

clustord Convergence Tutorial

Louise McMillan

Contents

Introduction	1
How to check convergence	1
Why is convergence important?	3
What to do if the algorithm has not converged	3
What if it still doesn't converge?	4

Introduction

The clustering algorithm in `clustord` is based on the EM algorithm. This is an iterative algorithm for finding maximum likelihood estimates. It is not possible to predict in advance how many iterations the algorithm will require to converge on a solution for a given dataset and clustering settings. But it will stop running after a specified number of iterations. The user can change this value, but it defaults to a middlingly low number (50) in order to prevent the algorithm from running too long on a typical run.

It is important to always check whether the algorithm has converged at the end of a given run.

How to check convergence

Let's use a simulated dataset to demonstrate how to check for convergence using the simplest row clustering model.

```
library(clustord)
set.seed(30)
long_df_sim <- data.frame(Y=factor(sample(1:3, 5*30, replace=TRUE)),
                          ROW=factor(rep(1:30, times=5)), COL=rep(1:5, each=30))
fit <- clustord(Y ~ ROWCLUST, model="POM", RG=2, long_df=long_df_sim,
               nstarts=2, control_EM = list(maxiter=2))
#> Converting factor ROW to numeric.
#> EM algorithm has not converged. Please try again, or with a different random seed, or w
```

You can check the convergence in three ways.

1. Display the results object:

```
fit
#>
#> Call:
#> clustord(formula = Y ~ ROWCLUST, model = "POM", RG = 2, long_df = long_df_sim,
#>   control_EM = list(maxiter = 2), nstarts = 2)
#>
#> Clustering mode:
#>   row clustering
#>
#> Converged:
#>   FALSE
#>
#> Cluster sizes:
#> Row clusters:  30 0
```

2. Display the summary of the results:

```
summary(fit)
#>
#> Call: clustord(formula = Y ~ ROWCLUST, model = "POM", RG = 2, long_df = long_df_sim,
#>   control_EM = list(maxiter = 2), nstarts = 2)
#>
#> Clustering mode: row clustering
#> Model: POM
#>
#> Converged: FALSE
#> AIC: 337.5446
#>
#> BIC: 349.5871
#>
#>
#> Cluster sizes:
#> Row clusters:  30 0
#> Average probabilities of membership:  0.94 0
#>
#> Parameter estimates:
#> $mu
#>   mu_1   mu_2
#> -0.5103  0.9408
#>
#> $rowc
#> rowc_1 rowc_2
#>  0.2454 -0.2454
```

3. Fetch the convergence check from the results object:

```
fit$EMstatus$converged
#> [1] FALSE
```

The convergence will be shown as TRUE if the algorithm has converged, and FALSE if it has not. The output during the completion of the algorithm run will also display a message indicating whether or not it has converged.

Why is convergence important?

If the algorithm has not converged, that means that it has not found the optimum solution. Therefore, the parameter estimates will not be accurate and the cluster memberships may not be accurate either.

What to do if the algorithm has not converged

If the algorithm has not converged, then the best solution is to keep running it from where it stopped, until it does converge. That is, do not start the algorithm again from scratch, but use the output from the run that stopped without converging as the input for a new run.

In order to do this, use the `rerun()` function. This allows you to pass in a previous run and use it as the starting point of a new run. You supply the output of the previous run, and the dataset, and any options you want in `control_EM`, which may include changing the number of iterations.

For diagnostic purposes, you also have the option of changing the `verbose` setting for the new run or supplying a different `control_optim` list. `control_optim` has an entry called `trace` which can make `optim()` produce much more verbose output during the M-step, but most users will not need this functionality.

For now, we will just ask for more iterations for the next run using `control_EM` since the previous run only used 5.

```
fit_continued <- rerun(fit, long_df=long_df_sim, control_EM=list(maxiter=20))
#> Converting factor ROW to numeric.
#> EM algorithm has successfully converged.
summary(fit_continued)
#>
#> Call: clustord(formula = Y ~ ROWCLUST, model = "POM", RG = 2, long_df = long_df,
#>   init_parvec = init_parvec, init_pi = init_pi, init_kappa = init_kappa,
#>   control_EM = control_EM, control_optim = control_optim, verbose = FALSE)
#>
#> Clustering mode: row clustering
#> Model: POM
#>
#> Converged: TRUE
#> AIC: 337.4933
#>
#> BIC: 349.5358
#>
#>
#> Cluster sizes:
#> Row clusters: 30 0
#> Average probabilities of membership: 0.94 0
#>
#> Parameter estimates:
#> $mu
#>   mu_1   mu_2
#> -0.5740 0.8747
#>
#> $rowc
#> rowc_1 rowc_2
#> 0.1696 -0.1696
```

What if it still doesn't converge?

If you still find that the algorithm does not converge within e.g. 500 iterations, then it would be worth running the algorithm from scratch (i.e. without supplying `init_parvec`) and increasing the number of starts in order to improve the chances of finding a better starting point that might allow the algorithm to converge faster. You should use at least 20 starting points for any clustering structure more complex than $Y \sim \text{ROWCLUST}$ (the simplest model).

Also note that if you have multiple cores available on your machine, or access to a high-performance computing cluster with many CPUs, you can leverage the `parallel_starts` option in `clustord` to speed up the process of running many starting points.