FPF-SB: A Scalable Algorithm for Microarray Gene Expression Data Clustering

Filippo Geraci[1,3], Mauro Leoncini[1,2], Manuela Montangero[1,2], 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)
[3] Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma 56 - 53100 Siena, (Italy)
filippo.geraci@iit.cnr.it, leoncini@unimo.it, montangero.manuela@unimo.it, marco.pellegrini@iit.cnr.it, elena.renda@iit.cnr.it

Abstract. 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 of the data processing software for clustering gene expression data into groups with homogeneous expression profile. In this paper we propose *FPF-SB*, a novel clustering algorithm based on a combination of the Furthest-Point-First (FPF) heuristic for solving the *k*-center problem and a stability-based method for determining the number of clusters *k*. Our algorithm improves the state of the art: it is scalable to large datasets without sacrificing output quality.

1 Introduction

Personalized Digital Human Modeling is one of the new challenges at the frontier of medical practice, medical research and computer science. The relatively recent family of microarray based technologies holds the promise of personalized diagnosis based on tracing the metabolism of cells of individual patients. However, several obstacles still lay on the path to exploiting the full potential of these technologies [23]. One issue is, for instance, the scalability of the data processing software for clustering gene expression data into groups with homogeneous expression profile. In this paper we tackle this problem by proposing *FPF-SB,* a clustering algorithm based on a combination of the Furthest-Point-First (FPF) heuristic for the *k*-center problem [11] and a stability-based method for determining the number of clusters *k* [22]. The experiments we report here demonstrate that our novel algorithm is scalable to large datasets without sacrificing output quality.

The FPF heuristic is known to attain a result that is within a factor two of the optimum clustering according to the *k*-center criterion. This theoretical guarantee, coupled with a small computational complexity and with a careful implementation, makes this algorithm an ideal candidate for attaining scalability without sacrificing

V.G. Duffy (Ed.): Digital Human Modeling, HCII 2007, LNCS 4561, pp. 606–615, 2007.
© Springer-Verlag Berlin Heidelberg 2007

quality. The FPF algorithm constructs the clusters incrementally (that is the *k*-clustering is obtained by a refinement of the (*k*-1)-clustering) but it needs to know the number *k* of clusters in advance. For this reason, we apply a stability-based technique for cluster validation by *prediction strength* [22] that guides the selection of the "best" number of clusters in which the dataset should be partitioned. Our experiments point out that FPF-SB does maintain the original FPF scalability properties.

We apply the FPF algorithm directly to the input real data, overcoming the limitations of a similar approach [7] in which only a restricted set of artificially generated data points is used (meant to uniformly cover the parametric space of possible experimental outcomes). This approach has been shown not to be scalable.

The scalability of our algorithm can find applications in a number of different settings, in terms of both dataset dimension and number of experiments that need to be performed (hence, running time). Concerning the former, 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 [17]). Such a technology, adapted towards the needs of personalized medicine, promises to be able to screen a vast range of different pathological conditions. FPF might be used in this context, where the great amount of data to be managed is one of the main issues for the existing techniques.

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.

State of the art. The papers by Eisen et al. [6], Alon et al. [1] and Wen et al. [24] 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.*, a general survey in [15]). Among the most popular approaches we can broadly find those "distance based" such as *k*-means [21], Self Organized Maps (SOM) [20], Hierarchical Agglomerative Clustering (HAC) [6], and several variants thereof. A second broad class of algorithms is graph-based (CLICK [18], CAST [3] and CLIFF [25] all use weighted min cuts (with variants among them)). Other large families are those models-based [16], fuzzy-logic-based [2], or based on principal component analysis (PCA) [12]. Among the main issues for clustering there are the problem of guessing the optimal number of clusters [22] and that of cluster validation [10][26]. In biological data analysis a further issue is to provide metrics supported by ad hoc external biological knowledge [14].

