6.047/6.878 Lecture 13: Gene Expression Clustering

Franck Dernoncourt (2012)
Arvind Thiagarajan (2011)
Tahin Syed (2010)
Barrett Steinberg and Brianna Petrone (2009)
Mia Y. Qia (2008)

October 30, 2012
List of Figures

1. Clustering compared to classification. In clustering we group observations into clusters based on how near they are to one another. In classification we want a rule that will accurately assign labels to new points. ................................................................. 4
2. Gene expression values from microarray experiments can be represented as heat maps to visualize the result of data analysis. ................................................................. 5
3. RNA-Seq reads mapping to a gene (c-fos) and its splice junctions. Densities along exon represent read densities mapping to exons (in log_{10}), arcs correspond to junction reads, where arc width is drawn in proportion to number of reads in that junction. The gene is downregulated in Sample 2 compared to Sample 1. ................................................................. 5
4. A sample matrix of gene expression values, represented as a heatmap and with hierarchal clusters. [1] ................................................................. 6
5. The k-means clustering algorithm ................................................................. 7
6. Examples of final cluster assignments of fuzzy k-means using k= 4 with centroids, correct clusters, and most probable assigned clusters marked as crosses, shapes shapes of points, and colors respectively. Note that the original data set is non-Gaussian. ................................................................. 8
7. K-Means as a Generative Model. Samples were drawn from normal distributions. ................................................................. 9
8. K-Means as an expectation maximization (EM) algorithm. ................................................................. 9
9. Comparison of clustering, HMM and motif discovery with respect to expectation minimization (EM) algorithm. ................................................................. 10
10. Hierarchical Clustering ................................................................. 11

1 Introduction

In this chapter, we consider the problem of discerning similarities or patterns within large datasets. Finding structure in such data sets allows us to draw conclusions about the process as well as the structure underlying the observations. We approach this problem through the application of clustering techniques. The following chapter will focus on classification techniques.

1.1 Clustering vs Classification

One important distinction to be made early on is the difference between classification and clustering. Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations or instances whose category membership is known. The training set is used to learn rules that will accurately assign labels to new observations. The difficulty is to find the most important features (feature selection).

In the terminology of machine learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly-identified observations is available. The corresponding unsupervised procedure is known as clustering or cluster analysis, and involves grouping data into categories based on some measure of inherent similarity, such as the distance between instances, considered as vectors in a multi-dimensional vector space. The difficulty is to identify the structure of the data. Figure 1 illustrates the difference between clustering and classification.

1.2 Applications

Clustering was originally developed within the field of artificial intelligence. Being able to group similar objects, with full implications of generality implied, is indeed a fairly desirable attribute for an artificial intelligence, and one that humans perform routinely throughout life. As the development of clustering algorithms proceeded apace, it quickly becomes clear that there was no intrinsic barrier involved in applying these algorithms to larger and larger datasets. This realization led to the rapid introduction of clustering to computational biology and other fields dealing with large datasets.

Clustering has many applications to computational biology. For example, let’s consider expression profiles of many genes taken at various developmental stages. Clustering may show that certain sets of genes line
Figure 1: Clustering compared to classification. In clustering we group observations into clusters based on how near they are to one another. In classification we want a rule that will accurately assign labels to new points.

up (i.e. show the same expression levels) at various stages. This may indicate that this set of genes has common expression or regulation and we can use this to infer similar function. Furthermore, if we find a uncharacterized gene in such a set of genes, we can reason that the uncharacterized gene also has a similar function through guilt by association.

Chromatin marks and regulatory motifs can be used to predict logical relationships between regulators and target genes in a similar manner. This sort of analysis enables the construction of models that allow us to predict gene expression. These models can be used to modify the regulatory properties of a particular gene, predict how a disease state arose, or aid in targeting genes to particular organs based on regulatory circuits in the cells of the relevant organ.

