MCLUST: Software for Model-Based Cluster and Discriminant Analysis [*]

C. Fraley and A. E. Raftery

Technical Report No. 342
November 25, 1998
latest revision: May 20, 1999

Department of Statistics
University of Washington
Box 354322
Seattle, WA 98195-4322 USA

MCLUST is a software package for cluster and discriminant analysis written in Fortran and interfaced to the S-PLUS commercial software package[1] and the freely available R language [2] which has a similar look and feel. It implements parameterized Gaussian hierarchical clustering algorithms [16, 1, 7] and the EM algorithm for parameterized Gaussian mixture models [5, 13, 3, 14] with the possible addition of a Poisson noise term. MCLUST also includes functions that combine hierarchical clustering, EM and the Bayesian Information Criterion (BIC) in a comprehensive clustering strategy [4, 8]. Methods of this type have shown promise in a number of practical applications, including character recognition [16], tissue segmentation [1], minefield and seismic fault detection [4], identification of textile flaws from images [2], and classification of astronomical data [3, 15]. A web page with related links can be found at

`http://www.stat.washington.edu/fraley/mclust/home.html`.

1 Models

In MCLUST, each cluster is represented by a Gaussian model

$$\phi_k(\mathbf{x} \mid \mu_k, \Sigma_k) = (2\pi)^{-\frac{p}{2}} |\Sigma_k|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(\mathbf{x}_i - \mu_k)^T \Sigma_k^{-1} (\mathbf{x}_i - \mu_k) \right\}, \tag{1}$$

where $\mathbf{x}$ represents the data, and k is an integer subscript specifying a particular cluster. Clusters are ellipsoidal, centered at the means μ_k. The covariances Σ_k determine their other geometric features.

[*]Funded by the Office of Naval Research under contracts N00014-96-1-0192 and N00014-96-1-0330.
[1]MathSoft, Inc., Seattle, WA USA — `http://www.mathsoft.com/splus`
[2]see `http://lib.stat.cmu.edu/R/CRAN`

Each covariance matrix is parameterized by eigenvalue decomposition in the form

$$\Sigma_k = \lambda_k D_k A_k D_k^T,$$

where D_k is the orthogonal matrix of eigenvectors, A_k is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_k, and λ_k is a scalar. The orientation of the principal components of Σ_k is determined by D_k, while A_k determines the shape of the density contours; λ_k specifies the volume of the corresponding ellipsoid, which is proportional to $\lambda_k^d \, |A_k|$, where d is the data dimension. Characteristics (orientation, volume and shape) of distributions are usually estimated from the data, and can be allowed to vary between clusters, or constrained to be the same for all clusters [16, 1, 3]. This parameterization includes but is not restricted to well-known models such as uniform spherical variance ($\Sigma_k = \lambda I$) which gives the sum of squares criterion [19], constant variance [9], and unconstrained variance [18].

Table 1 shows the various model options currently available in `MCLUST` for hierarchical clustering (denoted HC) and EM. The model identifiers code geometric characteristics of the model. For example, `EFV` denotes a model in which the volumes of all clusters are equal (`E`), the shapes of all clusters are fixed (`F`) in advance by the user, and the orientation is allowed to vary (`V`) among the clusters. Parameters associated with characteristics designated by `E` or `V` are determined from the data.

Table 1: Parameterizations of the covariance matrix Σ_k currently available in `MCLUST` for hierarchical clustering (HC) and/or EM ('×' in the appropriate column indicates availability).

ID	Model	HC	EM	Distribution	Volume	Shape	Orientation	Reference
`EI`	λI	×	×	Spherical	equal	equal	NA	[19, 16, 1, 3]
`VI`	$\lambda_k I$	×	×	Spherical	variable	equal	NA	[1, 3]
`EEE`	λDAD^T	×	×	Ellipsoidal	equal	equal	equal	[9, 18, 1, 3]
`VVV`	$\lambda_k D_k A_k D_k^T$	×	×	Ellipsoidal	variable	variable	variable	[18, 1, 3]
`EFV`[2]	$\lambda D_k \hat{A} D_k^T$	×		Ellipsoidal	equal	fixed	variable	[16, 1]
`EEV`	$\lambda D_k A D_k^T$		×	Ellipsoidal	equal	equal	variable	[3]
`VFV`[2]	$\lambda_k D_k \hat{A} D_k^T$	×		Ellipsoidal	variable	fixed	variable	[1]
`VEV`	$\lambda_k D_k A D_k^T$		×	Ellipsoidal	variable	equal	variable	[3]

2 Obtaining and Installing `MCLUST`

`MCLUST` can be obtained via the world wide web at

`http://www.stat.washington.edu/fraley/mclust/soft.shtml`.

