Gene clustering on the Unit Hypersphere with the Spherical K-means algorithm: coping with extremely large number of local optima

Nguyen Xuan Vinh
School of Electrical Engineering and Telecommunications
The University of New South Wales, Sydney 2052 Australia
BIOCOMP'08

Abstract

Normalizing gene expression profiles to zero mean and unit norm is a common normalization technique for microarray data. Working with this type of normalized data, the Spherical K-means algorithm (SPK-means) using the dot product similarity measure is a suitable gene clustering technique. Although it has been widely known that K-means type algorithms often converge to a local optimum, we noticed an extremely huge number of optima while clustering real microarray data sets using the Spherical K-means algorithm. e.g., running SPK-means 10,000 times with random seed initialization on a certain microarray data set with a moderate number of 423 genes, 7 experiments and 20 clusters can produce 10,000 different local optima. Since the local optima are very dense, we employ a simple Neighborhood Randomized Search technique as a post-processing step for SPK-means. This results in a two phases optimization algorithm called SPK-RNS. In the first phase, SPK-means with random seed initialization is run a number of times to locate a fairly good solution. In the second phase, Randomized Neighborhood Search technique is used to further refine the obtained solution. To test the optimality of the final solution, we run the SPK-means algorithm 100,000 more times. Experimental results show that SPK-RNS often archives "highly optimized" solutions.

Availability: Our implementation of the SPK-means and SPK-RNS algorithm in Matlab is freely available via http://vthesniper.googlepages.com/clusteringontheunithypersphere. Original data sets can be downloaded from the web sites mentioned in the paper. Processed data sets are available via our web site.

Keyword: Microarray, Gene clustering, Spherical K-means algorithm, Randomized Neighborhood Search
Contact: n.x.vinh@student.unsw.edu.au

I. Introduction

Clustering is one of the most important data mining techniques used to extract useful information from microarray data. Microarray data sets can be either clustered by samples or by genes. In this research we focus on the gene clustering problem. The objective of gene clustering is to group genes with similar expression patterns together with the common belief that those genes often have similar functions, participate in a particular pathway or response to a common environmental stimulus [2]. Although hundreds of clustering algorithms exist, the very simple K-means and its variants remain among the most widely used algorithms for gene clustering by biologists and practitioners. This surprising fact may be attributed to its especial ease of implementation and use.

When microarray data are normalized to zero mean and unit norm, a variant of the K-means algorithm has been introduced by [6]. Since the data points are on a unit hypersphere, the algorithm was called the Spherical K-means algorithm (SPK-means). With the above normalization scheme, it can be seen that certain similarity and distance measure such as the Squared Euclidean distance, the correlation coefficient, the Pearson correlation coefficient and the cosine similarity essentially coincide and depend only on the dot product of the normalized gene profiles [5]. We call the unified similarity measure the dot product similarity for normalized data. The Spherical K-means algorithm seeks to find the K cluster centers on the unit hypersphere (K prespecified by the user) such that the total similarity is maximized. It is widely known that K-means type algorithms can either converge to a local optimum or a saddle point of the

objective function. In this paper we prove a sufficient condition for the Spherical K-means algorithm to converge to a strict local optimum. It turns out that in practice, almost all of the SPK-means runs will converge to a strict local optimum of the clustering objective function. To avoid being trapped in a local optimum, a traditional strategy is to run the SPK-means algorithm multiple times and then retain the best solution.

Although it is often reported that the number of local optima of the clustering problem might be large, while working with real microarray data sets, we notice that it is hardly to see any two runs of the SPK-means algorithm resulting in the same objective values. Surprised by this fact, we did extensive experiments about the possible number of local optima in real gene clustering problems. Experimental result showed that the number of local optima is not only large but it is an extremely huge number. In one of our previous works [11], we investigated the ability to globally solve the Spherical K-means clustering problem using concave programming. However with the problems where the number of optima is huge like this, global optimization techniques turn out to be quite inefficient.

