2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology

# Graph-cut based Iterative Constrained Clustering

Masayuki Okabe
*Toyohashi University of Technology*
*Tenpaku 1-1, Toyohashi, Aichi, Japan*
*okabe@imc.tut.ac.jp*

Seiji Yamada
*National Institute of Infomatics*
*Chiyoda, Tokyo, Japan*
*seiji@nii.ac.jp*

***Abstract*—This paper proposes a constrained clustering method that is based on a graph-cut problem formalized by SDP (Semi-Definite Programming). Our SDP approach has the advantage of convenient constraint utilization compared with conventional spectral clustering methods. The algorithm starts from a single cluster of a complete dataset and repeatedly selects the largest cluster, which it then divides into two clusters by swapping rows and columns of a relational label matrix obtained by solving the maximum graph-cut problem. This swapping procedure is effective because we can create clusters without any computationally heavy matrix decomposition process to obtain a cluster label for each data. The results of experiments using a Web document dataset demonstrated that our method outperformed other conventional and the state of the art clustering methods in many cases. Hence we consider our clustering provides a promising basic method to interactive Web clustering.**

## I. Introduction

Constrained clustering is a semi-supervised learning approach that utilizes pre-given knowledge about data pairs to improve normal clustering accuracy [1]. The knowledge used is generally of two simple types: a constraint about data pairs that must be in the same cluster, and a constraint about data pairs that must be in a different cluster. These are usually called *must-link* and *cannot-link*, respectively.

Recent research about distance metric learning interprets the constraint information as the distance or kernel value of data pairs and tries to produce a new distance measure or kernel matrix for a complete dataset to ensure the distance of must-link is small and the distance of cannot-link is large [2]. In this research, we do not interpret the constraint as a distance or kernel value but rather as a relational label that indicates whether data pairs should be in the same cluster or not. Our objective is to predict the correct label for each data pair (not for each individual piece of data) by using sample labels converted from pre-given constraint information according to the framework of the transductive learning.

Our method is based on the graph-cut problem. Although graph-cut based clustering (e.g., spectral clustering) is a well known approach and many methods have been proposed so far [3], their solutions are mostly based on the graph spectrum obtained by eigen decomposition, which requires complicated processes to add in the constraint information. Our approach is to solve it as a semi-definite programming (SDP) problem. The advantage of SDP is that we can naturally incorporate constraints without any complicated processing and do not need any specific objective functions (e.g., normalized cut) to avoid a trivial solution (as is the case with many other spectral clustering methods).

In terms of formalization, our problem is the same as Li's [4] or Hoi's [5], although the introduction is completely different. The most critical difference is the interpretation of the SDP solution. They interpret the solution as a kernel matrix and use it for multi-class clustering, while we interpret it as a label matrix (as described above) and use it for two-class clustering. As we will show in the experiments, our two-class clustering approach performs better than the multi-class clustering approach. Our approach is based on the divide and conquer algorithm. It starts from a single cluster of a complete dataset and repeatedly selects the largest cluster, which it then divides into two clusters until we get the target numbers of clusters. In each iteration, we obtain relational labels for all data pairs from the solution of the SDP problem. We then use the label matrix to create clusters by swapping rows and columns to reduce the clusters' label distribution entropies. This swapping procedure is very effective because we can create clusters without any computationally heavy matrix decomposition processing.

In summary, we propose a constrained clustering method that has the following features.

- Clustering is performed based on the relational labels of all data pairs that are obtained by solving a graph-cut problem formalized by semi-definite programming. Our SDP approach has the advantage convenient constraint utilization compared with conventional spectral clustering methods.
- The interpretation of the obtained matrix is different from Li and Hoi's approaches, although the problem formalization is similar. They use the matrix as a kernel matrix for one-time multi-class clustering, while we use it as binary label matrix for divide and conquer-based two-class clustering.

These advantages make constrained clustering more efficient, especially in the case of small number of constraints. Thus we are planning to apply this clustering method to

978-0-7695-4513-4/11 $26.00 © 2011 IEEE
DOI 10.1109/WI-IAT.2011.42

interactive web clustering like [6].

## II. Graph Partitioning and its Solution

### A. Maximum Cut Problem

The objective of the problem is to divide a graph into two parts as its cut amount reaches the maximum. More formally, consider a graph $G = (V, E)$, where $V$ is a set of vertices and $E$ is a set of edges. The problem is to find partitioning $(V_1, V_2)$ such as $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \phi$ and a maximum cut amount of $\sum_{i \in V_1, j \in V_2} w_{ij}$. Here, $w_{ij}$ is the weight of an edge between data $i \in V_1$ and data $j \in V_2$. We decide on a "maximum" cut when $w_{ij}$ is defined by some distance (e.g., Euclid distance). In contrast, if $w_{ij}$ is defined by some similarity (e.g., Gauss kernel), the "minimum" cut is appropriate.