The S-PLUS version and R versions are available from Statlib; see

[3]Hierarchical clustering methods that estimate the shape from the data are possible [3, 7], but they cannot be computed as efficiently as their fixed-shape counterparts. The EM software for mixture models in `MCLUST` does estimate shape from the data.

http://lib.stat.cmu.edu/S/mclust

(MCLUST for S-PLUS version 3.4 for UNIX or version 4.5 for Windows), and

http://lib.stat.cmu.edu/R/CRAN/src/contrib/PACKAGES.html

(MCLUST for R, ported by Ron Wehrens http://www-sci.sci.kun.nl/cac/rwehrens).

The S-PLUS version and the associated Fortran code are also available via anonymous ftp from ftp.u.washington.edu in the directory public/mclust.

2.1 Using MCLUST with S-PLUS 3.4 for UNIX

The file MCLUST.tar.gz is a packed version of a directory containing all the necessary files for incorporating MCLUST into S-PLUS on UNIX systems. The commands to unpack it (and remove the tar file) are:

```
gunzip MCLUST.tar.gz
tar xvf MCLUST.tar
rm MCLUST.tar
```

This creates a directory called MCLUST, which should be moved to the working directory (in which MCLUST it is to be run with S-PLUS) if it is not already there. To compile the Fortran code for use with S-PLUS, do the following from the working directory:

```
cd MCLUST
Splus COMPILE mclust.o
```

This creates an object file mclust.o for loading into S-PLUS. Now return to the working directory and move mclust.o there:

```
cd ..
mv MCLUST/mclust.o .
```

The next step is to load the object code into S-PLUS. The following command statically loads mclust.o into S-PLUS (and removes the object file which is no longer needed): [4]

```
Splus LOAD mclust.o
rm mclust.o
```

This creates a file called local.Sqpe which is a local version of S-PLUS incorporating the Fortran functions from MCLUST. To use the S-PLUS functions in MCLUST, do the following:

```
cd MCLUST
cat mclust.S | Splus
```

These functions can now be found in MCLUST/.Data. Finally, use the command

```
> attach("MCLUST/.Data")
```

in S-PLUS to attach these functions (this command can be included in a .First function so that it will be executed when S-PLUS is invoked).

[4] On some UNIX systems, dynamic loading may also be possible.

2.2 Using MCLUST with S-PLUS 4.5 for Windows

The file `mclust.zip` contains the necessary files for running `MCLUST` with S-PLUS 4.5 for Windows. To install `MCLUST` on Windows, first go to the directory to be used as a working directory for S-PLUS, put `mclust.zip` there, open it to invoke `WinZip`, and then extract a folder called `mclust`. To put the S-PLUS functions from `MCLUST` into the working `_Data` directory, use the following S-PLUS command:

```
> source("mclust/mclust.S")
```

It may be preferable to source these functions into a separate directory and access them from S-PLUS through the `attach` command to avoid modifying them by accident. To load the compiled Fortran code for `MCLUST`, use the S-PLUS command

```
> dyn.load("mclust/mclust.obj")
```

This command needs to be executed every time `MCLUST` is to be used with S-PLUS, and can be included in a `.First` function so that it will be invoked automatically when S-PLUS is started in the working directory.

3 Hierarchical Clustering

`MCLUST` provides functions `mhtree` for computing classification trees via model-based hierarchical agglomeration, and `mhclass` for determining the resulting classifications. As an example of the use of `mhtree` and `mhclass`, consider Fisher's iris data [6], which is available as a data set in S-PLUS.[5] We first transform the data from a three-dimensional array to a matrix in which the species information is lost, then apply the hierarchical clustering algorithm for non-uniform spherical variances (`VI`):

```
> iris.matrix <- matrix(aperm(iris,c(1,3,2)),150,4,dimn=dimnames(iris)[1:2])
> cltree <- mhtree(iris.matrix, modelid = "VI")
```

The classification produced by `mhtree` for various numbers of clusters can be obtained with `mhclass`. For example, for the classifications corresponding to 2 and 3 clusters:

```
> cl <- mhclass(cltree, 2:3)
```

Classifications can be displayed with the data using `clpairs`:

```
> clpairs(iris.matrix, cl[,"2"])
> clpairs(iris.matrix, cl[,"3"])
```

Figure 1 shows the 3-cluster classification for the iris data with this model.

The function `mhtree` starts by default with every observation of the data in a cluster by itself, and continues until all observations are merged into a single cluster. However arguments `partition` and `min.clusters` can be used to initialize the process at a chosen nontrivial partition, and to stop it before it reaches the final stage of merging.

[5] A description of this data set is available from the command line via `help(iris)` or from the help window under '`datasets`'.

