Graph-cut based Constrained Clustering by Grouping Relational Labels

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

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

Abstract—This paper proposes a novel 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 whole 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 datasets from the ODP and WebKB corpus demonstrated that our method outperformed other conventional and the state of the art clustering methods in many cases. In particular, we discuss the difference between our approach and another similar one that uses the same SDP formalization as ours. Since the number of constraints used in the experiments is relatively small and can be practical for human feedback, we consider our clustering provides a promising basic method to interactive Web clustering.

I. INTRODUCTION

Clustering has long been one of the most essential and popular techniques in data mining [1]. It is used not only for visualization of huge data sets but also for image segmentation [2], medical applications, recommendation systems, and so on.

Constrained clustering is a semi-supervised learning approach that utilizes pre-given knowledge about data pairs to improve normal clustering accuracy [3], [4]. 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 whole dataset to ensure the distance of must-link is small and the distance of cannot-link is large [5], [6], [7]. 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 [8], [9], 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 [10] or Hoi’s [11], 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 such as interactive web clustering like [12].

The rest of the paper is organized as follows. First we explain the standard maximum graph cut problem and its solution by semi-definite programming relaxation in Section II. Next, we describe our clustering algorithm, which is based on the approximate solution of the SDP. We describe the entire clustering procedure including binarizing and swapping of the solution matrix in Section III. Section IV shows the results of experiments performed using datasets from the ODP and WebKB corpus. We discuss our methods in Section V, and finally we conclude our work in Section VI.

II. CONSTRAINED GRAPH CUT PROBLEM

Graph-cut formalization is a powerful clustering approach that many algorithms have adopted. In this section, we first formalize the maximum cut problem and then introduce a solution by semi-definite programming.

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 \( V_1 \cap V_2 = \emptyset \) 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**

maximize \[ \frac{1}{4} \sum_{i \in V_1} \sum_{j \in V_2} w_{ij} (1 - u_i u_j) \]

subject to \[ u_k^2 = 1 \quad (k \in V) \]

\[ u_k = \begin{cases} +1 & (k \in V_1) \\ -1 & (k \in V_2) \end{cases} \]

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. Formalization by SDP

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 \( u_j \).

\[
\sum_{i \in V_1} \sum_{j \in V_2} w_{ij} (1 - u_i u_j) = (\text{diag}(We) - W) \cdot X = L \cdot X
\]

\[
X = u^T u
\]

\[
u = (u_1, u_2, \ldots, u_n), \quad n = |V|
\]

\( L \) is the graph Laplacian matrix and \( e \) 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**

maximize \( L \cdot X \)

subject to \( E_{ii} \cdot X = 1, \quad (i = 1 \sim n) \)

\( E_{ij} \cdot X = 1, \quad (i, j) \in M \)

\( X \succeq O \)

\( E_{ij} \) 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*. 

© 2012 ACADEMY PUBLISHER
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 \( \hat{X} \). Although we need to decompose \( \hat{X} \) to obtain partitioning label \( u \), we found a way to complete partitioning by using only \( \hat{X} \). We explain this method in the next section.

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

In this section, we describe a concrete partitioning procedure using matrix \( \hat{X} \), which is given as the SDP solution, and then an entire clustering with an iterative dual-partitioning process.

As described in the previous section, we solve the maximum cut problem with relaxed SDP, so the elements of \( \hat{X} \) are assigned a real value ranging from -1 to 1. We therefore decided on a different approach in which we first binarize \( \hat{X} \) with 0-1 values, then swap rows and columns to maximize the evaluation measure, and finally determine the partitioning border.

The concrete procedures are as follows.

1. Each element of \( \hat{X} \) is binarized as follows.
   
   \[
   \hat{X}_{ij} =\begin{cases} 
   0, & \text{if } \hat{X}_{ij} < 0 \\
   1, & \text{if } \hat{X}_{ij} \geq 0 
   \end{cases}
   \]

   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). Figure 2(a) illustrates this procedure.

