Paper:

Learning Similarity Matrix from Constraints of Relational Neighbors

Masayuki Okabe* **and Seiji Yamada****

*Information and Media Center, Toyohashi University of Technology
1-1 Tenpaku, Toyohashi, Aichi 441-8580, Japan
E-mail: okabe@imc.tut.ac.jp
**National Institute of Informatics, the Graduate University for Advanced Studies (SOKENDAI)
2-1-2 Hitotsubashi, Chiyoda, Tokyko 101-8430, Japan

[Received December 15, 2009; accepted March 8, 2010]

This paper describes a method of learning similarity matrix from pairwise constraints assumed used under the situation such as interactive clustering, where we can expect little user feedback. With the small number of pairwise constraints used, our method attempts to use additional constraints induced by the affinity relationship between constrained data and their neighbors. The similarity matrix is learned by solving an optimization problem formalized as semidefinite programming. Additional constraints are used as complementary in the optimization problem. Results of experiments confirmed the effectiveness of our proposed method in several clustering tasks and that our method is a promising approach.

Keywords: learning similarity matrix, semidefinite programming, constrained clustering

1. Introduction

Interactive clustering is a significant method to realize intelligent Web interaction because it is very useful for interactive visualization, data mining and data analysis of the Web [1]. Interactive clustering consists of two main techniques: active learning and constrained clustering. The active learning selects effective data as judged by the user, then is used as constraints [2]. The constrained clustering categorizes the data under such constraints from a user. Both pursue the common requirement of dealing with constraints from the user [3].

In this paper, we propose a method of constrained clustering using as few constraints as possible by expanding them into richer constraints. Semi-supervised learning creating classifiers or clusters from limited supervised information and much unlabeled data has been vigorously researched [4], with constrained clustering used as a learning problem. In the typical setting for constrained clustering, several pairwise data are given as either must-link or cannot-link constraints. Must-link pair data must be in the same cluster, but cannot-link pair data cannot be.

One simple use of these constraints is to build a procedure into clustering algorithms to check whether clusters break constraints. COP-K-means proposed by Wagstaff [5] is one such representative method. COP-K-means is a modified version of the K-means algorithm. It first selects cluster centers to which it allocates data while ensuring that the given constraints are not broken, i.e., pairs of data with must-links are not allocated to different clusters and vice versa. While simple, this approach is too difficult for creating consistent clusters with constraints as must/cannot-link pairs increases. Since COP-K-means conducts only a greedy search, causing clustering to stop before a task is completed.

Meanwhile, another approach uses constraints to modify the similarities between data as the similarities of must-link pairs must be large and the similarities of cannot-link pairs must be small. There are several methods to realize this approach [6–11]. For example, Klein et al. proposed a clustering algorithm to deal with space-level constraints in addition to ordinary instance-level constrains with must/cannot-links [6]. They maintained that neighbors around given data constrained by must/cannot-links should be constrained similarly to data, and developed ways to concretely propagate space-level constraints by satisfying triangle inequality using all-pairs-shortest-paths and complete-link hierarchical agglomerative clustering. Li et al. [11] proposed a method to learn a kernel matrix that is obtained by solving an optimization problem as semidefinite programming. They integrate the must/cannot-link constraints into the optimization problem to propagate local constraints into the whole kernel matrix. How such constraints influence the kernel value of other pairs remains unclear, although it ensures only the kernel value of constrained pairwise data.

Based on these two methods, we propose a similarity learning method producing a similarity matrix by solving an optimization problem in the same way as Li et al. We additionally impose constraints on the neighbors of constrained pairwise data to move similar data together based on the movement of pairwise constrained data. This is similar to Klein's approach in propagating space-level constrains under complete-link hierarchical agglomerative clustering, although we need no special clustering algorithm such as agglomerative clustering and is independent of such algorithms. Our method controls constraint propagation coverage by changing the number of neighbors.