Since the number of local optima is huge, it is possible that close local optimizers will have close objective values. Departing from that conjecture, we employ a simple Randomized Neighborhood Search technique to further refine the solution obtained by SPK-means. This results in the so-called SPK-RNS, a two phase optimization algorithm. In the first phase we run the Spherical K-means algorithm for a number of times to explore a large region of the feasible domain and to locate a fairly good initial solution. In the second phase, the best solution from phase 1 is often quickly improved using a Randomized Neighborhood Search technique.

The rest of this paper is organized as follows. In section 2 we review the Spherical K-means algorithm using the dot product similarity measure for normalized data. We then prove a sufficient condition for the solution obtained by SPK-means to be a strict local optimizer. In section 3 we present our experimental result to give the readers an idea about the large number of local optima that SPK-means may encounter in a real gene clustering problem. SPK-RNS is presented in section 4 followed by experimental results that validate our method in section 5.

II. The Spherical K-means algorithm for normalized microarray data

A. The Spherical K-means algorithm

In the gene clustering problem, it is a common practice to group genes with similar expression patterns together. For this purpose the standard correlation coefficient has been found to work well since this statistic capture similarity in "shape" but place no emphasis on the magnitude of the two series of measurements [7]. The squared Euclidean distance however, does not generally perform very well on unnormalized data [9]. Gene microarray data are often L_2 normalized, i.e., scaling all the gene profiles to L_2 unit norm. In this case the squared Euclidean distance also works well. It is because that with normalized data, the squared Euclidean distance, the cosine similarity measure and the standard correlation coefficient coincide and depend only on the dot product of the normalized gene profile vectors. If further more, each gene profile is shifted by its means before normalizing to unit norm, we also have the Pearson correlation coefficient equals to the dot product of the normalized gene profile vectors [5]. Thus for normalized microarray data sets, the dot product can be suitably chosen as the similarity measure for gene profiles, denoted by $\langle ., . \rangle$ throughout this paper.

In this research we use the term "normalized data" for microarray data sets with all the gene profiles normalized to unit L_2 norm. The mean shifting step can be either performed or not. We denote a microarray data set as $A = \{a_1, a_2, \ldots, a_n\}$ with each a_i is a normalized gene profile vector. The gene clustering problem using the dot product similarity measure for normalized data can either be formulated as a combinatorial optimization problem [6] or as a continuous optimization problem as follows [11]. We need to find the K cluster centers $x = (x_1; x_2; \ldots; x_k)$ on the unit hypersphere so that the following total

similarity objective function is maximized:

$$f(x) = f(x_1; x_2; \ldots; x_K) = \sum_{i=1}^{n} \max_{l=1..K} \langle a_i, x_l \rangle \quad (1)$$

Finding the optimal solution to the above problem is NP-complete. A heuristic K-means type algorithm has been introduced by [6] which iterates through the two steps:

1. The membership assignment step: each data point is assigned to the nearest cluster (based on the dot product similarity). Let X_h be the set of indices of genes that belong to the cluster with corresponding center x_h then $X_h = \{i : \langle a_i, x_h \rangle = \max_{l=1...K} \langle a_i, x_l \rangle\}$

2. The center adjustment step: the cluster centers are relocated to the normalized gravity center of its new members:

$$x_l = \frac{\sum_{i \in X_l} a_i}{\| \sum_{i \in X_l} a_i \|}, l = 1 \ldots K$$

The algorithm can be proceeded until no further adjustment is made, or can be terminated early when a prespecified tolerance has been met. It has been proved that the spherical K-means never decreases the value of the objective function (1) [6].

Several notes about our SPK-means algorithm implementation:

- *Termination condition:* Our SPK-means version used throughout this research is let to run until the end, i.e., there are no possible more changes in the cluster centers. When performing on real microarray data set with the number of genes ranging from 400 to 1000 (after gene filtering), our "naive" implementation of SPK-means on Matlab (without any special technique to optimize the code) is reasonably fast. It often terminates after 15-40 iterations, taking only a fraction of second to a few seconds on a Pentium IV 3.0Ghz computer.