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 Hamming distance. String \( s_1 \) is the most
ourselves. We selected 12 subdirectories from the the

A. Datasets

extracted from the Open Directory Project (ODP) 1

1 http://www.dmoz.org/

Algorithm 1 section.

experimentally it works well, as described in the next

is no guarantee for obtaining global optimum. However,

is a combinatorial one and practically intractable. There

1 http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

2 http://www.math.nus.edu.sg/ mattohkc/sdpt3.html

Web sites listed in each directory. We removed tags and

stopwords from the pages, and stemmed each word. The

summary of each directory is listed in Table I.

We treated each directory as a target cluster, and

made four datasets using those clusters. One is a dataset

(odp_all) using all the directories in the corpus. The other

two (odp_odd and odp_even) are half size of odp_all. The

directories of odp_odd and odp_even are selected from the

odd and even number ones in Table I, respectively.

The final one (odp_small) is a set of small directories

(No.6~12) that include under 20 data. We summarize

about datasets in Table II.

We also used the WebKB corpus 2. This corpus consists

of seven datasets, and each one has fixed five clusters


and “misc”, respectively. Since the last “misc” cluster

consists of Web pages from miscellaneous universities

and lacks unity, we removed it from each dataset. We also

removed department dataset because it has only one page

for each cluster. The datasets are summarized in Table III.

We applied the same preprocessing as ODP corpus and

evaluated each method on all the datasets.

B. Compared methods

We compared our proposed method with other three

methods. The notation and brief introduction of each

method is listed below.

GCUT This is our proposed method. We calculated

the Euclid distance for the weight of the graph

edge, which is indicated as \(w_{ij}\) in the maximum graph cut problem in Section II. We used the

SDPT3 package 3 to solve SDP. The default

parameters are used for each run.

PCP PCP is one of the state of the art distance metric learning methods proposed by Li [10].

It learns a kernel matrix using the same SDP

formulation with ours. The difference between

two methods lies in the usage of the solution

matrix. PCP uses it as a kernel matrix for kernel

k-means while ours uses it for iterative graph

cut. The weight of the graph edge and the

parameters for SDPT3 are the same with GCUT.

Since this method can use both must/cannot-

link constraints, we conducted two trials. One is the trial using only must-link constraints, the

other uses both must/cannot-link constraints. the

2 http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/


similar to \(s_0\). Figure 2(b) illustrates this procedure.

4) Determine the partitioning boundary according to

the following measure.

\[
F(i,j) = \sum_{k=1}^{4} -p^{S_i} \log(p^{S_i}) - (1 - p^{S_j}) \log(1 - p^{S_j})
\]

\((i,j)\) 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. \(p^{S_i}_k\) and \(p^{S_j}_k\)

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 \(F(i,j)\).

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.

We describe an entire procedure of our clustering

algorithm in Algorithm 1.

IV. EXPERIMENTS

A. Datasets

We evaluated our proposed method on two Web page
corpora. One is a ODP corpus, a set of web pages
extracted from the Open Directory Project (ODP) 1 by
ourselves. We selected 12 subdirectories from the the

“Science” top directory, and downloaded top pages of the

TABLE I.

DIRECTORIES USED IN ODP CORPUS

<table>
<thead>
<tr>
<th>No.</th>
<th>Directory</th>
<th>#Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Anomalies and Alternative Science</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>Science in Society</td>
<td>45</td>
</tr>
<tr>
<td>3</td>
<td>Environment/Water Resources</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>Astronomy</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>Technology/Structural Engineering</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Agriculture</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>Biology/Genetics</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Social Sciences/Linguistics</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>Physics</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>Earth Sciences</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>Math</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>Chemistry</td>
<td>6</td>
</tr>
</tbody>
</table>

TABLE II.

DATASETS IN ODP CORPUS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Data</th>
<th>#Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>odp_odd</td>
<td>154</td>
<td>6</td>
</tr>
<tr>
<td>odp_even</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>odp_small</td>
<td>92</td>
<td>7</td>
</tr>
<tr>
<td>odp_all</td>
<td>274</td>
<td>12</td>
</tr>
</tbody>
</table>

TABLE III.