In the sections that follow, we first explain similarity learning and formalize it as a form of semidefinite programming in Section 2. We then discuss learning similarity matrix imposing additional constraints on the original optimization problem in Section 3. Section 4 presents experimental results in which the similarity matrix thus obtained is applied to a simple clustering task. Section 5 presents conclusions.

2. Basic Similarity Learning

In this section, we formalize similarity learning as a semidefinite programming problem.

Let P be a data collection where each data $\vec{p}_i \in R^m (i = 1,...,n)$ is a vector of dimension m. Let $S \in R^{n\times n}$ be its similarity matrix, where each $s_{ij} (0 \le s_{ij} \le 1)$ is the similarity between $\vec{p}_i$ and $\vec{p}_j$. Pairwise constraints are given as follows:

$$\begin{array}{rcl} M &=& \{(i,j) \mid (\vec{p}_i,\vec{p}_j) \text{ is a must-link pair}\} \\ C &=& \{(i,j) \mid (\vec{p}_i,\vec{p}_j) \text{ is a cannot-link pair}\} \end{array} \quad (1)$$

The objective of similarity learning is to create a new similarity matrix K satisfying the above constraints, i.e., data pairs in M must be similar ($k_{ij} = 1$) and data pairs in C must be dissimilar ($k_{ij} = 0$). Before formalizing an optimization problem to obtain K as semidefinite programming [12], we use a graph Laplacian [13] as a coefficient for K in the optimization problem, letting D be a diagonal matrix where $d_{ii} = \sum_{j=1}^{n} s_{ij}$ and defining the graph Laplacian L as follows:

$$L = D - S \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \quad (2)$$

Using normalized version $\bar{L} = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$, we formalize an optimization problem for the similarity learning as follows:

$$\begin{array}{ll} \min_K : & \bar{L} \bullet K \\ s.t. : & k_{ii} = 1, \quad i = 1,...,n \\ & k_{ij} = 1, \quad ^{\forall}(i,j) \in M, \\ & k_{ij} = 0, \quad ^{\forall}(i,j) \in C, \\ & K \succeq 0 \end{array} \; . \; . \; . \; . \; . \; . \quad (3)$$

$\bar{L} \bullet K$ represents the inner product between $\bar{L}$ and K. $\bar{L} \bullet K$ is calculated as follows.

$$\bar{L} \bullet K = \sum_{i=1}^{n} \sum_{j=1}^{n} \bar{l}_{ij} k_{ij} \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \quad (4)$$

$\bar{l}_{ij}$ is the element of $\bar{L}$. This objective function assigns a large or small value to a data pair whose original similarity is high or low to get a matrix satisfying constraints while maintaining the original dataset affinity by solving the optimization problem. $K \succeq 0$, a condition making K a semidefinite matrix guarantees a matrix K is a Mercer kernel matrix.

Several available software packages solve general semidefinite programming problems, as used below to solve this optimization problem.

3. Constraints Extended from Pairwise to Neighbors

New similarity matrix K returned by the above solver differs from original similarity matrix S. In this section, we question how the new similarity matrix is changed by constraints because we consider that constraint influence should be propagated locally, i.e., in local nonconstrained data. For this, we introduce additional pseudo constraints for local propagation created by given constraints, as shown in **Fig. 1**, where two pairs of data (one star and two diamonds) are constrained as a must-link and a cannot-link in 2-dimensional data space respectively. The original optimization problem takes into account only the relationship between constrained data pairs, so the solution may result in the data allocation in **Fig. 1(a)**, where only the star data moves. We posit, however, both constrained data pairs and their neighbors are affected by constraints in **Fig. 1(b)**, where star data and its neighbors – two diamonds called *relational neighbors* – move together. To achieve this influence of constraints, we impose additional constraints on the optimization problem, i.e., for data pair (i,j) with must-link or cannot-link, we impose the following constraints:

