Short introduction to Mfuzz package

Matthias E. Futschik
Institute for Theoretical Biology, Humboldt-University
URL: *http://itb.biologie.hu-berlin.de/~futschik/software/R/Mfuzz*

October 22, 2008

Contents

1 Overview

Clustering is an important tool in microarray data analysis. This unsupervised learning technique is commonly used to reveal structures hidden in large gene expression data sets. The vast majority of clustering algorithms applied so far produce *hard partitions* of the data, i.e. each gene is assigned exactly to one cluster. *Hard* clustering is favourable if clusters are well separated. However, this is generally not the case for microarray time-course data, where gene clusters frequently overlap. Additionally, hard clustering algorithms are often highly sensitive to noise.

To overcome the limitations of hard clustering, *soft* clustering can be applied offering several advantages to researchers [1]. First, it generates accessible internal cluster structures, i.e. it indicates how well corresponding clusters represent genes. This additional information can be used for a refined search for regulatory elements. Second, the overall relation between clusters, and thus a global clustering structure, can be defined. Additionally, soft clustering is more noise robust and *a priori* pre-filtering of genes can be avoided. This prevents the exclusion of biologically relevant genes from the data analysis.

This vignette gives a short introduction to soft clustering using the *Mfuzz* package. It misses some features (such as parameter selection and cluster stability) due to the size restrictions for Bioconductor vignettes. More information and a enlarged introduction can be found at the *Mfuzz* webpage:
http://itb.biologie.hu-berlin.de/~futschik/software/R/Mfuzz

2 Installation requirements

Following software is required to run the Mfuzz package:

- R ($>$ 2.0.0). For installation of R, refer to *http://www.r-project.org*.
- R-package: e1071. For installation of these add-on packages, refer to *http://cran.r-project.org*.
- Bioconductor package: Biobase. Refer to *http://www.bioconductor.org* for installation.

If these requirements are fulfilled, the *Mfuzz* add-on R-package can be installed. To see how to install add-on R-packages on your computer system, start *R* and type in *help(INSTALL)*. Optionally, you may use the R-function *install.packages()*. Once the *Mfuzz* package is installed, you can load the package by

```
> library(Mfuzz)
```

3 Data pre-processing

To illustrate our approach, we apply soft clustering to yeast cell cycle expression data by Cho *et al.* [2]. 6178 genes were monitored at 17 time points over a span of 160 minutes using Affymetrix chips. Note that we do not exclude here the array corresponding to the time-point of 90 mins which displays irregularities in the expression values measured. Additionally, the data set was modified by restricting the number of genes to 3000. Thus, results here might differ from those reported in reference [1].

```
> data(yeast)
```

3.1 Missing values

As a first step, we exclude genes with more than 25% of the measurements missing. Note that missing values should be denoted by `NA` in the gene expression matrix.

```
> yeast.r <- filter.NA(yeast, thres = 0.25)
```

```
49 genes excluded.
```

Fuzzy c-means like many other cluster algorithms, does not allow for missing values. Thus, we replace remaining missing values by the average values expression value of the corresponding gene.

```
> yeast.f <- fill.NA(yeast.r, mode = "mean")
```

Alternatively (and recommended), the (weighted) k-nearest neighbour method can be used (`mode='knn'/'wknn'`). These methods perform usually favourable compared to the simple method above, but are computationally intensive.

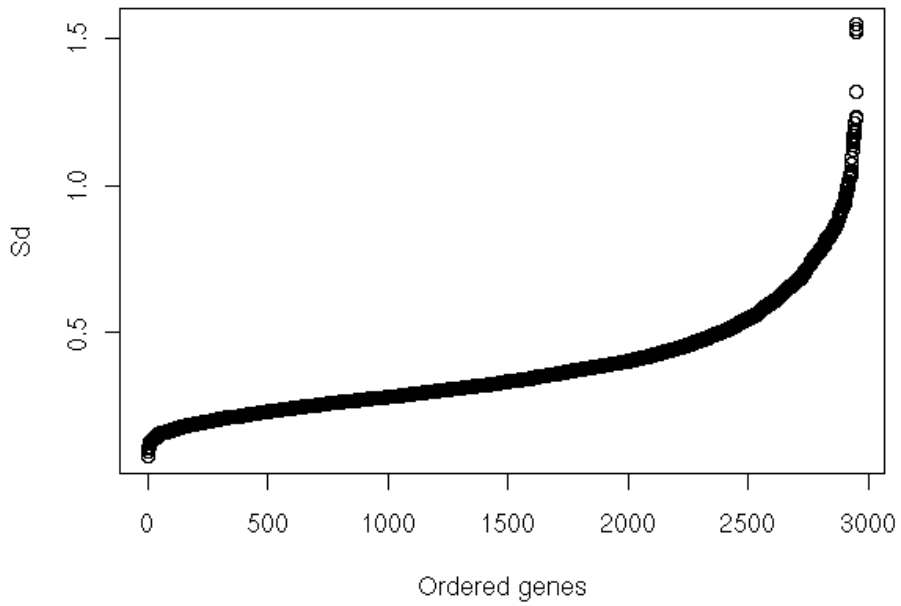

Figure 1: Standard deviation of gene expression vectors before standardisation. The genes were ordered by the standard deviation of the corresponding expression vector. A unique cut-off value for filtering is not prominent.

