PCluster: Probabilistic Agglomerative Clustering of Gene Expression Profiles

Nir Friedman
School of Computer Science & Engineering
Hebrew University
Jerusalem 91904, ISRAEL
nir@cs.huji.ac.il

Abstract

A central problem in analysis of gene expression data is *clustering* of genes with similar expression profiles. In this paper, I describe an hierarchical clustering procedure that is based on simple probabilistic model. This procedure clusters genes with respect to a target classification of conditions, so that genes that are expressed similarly in each group of conditions are clustered together.

1 Introduction

In recent years, technical breakthroughs in spotting hybridization probes and advances in genome sequencing efforts lead to development of *DNA microarrays*, which consist of many species of probes, either oligonucleotides or cDNA, that are immobilized in a predefined organization to a solid phase. By using DNA microarrays researchers are now able to measure the abundance of thousands of mRNA targets simultaneously [3, 8, 13]. Unlike classical experiments, where the expression levels of only a few genes were reported, DNA microarray experiments can measure *all* the genes of an organism, providing a "genomic" viewpoint on gene expression. As a consequence, this technology facilitates new experimental approaches for understanding gene expression and regulation [7, 11].

A central method in analysis of gene expression data is *clustering*; e.g., [1, 5, 9]. The goal of clustering is identify groups of genes with "similar" expression patterns. Clustering algorithm differ in their definition of similarity (e.g., Pearson correlation, Euclidean distance, etc.) and in how they use these measures. Another dimension is whether the algorithm returns one partition of genes, or a hierarchical partition that describes similarities at different scales.

In this note, I describe a simple clustering procedure that is based on probabilistic considerations. An important feature of this procedure is that it uses a likelihood function to determine clusters rather than a measure of similarity. Intuitively, a group of genes are clustered together if their measured expression values could have been sampled from the same stochastic source with a high probability.

A particular feature of the procedure is that the user specifies in advance a partition of the experimental conditions. Conditions within the same partition are viewed as indistinguishable by the algorithm. As such, a gene is described in terms of the distribution of values (mean and variance) it attains in each group of conditions. The ability to specify groups allows the clustering procedure to focus on the differences that are considered important by the user.

2 Scoring Method

We assume we receive a matrix of gene expression measurement $\mathcal{D} = \{e_{g,c} : g \in \mathsf{Genes}, c \in \mathsf{Conds}\}$ where Genes is a set genes and Conds is a set of conditions. For simplicity, we assume that there are no missing values in the matrix, although the analysis below can be easily extended to deal with missing values.

Suppose we have a partition $\mathcal{C} = \{C_1, \ldots, C_m\}$ of conditions in Conds and a partition $\mathcal{G} = \{G_1, \ldots, G_n\}$ of genes in Genes. We want to score the combined partition. For this purpose, we consider a generative probabilistic model that could have generated the data. The assumption of the model is if g and g' are in the same gene cluster, and c and c' in the same condition cluster, then the expression value $e_{g,c}$ and $e_{g',c'}$ are sampled from the same distribution.

Formally, we assume we assume that the likelihood function has the form:

$$\begin{aligned} L(\mathcal{G}, \mathcal{C}, \theta : \mathcal{D}) &= P(\mathcal{D} \mid \mathcal{G}, \mathcal{C}, \theta) \\ &= \prod_i \prod_k \left[\prod_{g \in G_i} \prod_{c \in C_k} P(e_{g,c} \mid \theta_{i,k}) \right] \end{aligned}$$

where $\theta_{i,k}$ are the parameters that describe the expression of genes in G_i in conditions in C_k.

Note that the likelihood is invariant to reshuffling of the expression values within each cell, since it is the product of the probability of each one of them, regardless of its exact position in the cell. Thus, suppose take $\mathcal{D}$ and create a new data set $\mathcal{D}'$ by reshuffling expression values of conditions in the same group according to $\mathcal{C}$ for each gene. It is easy to see that $L(\mathcal{G}, \mathcal{C}, \theta : \mathcal{D}) = L(\mathcal{G}, \mathcal{C}, \theta : \mathcal{D}')$ for any choice of $\mathcal{G}$ and θ. In other words, the likelihood term (and hence all scores that use it) is insensitive to differences between conditions that are in the same group according to $\mathcal{C}$. This feature allows the clustering to focus on genes whose expression patterns relates to the partition in $\mathcal{C}$.