$$\begin{array}{ll} k_{ir_t^j} \le -\bar{l}_{ir_t^j},\; k_{jr_t^i} \le -\bar{l}_{jr_t^i}, & if\ (i,j) \in M \\ k_{ir_t^j} \ge -\bar{l}_{ir_t^j},\; k_{jr_t^i} \ge -\bar{l}_{jr_t^i}, & if\ (i,j) \in C \end{array} \; . \; . \quad (5)$$

r_t^i is the t-th nearest neighbor of $\vec{p}_i$, r_t^j is the t-th nearest neighbor of $\vec{p}_j$, $k_{ir_t^j}$ is the similarity between $\vec{p}_i$ and r_t^j, $k_{jr_t^i}$ is the similarity between $\vec{p}_j$ and r_t^j, and $-\bar{l}_{ij} = \frac{s_{ij}}{\sqrt{d_{ii}d_{jj}}}$ is a regularized value of s_{ij}. r_t^i is thus a *relational neighbor* of $\vec{p}_j$, and r_t^j is a *relational neighbor* of $\vec{p}_i$, as shown in **Fig. 2**.

By adding these constraints,we expect that not only $\vec{p}_i$ and $\vec{p}_j$ move closer but also $\vec{p}_i$ and r_t^j, $\vec{p}_j$ and r_t^i move closer if $\vec{p}_i$ and $\vec{p}_j$ are must-link pair. Similarly, if they are cannot-link pair, we expect not only $\vec{p}_i$ and $\vec{p}_j$ move farther but also $\vec{p}_i$ and r_t^j, $\vec{p}_j$ and r_t^i move farther.

After all, the final optimization problem is formalized as follows:

$$\begin{array}{lll} \min_K : & \bar{L} \bullet K & \\ s.t. : & k_{ii} = 1, & i = 1,...,n \\ & k_{ij} = 1, & ^{\forall}(i,j) \in M, \\ & k_{ij} = 0, & ^{\forall}(i,j) \in C, \\ & k_{ir_t^j} \le -\bar{l}_{ir_t^j},\; k_{jr_t^i} \le -\bar{l}_{jr_t^i}, & ^{\forall}(i,j) \in M \\ & k_{ir_t^j} \ge -\bar{l}_{ir_t^j},\; k_{jr_t^i} \ge -\bar{l}_{jr_t^i}, & ^{\forall}(i,j) \in C \\ & K \succeq 0 & \end{array} \quad (6)$$

The fourth and fifth constraints are different from the original problem. By imposing these constraints, we en-