There is no clear winner in the area of clustering algorithms for gene expression data as any method has strong and weak points: high quality usually results into high computational costs, while high efficiency usually results into poor quality, in both cases affecting the scalability. In our approach we show that the two goals (quality and scalability) are not in contrast: high scalability need not entail lower quality.

Microarray for digital human modeling and personalized medicine. Research on Digital Human Modeling and Simulation[1] is expected to produce models of human biology at all relevant physical scales such as, among the others, proteins, cells, and tissues. Among the computational tools that will be required by digital human models, microarray algorithms will play an important role in order to, *e.g.,* analyze diseased versus normal tissues, profiling tumors and studying gene regulation during development [13]. Also, in case of disease modeling, data extraction and processing will have to be repeated many times (say, before and after a therapy or drug exposure), which is of course another reason for the availability of very efficient algorithms. In medical practice, the adoption of patients' molecular information can give a boost to personalized medicine, and the methods that will make this possible include gene variations and expression level analysis [27]. While cost effectiveness and the availability of the data processing facilities needed by personalized medicine may be a decade ahead from now [28], the coming years will likely see an increasing number of medical protocols based on patients genotype and gene expression level information.

2 Preliminaries

Clustering. Let $N=\{e_1,..,e_n\}$ a set of n vectors of m components, 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 a cluster $N_t\subseteq\tilde{N}$. Thus, mates are "similar" in a sense that may vary depending on the application.

Distance function. 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 (resp. different) if $d_{i,j}$ is small (resp. large). We use a distance measure based on the *Pearson Coefficient*, one of the most used in the context of gene expression microarray data clustering, defined as follows:

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

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

The Pearson Coefficient is a measure of similarity but it is not a distance. To come up with a measure suitable for the metric space method, we define $\delta_{i,j}= 1 - P(e_i,e_j)$, $0 \leq \delta_{i,j} \leq 2$. Even if $\delta_{i,j}$ is widely accepted as a valid dissimilarity measure in gene expression analysis, it still violates the triangle inequality constraint. However, since $d_{i,j}$, the square root of $\delta_{i,j}$, is proportional to the Euclidean distance between e_i and e_j [5], it can be used with algorithms that need a metric space.

[1] http://www.fas.org/dh/publications/white_paper.doc.

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, approximable within a factor 2 by FPF [11], and not approximable within a factor 2-ε for any ε >0 [8].

In our case, 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 R^m where $d_{i,j}$ is the distance measure defined above. We apply a variation of the FPF algorithm computing k by way of the *Stability-Based* method in [22].

3 Main Algorithm

In this section, we describe FPF-SB by first presenting a variant of the FPF algorithm and then our implementation of the stability-based method for determining k.

Clustering algorithm. FPF is based on a greedy approach: it increasingly computes the set of centers $C_1\subset\ldots\subset C_k$, 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 $p\in N\setminus C_i$ compute its closest center c_p, *i.e.*, c_p satisfies:

$$d(p,c_p)=\min_{c\in C_i} d(p,c)\,; \tag{2}$$

2. determine the point $p\in N\setminus C_i$ that is farthest form its closest center c_p and let it be c_{i+1}; *i.e.*, c_{i+1} satisfies:

$$d(c_{i+1}c_{c_{i+1}})=\max_{p\in N\setminus C_i} d(p,c_p); \tag{3}$$

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

$$N_j = \{ p \in N \mid c_p = c_j \}. \tag{4}$$

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 [22]. 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 mantain, 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 (4), 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 $d(p,c_i) \leq 0.5\ d(c_j,c_i)$ 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 $k \leq n^{1/2}$.

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 strenght method developed by Tibshirani et al. [22]. First we briefly describe the original prediction strength method, and then give details of our implementation. 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,...) repeat the following steps: (*i*) using the clustering algorithm, cluster both $N_{ds}=N \setminus N_r$ 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:

$$PS(t) = \min_{1 \leq \ell \leq t} \frac{1}{\#\, pairs\ in\ C_{t,\ell}(ds)} \sum_{\substack{i,j \in C_{t,\ell}(ds) \\ i<j}} D[i,j], \tag{5}$$

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 above outlined procedure terminates at the largest value of t such that $PS(t)$ is above a given threshold, setting k equal to such t.