3.2 Filtering

Most cluster analyses published include a filtering step to remove genes which are expressed at low levels or show only small changes in expression. Different filtering procedures have been proposed. A popular procedure is the setting of a minimum threshold for variation
Calculation the standard deviation shows, however, that the transition between low and high values for variation in gene expression is smooth and no particular cut-off point is indicated (figure 1).

```
> tmp <- filter.std(yeast.f, min.std = 0)
```

Thus, the value of a filtering threshold remains arbitrary. As no stringent filtering procedure currently exists, we avoided any prior filtering of gene data. This prevents the loss of genes that may be biologically important.

3.3 Standardisation

Since the clustering is performed in Euclidian space, the expression values of genes were standardised to have a mean value of zero and a standard deviation of one. This step ensures that vectors of genes with similar changes in expression are close in Euclidean space:

```
> yeast.s <- standardise(yeast.f)
```

4 Soft clustering of gene expression data

Clustering is often applied to reveal regulatory mechanisms underlying gene expression. It is well known that regulation of genes is generally not in an 'on-off', but gradual manner which allows a finer control of the genes' functions. A cluster algorithm should reflect this finding by differentiating how closely a gene follows the dominant cluster patterns. Soft clustering appears as a good candidate for this task since it can assign a gene i gradual degrees of membership μ_{ij} to a cluster j. The membership values can vary continuously between zero and one. This feature enables soft clustering to provide more information about the structure of gene expression data.

Soft clustering is implemented in the function `mfuzz` using the fuzzy c-means algorithm (of the *e1071* package) based on minimization of a weighted square error function [3]. For soft clustering, the cluster centroids $\mathbf{c}_j$ result from the weighted sum of all cluster members and show the overall expression patterns of clusters. The membership values μ_{ij} indicate how well the gene i is represented by cluster $\mathbf{c}_j$. Low values μ_{ij} point to a poor representation of gene i by $\mathbf{c}_j$. Large values μ_{ij} point to a high correlation of the expression of gene i with the cluster centroid $\mathbf{c}_j$. The membership values are color-encoded in the plots generated by `mfuzz.plot`. This can facilitates the identification of temporal patterns in gene cluster (figure 2).

```
> cl <- mfuzz(yeast.s, c = 16, m = 1.25)
> mfuzz.plot(yeast.s, cl = cl, mfrow = c(4, 4), time.labels = seq(0,
+     160, 10))
```

4.1 Cluster cores

Membership values can also indicate the similarity of vectors to each other. If two gene expression vectors have a high membership value for a specific cluster, they are generally similar to each other. This is the basis for the definition of the core of a cluster. We define that genes with membership values larger than a chosen threshold α belong to the α-*core* of the cluster. This allows us to define relationships between genes within a cluster. Similarly to hierarchical clustering, the internal structures of clusters become accessible.

The average within-cluster variation is considerably reduced setting $\alpha = 0.7$. The use of the α-threshold can therefore act as *a posteriori* filtering of genes. This contrasts with previously discussed procedures which demand the problematic setting of a threshold *a priori* to the cluster analysis To extract list of genes belonging to the cluster cores, the `acore` function can be used.