- *Empty cluster:* during the run of SPK-means one of the clusters might become empty. There are several possible solutions for this problem. In our implementation, we choose to split the largest cluster into two clusters.

- *Initialization:* for the traditional K-means algorithm, the two most popular initialization methods are random partition and random seed. There exist also a dozen of other more carefully seeding techniques. As for the Spherical K-means algorithm, there was no previously formally studied initialization methods available so we chose random initialization. From our observation, random seed and random partition give comparable result over a large number of runs, however random partition requires additional computation load, especially when the number of data points is large and when we need to run the algorithm many times. Thus we finally chose to implement random seed initialization: K cluster centers will be chosen randomly from the data points (normalized genes profile vectors).

B. Optimality characterization

In this section we give a sufficient condition for the solution returned by SPK-means to be a strict local maximizer of the objective function (1). First, the following definitions are needed:

Definition II.1. Strict assignment: A data point is strictly assigned if it can be assigned to one and only one cluster. I.e, the similarity from it to the nearest cluster center is strictly larger than the similarity to any other cluster center.

Definition II.2. Strict solution: a solution returned by the Spherical K-means algorithm is strict if all the data points are strictly assigned.

The following result has been proved in [11]:

Lemma II.1. *A necessary optimality condition for a feasible solution* $x = (x_1, x_2, \ldots, x_K)$ *of* (1) *is that:*

$$x_l = \frac{\sum_{i \in X_l} a_i}{\| \sum_{i \in X_l} a_i \|}, l = 1 \ldots K$$

This lemma simply states that at optimality, each cluster center is located at the normalized gravity center of its members. A well known similar result exists for the Euclidean K-means clustering problem. [6] also showed that for each cluster, the total similarity is maximized if and only if the cluster center is located at the normalized gravity of mass of its members. Now we are ready to give a sufficient condition for the optimality of the solution obtained by SPK-means:

Lemma II.2. *If a solution x^* returned by the K-means algorithm is strict then it is a strict local maximizer of the clustering function* (1)

Proof: By definition of a local maximizer, we need to prove that there exists an $\varepsilon > 0$ such that $f(x^*) \geq f(x)$ for all $x \in (x^*, \varepsilon) = \{x | \parallel x - x^* \parallel < \varepsilon, \parallel x_l \parallel = 1, l = 1 \dots K\}$.

Intuitively this means that by moving the cluster centers a "tiny bit" around their current positions (but still on the hypersphere), the objective function will always be decreased. Now consider moving the cluster centers to obtain a new cluster center set $\{x_1^* + \varepsilon_1, x_2^* + \varepsilon_2, \dots, x_K^* + \varepsilon_K\}$. To ensure that the centers are still on the unit hypersphere, we need to normalized them to $\widetilde{x} = \{\widetilde{x_l} = (x_1^* + \varepsilon_1)/(\parallel x_1^* + \varepsilon_1 \parallel), l = 1 \dots K\}$. Since the move is very "tiny" we expect that there should be no change in the membership of the data points. Indeed this is true thanks to the strict solution condition.

Let's consider any data point a_i which is assigned to cluster X_l. Since a_i is strictly assigned, its similarity to the cluster centers satisfy: $\langle a_i, x_l^* \rangle > \langle a_i, x_j^* \rangle, \forall l \neq j$. Now since the $\{\varepsilon_i, i = 1 \dots K\}$ can be chosen arbitrarily small, we can easily choose them so that the following still hold: $\langle a_i, \widetilde{x_l} \rangle > \langle a_i, \widetilde{x_j} \rangle, \forall l \neq j$. Put another way we can always choose the radius of the ball $B(x^*, \varepsilon)$ so tiny such that there is no change in the membership of all the data points.