In our implementation, we first run the FPF algorithm on Nr up to t= η, storing all the computed centers c1,…,cη. Such preprocessing costs O(η |Nr|) = O(η 2). We then run k-center with input set Nds. Suppose at step t we have computed the clusters

Ct,1(ds), ..., Ct,t(ds) and suppose, for each $e \in N_{ds}$, we keep the index i(e,t) of its closest center among c1,…,ct. Such index can be updated in constant time by comparing d(e,ci(e,t-1)) with d(e,ct), i.e., the distance of e from the "current" center and that to the new center c_t. Now, for each $C_{t,\ell}(ds)$, $\ell = 1,...,t$, we can easily count in time $O(|C_{t,\ell}(ds)|)$ the number of elements that are closest to the same center c_j, j=1,…, t, and finally compute the summations in formula (5) in time $O(|N_{ds}|)$. After the last iteration, we obtain the clustering of N by simply associating the points $c_1,\ldots,c_\eta$ to their closest centers in $C_k(ds)$. The overall cost of our modified procedure is $O(\eta^2 + k(n- \eta) + k\,\eta) = O(kn)$ for $\eta = O(n^{1/2})$. Note that, differently from the original technique, we stop this procedure at the first value of t such that $PS(t) < PS(t-1)$ and set $k=t-1$. We empirically demonstrate that our choice of termination condition gives good results.

4 Experiments

Evaluation. We performed experiments on yeast datasets and other species. We compare our algorithm with some of the most used and valid clustering algorithms for microarray gene expression data, namely CLICK, *k*-means, SOM, and the basic FPF algorithm. CLICK, *k*-means and SOM algorithms have been run under EXPANDER[2] (EXpression Analyzer and DisplayER) [18], 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 *k* à priori, and SOM requires the grid dimension (not always corresponding to the required number of clusters[3]). Since FPF-SB and CLICK usually suggest a different *k*, we run *k*-means, SOM and FPF with both values of *k*.

In order to assess the validity of our method we evaluate the clusterings by means of the *z*-score computed by ClusterJudge tool [10], also available on line[4]. 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)[5] and the Gene Ontology Consortium[6]. 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 $(MI_{real} - MI_{random})/\sigma_{random}$, 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

[2] http://www.cs.tau.ac.il/~rshamir/expander.
[3] Note, for example, that in Table 2 the grid for Spellman et al. dataset was set to 9x3 resulting in only 18 clusters instead of 27.
[4] http://llama.med.harvard.edu/cgi/ClusterJudge/cluster_judge.pl.
[5] http://www.yeastgenome.org.
[6] http://www.geneontology.org.

numerical values, however the ranking of the method is stable and consistent across different applications of the evaluation tool. For this reason, for each dataset 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.

We tested our algorithm on some of the most used yeast datasets in literature [4,6,19] 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.

Datasets. The algorithm was tested on three well studied yeast datasets. The first is the yeast cell cycle dataset described by Cho et al. in [4]. In their work the authors monitored the expression levels of 6,218 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[7]. The second dataset, described by Spellman et al. in [19], is a comprehensive catalog of 6178 yeast genes whose transcript levels vary periodically within the cell cycle (for a total of 82 conditions) [8]. The third dataset, described by Eisen et al. [6], 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)[9].

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.

Table 1 summarizes the properties (number of probes and number of conditions) of the three datasets, while experimental results are reported in Tables 2 and 3. For each experiment we report the number of clusters k, the computation time in seconds and the value of the z-score.

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 labeled cluster zero. Hence, to correctly evaluate CLICK results we created a new cluster for each of these singletons.

Concerning Table 2, we observe that FPF-SB achieves a better z-score on all three datasets using far less computation time. The first two rows of the table show the results obtained with FPF-SB and CLICK, the two algorithms that do not need k as input. It can be seen that the two algorithms make a very different choice of k, except for the Eisen et al. dataset. The results show that FPF-SB outperforms CLICK, both in terms of actual computational time (in seconds)[10] and z-score, even when the guess of k is the same. FPF-SB outperforms all the other algorithms when fed with CLICK's guess of k.