By introducing a cluster label variable $u_i$ for each vertex, we can formalize the maximum cut problem as follows.

Maximum Cut Problem

$$\begin{array}{ll} \text{maximize} & \frac{1}{4}\sum_{i \in V_1}\sum_{j \in V_2} w_{ij}(1 - u_i u_j) \\ \text{subject to} & u_k^2 = 1 \quad (k \in V) \\ & u_k = \begin{cases} +1 & (k \in V_1) \\ -1 & (k \in V_2) \end{cases} \end{array}$$

According to the standard method of spectral clustering or segmentation by the random walk model, we can solve this problem with the method of Lagrange multipliers. The $u_i$ labels are obtained as eigen vectors corresponding to the second largest eigen value.

Our aim is to incorporate given constraints into the above problem and find a method to solve constrained maximum cut problems. While there are constrained versions of spectral clustering methods, we adopt a different approach, solution by semi-definite programming (SDP), which is practically easier to use because it can handle constraints intrinsically.

### B. Solution by Semi-definite Programming Relaxation

Semi-definite programming is a kind of convex optimization that is used to relax several optimization problems such as combinatorial optimization, 0-1 integer programming, and non-convex quadratic programming. Since the maximum cut problem is an example of 0-1 integer programming, SDP can also relax it.

For the standard formalization of SDP, we transform the above objective function into a matrix representation with a weight matrix $W$ and a matrix $X$ whose element is the product of $u_i$ and $u_j$.

$$\begin{aligned} \sum_{i \in V_1}\sum_{j \in V_2} w_{ij}(1 - u_i u_j) &= (diag(W\mathbf{e}) - W) \bullet X \\ &= L \bullet X \end{aligned}$$

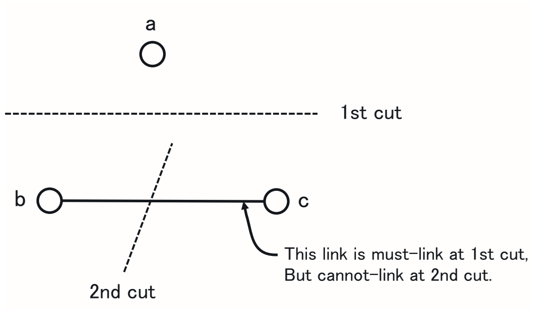

a
1st cut
b
c
This link is must-link at 1st cut,
But cannot-link at 2nd cut.
2nd cut

Figure 1. Cannot-link is not applicable in divide and conquer approach

$$X = \mathbf{u}^T\mathbf{u}$$

$$\mathbf{u} = (u_1, u_2, ..., u_n), \quad n = |V|$$

$L$ is the graph Laplacian matrix and $\mathbf{e}$ is a vector whose elements are all one. As a final step, we add *must-link* constraints to formalize the constrained maximum cut problem as follows.

Maximum Cut Problem with SDP Relaxation

$$\begin{array}{ll} \text{maximize} & L \bullet X \\ \text{subject to} & E_{ii} \bullet X = 1, \quad (i = 1 \sim n) \\ & E_{ij} \bullet X = 1, \quad (i, j) \in M \\ & X \succeq O \end{array}$$

$E_{ij}$ is an $n \times n$ matrix in which only the $(i, j)$ element is 1, and all others are 0. $M$ is a set of *must-link*.

Although available constraints are not limited to must-link, meaning we can also use *cannot-link*, it is very difficult to select applicable cannot-links during multi-class clustering because the partitioning order determined in the graph cut process is usually unpredictable in advance. Figure 1 gives an idea of the difficulty of using cannot-link. There are three data in the figure - $a, b, c$ - and cannot-link is applicable only at the second cut. It cannot be used at the first cut because $b$ and $c$ are in the same cluster at that time.

There are several freely available SDP solvers that we can use to obtain an approximate solution $\tilde{X}$. Although we need to decompose $\tilde{X}$ to obtain partitioning label $\mathbf{u}$, we found a way to complete partitioning by using only $\tilde{X}$. We explain this method in the next section.