Now we check the objective value at the new cluster centers. As we have discussed above, for each cluster the total similarity from the cluster center to all its members will be maximized if the cluster center is located at the normalized center of mass of all its points which is a unique point. Here since one or all of the cluster centers have been moved out of the normalized gravity center, the objective is strictly decreased and hence x^* is a strict local maximizer. ■

Interested readers can derive a similar result for the Euclidean K-means clustering problem. In practice, since machine round off will prevent most of the cases where the distance from a point to two or more nearest cluster centers are exactly equal, it is expected that almost all the runs of the SPK-means will return a strict local maximizer of the objective function (1).

III. The number of local optima in a real microarray data set

Since the Spherical K-means will most often converges to a local optima, it is a common practice to run the algorithms many times with different initialization and then retain the best solution. While working with microarray data sets, we notice that SPK-means encounter an extremely huge number of local optima.

A. Yeast data set 1: Saccharomyces cerevisiae

This dataset was first used by [4]. It contains gene expression profiles of 6400 Saccharomyces cerevisiae yeast genes, taken in 7 time points during the myotic shift. The dimension of the data is thus 6400 x 7. We filtered genes with missing values and genes with low absolute expression values to obtain a final data set containing 423 genes. Genes profiles are then normalize to zero mean and unit norm. The original data set is freely available in the demo section of the Bio-informatics toolbox for Matlab (version 7.0) or can be downloaded from http://www.broad.mit.edu.

B. Method and result

Since the true number of clusters is unknown, we assigned K several values in the range [5,30]. For each value of K, we ran the SPK-means algorithm with random seed initialization 10,000 times. The

TABLE I
THE NUMBER OF LOCAL OPTIMA IN THE YEAST DATASET

Runs	N^o of cluster	N^o of local optima	
		K-means	SPK-means
10000	5	112	1966
10000	6	286	4523
10000	7	1049	6740
10000	8	2492	8179
10000	9	4178	9038
10000	10	5535	9499
10000	15	9600	9999
10000	20	9998	10000
10000	25	10000	10000
10000	30	10000	10000

TABLE II
THE NUMBER OF LOCAL OPTIMA IN THE YEAST DATASET FOR VARYING NUMBER OF CLUSTERS AND GENES.

Runs	Clusters	Number of genes		
		100	200	300
10000	5	1985	1642	2238
10000	6	3545	2898	4026
10000	7	5033	4898	5852
10000	8	6528	6906	7355
10000	9	7985	8427	8531
10000	10	9244	9470	9433
10000	15	10000	10000	10000
10000	20	10000	10000	10000
10000	25	10000	10000	10000

obtained objective values are stored in an array. The array is then sorted and the different objective values are counted. The difference between two successive different optima in the sorted array ranges from 10^{-4} to 10^{-6} which is very well under the double precision capacity of Matlab. We also tested the Euclidean K-means algorithms (using the implementation from the Matlab Statistic toolbox) on the unnormalized yeast data. From table I it can be seen that both K-means and SPK-means encountered an extremely huge number of local optima, with the number of local optima encountered by SPK-means tends to be higher than that encountered by K-means. We counted the number of non-strict solutions of SPK-means for each value of K. As predicted above, this number is only a few per 10,000 runs, thus is negligible.

To find the connection between the number of data points, the number of clusters and the possible number of local optima, we picked the first 100, 200 and 300 genes in the datat set, gave K several value in [5,25] and run the SPK-means algorithm 10,000 times for each experiment scenario. The result is reported in table II.

As can be expected, the number of local optima encountered by SPK-means generally increases along with the increment in the number of data points and the number of clusters. Since this number is very large even for the moderate number of cluster $K = 10$, it is very unlikely that any two runs of those algorithms will produce identical results. We shall see in section 5 that the number of local optima is still much larger given a sufficiently large number of SPK-means runs.