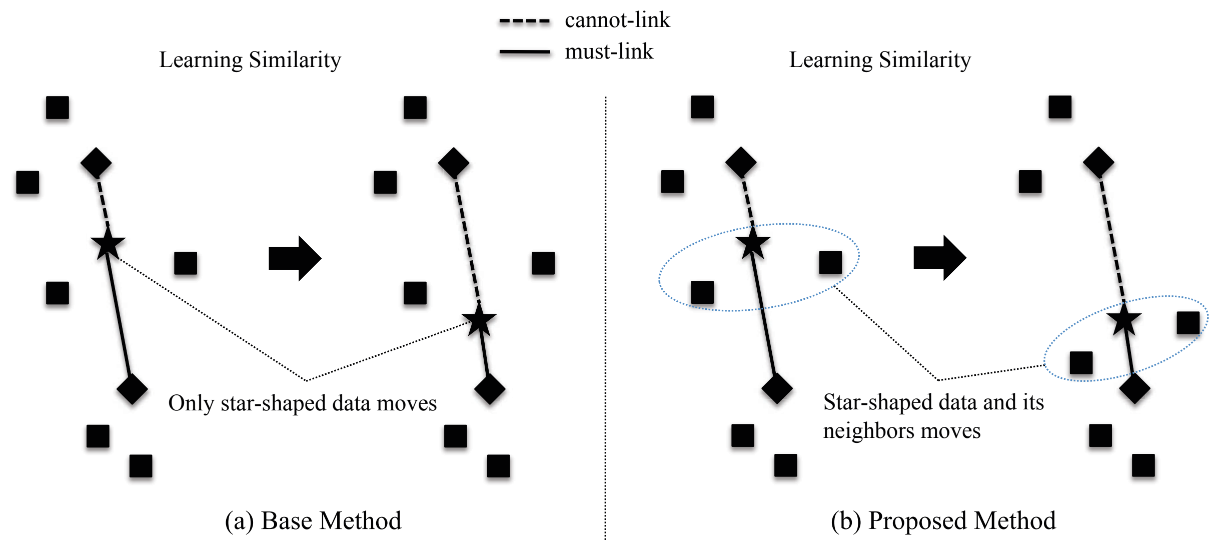

Fig. 1. Difference in neighbor's movement between base and proposed method.

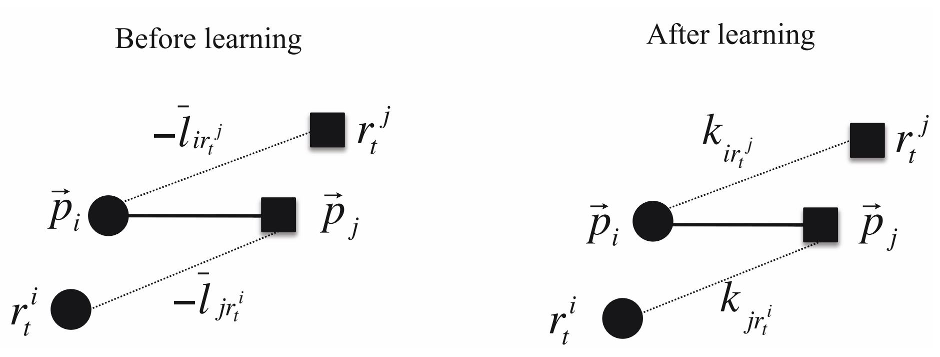

Fig. 2. Relational neighbors.

sure that similarities between constrained pairwise data and their relational neighbors change in the same way.

4. Experiments

In this section, we evaluate our proposed method on several clustering tasks. We used four datasets, Iris, Soybean, Wine, and Glass from the UCI Machine Learning Repository[1]. We standardize attribute values for the Wine dataset alone. The details about these datasets are summarized in **Table 1**. The experimental procedure consists of two main steps. First, a similarity matrix of a dataset is calculated from constraints. Second, a clustering algorithm is applied to the dataset with the above matrix. Then the results are evaluated. We use the k-medoids clustering algorithm in this procedure. A detailed description of the algorithm is shown as **Algorithm 1**.

The elements of the initial similarity matrix are calculated by using $s_{ij} = e^{-d_{ij}^2/\sigma^2}$, where s_{ij} is a value of the similarity between data i and j, d_{ij} is an Euclidean distance, and σ is the parameter shown in **Table 1**. We use the SDPA package[2] for solving the semidefinite programming.

Table 1. Datasets.

Dataset	Iris	Soybean	Wine	Glass
data	150	47	178	214
attribute	4	35	13	9
class	3	4	3	6
σ	0.271	0.5	0.055	0.473

We compare the following two methods of similarity learning to investigate the effect of additional constraints in our method.

- A method that uses only given constraints (data pairs of must/cannot-link) for learning a similarity matrix. (KK-MEANS)