A simple choice of parameterization for expression is using a Gaussian distribution. Thus, for each *cell* (i, k), we need to estimate a mean $\mu_{i,k}$ and a variance $\sigma^2_{i,k}$. Using the Gaussian parameterization, the likelihood has the form

$$L(\mathcal{G}, \mathcal{C}, \theta : \mathcal{D}) = \prod_i \prod_k \left[\prod_{g \in G_i} \prod_{c \in C_k} \frac{1}{\sqrt{2\pi\sigma^2_{i,k}}} e^{-\frac{1}{2\sigma^2_{i,k}}(e_{g,c} - \mu_{i,k})^2} \right]$$

To score a given partition, we can consider the *maximum likelihood* that we can achieve for it. Thus, we define

$$\mathrm{score}_{ML}(\mathcal{G}, \mathcal{C} \ : \ \mathcal{D}) = \max_\theta L(\mathcal{G}, \mathcal{C}, \theta : \mathcal{D})$$

We compute this score by plugging the maximum likelihood estimates for the mean and variance in each cell:

$$\begin{aligned} \mu_{i,k} &= \frac{1}{|G_i||C_k|} \sum_{g \in G_i} \sum_{c \in C_k} e_{g,c} \\ \sigma^2_{i,k} &= \frac{1}{|G_i||C_k|} \sum_{g \in G_i} \sum_{c \in C_k} (e_{g,c} - \mu_{i,k})^2 \end{aligned}$$

The maximum likelihood score measures how likely can we make the clustering seem if we optimize the parameters. This might be an overestimate of the likelihood of the clustering, since for each data we choose the best parameter sets.

There are many approaches to compensate for this overestimate. A simple one is the *Bayesian* approach. In this variant, we consider all possible parameterizations, and average the likelihood over all of them.

$$\mathrm{score}_B(\mathcal{G}, \mathcal{C} \ : \ \mathcal{D}) = \int P(\mathcal{D} \mid \mathcal{G}, \mathcal{C}, \theta) P(\theta \mid \mathcal{G}, \mathcal{C}) d\theta$$

where $P(\theta \mid \mathcal{G}, \mathcal{C})$ is the prior probability of the parameters for the given cluster conformation.

Although the integral required by the Bayesian score seems forbidding, when we make the proper assumptions, it can be computed in closed form. In our case, assume that

$$P(\theta \mid \mathcal{G}, \mathcal{C}) = \prod_i \prod_k P(\mu_{i,k}, \sigma^2_{i,k})$$

where $P(\mu_{i,k}, \sigma^2_{i,k})$ is a normal-gamma prior [2]. Under this assumption, the integral can be rewritten as a product of smaller integrals

$$\text{score}_B(\mathcal{G}, \mathcal{C} \ : \ \mathcal{D}) = \prod_i \prod_k \int \int \prod_{g \in G_i} \left[\prod_{c \in C_k} P(e_{g,c} \mid \mu_{i,k}, \sigma^2_{i,g}) \right] P(\mu_{i,k}, \sigma^2_{i,g}) d\mu_{i,k} d\sigma^2_{i,g}$$

Each of the double integrals is a *marginal likelihood* of a Gaussian distribution, and can be computed using a closed form formula [2].

In summary, I described two methods for scoring putative clusterings of the data, the maximum likelihood score, and the Bayesian score. A putative clustering receives a high score if the partitions of genes and conditions define cells with coherent distribution. If there is high within-cell variance, the clustering will be assigned a low score.

Both scores have additional attractive properties that will be crucial for the algorithmic section. First, as we can easily verify, both scores *decompose* into a product of cell-specific terms, each one measuring the "coherence" of the expression values in the cell G_i vs. C_k. Thus, for both scores, we can write

$$\text{score}(\mathcal{C}, \mathcal{G} \ : \ \mathcal{D}) = \prod_i \prod_k \text{LocalScore}(G_i, C_k \ : \ \mathcal{D})$$

where $\text{LocalScore}(G_i, C_k \ : \ \mathcal{D})$ is the *local score* of a particular cell. For the maximum likelihood score

$$\text{LocalScore}_{ML}(G_i, C_k \ : \ \mathcal{D}) = max_{\mu_{i,k}, \sigma^2_{i,k}} \prod_{g \in G_i} \prod_{c \in C_k} \frac{1}{\sqrt{2\pi\sigma^2_{i,k}}} e^{-\frac{1}{2\sigma^2_{i,k}}(e_{g,c} - \mu_{i,k})^2}$$

and for the Bayesian score, the local score is

$$\text{LocalScore}_B(G_i, C_k \ : \ \mathcal{D}) = \int \int \prod_{g \in G_i} \left[\prod_{c \in C_k} P(e_{g,c} \mid \mu_{i,k}, \sigma^2_{i,g}) \right] P(\mu_{i,k}, \sigma^2_{i,g}) d\mu_{i,k} d\sigma^2_{i,g}$$

Second, each of these cell-specific terms can be written as a function of a vector of sufficient statistics. Let $s(x) = \langle 1, x, x^2 \rangle$ be a vector valued function. Define

$$S_{i,k} = \sum_{g \in G_i} \sum_{c \in C_k} s(e_{g,c})$$

to be the sufficient statistics of the cell (i, k). We can write the likelihood and the Bayesian marginal likelihood scores as functions of these statistics [2]. Thus, to evaluate the local score of a particular cell, we need to accumulate these statistics, and then plug them in into a predefined analytical function.

