Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Gene expression

COBRAC: a fast implementation of convex biclustering with compression

Haidong Yi [1,*], Le Huang [2], Gal Mishne[3], and Eric C. Chi [4,*]

[1]Department of Computer Science, [2]Curriculum in Bioinformatics & Computational Biology, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599
[3]Halıcıoğlu Data Science Institute, University of California, San Diego, La Jolla, CA, 92093
[4]Department of Statistics, North Carolina State University, Raleigh, NC, 27607

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Biclustering is a generalization of clustering used to identify simultaneous grouping patterns in observations (rows) and features (columns) of a data matrix. Recently, the biclustering task has been formulated as a convex optimization problem. While this convex recasting of the problem has attractive properties, existing algorithms do not scale well. To address this problem and make convex biclustering a practical tool for analyzing larger data, we propose an implementation of fast convex biclustering called COBRAC to reduce the computing time by iteratively compressing problem size along the solution path. We apply COBRAC to several gene expression datasets to demonstrate its effectiveness and efficiency. Besides the standalone version for COBRAC, we also developed a related online web server for online calculation and visualization of the downloadable interactive results.

Availability: The source code is available at `https://github.com/haidyi/cvxbiclustr` and the web server is available at `http://cvxbiclustr.ericchi.com`.

Contact: haidyi@cs.unc.edu or eric_chi@ncsu.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Hierarchical clustering is a fundamental task in many bioinformatics problems ranging from cytogenetics to population genetics. Many clustering algorithms, including the popular k-means method, formulate the clustering task as a non-convex optimization problem. A serious issue with solving non-convex optimization problems is the presence of suboptimal local minima which often leads to suboptimal clustering assignments. This issue vanishes, however, when using the convex clustering algorithm introduced by Pelckmans *et al.* (2005). Solutions to the convex clustering problem trace out a tree organization of the data from the data points located at the leaves to a root as a nonnegative tuning parameter γ increases from zero to a finite positive value maximal tuning parameter value $\gamma_{\max}$. The convex clustering tree has several attractive properties, in particular the recovered hierarchical clustering is guaranteed to be stable to noise in the sense that small perturbations in the data are guaranteed to not lead to disproportionately large fluctuations in the output tree (Chi *et al.*, 2017). Convex clustering has proven useful for revealing hierarchical organizations in data from a wide range of applications from genetics (Chen *et al.*, 2015) to text mining (Weylandt *et al.*, 2020) and has been extended to biclustering case by Chi *et al.* (2017).

From a computational perspective, several optimization algorithms have been proposed for solving the convex clustering problem ranging from a variety of first order methods, (Hocking *et al.*, 2011; Chi and Lange, 2015; Panahi *et al.*, 2017), to second order methods (Yuan *et al.*, 2018), as well as a novel algorithm regularization path approach which approximates the solution path up to an arbitrarily small error with extremely inexact alternating direction method of multiplier updates (Weylandt *et al.*, 2020). Although these methods are successful for solving convex clustering problem, with the exception of the work by Weylandt *et al.* (2020), none of them are designed to solve the optimization problem with the goal of efficiently generating a final dendrogram.

In this paper, we introduce the COBRAC algorithm which iteratively performs two steps: (1) solves a weighted convex biclustering problem; (2) compresses the problem size. Compared with Weylandt *et al.*

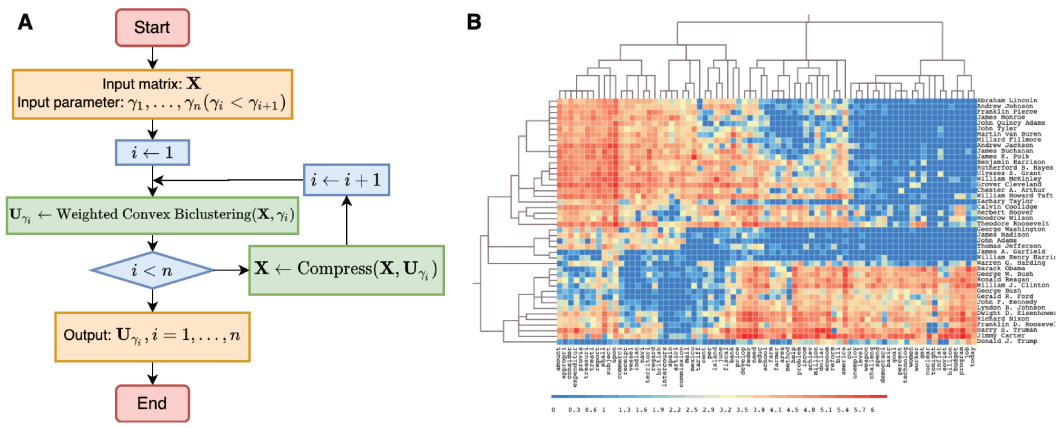

Fig. 1. (A) Flow chart of the algorithm. (B) The heatmap of the U.S. president dataset with 44 observations (rows) and 75 features (columns) and clustering dendrogram generated by COBRAC using $\gamma = 3, 5, 10, 20, 50, 100, 150, 200, 500, 1000$.