1. http://www.ics.uci.edu/˜mlearn/

2. http://sdpa.indsys.chuo-u.ac.jp/sdpa/

Algorithm 1 Clustering Procedure

```
1: Calculate initial similarity matrix $S$.
2: Solve the optimization problem described in Section 3 to obtain new similarity matrix $K$.
3: Select initial cluster centers $c_i(i = 1 \sim N_k)$.
4: For each data $p_i$, sort $c_i$ in descending order of similarity between $p_i$ and $c_i$. Assign $p_i$ to top $c_i$.
       $$D = \sum_{t=1}^{N_k} \sum_{p_i \in C_k} (p_i - c_t)^2$$
   $C_k$ is a cluster. Let $D$ as $D^{orig}$ If $D^{tmp} - D^{orig} < 0$, stock pair $(p_i, c_i)$. Then find pair $(p_i^*, c_i^*)$ that produces smallest $D^{tmp} - D^{orig}$. Replace $p_i^*$ and $c_i^*$. Let $D^{orig} = D^{tmp}(p_i^*, c_i^*)$, and go to Step 2. If no pair $(p_i, c_i)$ produces $D^{tmp} - D^{orig} < 0$, stop and return temporal clusters.
```

- A method that uses given constraints and constraints concerning the relational neighbors of given constraints. (NKK-MEANS). For this method, we test $t = 1 \sim 5$ (the number of relational neighbors) on each dataset.

We used *Normalized Mutual Information* (NMI) as the performance measure. This measure is defined as follows:

$$\mathrm{NMI}(C,T) = \frac{I(C,T)}{\sqrt{H(C)H(T)}} \quad \dots\dots\dots \quad (7)$$

C is the set of clusters returned by the K-means 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.

Following Li et al. [11], we test different sets of constraints. For the j-th constraint set, j must-link constraints for each class, and j cannot-link constraints for every two classes are randomly generated. Thus for a database of k classes, the j-th constraint set consists of a total of $j \times (k + k(k-1)/2)$ constraints. We also follow the setting of j that ranges from 1 to 10.

The results are shown in **Figs. 3-6**. **Fig. 3** shows the Iris dataset results. The horizontal axis shows the number of constraints, and the vertical axis represents the NMI value. For every method, the performance improves as the number of constraints increases with few exceptions. These exceptions suggest that some constraints do not help to improve the performance. In this dataset, our method outperforms the base one on almost every point. The Soybean in **Fig. 4** shows similar characteristics to the Iris dataset. In the Wine dataset (**Fig. 5**), our method performed pretty well although the constraints do not help to improve the performance. However, our method does not work well or performs worse in the Glass dataset where the base method maintains a stable performance. We consider the reason of the result and the effect of the constraints of relational neighbors in the next section.

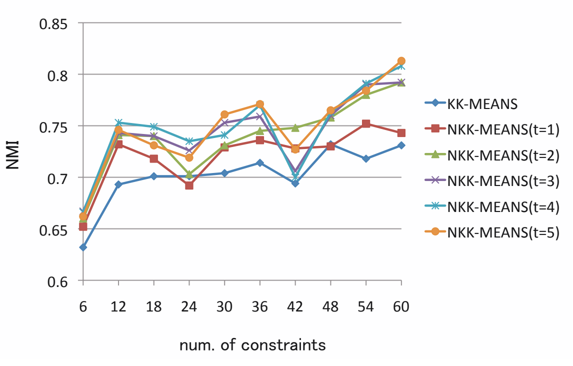

Fig. 3. Iris.

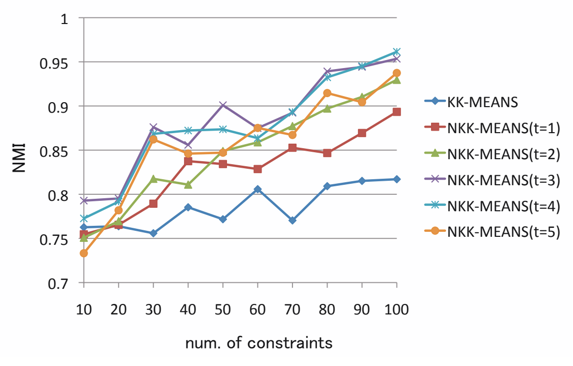

Fig. 4. Soybean.