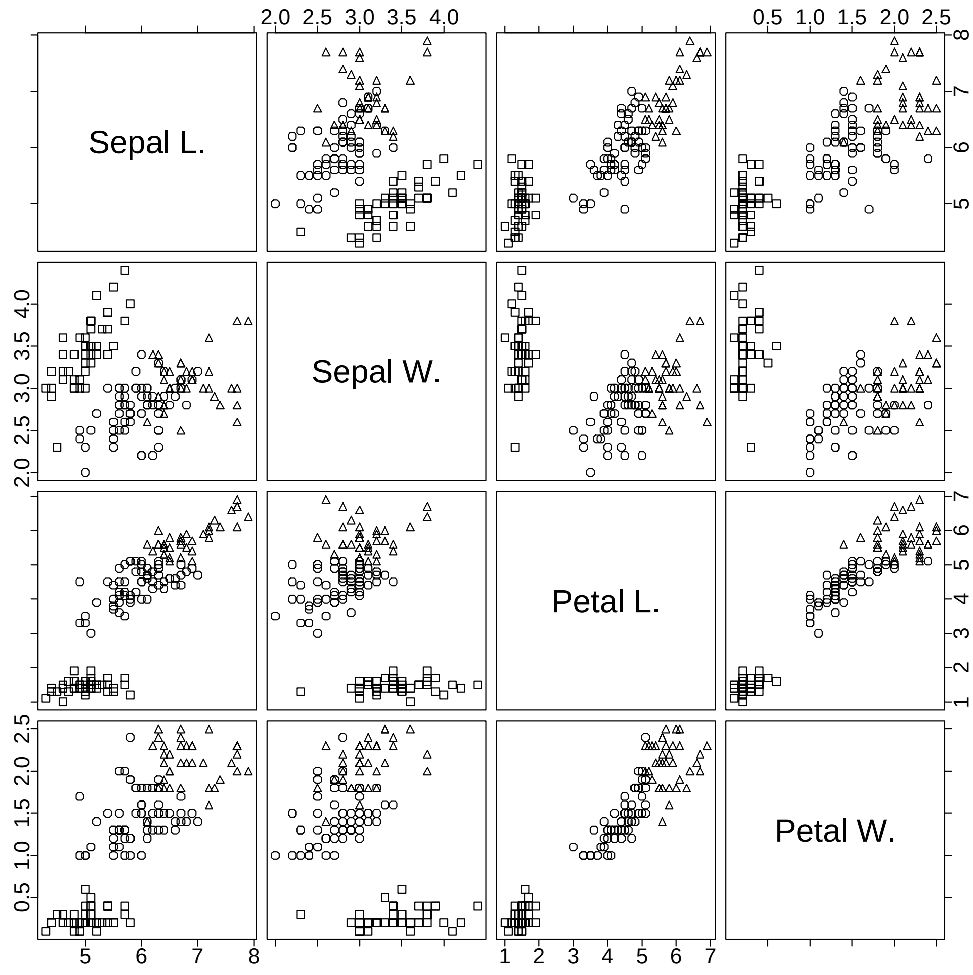

Figure 1: Pairs plot created with `clpairs` showing the 3-cluster classification of Fisher's iris data. This classification was produced by agglomerative hierarchical clustering using the criterion for a nonuniform spherical Gaussian model (`VI`).

4 EM for Mixture Models

`MCLUST` provides iterative EM (Expectation-Maximization) methods for maximum likelihood clustering with parameterized Gaussian mixture models. EM iterates between an 'E'-step, which computes a matrix z such that z_{ik} is an estimate of the conditional probability that observation i belongs to group k given the current parameter estimates, and an 'M-step', which computes maximum likelihood parameter estimates given z. In the limit, the parameters usually converge to the maximum likelihood values for the Gaussian mixture model

$$\prod_{i=1}^{n} \sum_{k=1}^{G} \tau_k \ \phi_k(\mathbf{x}_i \mid \mu_k, \Sigma_k),$$

and the sums of the columns of z converge to n times the mixing proportions τ_k, where n is the number of observations in the data. Here G is the number of groups in the data, which is assumed to be known for the purposes of the EM algorithm. The parameterizations of

Σ_k currently available for EM in MCLUST are listed in Table 1. They are a subset of the parameterizations discussed in [3], which gives details of the EM iteration for these models.

MCLUST provides functions me (iterated M-step followed by E-step), estep and mstep, implementing the EM algorithm for the parameterized Gaussian mixtures. Given the data, an initial estimate of z, and the model specification, me produces the values of z associated with maximum likelihood parameters. Initial estimates of z may be obtained from a discrete classification, resulting in a matrix that has only $0, 1$ entries with exactly one 1 per row. For example, me can be started with a classification produced by mhtree:

```
> cltree <- mhtree( iris.matrix, modelid = "VVV") # unconstrained model
> cl <- mhclass(cltree, 3)                       # 3-group mhtree classification
> z <- me( iris.matrix, modelid = "VVV", ctoz(cl)) # optimal z
```

The function ctoz converts a discrete classification into the corresponding z matrix. In general, the models used in mhtree and me need not be the same. It may in some cases be desirable to use one of the faster methods in mhtree (e. g. spherical or unconstrained models), followed by specification of a more complex model for EM.