Computational biology deals with increasingly large and open-access datasets. One such example is the ENCODE project [2]. Launched in 2003, the goal of ENCODE is to build a comprehensive list of functional elements in the human genome, including elements that act at the protein and RNA levels, and regulatory elements that control cells and circumstances in which a gene is active. ENCODE data are now freely and immediately available for the entire human genome: \texttt{http://genome.ucsc.edu/ENCODE/}. Using all of this data, it is possible to make functional predictions about genes through the use of clustering.

2 Microarrays

The most intuitive way to investigate a certain phenotype is to measure the functional proteins present at a given time in the cell. However, given the difficulty of measuring protein, due to their varying locations, modifications, and contexts in which they are found, as well as due to the incompleteness of the proteome, mRNA expression levels are often used as a good approximation, since it is much easier to measure and allows thousands of genes to be measured in one \textbf{microarray} experiment. Furthermore, given the Central Dogma of Molecular Biology, it is more desirable to measure mRNA since we are measuring the regulation that occurs at the level of the genome. By measuring proteins, we would be combining two regulatory steps.

2.1 Technology/Biological Methods

The basic principle behind microarrays is the hybridization of complementary DNA fragments. To begin, short segments of DNA, known as probes, are attached to a solid surface, commonly known as a gene chip. Then, the RNA population of interest, which has been taken from a cell, is reverse transcribed to cDNA (complementary DNA) via reverse transcriptase, which synthesizes DNA from RNA using the poly-A tail as a primer. For intergenic sequences which have no poly-A tail, a standard primer can be ligated to the ends of the mRNA. The resulting DNA has more complementarity to the DNA on the slide than the RNA. The cDNA is than washed over the chip and the resulting hybridization triggers the probes to fluoresce. This can be detected to determine the relative abundance of the mRNA in the target, as illustrated in figure 2.
Two basic types of microarrays are currently used. Affymetrix gene chips have one spot for every gene and have longer probes on the order of 100s of nucleotides. On the other hand, spotted oligonucleotide arrays tile genes and have shorter probes around the tens of bases.

2.2 Limits
There are numerous sources of error in the current methods and future methods seek to remove steps in the process. For instance, reverse transcriptase may introduce mismatches, which weaken interaction with the correct probe or cause cross hybridization, or binding to multiple probes. One solution to this has been to use multiple probes per gene, as cross hybridization will be different for each gene. Still, reverse transcription is necessary due to the secondary structure of RNA. The structural stability of DNA makes it less probable to bend and not hybridize to the probe. The next generation of technologies, such as RNA-Seq, sequences the RNA as it comes out of the cell, essentially probing every base of the genome.

3 RNA-Seq
RNA-Seq, also known as whole transcriptome shotgun sequencing, attempts to perform the same function that DNA microarrays have been used to perform in the past, but with greater resolution. In particular, DNA microarrays utilize specific probes, and creation of these probes necessarily depends on prior knowledge of the genome and the size of the array being produced. RNA-seq removes these limitations by simply sequencing all of the cDNA produced in microarray experiments. This is made possible by next-generation sequencing technology. The technique has been rapidly adopted in studies of diseases like cancer [4]. The data from
RNA-seq is then analyzed by clustering in the same manner as data from microarrays would normally be analyzed.

### 4 Gene Expression Matrices

The amount of data generated by a microarray or an RNA-seq is enormous. For example, taking the output of a gene prediction model and making probes for every single one, detects the expression of every gene in the cell. Furthermore, performing experiments across conditions, time courses, stages of development, phenotype, or any other factor increases the amount of data. The result is often represented as an $m \times n$ expression matrix, profiling the expression levels of $m$ genes across $n$ experiments. Clustering gene expression data enables data visualization to identify what different cancers look like, find functional similarities between genes, and can facilitate the development of a gene regulatory network model.

These matrices can be clustered hierarchically showing the relation between pairs of genes, pairs of pairs, and so on, creating a dendrogram in which the rows and columns can be ordered using optimal leaf ordering algorithms.

