Pattern Recognition 44 (2011) 236–251

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A spectral approach to clustering numerical vectors as nodes in a network

Motoki Shiga [a,b,*], Ichigaku Takigawa [a,b], Hiroshi Mamitsuka [a,b]

[a] Bioinformatics Center, Kyoto University, Gokasho, Uji 611-0011, Japan
[b] Institute for Bioinformatics Research and Development (BIRD), Japan Science and Technology Agency (JST), Japan

ARTICLE INFO

Article history:
Received 27 January 2010
Received in revised form
28 May 2010
Accepted 7 August 2010

Keywords:
Semi-supervised clustering
Heterogeneous data
Data integration
Spectral clustering

ABSTRACT

We address the issue of clustering examples by integrating multiple data sources, particularly numerical vectors and nodes in a network. We propose a new, efficient spectral approach, which integrates the two costs for clustering numerical vectors and clustering nodes in a network into a matrix trace, reducing the issue to a trace optimization problem which can be solved by an eigenvalue decomposition. We empirically demonstrate the performance of the proposed approach through a variety of experiments, including both synthetic and real biological datasets.

1. Introduction

We address the issue of clustering examples by integrating heterogeneous data sources, each being given independently. In particular, we focus on the case in which examples are numerical vectors and at the same time nodes in a network, where two nodes connected by an edge are more likely to be in the same cluster. This situation is general, being found in a lot of applications.

A typical application is web page clustering. Web pages can be clustered by their contents, say term frequencies, based on the assumption that if the content of a page is similar to that of another, these pages can be in the same cluster. At the same time, web pages are hyperlinked together, forming a network, in which nodes and edges correspond to web pages and hyperlinks, respectively, and can be clustered based on the hyperlink connectivity. We assume that a given graph and a given set of numerical vectors are independently observed, meaning that the contents of web pages and their links are independently generated. Under this assumption, combining texts with hyperlinks is helpful for clustering web pages [1].

Another application is gene clustering, which is useful for annotating gene functions. That is, once genes are clustered, we can assign functions to unknown genes by using known genes in the same cluster. Understanding gene function is important, since there are still a lot of functionally unknown genes. For example, the MIPS database categorizes genes (or open reading frames (ORFs)) into three classes: characterized, uncharacterized and dubious, and the MIPS database (as of January 25, 2010) shows 944 uncharacterized ORFs (14% of all 6607 genes) and 811 dubious ORFs (12% of all) even for *Saccharomyces cerevisiae*, a most well-investigated species in biology [2]. Genes are expressed and then function in a cell. Currently quantitative expression of thousands of genes can be measured simultaneously by using a technology in genetic engineering, called cDNA microarray. Thus by repeating the experiment of cDNA microarray under various conditions, we can have numerical vectors (generally called profiles) of genes to do clustering genes in terms of expressions in a cell. However, cDNA microarray data is very noisy and unreliable. Naturally we need another data source for more precise gene clustering, and we can have more reliable information on genes as a gene network. For example, literature information provides us with the co-occurrence frequencies of genes in medical documents which can be turned into a network of genes with the frequencies as edge weights. Similarly, metabolic or gene regulatory networks which are generated from literature are much more reliable than microarray data.

Our approach for this issue is based on spectral clustering, which has been popularly used for graph partitioning [3,4]. Standard (graph-cut) criteria of graph partitioning are *ratio cut* [5] and *normalized cut* [6], by which the number of inter-cluster edges is minimized, keeping clusters balanced. For usual graph cuts, the clustering *cost*, which should be minimized, is a matrix trace of *Rayleigh quotient*, written by using a cluster assignment matrix and a node affinity matrix. We thus can first define the clustering cost of each of two data types: numerical vectors and a network,

* Corresponding author at: Bioinformatics Center, Kyoto University, Gokasho, Uji 611-0011, Japan. Tel.: +81 774 38 3313; fax: +81 774 38 3037.
E-mail addresses: shiga@kuicr.kyoto-u.ac.jp (M. Shiga), takigawa@kuicr.kyoto-u.ac.jp (I. Takigawa), mami@kuicr.kyoto-u.ac.jp (H. Mamitsuka).