For any z, there is a corresponding discrete classification assigning each observation to the group represented by the column in which the z value for that observation is maximized. MCLUST provides a function ztoc for coverting z to this 'nearest' classification. The following call to clpairs plots the iris data along with its classification obtained from me in the above example:

```
> clpairs( iris.matrix, ztoc(z))
```

The uncertainty in the classification associated with z can be obtained by subtracting the probability of the most likely group for each observation from 1:

```
> uncer <- 1 - apply( z, 1, "max")
```

The S-PLUS function quantile applied to the uncertainty gives a measure of the quality of the classification.

```
> quantile(uncer)
   0%  25%          50%         75%      100%
    0    0 1.727162e-08 0.001429494 0.3288714
```

In this case the indication is that the majority of observations are well classified. Note, however, that when groups intersect, uncertain classifications would be expected in the overlapping regions.

Maximum likelihood parameters can be recovered from the z produced by me using mstep:

```
> pars <- mstep( iris.matrix, modelid = "VVV", z)
> pars
```

Once values of the parameters are available, projections of the data showing the means and standard deviations of the corresponding clusters (and optionally the partition or classification) may be plotted using mixproj:

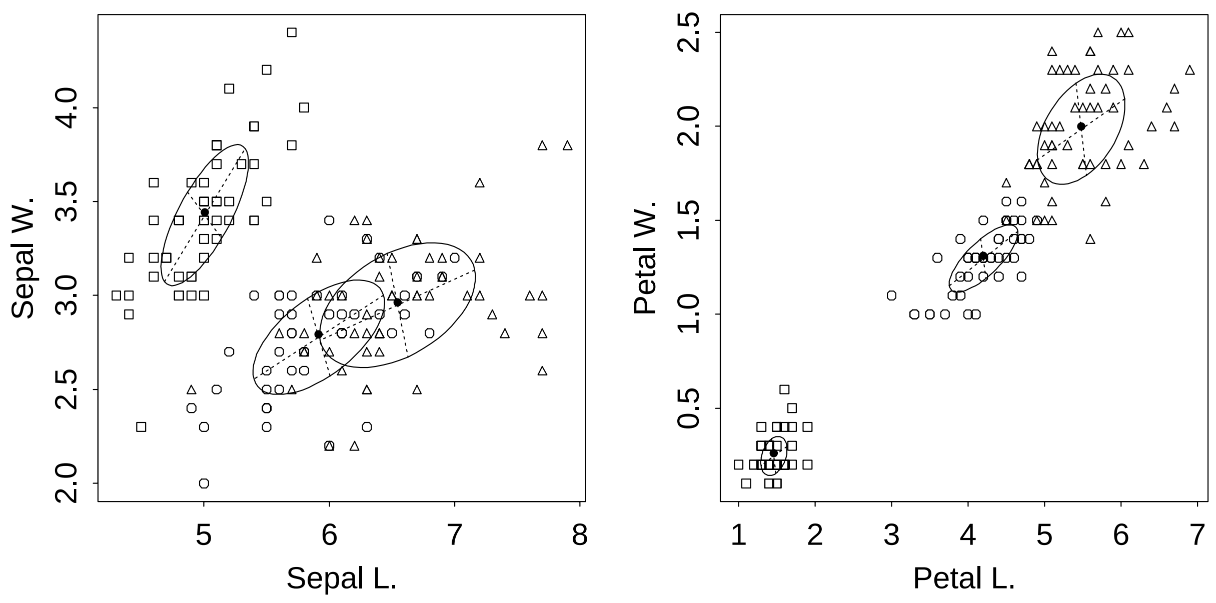

Figure 2: Plots created with `mixproj` showing the 3-cluster classification from EM for Fisher's iris data using an unconstrained Gaussian model (`VVV`).

```
> mixproj(iris.matrix, ms=pars, partition=ztoc(z), dimens = c(1,2))
> mixproj(iris.matrix, ms=pars, partition=ztoc(z), dimens = c(3,4))
```

The resulting plots are displayed in Figure 2.

`MCLUST` includes functions `estep` and `mstep` implementing the individual steps of the EM iteration. The iterative process can therefore be initialized with parameter estimates by calling `estep` before `me`, as well as with conditional probabilities as illustrated above.