3 Agglomerative Clustering

Assume that we receive as input a partition $\mathcal{C} = \{C_1, \ldots, C_m\}$ of conditions. We want to learn a clustering of genes. One approach to do this is using an agglomerative procedure. This procedure is a variant of classic clustering methods [4], and is motivated by agglomerative methods for probabilistic models [12, 6] and related models [10].

The basic algorithm is simple. We initialize the algorithm with maximal partition of genes, $\mathcal{G}^{(1)} = \{G_1, \ldots, G_{|\mathsf{Genes}|}\}$ where each G_i is a singleton.

At the t'th stage of the algorithm, we compute for each pair i, j of gene clusters

$$\begin{aligned}\Delta_{i,j}^{(t)} &= \mathrm{score}((\mathcal{G}^{(t)} - \{G_i, C_k\}) \cup (G_i \cup G_j) \ : \ \mathcal{D}) - \mathrm{score}(\mathcal{G}^{(t)}, \mathcal{C} \ : \ \mathcal{D}) \\ &= \prod_k \frac{\mathrm{LocalScore}(G_i \cup G_j, C_k \ : \ \mathcal{D})}{\mathrm{LocalScore}(G_i, C_k \ : \ \mathcal{D}) \cdot \mathrm{LocalScore}(G_j, C_k \ : \ \mathcal{D})}\end{aligned}$$

That is, $\Delta_{i,j}^{(t)}$ is the change in the score that results from merging the clusters G_i and G_j into one cluster.

After computing this quantity for all pairs i, j, we choose (i_t, j_t) to be the pair of clusters whose merger is the most beneficial according to the score:

$$(i_t, j_t) = \arg\max_{i,j} \Delta_{i,j}^{(t)}.$$

We then define

$$\mathcal{G}^{(t+1)} = \mathcal{G} \cup \{G_{i_t} \cup G_{j_t}\} - \{G_i, G_j\}$$

to be the partition resulting from merging G_i, and G_j.

We repeat these iterations until $t = |\mathsf{Genes}|$ and $\mathcal{G}^{(t)}$ contains a single cluster. The procedure returns the cluster tree that correspond the to sequence of cluster mergers. Each branch is annotated by the associated improvement in the score.

Several notes on the above algorithm. First, The actual computation is done using the logarithm of the score, which is much more numerically stable. Second, we note the sufficient statistics vector of a merged cluster is the sum of the sufficient statistics of the two individual clusters. Thus, once we have the sufficient statistics of each of the cells, the the cost of computing $\mathrm{LocalScore}(G_i \cup G_j, G_k \ : \ \mathcal{D})$ is a constant. Third, each step of the agglomerative clustering is in theory quadratic in the size of $\mathcal{C}^{(t)}$. However, in practice it is only linear in the size $\mathcal{C}^{(t)}$, since for most cluster pairs, $\Delta_{i,j}^{(t)}$ does not change. The only values we need to compute are these that involve the new cluster created in the last step of the algorithm. Putting these last two observations together, we conclude that that the overall cost of the algorithm is $O(|\mathsf{Genes}||\mathsf{Conds}| + |\mathsf{Genes}|^2|\mathcal{C}|)$. The first term is the cost of computing the sufficient statistics for the initial clustering, and second term is the cost of computing the initial pairwise merger costs. Finally, the cost of all additional iterations is is also $O(|\mathsf{Genes}|^2|\mathcal{C}|)$.

A final note on regularization. When estimating parameters from few examples, the estimate can be highly non-robust. This is particularly true for estimate of the variance. To avoid this problem, we use the maximum a posteriori estimate for the mean and variance (using a normal-gamma prior) instead of the maximum likelihood one. This estimator results in almost identical parameter values for a large sample sizes. However, for small ones it serves to regularize the estimate.