(2020), COBRAC stores the solution over a much smaller set of γ, leading to significantly smaller memory usage to generate the clustering dendrograms. In addition, the compression procedure in COBRAC can be easily incorporated to other convex clustering algorithms to further accelerate computations. Our contributions to convex biclustering are as follows: (i) We present COBRAC, a fast implementation of convex biclustering that can identify possible biclusters and generate the corresponding row and column clustering dendrograms; (ii) While a similar strategy has been proposed for convex clustering before (Hocking *et al.*, 2011), to our best of knowledge, COBRAC is the first implementation of convex biclustering that utilizes this compression strategy to generate the full path solution; (iii) We also develop a webserver for users to run COBRAC online and visualize the dynamic clustering process along the rows and columns of a matrix.

2 Algorithm

Figure 1**A** summarizes COBRAC at a high level. COBRAC solves a sequence of weighted convex biclustering problem over a series of parameters $\gamma_1, \ldots, \gamma_n$, and uses the solutions $\mathbf{U}_{\gamma_1}, \ldots, \mathbf{U}_{\gamma_n}$ to generate row and column clustering dendrograms. The core idea of COBRAC is to reduce the size of matrix $\mathbf{X}$ iteratively because when some rows and columns are clustered at a given γ, they will remain clustered for all larger γs (Chi and Steinerberger, 2019). After compression, the objective function is still a weighted convex biclustering problem, and the only difference is the size of $\mathbf{X}$ and weight coefficients. For example, suppose for a given parameter γ, the rows and columns of $\mathbf{X}$ have been clustered into r and c clusters respectively. Then, $\mathbf{X}$ will be compressed into an $r \times c$ matrix for next larger γ in the series, enabling the efficient computation of the entire solution path. For detailed formulation and derivation, see Section 1 in Supplementary data.

3 Implementation

The COBRAC program is written in C/C++ and a Python wrapper is also provided for easy usage. In our implementation, we optimize the dual problem of weighted convex biclustering using an accelerated proximal gradient algorithm FASTA (Goldstein *et al.*, 2014). The input of COBRAC is a data matrix in csv format and a list of γ to run the convex biclustering algorithm. The output of COBRAC is a json file that contains both the solution matrices for different γs and the corresponding clustering dendrograms. In addition to the standalone program, we also develop a web server at `http://cvxbiclustr.ericchi.com` for people to run COBRAC and visualize the results online.

4 Evaluation

As an example, we run COBRAC on the President dataset that contains log-transformed word counts of the 75 most variable words taken from the aggregated major speeches of the 44 U.S. presidents through mid-2018 (Weylandt *et al.*, 2020). Figure 1**B** shows that COBRAC identifies a strong biclustering structure among word usage and presidents. To investigate the speed-ups attainable by introducing compression, we also run COBRAC on several different genomic expression data including Lung100, Lung500 (Lee *et al.*, 2010) and TCGA Breast (Koboldt *et al.*, 2012) datasets. COBRAC achieves a 2.5∼5x speed-up to solve the entire solution path over these data (Table 1 in Supplementary data). The detailed descriptions of the data and experimental set-up are provided in Supplementary data. The heatmaps with dendrogram of Lung100 and Lung500 are provided in our webserver and Supplementary data (Figures 2&3). Furthermore, we also test the performance of COBRAC using different number of CPUs. The results show that parallel computing can further reduce the running time by nearly 40% (Figure 1 in Supplementary data). All the example data are available on our webserver for users to reproduce the results.

Funding

This work has been supported by in part by National Science Foundation grant DMS-1752692 and by the National Institutes of Health grant no. R01EB026936 and grant no. R01GM135928.

References

Chen, G. K. *et al.* (2015). Convex clustering: An attractive alternative to hierarchical clustering. *PLoS Computional Biology*, **11**(5), e1004228.

Chi, E. C. and Lange, K. (2015). Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, **24**(4), 994–1013.

Chi, E. C. and Steinerberger, S. (2019). Recovering trees with convex clustering. *SIAM Journal on Mathematics of Data Science*, **1**(3), 383–407.

Chi, E. C. *et al.* (2017). Convex biclustering. *Biometrics*, **73**(1), 10–19.

Goldstein, T. *et al.* (2014). A field guide to forward-backward splitting with a fasta implementation. *arXiv preprint arXiv:1411.3406*.

Hocking, T. *et al.* (2011). Clusterpath: an algorithm for clustering using convex fusion penalties. In *International Conference on Machine Learning*, pages 745–752.

Koboldt, D. C. *et al.* (2012). Comprehensive molecular portraits of human breast tumours. *Nature*, **490**(7418), 61–70.

Lee, M. *et al.* (2010). Biclustering via sparse singular value decomposition. *Biometrics*, **66**(4), 1087–1095.

Panahi, A. *et al.* (2017). Clustering by sum of norms: Stochastic incremental algorithm, convergence and cluster recovery. In *International Conference on Machine Learning*, pages 2769–2777.

Pelckmans, K. *et al.* (2005). Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*.

Weylandt, M. *et al.* (2020). Dynamic visualization and fast computation for convex clustering via algorithmic regularization. *Journal of Computational and Graphical Statistics*, **29**(1), 87–96.

Yuan, Y. *et al.* (2018). An efficient semismooth Newton based algorithm for convex clustering. In *International Conference on Machine Learning*, pages 5718–5726.