5 Bayesian Information Criterion

`MCLUST` provides a function `bic` to compute the Bayesian Information Criterion (BIC) [17] given the data and a model along with conditional probability estimates. This allows comparison of models with differing parameterizations and/or differing numbers of clusters. In general the larger the value of the BIC, the stronger the evidence for the model and number of clusters. A standard convention for calibrating BIC differences is that differences of less than 2 correspond to weak evidence, differences between 2 and 6 to positive evidence, differences between 6 and 10 to strong evidence, and differences greater than 10 to very strong evidence [11, 12]. The following shows the BIC calculation in MCLUST for the 2 and 3-cluster classifications Fisher's iris data with the uniform and nonuniform spherical models:

```
> cltree <- mhtree(iris.matrix)  # uses default unconstrained model
> cl <- mhclass(cltree, 2:3)

> z <- me( iris.matrix, modelid = "EI", ctoz(cl[,"2"])) # uniform spherical
> bic(iris.matrix, modelid = "EI", z)
[1] -1123.411

> z <- me( iris.matrix, modelid = "EI", ctoz(cl[,"3"]))
> bic(iris.matrix, modelid = "EI", z)
[1] -878.7639

> z <- me( iris.matrix, modelid = "VI", ctoz(cl[,"2"])) # spherical
> bic(iris.matrix, modelid = "VI", z)
[1] -1012.235

> z <- me( iris.matrix, modelid = "VI", ctoz(cl[,"3"]))
> bic(iris.matrix, modelid = "VI", z)
[1] -853.809
```

In both models, the BIC favors the 3 group classification over 2 groups; overall, of the four possibilities considered in this section (EI or VI model; 2 or 3 groups), the best choice is VI (nonuniform spherical), with 3 groups.

6 Cluster Analysis

MCLUST provides two functions, emclust and emclust1, for cluster analysis with BIC. Both initialize EM using hierarchical clustering for various parameterizations of the Gaussian model. The input to emclust is the data, the desired numbers of groups, and a list of models to apply in the EM phase (initialized with hierarchical clustering using the unconstrained model). It returns the BIC values for all of the chosen models and number of clusters, together with auxiliary information that is used by the corresponding summary method for recovering parameter values. The following is an example of the use of emclust with Fisher's iris data:

```
> bicvals <- emclust( iris.matrix, nclus = 1:6, modelid = c("VVV","EEV","VEV"))
> bicvals

 BIC:
           1         2         3         4         5         6
VVV -829.9782 -574.0178 -580.8389 -628.9564 -683.8114 -711.5657
EEV -829.9782 -644.5997 -610.0836 -645.9950 -621.6901 -669.7069
```

```
VEV -829.9782 -561.7285 -562.5507 -589.3510 -635.2051 -681.2976

 sample noise equal
      F     F     F
> plot(bicvals)
```

The BIC values for this example are shown in Figure 3. Application of the `summary` function to this result reveals further informartion:

```
> sumry <- summary(bicvals, iris.matrix) # summary object for emclust()
> sumry

 best classification:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[112] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[149] 2 2

 uncertainty (quantiles):
   0%  25%  50%          75%         100%
    0    0    0 8.917506e-12 0.0002025599

 best BIC values:
     VEV,2     VEV,3     VVV,2
 -561.7285 -562.5507 -574.0178

 best model: uniform shape

 sample noise equal
      F     F     F
```

The best model among those fitted by `emclust` is the uniform shape model `VEV`, with 2 clusters. The same model with 3 clusters has a BIC value that is little different from the maximum; the conclusion is that there are either 2 or 3 clusters in the data under these models. The 2 cluster EM result separates the first species from the other two, while the 3 cluster result nearly separates the three species (there are 5 misclassifications out of 150).

The function `emclust1` is similar to `emclust`, except that in addition to the data and the desired numbers of groups, it takes as input a pair of models, the first to be used in the initial hierarchical clustering phase, and the second to be used in the EM phase. It returns the BIC values for each number of groups under the chosen model, as well as auxiliary information to be used by the associated `summary` method for recovering parameter values. Here is another analysis of the iris data with `emclust1` that uses the uniform spherical model (`EI`) in the hierarchical clustering phase, and the uniform variance model (`EEE`) in the EM phase:

```
> bicvals1 <- emclust1( iris.matrix, nclus = 1:6, modelid = c("EI","EEE"))
> bicvals1
```