This predictive and analytical power is increased due to the ability of biclustering the data; that is, clustering along both dimensions of the matrix. The matrix allows for the comparison of expression profiles of genes, as well as comparing the similarity of different conditions such as diseases. A challenge, though, is the curse of dimensionality. As the space of the data increases, the clustering of the points diminishes. Sometimes, the data can be reduced to lower dimensional spaces to find structure in the data using clustering to infer which points belong together based on proximity.

![Figure 4: A sample matrix of gene expression values, represented as a heatmap and with hierarchal clusters.](image)

Interpreting the data can also be a challenge, since there may be other biological phenomena in play. For example, protein-coding exons have higher intensity, due to the fact that introns are rapidly degraded. At the same time, not all introns are junk and there may be ambiguities in alternative splicing. There are also cellular mechanisms that degrade aberrant transcripts through non-sense mediated decay.

### 5 Clustering Algorithms

There are two types of clustering algorithms: partitioning and agglomerative. Partitional clustering divides objects into non-overlapping clusters so that each data object is in one subset. Alternatively, agglomerative
clustering methods yield a set of nested clusters organized as a hierarchy representing structures from broader to finer levels of detail.

5.1 **K-Means Clustering**

The k-means algorithm clusters \( n \) objects based on their attributes into \( k \) partitions. This is an example of partitioning, where each point is assigned to exactly one cluster such that the sum of distances from each point to its correspondingly labeled center is minimized. The motivation underlying this process is to make the most compact clusters possible, usually in terms of a Euclidean distance metric.

![Figure 5: The k-means clustering algorithm](Image)

The k-means algorithm, as illustrated in figure 5, is implemented as follows:

1. Assume a fixed number of clusters, \( k \)

2. **Initialization**: Randomly initialize the \( k \) means \( \mu_k \) associated with the clusters and assign each data point \( x_i \) to the nearest cluster, where the distance between \( x_i \) and \( \mu_k \) is given by \( d_{i,k} = (x_i - \mu_k)^2 \).

3. **Iteration**: Recalculate the centroid of the cluster given the points assigned to it: 
   \[
   \mu_k(n+1) = \frac{1}{\sum_{x_i \in k} x_i} \sum_{x_i \in k} x_i
   \]
   where \( x_k \) is the number of points with label \( k \). Reassign data points to the \( k \) new centroids by the given distance metric. The new centers are effectively calculated to be the average of the points assigned to each cluster.

4. **Termination**: Iterate until convergence or until a user-specified number of iterations has been reached. Note that the iteration may be trapped at some local optima.

There are several methods for choosing \( k \): simply looking at the data to identify potential clusters or iteratively trying values for \( n \), while penalizing model complexity. We can always make better clusters by increasing \( k \), but at some point we begin overfitting the data.

We can also think of k-means as trying to minimize a cost criterion associated with the size of each cluster, where the cost increases as the clusters get less compact. However, some points can be almost halfway between two centers, which doesn’t fit well with the binary belonging k-means clustering.

5.2 **Fuzzy K-Means Clustering**

In fuzzy clustering, each point has a probability of belonging to each cluster, rather than completely belonging to just one cluster. Fuzzy k-means specifically tries to deal with the problem where points are somewhat in between centers or otherwise ambiguous by replacing distance with probability, which of course could be some function of distance, such as having probability relative to the inverse of the distance. Fuzzy k-means uses a weighted centroid based on those probabilities. Processes of initialization, iteration, and termination are
the same as the ones used in k-means. The resulting clusters are best analyzed as probabilistic distributions rather than a hard assignment of labels. One should realize that k-means is a special case of fuzzy k-means when the probability function used is simply 1 if the data point is closest to a centroid and 0 otherwise.

Figure 6: Examples of final cluster assignments of fuzzy \( k \)-means using \( k = 4 \) with centroids, correct clusters, and most probable assigned clusters marked as crosses, shapes shapes of points, and colors respectively. Note that the original data set is non-Gaussian.