0031-3203/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2010.08.010

as a matrix trace of Rayleigh quotient, by which two inputs can be combined naturally. We then follow the procedure of a standard spectral approach, which converts the trace optimization problem into a minimization problem with a constraint which can be solved by Lagrange multipliers, resulting in an eigenvalue problem. Resultant eigenvectors are used for final cluster assignment by a simpler clustering method such as *k*-means.

Our problem setting is closely related with that of *semi-supervised clustering* [7], where examples (or nodes in a graph) are clustered under some constraints, such as *must-link* which makes two nodes be in the same cluster, implying that some approaches of semi-supervised clustering can be applied to our problem setting. Kernel *k*-means is applicable to semi-supervised clustering by using a kernel which can be generated from two types of data: numerical vectors and a network corresponding to constraints [8]. This approach, which can be applied to our problem setting as well, is a competing method to which the most attention should be paid. However, the performance of kernel *k*-means is unclear even in graph partitioning, because a graph cut which is generated from a given node affinity matrix over a graph may neither be a kernel (i.e. a positive semi-definite matrix) nor a well-considered kernel even if it is positive semi-definite.

In this light, the main contribution of this paper can be summarized into three folds: (1) For graph partitioning, we empirically confirm the performance advantage of spectral clustering in accuracy and running time over other methods including kernel *k*-means. Experimental results demonstrate that spectral clustering is the most promising approach among the competing methods for graph partitioning as well as our problem setting. (2) We propose a new method for clustering examples, represented by numerical vectors and nodes in a network. The proposed method directly integrates the two costs: (1) clustering numerical vectors and (2) clustering nodes in a network. Our approach of combining two costs allows to avoid possible problems in existing semi-supervised clustering, such as the issue of positive semidefiniteness in kernel *k*-means. (3) In our problem setting, we empirically examine a variety of costs for clustering nodes in a network in the proposed method, comparing with other approaches such as those for semi-supervised clustering. This experiment reveals that our spectral approach outperforms other methods throughout all settings in our experiment.

2. Related work

Graph partitioning or grouping examples over associations is an NP-complete problem to which considerable work have been devoted for more than twenty years [9,10,6]. Currently spectral clustering is the most well-accepted approach for graph partitioning, with a variety of publications (including reviews [3,4]), software [11] and applications [6,12]. Spectral clustering has a lot of variations in algorithms (such as hierarchical, recursive bipartitioning [11,13]) and in graph cut criteria (such as ratio cut [5], normalized cut [6], Min-Max cut [14] and other more recent criteria [15,13,16]). Another approach for graph partitioning is kernel *k*-means [17], which can be connected to spectral clustering [18]. However, a graph-cut from a given affinity matrix can neither be a kernel (i.e. a positive semi-definite matrix) nor be an appropriate kernel even if it is positive semi-definite. A possible way to make an affinity matrix a semi-definite matrix is a diagonal shift [19]. Another approach, being related with kernel *k*-means, is first variation [20,21], which attempts to minimize the same cost function of kernel *k*-means.

Our problem setting is very similar to that of semi-supervised clustering (or constrained clustering), where examples are clustered under some given constraints. Constraints are two types: *hard constraints*, where constraints must be always kept [22], and *soft constraints*, being more effective in real applications. An existing approach of semi-supervised clustering is probabilistic model-based learning, such as hidden Markov random field [7]. However, this approach usually needs a large amount of computational cost, making it very hard to apply to a practical application. It is then suggested to apply kernel *k*-means to semi-supervised clustering [8], i.e. clustering over a kernel, which was generated from two matrices, one from numerical vectors and the other from constraints. However, as mentioned in Introduction already, it is unclear that kernel *k*-means works properly even on the input with a graph only, i.e. graph partitioning. Another existing approach is to use constraints to alter the corresponding values in the affinity matrix of given numerical vectors [23]. For example, the affinity between two examples is set at some maximum value if there is a must-link between these examples [23]. In other words, this is equivalent to taking the *maximum* of two affinity matrices of given examples and network constraints. Similarly simple operations can be considered on combining two affinity matrices, such as the *weighted sum*, the *union* or the *intersection*, and perform a clustering algorithm over the combined matrix, resulting in semi-supervised clustering. We note that these approaches for semi-supervised clustering can be applied to our problem setting.