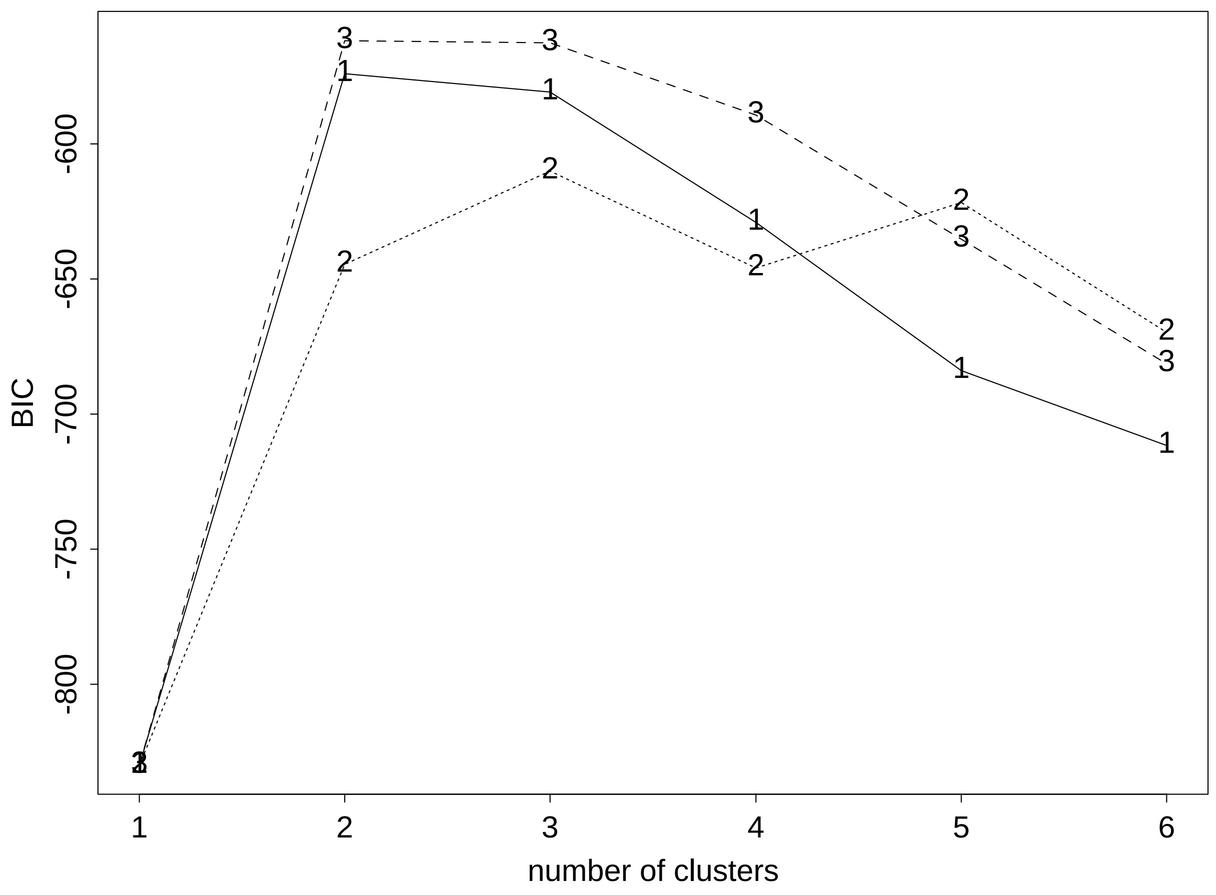

Figure 3: BIC values from `emclust` for the models 1 - `VVV`, 2 - `EEV`, and 3 -`VEV` with up to six clusters applied to Fisher's iris data.

```
 BIC:
         1          2          3          4          5          6 
 -829.9782 -688.0972 -632.9633 -591.4057 -604.9243 -621.8101 

                       HC                         EM 
 "uniform spherical (EI)" "uniform variance (EEE)" 

 sample noise equal 
      F     F     F 
> plot(bicvals1)
> sumry1 <- summary(bicvals1, iris.matrix)
> sumry1

 best classification:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3
 [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 2 4 2 4 2 2 4 4
[112] 4 4 4 4 4 2 2 2 3 4 4 2 4 4 2 4 4 4 2 2 2 4 3 2 4 4 2 4 4 4 4 4 4 4 4 4 4
[149] 4 4

 uncertainty:
   0%  25%          50%          75%      100%
    0    0 2.93774e-06 0.001037691 0.5021521

 best BIC values:
         4         5
 -591.4057 -604.9243

 model:
                       HC                         EM
 "uniform spherical (EI)" "uniform variance (EEE)"

 sample noise equal
      F     F     F
```

With this model and initialization scheme, a 4-cluster solution is preferred (note, however, that higher BIC values were obtained for other models with fewer clusters in the `emclust` example above).

Because of the size of the objects involved, optimal parameter and z values are not available from `emclust` or `emclust1`. Instead, they can be obtained through `summary` functions, which have arguments allowing the summarizing information to be restricted to a subset of the number of clusters (and models, in the case of `emclust`). The `summary` functions require the data to be supplied as an argument in addition to the output from `emclust` and `emclust1`. In the above example, the best classification (according to the BIC) is recovered from `summary` by default. We can also see that the next best classification is the 3 group classification with the constant-shape model `VEV`. Parameters associated with this classification can be recovered via `summary` as follows:

```
> nextbest <- summary(bicvals, iris.matrix, nclus = 3, modelid = "VEV")
```

Those who want to extend the methods provided in `MCLUST` to suit their own special needs can do so via `unclass`, which recovers the underlying list structure of an S-PLUS objects:

```
> unclass(bicvals)
> unclass(sumry)
```

For large data sets, there is the option of using only a subset of the data in the initial hierarchical clustering phase in both `emclust1` and `emclust`.

