BIOINFORMATICS APPLICATIONS NOTE Vol. 20 no. 9 2004, pages 1453–1454
DOI: 10.1093/bioinformatics/bth078

Open source clustering software

M.J.L. de Hoon[1,*], S. Imoto[1], J. Nolan[2] and S. Miyano[1]

[1]Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo, 108-8639 Japan and [2]University of California, Santa Cruz Extension in Silicon Valley, 10420 Bubb Road, Cupertino, CA 95014, USA

Received on June 17, 2003; revised on July 27, 2003, accepted on July 31, 2003
Advance Access publication February 10, 2004

ABSTRACT

Summary: We have implemented *k*-means clustering, hierarchical clustering and self-organizing maps in a single multipurpose open-source library of C routines, callable from other C and C++ programs. Using this library, we have created an improved version of Michael Eisen's well-known Cluster program for Windows, Mac OS X and Linux/Unix. In addition, we generated a Python and a Perl interface to the C Clustering Library, thereby combining the flexibility of a scripting language with the speed of C.

Availability: The C Clustering Library and the corresponding Python C extension module Pycluster were released under the Python License, while the Perl module Algorithm::Cluster was released under the Artistic License. The GUI code Cluster 3.0 for Windows, Macintosh and Linux/Unix, as well as the corresponding command-line program, were released under the same license as the original Cluster code. The complete source code is available at http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster. Alternatively, Algorithm::Cluster can be downloaded from CPAN, while Pycluster is also available as part of the Biopython distribution.

Contact: mdehoon@ims.u-tokyo.ac.jp

INTRODUCTION

Clustering techniques are widely used in gene expression data analysis. By grouping genes together based on the similarity in their gene expression profile, we may find functionally related genes, and potentially the function of genes whose role is presently unknown.

Several programs are currently available to analyze gene expression data. The closed source Java program GeneCluster (Tamayo *et al*., 1999; Golub *et al*., 1999) implements two-dimensional (2D) self-organizing maps (SOMs) (Kohonen, 1990, 2001). The widely used Cluster/TreeView code (Eisen *et al*., 1998), written in C++ for the Microsoft Windows platform, focuses on hierarchical clustering methods, while 1D SOMs, principal component analysis and *k*-means clustering are also implemented. The source code for Cluster/TreeView, minus the routines that are covered by the Numerical Recipes license (Press *et al*., 1992), is available for the academic and non-profit community.

Clustering routines suitable for usage with scripting languages such as Python (http://www.python.org) or Perl (http://www.perl.com) are often not available. Scripting languages are heavily used in other fields of bioinformatics, as they provide a flexible, platform-independent, mature and easily extendible basis for data analysis.

THE C CLUSTERING LIBRARY

We have implemented hierarchical (pairwise single-, average-, maximum- and centroid-linkage) clustering techniques, SOMs on a 2D rectangular grid and the *k*-means clustering algorithm as a library of C routines. The similarity between gene expression data can be measured by the Pearson correlation and the uncentered correlation, the city-block distance, the Euclidean distance, the harmonically summed Euclidean distance, Spearman's rank correlation and Kendall's τ. All the clustering routines and distance measures commonly used in expression data analysis are thus available in a single open-source library.

The clustering library was written in ANSI-compliant C for reasons of portability, interoperability with other languages (in particular scripting languages), its speed compared with Java and pure Python or Perl codes, and compiler availability. An example C program that makes use of the C Clustering Library is included with the source code.

Particular attention was paid to the implementation of the *k*-means routine (Hastie *et al*., 2001; Tavazoie *et al*., 1999). The algorithm starts from a random initial clustering, then iterates by calculating the cluster centroids and reassigning elements to the cluster with the closest centroid, and is halted when no more reassignments are made. The aim is to find the clustering solution that minimizes the within-cluster sum of distances. As the initial clustering is random, a different (and probably suboptimal) clustering solution may be found each time the algorithm is executed. To find the optimal *k*-means clustering solution, the algorithm should be repeated many times with different initial random clusterings

*To whom correspondence should be addressed.

(Hastie *et al*., 2001). The *k*-means routine as implemented in the C Clustering Library automatically executes such repetitions of the algorithm, where the number of repetitions is specified by the user. The clustering solution with the smallest within-cluster sum of distances is returned, as well as the number of runs in which that solution was found. If it was found in only a few runs of the algorithm, better *k*-means clustering solutions are likely to exist. In that case, the *k*-means routine should be executed again with a larger number of repetitions.