3. Method

3.1. Preliminaries and notations

We describe the notations that will be used throughout this paper. Let $\boldsymbol{X} \coloneqq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ be given numerical vectors. Let $\boldsymbol{I}$ be the identity matrix of size N. Each $\boldsymbol{x}_n$ has p entries, and let $x_n(i)$ be the i-th entry of $\boldsymbol{x}_n$. We define $\boldsymbol{x}_n^2 \coloneqq \boldsymbol{x}_n^T \boldsymbol{x}_n$. Let $\boldsymbol{Y}$ be a $(N \times N)$-matrix where its (i,j)-element is given by $\boldsymbol{Y}_{ij} = (1/2N)(\boldsymbol{x}_i - \boldsymbol{x}_j)^2$. Let $\boldsymbol{G}$ be a given network with N nodes and edges. Let $\boldsymbol{W} \in \Re^{N \times N}$ be a non-negative, symmetric matrix whose (i,j) entry, w_{ij} is a non-negative weight between nodes i and j. If there is no edge between nodes i and j, w_{ij} is zero. We note that in our problem setting, $\boldsymbol{W}$ is an input having all information on a given graph $\boldsymbol{G}$ and is called an *affinity matrix*. Let $\boldsymbol{D}_d$ be a $N \times N$ diagonal matrix whose (i,i) entry d_i satisfies that $d_i = \sum_{j=1}^{N} w_{ij}$. Let $\boldsymbol{D} \coloneqq \boldsymbol{d}\boldsymbol{d}^T$ where $\boldsymbol{d} \coloneqq (d_1, \ldots, d_N)^T$. Let K be the number of clusters which is an input. Let $\boldsymbol{I}_K$ be the identity matrix of size K. Let $\boldsymbol{Z} \coloneqq (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K)$ be an unsigned cluster assignment in which $\boldsymbol{z}_k^T = (z_{1,k}, \ldots, z_{N,k})$ where $z_{n,k} (\in \{0,1\})$ is 1 if $\boldsymbol{x}_n$ is in cluster k, otherwise zero. Let $\boldsymbol{\mu} \coloneqq (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$ where $\boldsymbol{\mu}_k$ be the representative (or the cluster center) of cluster k. Let $\mathcal{Z}_k$ be a set of nodes in a given graph (or numerical vectors) in cluster k, and let $|\mathcal{Z}_k|$ be the number of all nodes (or vectors) in cluster k. That is, $i \in \mathcal{Z}_k$ if $z_{i,k}=1$, otherwise $i \notin \mathcal{Z}_k$. $\mathcal{Z} \coloneqq \cup_{k=1}^{K} \mathcal{Z}_k$. $L(\mathcal{Z}_k, \mathcal{Z}_{k'}) \coloneqq \sum_{i \in \mathcal{Z}_k} \sum_{j \in \mathcal{Z}_{k'}} w_{ij}$, and $L \coloneqq L(\mathcal{Z}, \mathcal{Z})$. Let J be a cost of clustering numerical vectors $\boldsymbol{X}$ (or/and nodes in network $\boldsymbol{G}$). Let ω be a numerical parameter which takes a value between zero and one, and balances the two data sources, i.e. numerical vectors $\boldsymbol{X}$ and network $\boldsymbol{G}$. Let $\boldsymbol{M}$ be a matrix. We write the matrix trace of $\boldsymbol{M}$ as $\mathrm{tr}(\boldsymbol{M}) = \sum_{i=1}^{N} \boldsymbol{M}_{ii}$. Using this notation, we can easily derive the following: $\mathrm{tr}(\boldsymbol{Z}^T \boldsymbol{M} \boldsymbol{Z}) = \sum_{k=1}^{K} \boldsymbol{z}_k^T \boldsymbol{M} \boldsymbol{z}_k$. Furthermore we define the following notation:

$$\mathrm{tr}\left(\frac{\boldsymbol{Z}^T \boldsymbol{M} \boldsymbol{Z}}{\boldsymbol{Z}^T \boldsymbol{Z}}\right) \coloneqq \sum_{k=1}^{K} \frac{\boldsymbol{z}_k^T \boldsymbol{M} \boldsymbol{z}_k}{\boldsymbol{z}_k^T \boldsymbol{z}_k}.$$

3.2. Proposed method

3.2.1. k-Means: clustering numerical vectors

We briefly review the *k*-means clustering algorithm, which has been widely used in a lot of applications for clustering numerical