IV. SPK-RNS, A TWO PHASES ALGORITHM

A. Algorithms review

Since SPK-means is a local optimization algorithm, it will be trapped in various local optima of the objective function. Integrating SPK-means with a global optimization scheme will help to improve the solution. In [11] we formulated the SPK-means clustering problem as a Concave Programming problem and proposed a conical Branch and Bound algorithm. This deterministic global optimization algorithm has certain merits such as it guarantees an $\epsilon-$global suboptimal solution in a finite number of steps, however

the extremely rugged patterns of the objective function makes Branch and Bound quite inefficient and as a result, cannot be fruitfully applied to real gene clustering problems. Several other stochastic global optimization schemes such as Genetic algorithm, Evolutionary algorithm, Simulated Annealing and Tabu search might also be employed. Those algorithms guarantee global optimal solution in a probabilistic sense.

The main common point of the global optimization algorithms mentioned above, deterministic or probabilistic, is that they require a certain level of expertise to properly implement and fine tune the parameters. Such difficulties might hinder the wide spread use of the algorithms within a broader biology community. A simpler approach which is often chosen in practice is to run SPK-means for a number of times and then retain the solution with the highest objective value. Generally the more number of runs, the better chance we obtain a better solution. However with a limited amount of time, running random SPK-means alone would not be the most effective way. We shall see in the next section that a simple post processing step often improves efficiency especially for problems with very large number of local optima.

B. SPK-RNS

As can be seen in the previous experiment, the clustering objective function for a real gene clustering problem is extremely rugged with a huge number of local optima. This special structure should be taken into account when designing efficient clustering algorithms. In this section we present a simple Randomized Neighborhood Search technique to exploit such structure. This results in the following two phases SPK-RNS algorithm.

The SPK-RNS algorithm:

Phase 1: SPK-means. The SPK-means algorithm is run for $N1$ times with random seed initialization. The K initial cluster centers are randomly chosen from the set of normalized gene profiles. The best solution obtained x^* in phase 1 is retained for further refinement in phase 2.

Phase 2: Randomized neighborhood search. In this phase a large number of random seeds $\widetilde{x}$ for SPK-means will be generated around the current best solution x^* as follows: $\widetilde{x} = \{\widetilde{x}_l = x^*_l + \delta.r, l = 1 \dots K\}$ where r is a random vector with each element uniformly distributed in the range [-0.5,0.5]. δ is the radius parameter which define the "radius" of the neighborhood area. The vectors $\widetilde{x}_l$ are then normalized to unit norm and SPK-means is started from this seed. A counter is set up. If any better solution is detected, the current best solution is updated, the counter is reset and the process continues again around the new best solution. Phase 2 stops when the counter reach N_2, i.e., N_2 seeds has been generated around the current best solution without any improvement.

The rationale behind this algorithm is that we observe that when SPK-means with random initialization has located a fairly good solution, the improvement will become slower as the next runs will generally discover many optima but with lower objective value. As a result, the algorithm will soon run out of the prelocated time without discovering any better solution. More over, in the gene clustering problem, since the number of optima is huge, it is likely that close optimizers may have close objective value. Thus it is reasonable to look around the current best solution for possible better solutions. In SPK-RNS, we use phase 1 to locate a good area of the feasible domain where the objective function is generally higher than the other area and then use phase 2 to locally refine the solution within that good area. There are only three parameters for this simple algorithm namely $N1, N2$ and δ with very intuitive meanings.

V. Experimental result

A. Data sets

1) Yeast data set 2: This dataset, consists of 6220 gene expression collected at 17 times points, was first used by [3] to discover cell cycle-dependent gene expression. Following [10] the 90-minutes time point has been excluded. We then use a variation filter to filter out genes which do not show a relative

TABLE III
SPK-RNS ON YEAST DATA SET 1

Clusters	SPK			RNS			Test phase		
	Ave loop	Time	Obj	Ave loop	Time	Obj	Runs	Optima	Obj
10	15.68	196s	407.0824	6.03	190s	407.0850	100,000	83,183	-
15	16.41	266s	410.9673	7.75	107s	411.0213	100,000	99,961	-
20	15.53	323s	412.6402	6.13	247s	412.8166	100,000	100,000	-