The fuzzy k-means algorithm is the following:

1. Assume a fixed number of clusters \( k \)
2. Initialization: Randomly initialize the \( k \) means \( \mu_k \) associated with the clusters and compute the probability that each data point \( x_i \) is a member of a given cluster \( k \), \( P(\text{point } x_i, \text{has label} k | x_i, k) \).
3. Iteration: Recalculate the centroid of the cluster as the weighted centroid given the probabilities of membership of all data points \( x_i \):
   \[
   \mu_k(n + 1) = \frac{\sum_{x_i \in k} x_i \times P(\mu_k | x_i)^b}{\sum_{x_i \in k} P(\mu_k | x_i)^b}
   \]

4. Termination: Iterate until convergence or until a user-specified number of iterations has been reached (the iteration may be trapped at some local maxima or minima)

5.3 K-Means as a Generative Model

A generative model is a model for randomly generating observed data, given some hidden parameters. While a generative model is a probability model of all variables, a discriminative model provides a conditional model only of the target variable(s) using the observed variables.

When we perform clustering, we are assuming something about the underlying data. In the case of k-means and fuzzy k-means, we are assuming that a set \( k \) centers (parameters) generate the data points using a Gaussian distribution for each \( k \), potentially generating a stochastic representation of the data, as shown in figure 7. In the case where \( k = 2 \), this can be thought of as flipping a coin to choose one of the two centers, then randomly placing a point, according to a Gaussian distribution, somewhere near the center. Unfortunately, since k-means assumes independence between the axes, covariance and variance are not accounted for using k-means, so models such as oblong distributions are not possible. In the clustering processes discussed containing a set of labeled data points \( x \), we want to choose the most probable center (parameter) for \( x \); that is, we want to maximize the probability of the clustering model. This is the maximum likelihood setting of \( \mu_k \), given the data. Given a set of observations \( x_i \) and all labels \( k \), we can find the maximum likelihood \( \mu_k \) as follows:

\[
\arg \max_{\mu} \left\{ \log \prod_i P(x_i | \mu) \right\} = \arg \max_{\mu} \sum_i \left\{ -\frac{1}{2} (x_i - \mu)^2 + \log\left(\frac{1}{\sqrt{2\pi}}\right) \right\} = \arg \min_{\mu} \sum_i (x_i - \mu)
\]
Probability in this case is a function of distance of each data point from each center. After that, we want to find the best new parameter, the maximum likelihood for the next iteration of the algorithm using the same processes.

These same principles apply in reverse to determine labels from known centers, similarly using the argmin function. In this case, we attempt to find the best label that maximizes the likelihood of the data. We find that this is the same as simply finding the nearest center.

If neither labels nor centers are known, a common solution to estimate both is to start with \( k \) arbitrary centers, calculate the most likely labels given the centers, use these labels to choose new centers, and iterate until a local maximum of probability is reached.

K-means can be seen as an example of EM (expectation maximization algorithms), as shown in figure 8 where expectation consists of estimation of hidden labels, \( Q \), and maximizing of expected likelihood occurs given data and \( Q \). Assigning each point the label of the nearest center corresponds to the E step of estimating the most likely label given the previous parameter. Then, using the data produced in the E step as observation, moving the centroid to the average of the labels assigned to that center corresponds to the M step of maximizing the likelihood of the center given the labels. This case is analogous to Viterbi learning. A similar comparison can be drawn for fuzzy \( k \)-means, which is analogous to Baum-Welch from HMMs. Figure 9 compares clustering, HMM and motif discovery with respect to expectation minimization algorithm.

EM is guaranteed to converge and guaranteed to find the best possible answer, at least from an algorithmic point of view. The notable problem with this solution is that of local maxima of probability distributions in which only uphill movement is possible. Thus an absolute maximum may never be determined. This problem may be hopefully avoided by attempting multiple initializations to better determine the landscape of probabilities.