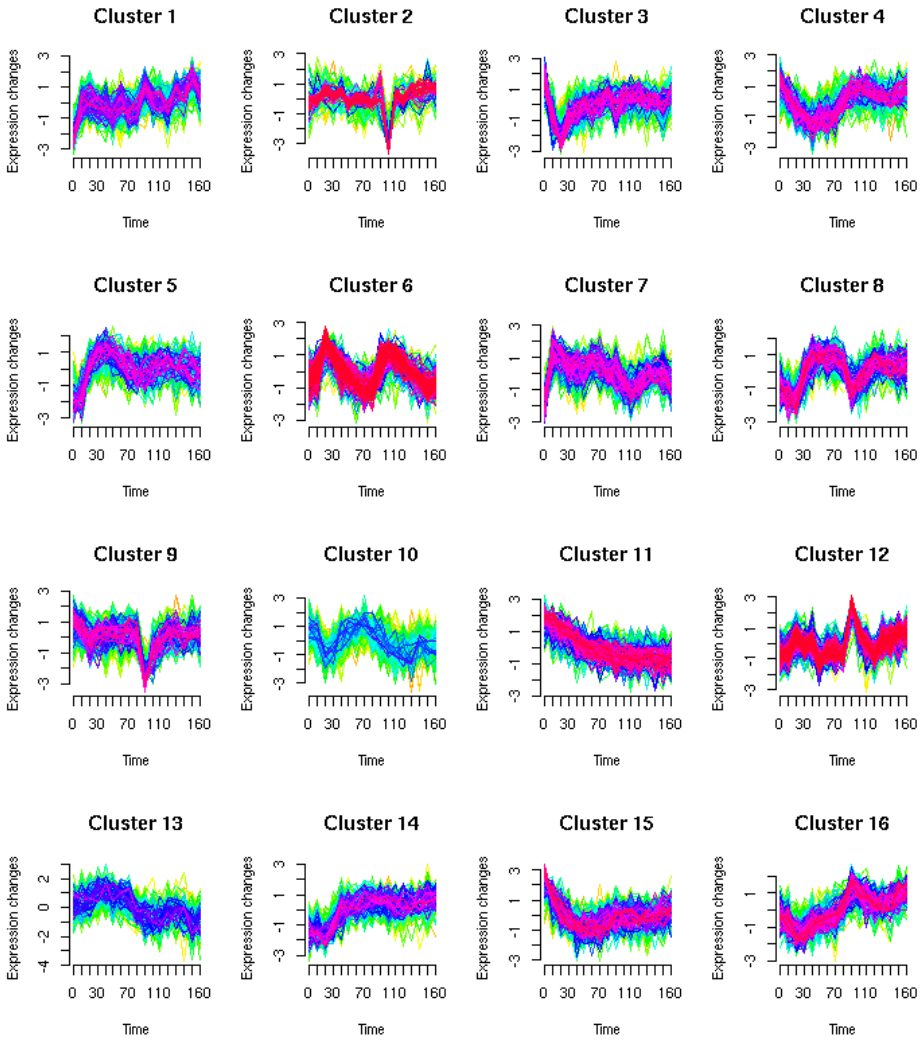

Figure 2: Soft clusters of yeast cell cycle expression data. Yellow or green colored lines correspond to genes with low membership value; red and purple colored lines correspond to genes with high membership value. Note the peaks in negative and positive direction for some clusters at time-point 90 min which may correspond to an experimental artifact.

5 Cluster stability

Variation of the FCM parameter m also allows investigation of the stability of clusters. We define here stable clusters as clusters that show only minor changes in their structure with variation of the parameter m. Stable clusters are generally well distinct of other clusters and compact. This is contrasted by weak clusters that lose their internal structure or disappear if m was increased.

```
> cl2 <- mfuzz(yeast.s, c = 16, m = 1.35)
> mfuzz.plot(yeast.s, cl = cl2, mfrow = c(4, 4), time.labels = seq(0,
+     160, 10))
```

6 Global clustering structures

An interesting feature of soft clustering is the overlap or coupling between clusters. The coupling coefficient V_{kl} between cluster k and cluster l can be defined by

$$V_{kl} = \frac{1}{N} \sum_{i=1}^{N} \mu_{ik} \mu_{il} \tag{1}$$

where N is the total number of gene expression vectors. The coupling indicates how many genes are shared by two clusters. Clusters which have a low coupling show distinct overall patterns. If the coupling is large, clusters patterns are more similar. Hence, the coupling defines a similarity measure for pairs of clusters.

```
> O <- overlap(cl)
> Ptmp <- overlap.plot(cl, over = O, thres = 0.05)
```

This allows the analysis of global clustering structures obtained by soft clustering, since relationships between clusters are defined. Similarly to hierarchical clustering, the global clustering structure can be examined at different resolutions determined by the cluster number c. For a small c, only the major clusters present in the data are obtained.

```
> cl3 <- mfuzz(yeast.s, c = 10, m = 1.25)
> mfuzz.plot(yeast.s, cl = cl3, mfrow = c(3, 4))
> O3 <- overlap(cl3)
> overlap.plot(cl3, over = O3, P = Ptmp, thres = 0.05)
```

If c is increased, sub-clusters with distinct patterns emerge Sub-clusters derived from a major cluster are generally strongly coupled, since they share the overall expression pattern. Finally, soft clustering produces empty clusters for further increase of c.

```
> cl4 <- mfuzz(yeast.s, c = 25, m = 1.25)
> mfuzz.plot(yeast.s, cl = cl4, mfrow = c(5, 5))
> O4 <- overlap(cl4)
> overlap.plot(cl4, over = O4, P = Ptmp, thres = 0.05)
```

References

[1] M.E. Futschik and B. Charlisle, Noise robust clustering of gene expression time-course data, Journal of Bioinformatics and Computational Biology, Vol. 3, No. 4, 965-988, 2005

[2] Cho RJ, Campbell MJ, Winzeler EA, Steinmetz L, Conway A, Wodicka L, Wolfsberg TG, Gabrielian AE, Landsman D, Lockhart DJ, Davis RW, A genome-wide transcriptional analysis of the mitotic cell cycle, *Mol Cell*, **2**:65–73, 1998

[3] Bezdak JC, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981