---

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

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

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

[10] As expected, since CLICK asymptotically runs in $O(n^3)$.

Table 1. Summary of dataset properties

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

Table 2. Experimental results comparing FPF-SB with algorithms that need k as input parameter (using CLICK's guess of k)

Dataset	**Cho et al.**			**Spellman et al.**			**Eisen et al.**		
Algorithm	***k***	**Time**	***z*-score**	***k***	**Time**	***z*-score**	***k***	**Time**	***z*-score**
FPF-SB	8	12	77	16	94	61.6	8	15	62.9
CLICK	30	660	52.3	27	4200	52.9	8	240	38
***k*-means**	30	720	66.3	27	1140	52.5	8	120	34.2
SOM	29	300	54.9	18	240	49.1	8	60	59
FPF	30	22	46.8	27	80	47.9	8	7	62.9

Table 3. Experimental results comparing FPF-SB with algorithms that need k as input parameter (using FPF-SB's guess of k)

Dataset	**Cho et al.**			**Spellman et al.**			**Eisen et al.**		
Algorithm	***k***	**Time**	***z*-score**	***k***	**Time**	***z*-score**	***k***	**Time**	***z*-score**
FPF-SB	8	12	77	16	94	61.6	8	15	62.9
***k*-means**	8	420	92.5	16	900	63	8	120	34.2
SOM	8	180	75.9	16	420	52.1	8	60	59
FPF	8	5	77	16	43	61.6	8	7	62.9

The results reported in Table 3 show that, with the number of clusters suggested by FPF-SB, the quality of the clusterings computed by k-means and SOM, as judged by the z-score, is always better than in the previous case. This suggests that our implementation of the stability-based method produces a good guess of k in a computationally efficient way. Note also that we can evaluate the time spent by FPF-SB for the computation of k by comparing its running time with that of the basic FPF algorithm when computing the same number of clusters. Empirically, we evaluated this time as 55% of the total time on the average. However, the overall computation time is still much less than that of all the other algorithms.

The results also show the scalability of FPF and FPF-SB in terms of actual performance. Note that the size of a dataset is correctly measured as the product $n \cdot m$ (the dimension of the point space where the clusterings are computed). The actual running time is thus proportional, to first order, to the product $m \cdot n \cdot k$. It can be easily seen that the increase in running time of the Spellman et al. and Eisen et. al. datasets over the Cho et al. (the smallest one) can be quite accurately predicted using the ratio of the corresponding asymptotic costs, plus the time needed to compute k (the

stability-based part). For instance, FPF-SB prediction for Spellman et al. dataset is 96 seconds[11].

Moreover, our experiments give us a way to estimate the multiplicative constant hidden by the big-O notation and, thus, make a reliable prediction of the running time of FPF and FPF-SB in actual seconds, for any given dataset. This constant can be obtained by taking the maximum ratio, among all datasets, between the measured time (expressed in μsecs) and $m \cdot n \cdot k$. We obtained the constants 13.4 for FPF-SB and 6.5 for FPF. Hence, given any dataset consisting of n genes and m conditions, we can estimate (an upper bound to) the time needed by FPF-SB (resp. FPF) to compute a k-clustering in 13.4 x 10^{-6}x $m \cdot n \cdot k$ seconds (resp. 6.5 10^{-6}x $m \cdot n \cdot k$ seconds) [11].

5 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 of the data processing software for clustering gene expression data into groups with homogeneous expression profile. In this paper we propose *FPF-SB*, 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.).