For a complete analysis, it may be desirable to try various models, initialization strategies for EM, permutations or subsets of the observations, and/or to perturb the data, to see if the classification remains stable. Scaling or otherwise transforming the data may also affect the results. It is advisable to examine the data beforehand, in case (for example) the dimensions can be reduced due to highly correlated variables.

Finally, it is important to take into account numerical issues in cluster analysis. The EM computations break down when the covariance corresponding to one or more components becomes ill-conditioned (singular or nearly singular). In general they cannot proceed if clusters contain only a few observations or if the observations they contain are very nearly colinear. If EM for a model having a certain number of components is applied to a mixture in which there are actually fewer groups, then it may fail due to ill-conditioning.

The EM functions in `MCLUST` compute and monitor an estimate of the reciprocal of the condition number of the covariance matrices (the condition number is the ratio of the largest to the smallest eigenvalue). Reciprocal condition estimates fall in the interval $(0, 1)$, and values very near zero indicate ill-conditioning. Computations will terminate with a warning message and return missing values (`NA`s) if a reciprocal condition estimate falls below a prescribed tolerance. Computations are less reliable for ill-conditioned problems, and may cause anomalies before a computation reaches the point of actual failure. Reciprocal condition estimates are available from the output for all functions in `MCLUST` (including `bic`) that involve EM. The reciprocal condition estimates for the `emclust` example given above may be obtained as follows:

```
> attr(bicvals, "rcond")
             1           2           3           4           5            6
VVV 0.01076579 0.005078553 0.003693874 0.002470445 0.004458351 1.489486e-05
EEV 0.01076579 0.013875312 0.019530343 0.024492144 0.019325741 1.845156e-02
VEV 0.01076579 0.024401829 0.029235848 0.029920233 0.028300937 2.760809e-02
```

In this example, the 6-group unconstrained (`VVV`) model has at least one ill-conditioned covariance matrix. The reciprocal condition estimate is always at least as large as the true reciprocal condition number, so that if ill-conditioning is suspected but not necessarily indicated by these estimates, it can be further checked by obtaining the covariance matrix using `mstep` and computing its eigenvalues. For more information about condition estimates and their interpretation, see [10].

7 Discriminant Analysis

The `MCLUST` function `estep` implementing the E-step of EM for Gaussian mixtures can be used for discriminant analysis. It takes as input parameters of a Gaussian mixture, which could come from the result of cluster analysis via `MCLUST`, for example. As an illustration, consider the following two-dimensional data set which consists of two intersecting Gaussian clusters centered at the origin:

```
> n <- 100
> set.seed(0)  # for reproducibility
> x <- rbind(matrix(rnorm(n * 2), n, 2) %*% diag(c(1, 9)),
            (matrix(rnorm(n * 2), n, 2) %*% diag(c(1, 9)))[,2:1])
```

We first perform a cluster analysis via `emclust`:

```
> xbic <- emclust(x)
> plot(xbic)  # plot bic
> sumry <- summary(xbic, x) # recover information for best classification
> clpairs(x, ztoc(sumry$z), symbols = c("1", "2")) # plot classes
```

To classify new data points (0,5) and (5,0), we first compute the parameters corresponding to the bets classification from `emclust` using `mstep`:

```
> pars <- mstep( x, modelid = attr(sumry, "modelid"), z = sumry$z),
```

and then determine conditional probabilities for the new observaton with `estep`:

```
> znew <- estep( matrix(rbind(c(0,5),c(5,0)), nrow = 2),
                 modelid = attr(sumry, "modelid"),
                 mu = pars$mu, sig = pars$sig, pro = pars$pro)
> znew
> ztoc(znew) # classification of the new data points
> 1  - apply(znew, 1, max) # uncertainty in classification
```

Figure 4 shows a plot of the initial clustering from `emclust` along with the two additional points classified via discriminant analysis with `estep`.

8 Large Data Sets

The discrimination capability of MCLUST can be used for classification of large data sets. First, cluster analysis with the methodolgy of `emclust` and `emclust1` can be performed on a subset of the data, and the optimal parameters calculated via `mstep`. Then the remaining data points can then be classified (in reasonable sized blocks) using `estep` as illustrated in section (7).

Functions `emclust` and `emclust1` also include a provision for using a subsample of size k of the data in the hierarchical clustering phase before applying EM to the full data set. This strategy is often adequate for data sets that are not extremely large.