TABLE IV
SPK-RNS ON THE RAT CNS DATA SET

Clusters	SPK			RNS			Test phase		
	Ave loop	Time	Obj	Ave loop	Time	Obj	Runs	Optima	Obj
4	9.68	11s	87.6025	3.02	5s	-	100,000	2,162	-
5	9.19	15s	90.9095	3.11	8s	90.9498	100,000	10,272	-
6	9.06	32s	93.2462	3.25	10s	-	100,000	34,892	-
7	8.02	36s	94.9720	3.58	13s	94.9781	100,000	65,561	-

change of x and an absolute change of y units, with (x,y)=(3,100). The final data set containing 1060 genes are then normalized to have zero mean and unit norm. The original data set can be downloaded from http://genomics.standford.edu.

2) Serum data: The serum data set was described and used in [8] containing the expression levels of 8613 human genes. Following [7], 517 genes whose expression levels change substantially across experiments have been chosen for further analysis. The original and filtered data set can be downloaded from http://genome-www.stanford.edu/serum/data.html.

3) Rat CNS: This data set containing 112 genes taken in 9 time points during rat central nervous system development can be download at http://faculty.washington.edu/kayee/cluster. For this small data set, the number of cluster has been previously reported to be from 4 to 6 ([12], [1]), thus in our experiments on this data set we assign the number of clusters K=[4,5,6,7].

B. Experiment scenario

We empirically set the parameters for SPK-RNS to $N1 = 500, N2 = 200$ and $\delta = 1/10$ throughout the experiments. Although N2 was set to 200, it can be seen later from the experimental result that phase 2 of the algorithm executes quite fast. The reason is that since the seeds generated in phase 2 are generally close to the optimum points, SPK-means would need fewer iterations to converge than it did in phase 1 of the algorithm. We reported the number of average SPK-means iterations in our experiments.

Since the true number of gene clusters is unknown, we assigned K several values in the range [10,20]. We call each combination of K and a dataset an "experiment scenario". To test the optimality of the

TABLE V
SPK-RNS ON THE YEAST DATA SET 2

Clusters	SPK			RNS			Test phase		
	Ave loop	Time	Obj	Ave loop	Time	Obj	Runs	Optima	Obj
10	18.39	358s	449.7272	12.92	247s	449.7521	100,000	92,059	-
15	16.96	398s	467.7945	4.90	93s	467.9875	100,000	100,000	-
20	15.27	446s	476.4643	6.70	301 s	476.9774	17,684	17,684	477.0733

TABLE VI
SPK-RNS ON THE HUMAN FIBROBLASTS SERUM DATA SET

Clusters	SPK			RNS			Test phase		
	Ave loop	Time	Obj	Ave loop	Time	Obj	Runs	Optima	Obj
10	15.00	233s	422.3469	7.39	124s	422.4672	100,000	99,982	-
15	14.92	315s	434.5683	10.21	98s	435.0351	100,000	100,000	-
20	13.65	413s	441.4696	13.40	423s	442.9482	100,000	100,000	-

obtained solution, we run a test phase with random SPK-means initialization until a better solution has been found or 100,000 runs has been performed.

For each experiment scenario we report the followings:

- SPK phase: Ave Loop (the average number of SPK-means iterations in each run starting from a random seed initialization), time and the best objective value obtained.

- RNS phase: Ave Loop (the average number of SPK-means iterations in each run starting from a seed generated using RNS technique), time and the best objective. If no better objective than the SPK phase were found, the field will be marked by '-'.

- Test phase: The number of actual run (maximum 100,000; this phase will be terminated early if a better solution than SPK-RNS were found), the number of different optima and the best objective value (marked by '-' if no better solution were found).

It can be observed that in all experiments, SPK-means either found a better solution but after a very large number of runs or more often, exceed the 100,000 runs limit without locating any better solution than the one found by SPK-RNS. The experiments further show that the number of local optima discovered is huge given sufficient runs, e.g., 100,000 different local optima in 100,000 runs for some scenarios. In our experiments, SPK-RNS often requires a few minutes to process a common real microarray data set. This is acceptable considering that microarray data mining is not of time critical application.