## III. Clustering through Swapping Rows and Columns in a Label Matrix

As described in the previous section, we solve the maximum cut problem with relaxed SDP, so the elements of $\tilde{X}$ are assigned a real value ranging from -1 to 1. Though we can use those values as similarity and run k-means, we found the resulting performance was very low. We therefore decided on a different approach in which we first binarize $\tilde{X}$ with 0-1 values, then swap rows and columns to

**Algorithm 1** Clustering Procedure

```
1: Input: $D$      // Dataset
2:        $W$      // Weight Matrix
3:        $M$      // Must-Link Set
4:        $K$      // Number to be Clustered
5: Output: $\{D_1, D_2, ..., D_K\}$      // $K$ clusters
6:
7: Let $D^0_{sub} = D$
8: Select the largest cluster $D^t_{sub}$
9: Select a set of must-link applicable to divide $D^t_{sub}$
10: Solve maximum graph cuts problem for $D^t_{sub}$ and divide $D^t_{sub}$
    into $D^t_{sub1}$ and $D^t_{sub2}$
11: Return STATE 8 and Repeat K-1 times
```

maximize the evaluation measure, and finally determine the partitioning border.

The concrete procedures are as follows.

1) Each element of $\tilde{X}$ is binarized as follows.

$$x_{ij} = \begin{cases} 1, & \text{if} \quad x_{ij} \geq 0 \\ 0, & \text{if} \quad x_{ij} < 0 \end{cases}$$

The value does not matter because we treat 0 and 1 as a character in the following steps.

2) For each row, treat the column's value as a character (0 or 1) and make a string (or pattern) by concatenating each character in the original order. Next, make groups of the same string (select a representative of each kind of string).

3) Determine the most frequent string $s_0$, and calculate the Hamiltonian distance between $s_0$ and the other strings. Next, align other strings in descending order of the distance. String $s_1$ is the most similar to $s_0$.

4) Determine the partitioning boundary according to the following measure.

$$F(i,j) = \sum_{k=1}^{4} -p^k_0 \log(p^k_0) - p^k_1 \log(p^k_1)$$

$(i,j)$ represents the partitioning boundary. If we partition the above aligned matrix in the boundary between $i$'s row and $j$'s row (i.e., also in the boundary of $i$'s column and $j$'s column ), there are four partitioned areas in the matrix. $p^k_0$ and $p^k_1$ are the probabilities of 0 and 1, respectively, that appears in the area $k$. Thus, $F(i,j)$ is the sum of the entropy in four areas when partitioned between $i$th and $j$th row (and column). The more clearly partitioned, $F(i,j)$ becomes lower.

5) Determine the boundary at the lowest $F(i,j)$.

Figure 2 illustrates this procedure. The figure shows the case of two class clustering. The clusters are made by the boundary between 0 and 1 in the matrix.

This is a heuristic method because the original problem is a combinatorial one and practically intractable. There is no guarantee for obtaining global optimum. However, experimentally it works well, as described in the next section.

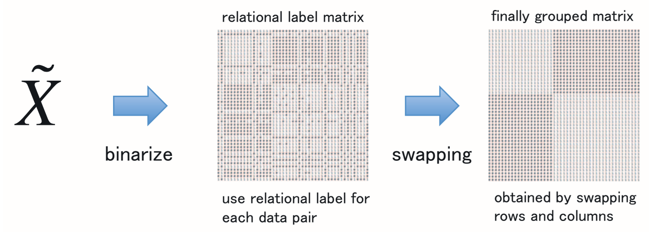

$\tilde{X}$
binarize
relational label matrix
use relational label for each data pair
swapping
finally grouped matrix
obtained by swapping rows and columns

Figure 2. Clustering process by swapping rows and columns in a label matrix

Table I
DATASETS

| No. | Category | #data |
|---|---|---|
| 1 | Anomalies and Alternative Science | 47 |
| 2 | Science in Society | 45 |
| 3 | Environment/Water Resources | 42 |
| 4 | Astronomy | 24 |
| 5 | Technology/Structural Engineering | 24 |
| 6 | Agriculture | 19 |
| 7 | Biology/Genetics | 16 |
| 8 | Social Sciences/Linguistics | 15 |
| 9 | Physics | 15 |
| 10 | Earth Sciences | 11 |
| 11 | Math | 10 |
| 12 | Chemistry | 6 |

We describe an entire procedure of our clustering algorithm in **Algorithm 1**.

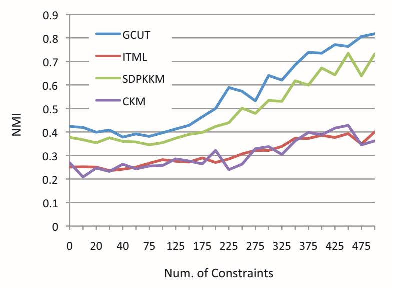
## IV. Experiments

We evaluated our proposed method using a Web document dataset extracted from the Open Directory Project (ODP)[1]. The categories we used are summarized in Table I. They are all sub-categories of the "Science" top category. In each sub-category, there are some registered sites. We used top pages of those sites as data. We removed tags and stop words from original web pages, and then made a tf-idf vector for each data.