WEBKB CORPUS

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Data</th>
<th>#Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>student</td>
<td>558</td>
<td>4</td>
</tr>
<tr>
<td>faculty</td>
<td>153</td>
<td>4</td>
</tr>
<tr>
<td>staff</td>
<td>46</td>
<td>4</td>
</tr>
<tr>
<td>course</td>
<td>234</td>
<td>4</td>
</tr>
<tr>
<td>project</td>
<td>86</td>
<td>4</td>
</tr>
<tr>
<td>other</td>
<td>3033</td>
<td>4</td>
</tr>
</tbody>
</table>
Figure 4. Results of ODP corpus (horizontal axis is the number of constraints and vertical axis is NMI)

former is denoted by PCP(m) and the latter is PCP(n/c).

**ITML** ITML is another distance metric learning method proposed by Jain [13]. It learns a transform matrix to calculate desired distance. Since this method follows online learning process, we tuned its learning parameter $\eta$ by selecting the best value from 0.1 to 1.0 with 0.1 steps. Clustering is done by normal k-means with learned Euclidian distance. Since cannot-link on this method showed terrible performance deterioration, we applied only must-link for it.

**CKM** CKM is the constrained k-means clustering algorithm (called COP-Kmeans) proposed by Wagstaff [14]. Since cannot-link often caused deadlock and stopped all the algorithm procedure, we applied must-link only as well as ITML.

### C. Other settings

We use normalized mutual information (NMI) to measure the clustering accuracy. NMI is calculated by the following formula.

$$\text{NMI}(C, T) = \frac{I(C, T)}{\sqrt{H(C)H(T)}}$$

where $C$ is the set of clusters returned by each algorithm and $T$ is the set of true clusters. $I(C, T)$ is the mutual information between $C$ and $T$, and $H(C)$ and $H(T)$ are the entropies.

Constraints are selected randomly. We changed the number of constrains from 0 to 100 with 10 steps. For each number of constrains, we selected 10 different sets of constraints and used the same sets for each method. The NMI is calculated as the average value of those 10 sets.

### D. Results

Figure 4 is the results of the ODP corpus. The horizontal axis is the number of constraints and the vertical axis is the value of normalized mutual information (NMI).

In this corpus, GCUT outperformed other methods in all the datasets, especially at the points where the number of constrains is small.

PCP(m/c) showed comparable or slightly better performance at some points in the odp_odd, the odp_even and
Figure 5. Results of the WebKB corpus (horizontal axis is the number of constraints and vertical axis is NMI)

odp small datasets though it remained the second best or worse in other points. PCP(m) is slightly worse than PCP. CKM outperformed other methods without GCUT in the odp_all dataset. ITML did not show the good results in this corpus.

Figure 5 shows the results of the WebKB corpus. The meaning of the horizontal and vertical axis is the same as the ODP corpus. In this corpus, GCUT indicated the best results in four datasets. The results in the other two datasets is comparable and slightly worse than the other methods. GCUT showed clearly better performance especially in the student and “other” datasets whose number of data is relatively larger than others. The performance of all methods in “other” dataset is low because the topic of this dataset is diverse and does not have unity as a cluster. GCUT also showed better performance than other methods in the staff and project datasets whose number of data is relatively small. In those datasets, performance gap between GCUT and others widened as the number of constraints increased. PCP showed comparable and slightly better performance in the course and faculty datasets. We did not observe significant difference between PCP(m/c) and PCP(m) in this corpus. The performance of CKM and ITML remained worse than GCUT and PCP in all the datasets.

The performance of GCUT as well as all the other
methods does not grow monotonically and sometimes drops despite the increase of constraints. This is because the effectiveness of constraints are generally quite uneven. Some sets of constraints may work better for some methods. On the other hand some sets may bring negative effect (e.g. because of must-link constraints related outliers).

V. DISCUSSIONS

Our proposed clustering method (GCUT) repeatedly divides the largest cluster into two sub-clusters until the given number of clusters is obtained. This procedure delivers a good property compared to a one-time multi-class clustering approach represented by PCP as shown in the experiments. The results indicate that we can obtain more accurate clusters if we interpret the SDP problem described in Section II as a constrained graph cut problem and use its solution for iterative two-class clustering.