References

[1] Alon, U., et al.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Natl Acad Sci USA 96(12), 6745–6750 (1999)

[2] Belacel, N., Cuperlovic-Culf, M., Laflamme, M., Ouellette, R.: Fuzzy J-Means and VNS methods for clustering genes from microarray data. Bioinf. 20(11), 1690–1701 (2004)

[3] Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. J Comput Biol. 6(3-4), 281–297 (1999)

[4] Cho, R.J., et al.: A genome-wide transcriptional analysis of the mitotic cell cycle. Mol Cell. 2(1), 65–73 (1988)

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

[6] Eisen, M.B., Spellman, P.T., Browndagger, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. PNAS 95(25), 14863–14868 (1998)

[7] Ernst, J., Naur, G.J., Bar-Joseph, Z.: Clustering short time series gene expression. Bioinf. 21(1), i159–i168 (2005)

[11] Note that these are our workstation estimation.

[8] Feder, T., Greene, D.H.: Optimal algortihms for approximate clustering. In: Proc. of 20th ACM Symposium on Theory of Computing, pp. 434–444 (1988)
[9] Geraci, F., Pellegrini, M., Sebastiani, F., Pisati, P.: A Scalable Algorithm for High-Quality Clustering of Web Snippets. In: Proc. of 21st ACM Symposium on Applied Computing 2006 (2006)
[10] Gibbons, F.D., Roth, F.P.: Judging the Quality of Gene Expression-Based Clustering Methods Using Gene Annotation. Genome Research 12, 1574–1581 (2000)
[11] Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theoretical Computer Science, pp. 293–306 (1985)
[12] Hastie, T., et al.: Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. Genome Biol. 1(2) (2000)
[13] Holloway, A.J., et al.: Options available - from start to finish - for obtaining data from DNA microarrays II. Nature Gen. Suppl. 32, 481–489 (2002)
[14] Huang, D., Pan, W.: Incorporating biological knowledge into distance-based clustering analysis of microarray gene expression data. Bioinf. 22(10), 1259–1268 (2006)
[15] Jiang, D., Tang, C., Zhang, A.: Cluster Analysis for Gene Expression Data: A Survey. IEEE Trans. on Knowledge and Data Eng. 16(11), 1370–1386 (2004)
[16] Ramoni, M.F., Sebastiani, P., Kohane, I.S.: Cluster analysis of gene expression dynamics. Proc. Nat Acad Sci USA 99(14), 9121–9126 (2002)
[17] Schadt, E.E., et al.: A comprehensive transcript index of the human genome generated using microarrays and computational approaches. Genome Biology 5(10), R73 (2004)
[18] Sharan, R., Maron-Katz, A., Shamir, R.: CLICK and EXPANDER: A System for Clustering and Visualizing Gene Expression Data. Bioinf. 19(14), 1787–1799 (2003)
[19] Spellman, P.T., et al.: Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization. Mol Biol Cell. 9, 3273–3297 (1998)
[20] Tamayo, P., et al.: Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. Proc. Natl Acad Sci USA 96(6), 2907–2912 (1999)
[21] Tavazoie, S., Hughes, J.D., Campbell, M.J., Cho, R.J., Church, G.M.: Systematic determination of genetic network architecture. Nature Genetics 22, 281–285 (1999)
[22] Tibshirani, R., Walther, G., Botstein, D., Brown, P.: Cluster validation by prediction strength. Journal of Computational & Graphical Statistics 14, 511–528 (2005)
[23] Trent, J.M., Bexevanis, A.D.: Chipping away at genomic medicine. Nature Genetics (Suppl), p. 426 (2002)
[24] Wen, X., et al.: Large-scale temporal gene expression mapping of central nervous system development. Proc. Natl Acad Sci USA 95(1), 334–349 (1988)
[25] Xing, E.P., Karp, R.M.: CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. Bioinf. 17(1), 306–315 (2001)
[26] Yeung, K.Y., Haynor, D.R., Ruzzo, W.L.: Validating clustering for gene expression data. Bioinf. 17(4), 309–318 (2001)
[27] WWW, Personalized Medicine Coalition, The case for Personalised Medicine http://www.personalizedmedicinecoalition.org
[28] WWW, The Royal Society, Personalised medicines: hopes and realities http://www.royalsoc.ac.uk