expression data, that is capable of clustering and visualizing the correspondent results. Among the clustering algorithms used for comparison, CLICK is the only one that does not need to know the number of clusters in advance, while both k-means and the basic FPF need the value of k as input. The SOM method requires the grid dimension (not always corresponding to the required number of clusters[2]). Since *K-Boost* and CLICK usually suggest a different value of k, we run experiments in tables 3 and 4 with both values of k.

In order to assess the validity of our method with an external measure we evaluate the clusterings by means of the z_{score} computed by ClusterJudge tool (Gibbons and Roth., 2000), also available on line[3]. This tool scores yeast (Saccharomyces cerevisiae) genes clusterings by evaluating the mutual information between a gene's membership in a cluster, and the attributes it possesses, as annotated by the Saccharomyces Genome Database (SGD[4]) and the Gene Ontology Consortium[5]. In particular, ClusterJudge first determines a set of gene attributes among those provided by Gene Ontology, that are independent and significant; then it computes the mutual information of the proposed clustering and that of a reference random clustering. Finally it returns the z_{score}:

$$z_{\text{score}} = \frac{MI_{\text{real}} - MI_{\text{random}}}{\sigma_{\text{random}}},$$

where MI_{random} is the mean of the mutual information score for the random clustering used, and σ_{random} is the standard deviation. The higher the z_{score} the better the clustering. Given the randomized nature of the test, different runs produce slightly different numerical values, however the ranking of the methods is stable and consistent across different applications of the evaluation tool. For this reason, for each data set used we repeated three times the evaluation of the output of all the different algorithms, here reporting the average z_{score}. Even if the ClusterJudge methodology is available only for yeast genes, it is independent of both the algorithm and the metric used to produce the clustering, and thus is in effect validating both choices.

. As an internal measure of quality we use *homogeneity* and *separation* as defined in (Sharan *et al.*, 2003). More precisely, calling M the set of indices of points forming unordered mate pairs, the *average homogeneity* is

$$H_{ave} = \frac{1}{|M|} \sum_{(i,j) \in M} P(e_i, e_j).$$

The *average separation* is:

$$S_{ave} = \frac{1}{\binom{n}{2} - |M|} \sum_{(i,j) \notin M} P(e_i, e_j).$$

Both homogeneity and separation have values in the range [-1,1]. Higher value of homogeneity indicate higher quality. Lower values of separation indicate higher quality. Note that singletons do not contribute to the average homogeneity, but do contribute to the separation. Since both measures are greatly influenced by the number of clusters, they are most significant in comparing solutions having the same value of k.

We tested our algorithm on some of the most used yeast datasets in literature (Cho *et al.*, 1998; Eisen *et al.*, 1998; Spellman *et al.*, 1998) and our results show that, on the average, we achieve a better score than that obtained by the other clustering algorithms, while using far fewer resources, especially time.

[1] http://www.cs.tau.ac.il/ rshamir/expander.

[2] Note, for example, that in Table 3 the grid for Spellman et al. dataset was set to 9×3 but the execution returned only 18 clusters instead of 27.

[3] http://llama.med.harvard.edu/cgi/ClusterJudge/cluster_judge.pl.

[4] http://www.yeastgenome.org.

[5] http://www.geneontology.org.

3.4.2 Datasets for yeast The algorithms were tested on three well studied yeast datasets. The first is the yeast cell cycle dataset described by Cho et al. in (Cho *et al.*, 1998). In their work the authors monitored the expression levels of 6218 Saccharomyces cerevisiae putative gene transcripts (ORFs). Probes were collected at 17 time points taken at 10 min intervals (160 minutes), covering nearly two full cell cycles[6]. The second dataset, described by Spellman et al. in (Spellman *et al.*, 1998), is a comprehensive catalog of 6178 yeast genes whose transcript levels vary periodically within the cell cycle (for a total of 82 conditions)[7]. The third dataset, described by Eisen et al. (Eisen *et al.*, 1998), contains 2467 probes under 79 conditions, and consists of an aggregation of data from experiments on the budding yeast Saccharomyces cerevisiae (including time courses of the mitotic cell division cycle, sporulation, and the diauxic shift)[8]. Table 1 summarizes the properties (number of probes and number of conditions) of the three datasets.

Dataset	**Cho et al.**	**Eisen et al.**	**Spellman et al.**
Probes	6601	2467	6178
Conditions	17	79	82

Table 1. Summary of dataset properties.

3.4.3 Experimental results The results reported here have been obtained on an 1.4 GHz AMD ATHLON XP workstation with 1.5 GB RAM running Linux kernel 2.6.11.4.