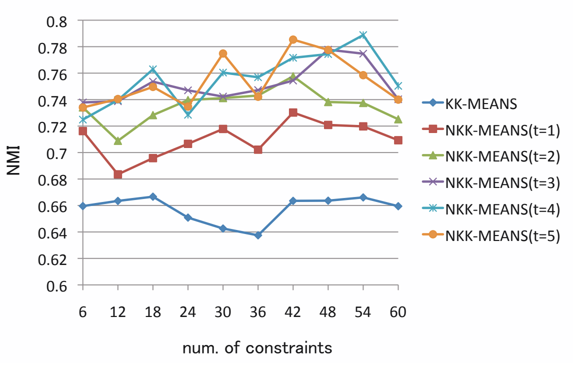

Fig. 5. Wine.

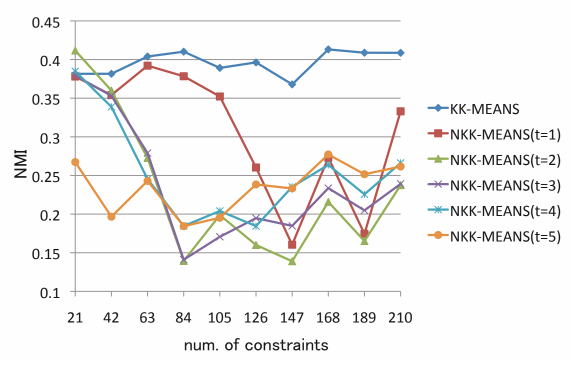

Fig. 6. Glass.

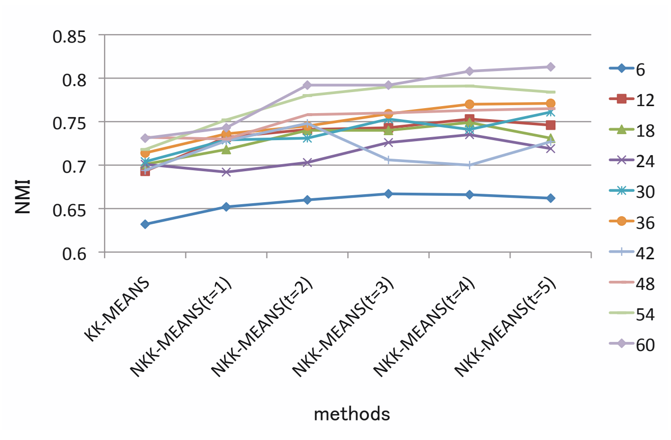

Fig. 7. Iris.

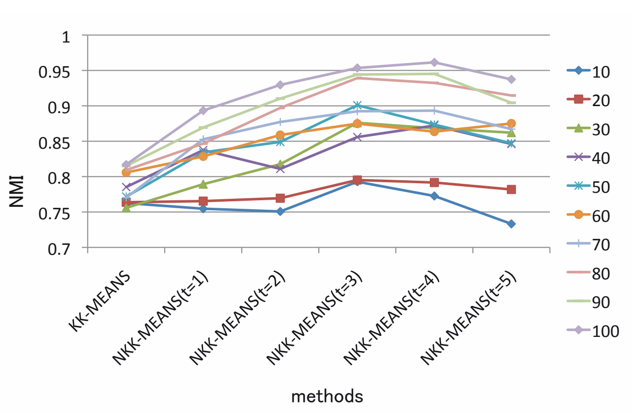

Fig. 8. Soybean.

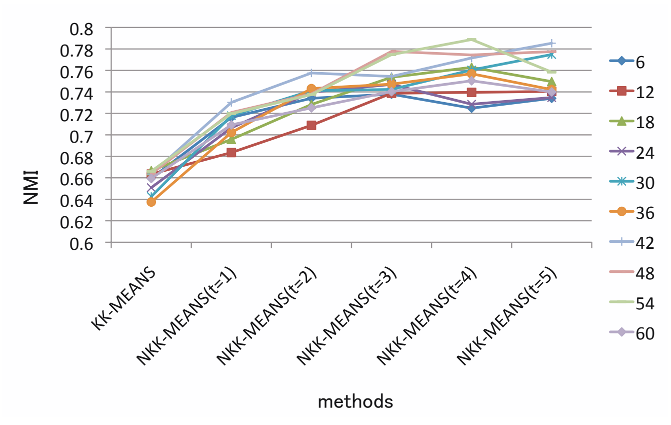

Fig. 9. Wine.