9 High Dimensional Data

Models in which the orientation is allowed to vary between clusters (e.g. `EEV`, `VEV`, `VVV`), have $\mathcal{O}(d^2)$ parameters per cluster, where d is the dimension of the data. For this reason, `MCLUST` may not work well or may otherwise be inefficient for these models when applied to high-dimensional data. It may still be possible to analyze such data with `MCLUST` by restriction to models with fewer parameters (either the spherical models `EI` and `VI` or the constant variance model `EEE` in MCLUST), or else by applying a dimension-reduction technique such as principal components.

Note that none of the methods in `MCLUST` can handle datasets in which the number of observations is smaller than the data dimension.

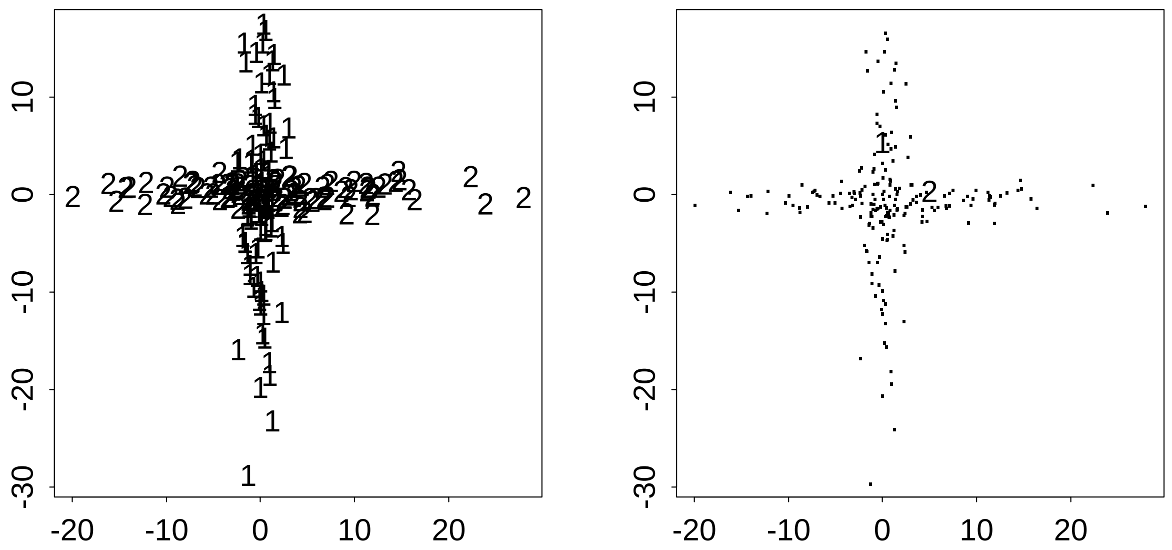

Figure 4: Left: best classification according to `emclust` of a simulated data set. Right: Classification of two additional data points via discriminant analysis using `estep`.

References

[1] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[2] J. G. Campbell, C. Fraley, F. Murtagh, and A. E. Raftery. Linear flaw detection in woven textiles using model-based clustering. *Pattern Recognition Letters*, 18:1539–1548, December 1997.

[3] G. Celeux and G. Govaert. Gaussian parsimonious clustering models. *Pattern Recognition*, 28:781–793, 1995.

[4] A. Dasgupta and A. E. Raftery. Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American Statistical Association*, 93(441):294–302, March 1998.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood for incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[6] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[7] C. Fraley. Algorithms for model-based Gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20(1):270–281, 1998.

[8] C. Fraley and A. E. Raftery. How many clusters? Which clustering method? - Answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[9] H. P. Friedman and J. Rubin. On some invariant criteria for grouping data. *Journal of the American Statistical Association*, 62:1159–1178, 1967.

[10] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins, 3rd edition, 1996.

[11] H. Jeffreys. *Theory of Probability.* Clarendon, 3rd edition, 1961.

[12] R. E. Kass and A. E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90:773–795, 1995.

[13] G. J. McLachlan and K. E. Basford. *Mixture Models : Inference and Applications to Clustering.* Marcel Dekker, 1988.

[14] G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* Wiley, 1997.

[15] S. Mukerjee, E. D. Feigelson, G. J. Babu, F. Murtagh, C. Fraley, and A. E. Raftery. Three types of gamma ray bursts. *The Astrophysical Journal*, 508:314–327, November 1998.

[16] F. Murtagh and A. E. Raftery. Fitting straight lines to point patterns. *Pattern Recognition*, 17:479–483, 1984.

[17] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6:461–464, 1978.

[18] A. J. Scott and M. J. Symons. Clustering methods based on likelihood ratio criteria. *Biometrics*, 27:387–397, 1971.

[19] J. H. Ward. Hierarchical groupings to optimize an objective function. *Journal of the American Statistical Association*, 58:234–244, 1963.