4 Double Clustering

The procedure I describe above builds a full agglomeration tree starting with the richest partition into singletons and ending with the poorest partition that contains a single set. In practice, we often want the procedure

to select for us the "best" partition. One possible way of doing so is to track the sequence of partitions $\mathcal{G}(1), \ldots, \mathcal{G}^{|\mathsf{Genes}|}$, and select the partition with the highest score. Note that although in theory the maximum likelihood score should select $\mathcal{G}(1)$, in practice, due to the regularization, it selects a partition in a much later stage. The intuition is that the best scoring partition strikes a tradeoff between finding groups of genes, so that each is homogeneous, and there distinct differences between them.

Using this stopping criteria, we can start with a condition partition $\mathcal{C}$, and find a new gene partition $\mathcal{G}$ that maximizes $\mathrm{score}(\mathcal{G}, C : \mathcal{D})$ among the partitions examined during the agglomeration of genes .

It is easy to consider a symmetric procedure that agglomerates conditions. Using this procedure, we can start with a gene partition $\mathcal{G}$, and perform agglomeration of conditions to find a condition partition $\mathcal{C}$ that maximizes $\mathrm{score}(\mathcal{G}, C : \mathcal{D})$ among the partitions examined during the agglomeration of conditions.

This suggests a simple iterative procedure to cluster both genes and conditions at the same time (using a variant of ideas presented in [**?**]). We start with some partition of the conditions (say the one where each is a singleton). We then perform gene agglomeration and select the "best" scoring gene partition. We fix this gene partition, and perform agglomeration on conditions (starting again from the richest partition on conditions) to find the best scoring condition partitions. We repeat these iterations until we reach a fixed point. Intuitively, each step improves the score, and thus this procedure should converge. Although this is true in practice, this is necessarily true in theory, since the agglomeration step is not guaranteed to find the best partition (among all possible ones). Thus, it is conceivable that some of the iterations reduce the score, and that this process will result in an infinite cycle. In practice, however, the procedure always converges.

The result of this iterative process is two hierarchies. One over genes, and the other over conditions. Each of these hierarchies is "tuned" to the best partition of the second one. Thus, the structure of the two hierarchies is strongly geared toward the cells defined by the gene and condition partitions we selected in the final iterations.

5 Summary

I presented a simple agglomerative approach to construct hierarchical clustering. This procedure is efficient and straightforward to implement. The main advantage of this procedure is that it can take as input the "relevant" distinctions among the conditions. It then uses these so that differences between genes are based on their distribution of values in each set of conditions. Since the procedure uses an adaptive probabilistic models, it can zoom in on the "interesting" distinctions by learns different variances for different cells. Finally, I described an iterative approach for constructing joint clustering of both genes and conditions at the same time.

Both strategies are implemented in the PCluster and DoublePCluster program of the ScoreGenes package, see `compbio.cs.huji.ac.il/scoregenes`.

Acknowledgments The development of the ideas presented here benefited from discussions with Amir Ben-Dor, Gal Elidan, Naftali Kaminski, Eran Segal, and Ilan Wapinski.

References

[1] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *J. Comp. Bio.*, 6(3-4):281–97, 1999.

[2] M. H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

[3] J. DeRisi., V. Iyer, and P. Brown. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 282:699–705, 1997.

[4] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[5] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95(25):14863–8, 1998.

[6] G. Elidan and N. Friedman. Learning the dimensionality of hidden variables. In *Proc. Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI '01)*. 2001.

[7] V.R. Iyer, M.B. Eisen, D.T. Ross, G. Schuler, T. Moore, J.C.F. Lee, J.M. Trent, L.M. Staudt, J. Hudson, 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*, 283:83–87, 1999.

[8] D. J. Lockhart, H. Dong, M. C. Byrne, M. T. Follettie, M. V. Gallo, M. S. Chee, M. Mittmann, C. Want, M. Kobayashi, H. Horton, and E. L. Brown. DNA expression monitoring by hybridization of high density oligonucleotide arrays. *Nature Biotechnology*, 14:1675–1680, 1996.

[9] R. Sharan and R. Shamir. CLICK: A clustering algorithm with applications to gene expression analisys. In *ISMB'00*, 2000.

[10] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Advances in Neural Information Processing Systems 12*. MIT Press, Cambridge, Mass., 2000.

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

[12] A. Stolcke and S. Omohundro. Hidden Markov Model induction by bayesian model merging. In Stephen José Hanson, Jack D. Cowan, and C. Lee Giles, editors, *Advances in Neural Information Processing Systems*, volume 5, pages 11–18. Morgan Kaufmann, San Mateo, CA, 1993.

[13] X. Wen, S. Furhmann, G. S. Micheals, D. B. Carr, S. Smith, J. L. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. *PNAS*, 95:334–339, 1998.