Experimental results are reported in Tables 2, 3 and 4. For each experiment we report (all or some of) the following parameters: the number of clusters *k* (either computed or fed as input), the number of singletons left out from the clusters, the computation time in seconds, the $z_{\rm score}$ (external measure), the homogeneity and separation (internal measures).

Note that CLICK is the only algorithm, among those that we have tested, that sometimes produces singletons in its clustering (136 in Cho et al. dataset, 17 in Spellman et al. dataset and none in Eisen et al. dataset) and put them into a single cluster labelled cluster zero. Hence, to correctly evaluate CLICK results we created a new cluster for each of these singletons..

In Table 2[9] we observe that *K-Boost* achieves a significantly better $z_{\rm score}$ on all three yeast datasets using far less computation time (by a factor from 5 to 10) than CLICK. FPF-SB is still faster but it attains lower $z_{\rm score}$ than *K-Boost*, though sometimes it beats CLICK. On larger data sets the time-gap is due to increase since CLICK asymptotically runs in $O(n^2m+n^3)$. The actual number of clusters computed by CLICK has little influence on its speed since the bulk of the cost is paid for in the set up and the initial iterations. In contrast *K-Boost* (and FPF-SB) has a lower asymptotic cost $O(nmk)$.

In two out of three cases CLICK and *K-Boost* make different choices as to the optimal value for k. A hint to the fact that our choice of k might be closer to the natural value for the data sets considered can be drawn comparing the $z_{\rm score}$ of k-means and SOM in Tables 3 and 4: when fed with *K-Boost*'s estimate they attain equal or higher $z_{\rm score}$ value. Note that internal measures are not suitable for such a comparison since both homogeneity and separation have drift due to the number of clusters.

In Table 3 we use for all five methods the value for k estimated by CLICK. For the data set of Cho et al *K-Boost* scores better than all the other methods. For the data set of Eisen et al. *K-Boost* and SOM tie in terms of $z_{\rm score}$, while *K-Boost* exhibit slightly better homogeneity. For the data set of Spellman et al. *K-Boost* and CLICK tie in terms of $z_{\rm score}$ and homogeneity.

In Table 4 (and Table 3 for the data set Eisen et al.) we can compare four algorithms (*K-Boost*, FPF-SB, k-means and SOM) when fed with the same estimate for k (computed by *K-Boost*). *K-Boost* attains better results in terms of $z_{\rm score}$ and homogeneity on all data sets (except for the $z_{\rm score}$ on Eisen et al. where SOM and *K-Boost* tie). On the the datas et of Cho et al. k-means come very close to *K-Boost* in terms of $z_{\rm score}$ but is far below in terms of homogeneity.

In terms of separation in tables 3 and 4 *K-Boost* exhibit often the best or second best (lower) value among the algorithms tested (with the exception of the data set Cho et al. in table 3 where CLICK and k-means and SOM are slightly better than *K-Boost*).

Overall, on the tree yeast data sets, when k is to be determined *K-Boost* is superior in terms of time and quality. In the case when the advantage of an educated guess is allowed (either using one from CLICK or one from *K-Boost*) *K-Boost* is always superior or ties with one of the competitors in one or two of the quality measures. Almost always *K-Boost* is significantly faster even for the relatively small data sets employed.

4 CONCLUSIONS

Efficient and effective analysis of large datasets from microarray gene expression data is one of the keys to time-critical personalized medicine. The issue we address here is the scalability and quality of the data processing software for clustering gene expression data into groups with homogeneous expression profile. In this paper we propose *K-Boost*, a new clustering algorithm that efficiently applies to microarray data analysis, being scalable to large datasets without sacrificing output quality.

In order to validate both the choice of the algorithm and the metric used to produce the clusterings we used ClusterJudge, an independent tool that only scores yeast (Saccharomyces cerevisiae) genes clusterings. Therefore, one of our future tasks will be to find methodologies for the evaluation of clusterings of gene datasets from other species (human, mouse, rat, etc.).

[6] The complete dataset, now containing 6601 Saccharomyces cerevisiae putative gene transcripts (ORFs), is available at http://genomics.stanford.edu/yeast_cell_cycle/cellcycle.html.

[7] The complete dataset and description are available at http://cellcycle-www.stanford.edu.

[8] The data is fully downloadable from http://genome-www.stanford.edu/clustering.

[9] We report the number of singletons generated by each clustering algorithm only in this table. This is because CLICK, in these sets of experiments, is the only algorithm tested generating singletons.