### 5.4 The limitations of the \( k \)-Means algorithm

The \( k \)-means algorithm has a few limitations which are important to keep in mind when using it and before choosing it. First of all, it requires a metric. For example, we cannot use the \( k \)-means algorithm on a set of words since we would not have any metric.

The second main limitation of the \( k \)-means algorithm is its sensitivity to noise. One way to try to reduce the noise is to run a principle component analysis beforehand. Another way is to weight each variable in order
to give less weight to the variables affected by significant noise: the weights will be calculated dynamically at each iteration of the algorithm K-means [3].

Third limitation, the choice of initial centers influences the results. There exist heuristics to select the initial cluster centers, but none of them are perfect.

Lastly, we need to guess a priori the number of classes. As we have seen, there are ways to circumvent this problem, essentially by running several times the algorithm while varying k or using the rule of thumb $k \approx \sqrt{n/2}$ if we are short on the computational side. http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set summarizes well the different techniques to select the number of clusters. Hierarchical clustering provides a handy approach to choosing the number of cluster.

### 5.5 Hierarchical Clustering

While the clustering discussed thus far often provide valuable insight into the nature of various data, they generally overlook an essential component of biological data, namely the idea that similarity might exist on multiple levels. To be more precise, similarity is an intrinsically hierarchical property, and this aspect is not addressed in the clustering algorithms discussed thus far. Hierarchical clustering specifically addresses this in a very simple manner. As illustrated in figure 10, it is implemented as follows:

1. **Initialization**: Initialize a list containing each point as an independent cluster.

2. **Iteration**: Create a new cluster containing the two closest clusters in the list. Add this new cluster to the list and remove the two constituent clusters from the list.

Of course, a method for determining distances between clusters is required. The particular metric used varies with context, but some common implementations include the maximum, minimum, and average distances between constituent clusters, and the distance between the centroids of the clusters.

### 5.6 Evaluating Cluster Performance

The validity of a particular clustering can be evaluated in a number of different ways. The overrepresentation of a known group of genes in a cluster, or, more generally, correlation between the clustering and confirmed biological associations, is a good indicator of validity and significance. If biological data is not yet available, however, there are ways to assess validity using statistics. For instance, robust clusters will appear from clustering even when only subsets of the total available data are used to generate clusters. In addition, the statistical significance of a clustering can be determined by calculating the probability of a particular distribution

6 Current Research Directions

The most significant problems associated with clustering now are associated with scaling existing algorithms cleanly with two attributes: size and dimensionality. To deal with larger and larger datasets, algorithms such as canopy clustering have been developed, in which datasets are coarsely clustered in a manner intended to pre-process the data, following which standard clustering algorithms (e.g. \( k \)-means) are applied to sub-divide the various clusters. Increase in dimensionality is a much more frustrating problem, and attempt to remedy this usually involve a two stage process in which appropriate relevant subspaces are first identified by appropriate transformations on the original space and then subjected to standard clustering algorithms.

7 Further Reading


8 Resources

- Cluster 3.0: open source clustering software that implements the most commonly used clustering methods for gene expression data analysis.
- Orange is a free data mining software suite (see module orngClustering for scripting in Python): [http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm](http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm)
- R (see Cluster Analysis and Finite Mixture Models)
- SAS CLUSTER
9 What Have We Learned?

To summarize, in this chapter we have seen that:

- In **clustering**, we identify structure in unlabeled data. For example, we might use clustering to identify groups of genes that display similar expression profiles.
  - Partitioning clustering algorithms, construct non-overlapping clusters such that each item is assigned to exactly one cluster. Example: k-means
  - Agglomerative clustering algorithms construct a hierarchical set of nested clusters, indicating the relatedness between clusters. Example: hierarchical clustering

- In **classification**, we partition data into known labels. For example, we might construct a classifier to partition a set of tumor samples into those likely to respond to a given drug and those unlikely to respond to a given drug based on their gene expression profiles. We will focus on classification in the next chapter.

References