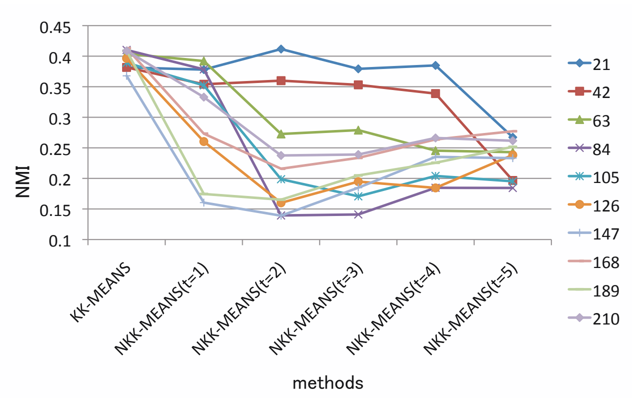

Fig. 10. Glass.

5. Discussion

In the previous section, we stated that our method outperforms the base method in three datasets. We first investigated the effect of additional constraints that are used in our proposed method in more detail. **Figs. 7-10** show the performance of each method for each set of constraints. The number listed in the right side of each graph is the number of constraints. Each method is shown along horizontal axis. For example, NKK-MEANS ($t = 2$) is our proposed method where two relational neighbors are considered for each other's pairwise data. Thus, in this case, a total of four additional constraints is appended when calculating a similarity matrix.

These graphs show the effect of the number of relational neighbors. Although the appropriate number of relational neighbors may differ from dataset to dataset, 3∼5 is the proper range for the datasets used in this experiment (without Glass dataset). It seems that the effect of the number of relational neighbors is not related to the number of constraints. In the Iris dataset, the performance partially decreases when the number of constraints was 24 and 42. However, the strength of the influence differs from the number of relational neighbors. For example, when the number of constraints is 24, the influence is bad at $t = 1,2$, however, not so much at $t = 3,4$. In contrast, when the number of constraints is 42, the influence is limited at $t = 1,2$, but, terrible at $t = 3,4$. These results indicate that there may be good or bad constraints combination. Thus, we need to select an appropriate set of constraints, and this is our open problem.

Table 2. Distribution of class members.

Iris	(50,50,50)
Soybean	(10,10,10,17)
Wine	(48,59,71)
Glass	(9,13,17,29,70,76)

Finally, we investigated why our method does not work on the Glass dataset. **Table 2** lists the distributions of the number of class members in each dataset. For example, the distribution $(48, 59, 71)$ in the Iris dataset represents that it has three classes and the number of members in each class is $48, 59, 71$, respectively. The Glass dataset is relatively different from the others. It has more classes (6 classes), and the distribution bias is relatively large compared with the other datasets. Our method takes into account the same number of relational neighbors even if there is a lot of bias between the number of class members for a pairwise data. Although this may not become a problem when a pairwise data is must-linked, it may be a problem when the pair is cannot-linked because the probability of the member of a small class may be highly constrained, and then they may be included in the conflicted constraints. Given this perspective, we need to create a function that controls the number of relational neighbors according to the size of a class that a constrained data pair belongs.

6. Conclusion

We proposed a method for learning a similarity matrix from pairwise constraints. Our method is based on the same approach proposed by Li et al., which produces a similarity matrix by solving an optimization problem as semidefinite programming. However, we impose additional constraints whereas the neighbors of constrained pairwise data are also influenced by the constraints, i.e., neighbors of a must-linked pair also become similar to that pair, and the neighbors of a cannot-link pair also become dissimilar. Experimental results for several clustering tasks show that our approach is promising even though we must test it on other test beds and analyze it in detail.