Dataset	Cho et al.				Eisen et al.				Spellman et al.			
Method	k	Sg	Time	z-score	k	Sg	Time	z-score	k	Sg	Time	z-score
K-boost	8	0	68	85.77	8	0	63	58.70	16	0	421	84.57
Click	30	136	660	62.77	8	0	240	41.40	27	17	4200	65.17
FPF-SB	8	0	12	70.53	8	0	15	53.93	16	0	94	54.53

Table 2. Experimental results comparing algorithms that compute the optimal k. The results shown are the average of three independent runs. For each algorithm and data set we report the number k of clusters, the number Sg of singleton data points, the running time in seconds, and the z_{score} computed by ClusterJudge.

Dataset	Cho et al.					Eisen et al.					Spellman et al.				
Method	k	Time	z-score	Hom	Sep	k	Time	z-score	Hom	Sep	k	Time	z-score	Hom	Sep
Kboost	30	41	72.63	0.696	0.000	8	63	60.97	0.547	-0.019	27	224	69.13	0.539	0.045
Click	30	660	59.70	0.681	-0.035	8	240	42.07	0.505	-0.174	27	4200	71.17	0.545	0.027
Kmeans	30	720	68.17	0.331	-0.019	8	120	27.63	0.390	0.121	27	1140	54.0	0.436	0.084
SOM	29*	300	64.40	0.329	-0.050	8	60	61.93	0.513	0.030	18*	240	49.73	0.477	0.077
FPF	30	22	50.13	0.638	0.005	8	15	55.57	0.514	-0.018	27	80	52.27	0.487	0.065

Table 3. Experimental results comparing algorithms that take k as input with the value computed by CLICK. The results shown are the average of three independent runs. For each algorithm and data set we report the number k of clusters, the number Sg of singleton data points, the running time in seconds, the z-score computed by ClusterJudge, Homogeneity and Separation. * denotes the number of clusters generated by SOM.

REFERENCES

Alon, U., Barkai, N., Notterman, D., Gish, K., Ybarra, S., Mack, D., and Levine, A. (1999). Board patterns of gene expression revealed by clustering analysis of tumoral and normal colon tissues probed by oligonucleotide array. In *Proc. of National Academy of Science of the United States of America*, pages 6745–6750.

Bar-Joseph, Z. (2004). Analyzing time series gene expression data. *Bioinformatics*, **20**(16), 2493–2503.

Belacel, N., Cuperlovic-Culf, M., Laflamme, M., and Ouellette, R. (2004). Fuzzy j-means and vns methods for clustering genes from microarray data. *Bioinformatics*, **20**(11), 1690–701.

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

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 transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, **2**, 65–73.

Clarkson, K. L. (2006). Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press.

Dugas, M., Merk, S., Breit, S., and Dirschedl, P. (2004). mdclust-exploratory microarray analysis by multidimensional clustering. *Bioinformatics*, **20**(6), 931–936.

Eisen, M., Spellman, P., Brown, P., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. In *Proc. of National Academy of Science of the United States of America*, pages 14863–14868.

Ernst, J., Naur, G., and Bar-Joseph, Z. (2005). Clustering short time series gene expression. *Bioinformatics*, **21**(1), i159–i168.

Feder, T. and Greene, D. (1988). Optimal algortihms for approximate clustering. In *Proc. of* 20^{th} *Annual ACM Symposium on Theory of Computing*, pages 434–444.

Gat-Viks, I., Sharan, R., and Shamir, R. (2003). Scoring clustering solutions by their biological relevance. *Bioinformatics*, **19**(18), 2381–2389.

Geraci, F., Pellegrini, M., Sebastiani, F., and Pisati, P. (2006). A scalable algorithm for high-quality clustering of web snippets. In *Proc. of the 21st Annual ACM Symposium on Applied Computing (SAC 2006)*, pages 0–0.

Geraci, F., Leoncini, M., Montangero, M., Pellegrini, M., and Renda, M. E. (2007). *PF-SB* : A scalable algorithm for microarray gene expression data clustering. In *HCI (12)*, volume 4561 of *Lecture Notes in Computer Science*, pages 606–615. Springer.

Gibbons, F. and Roth., F. (2000). Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Research*, **12**, 1574–1581.

Giurcaneanu, C., Tabus, I., Shmulevich, I., and Zhang, W. (2003). Stability-based cluster analysis applied to microarray data. In *Proceedings. Seventh International Symposium on Signal Processing and Its Applications 2003*, volume 2, pages 57–60. IEEE Press.

Gonzalez, T. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, **38**, 293–306.

Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. (2002). Co-clustering of biological networks and gene expression data. *Bioinformatics*, **18**(90001), 145S–154.