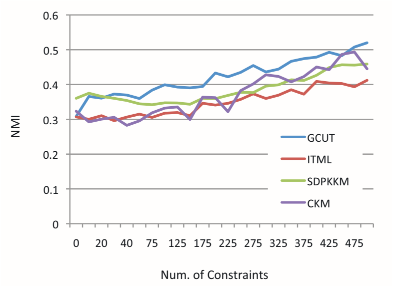
We compared the following four methods.

- **GCUT**: This is our proposed method. We use the SDPA package[2] for solving the semi-definite programming. We adopted the Euclid distance for the weight $w_{ij}$ of a graph edge in the maximum graph-cut problem.
- **CKM**: This is the COP-Kmeans constrained clustering algorithm proposed by Wagstaff [7]. We used ten seeds for the k-means and took the average of those results.
- **ITML**: This is a state of the art metric learning (Information Theoretic Metric Learning) method proposed by Jain [8]. We used improved online version of this method because it was too difficult to tune parameters and thereby obtain stable results with the original batch version. We changed the learning parameter $\eta$ from

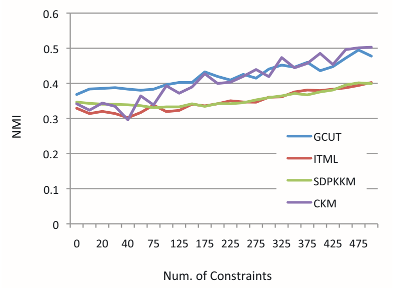
[1]http://www.dmoz.org/
[2]http://sdpa.indsys.chuo-u.ac.jp/sdpa/



(a) Category 1∼4



(b) Category 1∼8



(c) Category 1∼12

Figure 3. Results

0.1∼0.9 with 0.1 steps and selected the best results for the evaluation. The clustering is done by a normal k-means algorithm. We used ten seeds for the k-means and took the average of those results.

- **SDPKKM**: We also compared the method proposed by Li or Hoi [4], [5], which has similar SDP formalization of the problem to ours but there is a big difference for the use of the solution matrix. We adopted the the innner product of the feature vector $i$ and $j$ for the weight $w_{ij}$ of a graph edge used during the kernel matrix learning because it was too difficult to tune the radius parameter for the Gaussian kernel. The clustering is done by a conventional kernel k-means algorithm. We used ten seeds for the k-means and took the average of those results.

We use normalized mutual information (NMI) to measure the clustering accuracy.

Figure 3 shows the results. The horizontal axis is the number of constraints and the vertical axis is the value of normalized mutual information (NMI). We increase the number of must-link constraints from 0 to 500, which is randomly selected. We tested each method with ten different sets of constraint and calculate those average for the results at each number of constraint.

We changed the number of categories to test as shown in the figure. Though SDPKKM and CKM showed comparable performance in Figure 3 (a) and (c) respectively, our proposed method mostly outperformed other methods in any category set.

## V. Conclusions

In this paper, we proposed a constrained clustering method that is based on a graph-cut problem formalized by semi-definite programming and deterministic iterative two-class partitioning approach. While graph-cut based clustering is a particularly promising way to improve conventional techniques like k-means method, few methods have been proposed, which can naturally incorporate constraint like must-link and cannot-link.

Our method has the advantages of more convenient constraint incorporation compared to other graph-cut based method like spectral clustering and more appropriate SDP's solution matrix utilization compared with other SDP-based methods. Results showed that our proposed clustering method constantly outperformed conventional methods and utilized constraints effectively.

The advantages of our proposed clustering is efficiency, especially in the case of small number of constraints. Thus we are planning to apply this clustering method to interactive (Web) clustering with GUI [6].

## References

[1] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*. The MIT Press, 2006.

[2] W. Tang, H. Xiong, S. Zhong, and J. Wu, "Enhancing semi-supervised clustering: A feature projection perspective," in *Proceedings of the 13th International Conference on Knowledge Discovery and Data Mining*, 2007, pp. 707–716.

[3] S. X. Yu and J. Shi, "Segmentation given partial grouping constraints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 173–183, 2004.

[4] Z. Li, J. Liu, and X. Tang, "Pairwise constraint propagation by semidefinite programming for semi-supervised classification," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 576–583.

[5] S. C. H. Hoi, R. Jin, and M. R. Lyu, "Learning nonparametric kernel matrices from pairwise constraints," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 361–368.

[6] M. Okabe and S. Yamada, "An interactive tool for human active learning in constrained clustering," *Journal of Emerging Technologies in Web Intelligence*, vol. 3, no. 1, pp. 20–27, 2011.

[7] K. Wagstaff and S. Roger, "Constrained k-means clustering with background knowledge," in *Proceedings of the 18th International Conference on Machine Learning*, 2001, pp. 577–584.

[8] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *Proceedings of Neural Information Processing Systems*, 2008, pp. 761–768.