As described in Section I, our method uses the same SDP formulation as PCP. However the purpose, derivation and interpretation of the solution for our method is different from PCP. In this section, we explain a brief introduction of PCP and clarify the difference from our method, and then discuss the cause of the experimental results in the previous section.

PCP is a method to produce a kernel matrix of a dataset by projecting original feature vectors to a higher dimensional space. Constraints are used as desired inner product values for some selected data pairs in the projected feature space. In order to propagate the effect of the constraints to other data pairs, Li et al. formulated an optimization problem according to a well-known regularization in spectral graph theory.

Optimization Problem derived in PCP

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left( \frac{\phi(x_i)}{d_{ii}} - \frac{\phi(x_j)}{d_{jj}} \right)^2_F \\
\text{subject to} & \quad < \phi(x_i), \phi(x_j) >_F = 1 \quad i, j = 1, 2, ..., n \\
& \quad < \phi(x_i), \phi(x_j) >_F = 1 \quad \forall (x_i, x_j) \in M \\
& \quad < \phi(x_i), \phi(x_j) >_F = 0 \quad \forall (x_i, x_j) \in C
\end{align*}
\]

Here, \( \phi(x_i) \) is a feature vector in the projected space, \( w_{ij} \) is an initial similarity between data \( i \) and \( j \), \( d_{ii} \) is \( \sum_k w_{ik} \), and \( < .., .. >_F \) indicates inner product calculation in the projected space. The objective function of the above problem can be finally transformed into the formula \( L \bullet X \) that is the same one as described in Section II.

In this way, GCUT and PCP solve the same SDP problem except for the use of cannot-link. However their derivations of the problem are different from each other like following.

1) PCP places the SDP problem as a basis to define a proximity measure in a dataset. Thus it solves a SDP problem only once for a clustering trial. The proximity measure is actually got as a kernel matrix.

2) The SDP problem in GCUT is a constrained graph cut clustering problem itself. Thus it iterates to solve SDP \( k - 1 \) times. \( k \) is a target number of clusters.

Though the reason for the performance increase of GCUT compared with PCP superficially seems to be multiple SDP executions, it is more important that the derivations of the SDP problem in both methods are different. GCUT derives from a constrained graph cut problem while PCP derives from a problem to define a desired proximity measure.

VI. 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.

Our method has the advantages of more convenient constraint incorporation compared to other graph-cut based method such as spectral clustering and can utilize SDP’s solution matrix more appropriately compared with other SDP-based methods. Since our method adopts deterministic clustering approach unlike k-means using random seeds, the performance is stable and robust to outliers. These advantages were clearly demonstrated through experiments using many datasets from two Web corpus. Results showed that our proposed clustering method constantly outperformed conventional methods in many cases and utilized constraints effectively.

A few problems still remain in this work. First, we need to investigate how the constraint quality influences the clustering. This is an active learning problem and in the future we aim to develop a new active-learning technique by utilizing the properties of our clustering methods. Second, we need to develop a constraint propagation method to improve clustering accuracy even if the number of constraint is small. There are already various propagation methods in place [15], but we need something more powerful to improve the effectiveness of very small constraints.

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 [12].

REFERENCES


**Masayuki Okabe** is an assistant professor at Toyohashi University of Technology. He received B.S. (1996) degree from Soka University and M.S. (1998) and the Ph.D. (2001) degrees from Tokyo Institute of Technology. His research interests include information retrieval, machine learning and data mining. He is a member of The Japanese Society for Artificial Intelligence.

**Seiji Yamada** is a professor at the National Institute of Informatics. Previously he worked at Tokyo Institute of Technology. He received B.S. (1984), M.S. (1986) and the Ph.D. (1989) degrees in artificial intelligence from Osaka University. His research interests are in the design of intelligent interaction including Human-Agent Interaction, intelligent Web interaction and interactive machine learning. He is a member of IEEE, AAAI, ACM, JSAI, IPSJ and HIS.