In future work, we plan to investigate the influence of the number of neighbors and develop how to control it, and will try to find more effective constraints on the neighbors.

References:

[1] W. Wu, C. Yu, A. Doan, and W. Meng, "An interactive clustering-based approach to integrating source query interfaces on the deep Web," In Proc. of the 2004 ACM SIGMOD Int. Conf. on Management of data, pp. 95-106, 2004.

[2] D. Lewis and W. Gale, "A sequential algorithm for training text classifiers," In Proc. of the Seventeenth ACM-SIGIR Conf., pp. 3-12, 1994.

[3] S. Basu, I. Davidson, and K. L. Wagstaff, "Constrained Clustering," CRC Press, 2008.

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

[5] K. Wagstaff and S. Roger, "Constrained K-means Clustering with Background Knowledge," In Proc. of the 18th Int. Conf. on Machine Learning, pp. 577-584, 2001.

[6] D. Klein, S. D. Kamvar, and C. D. Manning, "From Instance-level Constraints to Space-Level Constraints: Making the Most of Prior Knowledge in Data Clustering," In Proc. of the 19th Int. Conf. on Machine Learning, pp. 307-314, 2002.

[7] S. Shwartz, Y. Singer, and A. Y. Ng, "Online and Batch Learning of Pseudo-Metrics," In Proc. of the 21st Int. Conf. on Machine Learning, pp. 94-101, 2004.

[8] J. Davis et al., "Information-Theoretic Metric Learning," In Proc. of the 24th Int. Conf. on Machine Learning, pp. 209-216, 2007.

[9] W. Tang, H. Xiong, S. Zhong, and J. Wu, "Enhancing Semi-Supervised Clustering: A Feature Projection Perspective," In Proc. of the 13th Int. Conf. on Knowledge Discovery and Data Mining, pp. 707-716, 2007.

[10] S. C. H. Hoi, R. Jin, and M. R. Lyu, "Learning Nonparametric Kernel Matrices from Pairwise Constraints," In Proc. of the 24th Int. Conf. on Machine Learning, pp. 361-368, 2007.

[11] Z. Li, J. Liu, and X. Tang, "Pairwise Constraint Propagation by Semidefinite Programming for Semi-supervised Classification," In Proc, of the 25th Int. Conf. on Machine Learning, pp. 576-583, 2008.

[12] K. Fujisawa, M. Kojima, and K. Nakata, "Exploiting sparsity in primal-dual interior-point methods for semidefinite programming," Mathematical Programming, Series B 79, pp. 235-253, 1997.

[13] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," In Proc. of the seventh ACM SIGKDD Conf., pp. 269-274, 2001.

Name:
Masayuki Okabe

Affiliation:
Toyohashi University of Technology

Address:
1-1 Tenpaku, Toyohashi, Aichi 441-8580, Japan

Brief Biographical History:
2001- Researcher of JST at Kyoto University
2003- Research Associate, Toyohashi University of Technology

Main Works:
• Information retrieval
• Machine learning

Membership in Academic Societies:
• The Japanese Society for Artificial Intelligence (JSAI)

Name:
Seiji Yamada

Affiliation:
National Institute of Informatics, the Graduate University for Advanced Studies (SOKENDAI)

Address:
2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan

Brief Biographical History:
1989- Research Associate, Osaka University
1991- Lecturer, Osaka University
1996- Associate Professor, Tokyo Institute of Technology
2002- Professor, National Institute of Informatics, SOKENDAI

Main Works:
• Human-agent interaction
• Intelligent interactive systems
• Interactive machine learning

Membership in Academic Societies:
• The Institute of Electrical and Electronics Engineers, Inc. (IEEE)
• Association for the Advancement of Artificial Intelligence (AAAI)
• The Japanese Society for Artificial Intelligence (JSAI)