VI. Discussion and Conclusion

Throughout this paper we have, for the first time, empirically counted the number of local optima that might be encountered by a certain local optimization techniques such as K-means or SPK-means. It turned out that for a real gene clustering problem, the number of local optima is often extremely huge. For this special structure of the objective function, we proposed using a simple Randomized Neighborhood Search technique as a post processing step for the SPK-means algorithm. This results in the two phases SPK-RNS algorithm. In our opinion, the solution obtained by SPK-RNS is satisfactory for several reason:

- Even the global solution of this (mathematical) clustering problem does not necessary yield the true cluster structure. Thus a "highly optimized" solution like the one obtained with SPK-RNS is satisfactory for practical purpose.

- SPK-RNS is conceptually very simple, easy to implement and use. This might help it to gain wide acceptance from biologists and practitioners, just like the simple K-means algorithm which is nowadays still a popular data mining tools.

The SPK-RNS framework can be used in conjunction with the other more sophisticated initialization techniques, i.e., initialization for the SPK-means algorithm in phase 1. This will result in even more efficient algorithms.

Funding

This work is supported in part by the a University of New South Wales international support scholarship.

Acknowledgement

The author is grateful to Dr. Pablo Tamayo from the Broad Institute of MIT and Harvard for his help in implementing the gene filter.

References

[1] S. Bandyopadhyay, A. Mukhopadhyay, and U. Maulik, "An improved algorithm for clustering gene expression data," *Bioinformatics*, vol. 25, p. 25, 2007.

[2] D. P. Berrar, W. Dubitzky, and M. Granzow, *A Practical Approach to Microarray Data Analysis*. Kluwer Academic Publishers Boston, 2003.

[3] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis, "A genome-wide transcriptional analysis of the mitotic cell cycle," *Mol Cell*, vol. 2, no. 1, pp. 65–73, 1998.

[4] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278, no. 5338, pp. 680–686, 1997.

[5] I. S. Dhillon, E. M. Marcotte, and U. Roshan, "Diametrical clustering for identifying anti-correlated gene clusters," *Bioinformatics*, vol. 19, no. 13, pp. 1612–1619, 2003.

[6] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1-2, pp. 143 – 175, January-February 2001.

[7] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proc Natl Acad Sci U S A*, vol. 95, no. 25, pp. 14 863–14 868, 1998.

[8] V. R. Iyer, M. B. Eisen, D. T. Ross, G. Schuler, T. Moore, J. C. Lee, J. M. Trent, L. M. Staudt, J. Hudson, J., M. S. Boguski, D. Lashkari, D. Shalon, D. Botstein, and P. O. Brown, "The transcriptional program in the response of human fibroblasts to serum," *Science*, vol. 283, no. 5398, pp. 83–87, 1999.

[9] W. Shannon, R. Culverhouse, and J. Duncan, "Analyzing microarray data using cluster analysis," *Pharmacogenomics*, vol. 4, no. 1, pp. 41–52, 2003.

[10] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitareewan, E. Dmitrovsky, E. S. Lander, and T. R. Golub, "Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation," *Proc Natl Acad Sci U S A*, vol. 96, no. 6, pp. 2907–2912, 1999.

[11] N. X. Vinh, H. D. Tuan, and H. Tuy, "Clustering on the unit hypersphere via concave programming," in *Proceeding of the International Conference on Non Convex Programming: Global and Local approach*, Rouen, France (to be appeared), 12 2007.

[12] X. Wen, S. Fuhrman, G. S. Michaels, D. B. Carr, S. Smith, J. L. Barker, and R. Somogyi, "Large-scale temporal gene expression mapping of central nervous system development," *Proc Natl Acad Sci U S A*, vol. 95, no. 1, pp. 334–339, 1998.