Hastie, T., Tibshirani, R., Eisen, M. B., Alizadeh, A., Levy, R., Staudt, L., Chan, W. C., Botstein, D., and Brown, P. (2000). 'gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology*, **1**, research00.

Huang, W. P. (2006). Incorporating biological knowledge into distance-based clustering analysis of microarray gene expression data. *Bioinformatics*, **22**(10), 1259–68.

J Taylor, R. T. (2006). A tail strength measure for assessing the overall univariate significance in a dataset. *Biostatistics*, **7**(2), 167–181.

Jiang, D., Tang, C., and Zhang, A. (2004). Cluster analysis for gene expression data: A survey. *IEEE Transaction on Knowledge and Data Engineering*, **16**(11), 1370–1386.

Madeira, S. and Oliveira, A. (2004). Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **1**, 24–45.

Ramoni, M., Sebastiani, P., and Kohane, I. (2002). Cluster analysis of gene expression dynamics. *Proc. Nat Acad Sci USA*, **99**(14), 9121–6.

Schadt, E., Edwards, S., GuhaThakurta, D., Holder, D., Ying, L., Svetnik, V., Leonardson, A., Hart, K., Russell, A., Li, G., Cavet, G., Castle, J., McDonagh, P., Kan, Z., Chen, R., Kasarskis, A., Margarint, M., Caceres, R., Johnson, J., Armour, C., Garrett-Engele, P., Tsinoremas, N., and Shoemaker, D. (2004). A comprehensive transcript index of the human genome generated using microarrays and computational approaches. *Genome Biol*, **5**(10), R73.

Shamir, R. and Sharan, R. (2002). *Current Topics in Computational Molecular Biology*, chapter Algoritmic Approaches to Clustering Gene Expression Data, pages 269–299. MIT Press.

Sharan, R., Maron-Katz, A., and Shamir, R. (2003). Click and expander: A system for clustering and visualizing gene expression data. *Bioinformatics*, **19**(14), 1787–1799.

Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., and Futcher, B. (1998). Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, **9**(12), 3273–97.

Tamayo, P., Slonim, D., Mesirov, J., Zhu, Q., Kitareewan, S., Dmitrovsky, E., Lander, E. S., and Golub, T. R. (1999). Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. In *Proceedings of National Academy of Science of the United States of America*, pages 2907–2912.

Tanay, A., Sharan, R., and Shamir, R. (2006). *Handbook of Computational Molecular Biology*, chapter Biclustering Algorithms: A Survey. Chapman and Hall / CRC Press.

Dataset	**Cho et al.**					**Spellman et al.**				
Method	k	Time	z-score	Hom	Sep	k	Time	z-score	Hom	Sep
K-boost	8	68	109.00	0.601	-0.085	16	421	83.17	0.519	0.031
FPF-SB	8	12	71.50	0.546	-0.075	16	94	49.83	0.463	0.049
K-means	8	420	104.17	0.264	-0.274	16	900	51.23	0.406	0.069
SOM	8	180	90.4	0.272	-0.079	16	420	54.37	0.467	0.091

Table 4. Experimental results comparing algorithms that take k as input with the value computed by K-boost. Click is not listed since we cannot force a given value of k. The experiments for the Eisen et al. data set coincide with those in Table 3. The results shown are the average of three independent runs. For each algorithm and data set we report the number k of clusters, the number Sg of singleton data points, the running time in seconds, the z-score computed by ClusterJudge, Homogeneity and Separation.

Tavazoie, S., Hughes, J., Campbell, M., Cho, R., and Church., G. (1999). Systematic determination of genetic network architecture. *Nature Genetics*, **22**, 281–285.

Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **63**(2), 411–423.

Tibshirani, R., Walther, G., Botstein, D., and Brown., P. (2005). Cluster validation by prediction strength. *Journal of Computational & Graphical Statistics*, **14**, 511–528.

Trent, J. and Bexevanis, A. (2002). Chipping away at genomic medicine. *Nature Genetics (Suppl)*, page 426.

Wen, X., Fuhrman, S., Michaelsdagger, G. S., Carrdagger, D. B., Smith, S., Barker, J. L., , and Somogyi, R. (1988). Large-scale temporal gene expression mapping of central nervous system development. *Natl Acad Sci U S A*, **95**(1), 334–349.

Xing, E. and Karp, R. (2001). Cliff: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics*, **17**(1), 306–315.

Xu, Y., Olman, V., and Xu, D. (2002). Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees. *Bioinformatics*, **18**(4), 536–545.

Yeung, K., Haynor, D., and Ruzzo., W. (2001). Validating clustering for gene expression data. *Bioinformatics*, **17**(4), 309–18.