We noticed that for some sets of initial cluster assignments, after a few iterations the same clustering solution reappears periodically, with periods of up to 10 iterations or so. As a result, the algorithm does not converge. The *k*-means algorithm in the C Clustering Library therefore checks for periodically reappearing clustering solutions, and halts the iteration if such periodicity is detected.

CLUSTER 3.0 FOR WINDOWS, MACINTOSH AND LINUX/UNIX

Cluster 3.0 is an improved version of the Cluster program (Eisen *et al*., 1998) using the C Clustering Library, thereby avoiding the need for the proprietary routines in Numerical Recipes (Press *et al*., 1992). GUI versions for Windows, Mac OS X and Linux/Unix platforms, as well as a command-line version are available. Cluster 3.0 provides several improvements over the original Cluster code, such as a choice of distance measures for *k*-means clustering and SOMs, automatic file format checking when loading a data file, as well as improved accuracy and memory usage. In addition, Cluster 3.0 provides periodicity checks and automatic repetitions of the *k*-means clustering algorithm, as described above, which facilitates the search for the optimal clustering solution.

The complete source code for Cluster 3.0, as well as an installer for Windows, is available from our Website. On all three platforms, Cluster 3.0 can be compiled with the GNU C compiler (http://gcc.gnu.org); no commercial compiler is required.

USING THE C CLUSTERING LIBRARY WITH PYTHON OR PERL

Analysis tools for gene expression data are usually written in the form of stand-alone GUI programs. In other fields of bioinformatics, scripting languages such as Python (http://www.python.org) or Perl (http://www.perl.com) are heavily used, as exemplified by the Biopython (http://www.biopython.org) and Bioperl (http://www.bioperl.org) projects.

Data analysis using scripting languages is performed by issuing a series of commands to an interpreter. Commands can be either issued individually by hand or stored in a script file. Scripting languages and their extensions typically already contain routines for file input and output, database access, data filtering and processing and visualization. This greatly reduces the code development time, as only the routines specific for bioinformatics need to be developed. Such routines are typically stored in modules written in the scripting language itself or in a compiled language such as C or Fortran. By combining native scripting commands and calls to external modules, the entire data analysis can be captured in a single script. The Numerical Python project (Ascher *et al*., 2001, http://sourceforge.net/projects/numpy/) makes Python particularly suitable for analyzing numerical data, such as those produced in gene expression experiments.

To make the advantages of scripting languages available for gene expression data analysis, we created the Python C extension module Pycluster, as well as the Perl module Algorithm::Cluster, such that the routines in the C Clustering Library can be called directly from these scripting languages. Pycluster is available at our Website as a source distribution and as an installer for Windows; it has also been integrated with Biopython. For Perl users, we created the module Algorithm::Cluster. This module is available from our Website, as well as from the Comprehensive Perl Archive Network CPAN (http://www.cpan.org).

REFERENCES

Ascher,D., Dubois,P.F., Hinsen,K., Hugunin,J. and Oliphant,T. (2001) *Numerical Python*. Lawrence Livermore National Laboratory.

Eisen,M., Spellman,P., Brown,P. and Botstein,D. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci., USA*, **95**, 14863–14868.

Golub,T.R., Slonim,D.K., Tamayo,P., Huard,C., Gaasenbeek,M., Mesirov,J.P., Coller,H., Loh,M.L., Downing,J.R., Caligiuri,M.A., Bloomfield,C.D. and Lander,E.S. (1999) Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Hastie,T., Tibshirani,R. and Friedman,J. (2001) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York.

Kohonen,T. (1990) The self-organizing map. *Proc. IEEE*, **78**, 1464–1480.

Kohonen,T. (2001) *Self-Organizing Maps*, 3rd edn. Springer-Verlag, Berlin.

Press,W.H., Teukolsky,S.A., Vetterline,W.T. and Flannery,B.P. (1992) *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK.

Tamayo,P., Slonim,D., Mesirov,J., Zhu,Q., Kitareewan,S., Dmitrovsky,E., Lander,E. and Golub,T. (1999) Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation. *Proc. Natl Acad. Sci., USA*, **96**, 2907–2912.

Tavazoie,S., Hughes,J.D., Campbell,M.J., Cho,R.J. and Church,G.M. (1999) Systematic determination of genetic network architecture. *Nat. Genet.*, **22**, 281–285.