2012 IEEE 12th International Conference on Data Mining

Clustering by Learning Constraints Priorities

Masayuki Okabe
Toyohashi University of Technology
Toyohashi, Japan
okabe@imc.tut.ac.jp

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

***Abstract*—A method for creating a constrained clustering ensemble by learning the priorities of pairwise constraints is proposed in this paper. This method integrates multiple clusters produced by using a simple constrained K-means algorithm that we modify to utilize the constraints priorities. The cluster ensemble is executed according to a boosting framework, which adaptively learns the constraints priorities and provides them for the modified constrained K-means to create diverse clusters that finally improve the clustering performance.**

The experimental results show that our proposed method outperforms the original constrained K-means and is comparable to several state-of-the-art constrained clustering methods.

***Keywords*-Constrained Clustering; Cluster Ensemble; Boosting; Learning Kernel Matrix; K-means;**

I. Introduction

Constrained clustering [1] is a semi-supervised learning approach that uses pre-given knowledge about data pairs to improve the normal clustering accuracy. There are generally two simple types of knowledge used: constraints concerning data pairs that must (or cannot) be in the same cluster. These are usually called must-link and cannot-link, respectively. Recent research on distance metric learning interprets the constraint information as the distance of data pairs and tries to produce a new distance measure for a complete dataset to ensure the distance of a must-link is small and the distance of a cannot-link is large [2], [3].

They usually use these constraints "softly", permitting certain violation to be overlooked when dealing with optimization problems such as SDP. Unfortunately, such approaches often consume a lot of computational time [4]. Moreover, most of the algorithms for distance metric learning treat constraints equally because no such information is pre-given. However, constraints may not always help to improve clustering performance [5]. It may happen that some sets of constraints works well but some others do not. We focus on these problems and try to create a constrained clustering algorithm that can adaptively select useful constraints as well as being less time consuming.

Our method is based on the COP-Kmeans algorithm [6], which is simple and relatively less time consuming. Since it is hard-constrained clustering algorithms, it sometimes halts along the way and returns no result, especially when the number of constraints is large. We complement such defects and improve its performance using boosting [7] as the way of creating a cluster ensemble [8].

In boosting, we treat COP-Kmeans as a weak learner to predict whether or not each pairwise data is in the same cluster. In order to make COP-Kmeans work as a weak learner, we modify it not to halt even if it violates some constraints, i.e. it may only satisfy partial constraints. However, we positively utilize this property to produce a variety of clustering results by changing the data assignment order with constraints priorities controlled by the boosting. These multiple clustering results are combined as a kernel matrix. Then it is used for the final clustering. The main originality of our proposed methods compared with other boosting based techniques for distance metric or kernel matrix learning [9], [10], [11] is the use of COK-Kmeans. We adjust its algorithm to work as a weak learner for the Adaboost.

Using COP-Kmeans and boosting will produce other benefits. It generally produces a monotonically increasing performance with an increase of the boosting steps. Thus, it can find the trade-off between the performance and the computational time. If we consume more time we can get an improved performance. This is an important feature of the anytime algorithm [12] , which will be required, especially in practical situations.

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

- Using a cluster ensemble technique based on the boosting and a modified COP-Kmeans algorithm that can use the adaptively learned constraints priorities, and it finally produces clusters that satisfy the given constraints as much as possible.
- We can estimate and maintain the balance between the cost and performance because the computational cost of the proposed algorithm linearly increases along with the total number of boosting steps.

II. Overview of the COP-Kmeans

COP-Kmeans is a representative algorithm for the constrained partitional clustering. It is based on a K-means algorithm whose algorithm consists of the following three steps.

1) Selection of initial cluster centers.

1550-4786/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDM.2012.150

2) Assignment of each data to one of the cluster centers.
3) Updating of cluster centers.

COP-Kmeans additionally considers not violating any constraints when assigning data in Step 2. Thus a data will not necessarily be assigned to its nearest cluster center. It may be assigned to the second or nth nearest cluster center to avoid committing constraint violations. Such usage of constraints has the following serious problems. One is the failure of the algorithm. If no cluster centers exist that satisfy all the constraints, the algorithm halts and returns no result. This often occurs when there are many constraints to be satisfied. The other is an unstable performance because clustering results depends on the order of data assignment. Some orders perform well, but some may be bad.

We get rid of such drawbacks according to the following approach, which aims to utilize the power of boosting-based cluster ensemble technique.

- We modify COP-Kmeans not to halt even if it violates some constraints. Instead, we repeat to run the modified COP-Kmeans a certain number of times by changing the data assignment order, and then integrate such produced multiple clustering results as a kernel matrix.
- We adopt the Boosting approach to integrate multiple clustering results in the process. We use it to control the order of data assignment in the modified COP-Kmeans. Boosting calculates the weight of each constraint, which is used as the priority to decide the order of data assignment in the modified COP-Kmeans.

III. Learning Constraints Priorities by Boosting

Following the approaches described in the previous section, we propose a constrained clustering method that consists of a modified COP-Kmeans algorithm and a cluster ensemble framework based on the boosting. We first introduce a boosting-based cluster ensemble framework, which calculates the constraints priorities for our modified COP-Kmeans algorithm to produce various clustering results in each round, and finally integrates them as a kernel matrix to produce the final clusters. Then, we describe our modified COP-Kmeans algorithm in detail, particularly concerning its data assignment procedure using constraints priorities.

A. Boosting Algorithm

Boosting is one of the ensemble learning techniques used to produce a classifier by integrating weak hypotheses generated by a weak learner that outperforms random classifiers to some extent. Although various kinds of boosting techniques have been proposed, our method is based on Adaboost[7], which is a well-known framework to enhance the classifier ability by flexibly changing the weights of the training data.

Algorithm 1 is our clustering algorithm based on Adaboost. According to the normal description of the boosting, we can correspond each element of our algorithm as a

- Weak Learner → modified COP-Kmeans
- Training data → Constraints (must/cannot-link)
- Weights for training data → Priorities for constraints.

Algorithm 1 Boosted Constrained K-means

INPUT: Dataset $X = \{x_1, ..., x_{|X|}\}$,
Constraints $S = \{(i_1, j_1, y_1), ..., (i_{|S|}, j_{|S|}, y_{|S|})\}$
(i, j) is a pair of data index and $y \in \{+1, -1\}$,
Parameters for loss function ρ, ξ, No. of clusters k

OUTPUT: Clusters $C = \{C_1, C_2, ..., C_k\}$

$w_n^0 \leftarrow \frac{1}{|S|} \quad (n = 1 \sim |S|)$

for $t = 1$ to T **do**

Run modified COP-Kmeans algorithm. Then, create kernel matrix K^t

$$K^t(i,j) = \begin{cases} 1 & (x_i, x_j) \text{ belongs to same cluster} \\ -1 & (x_i, x_j) \text{ belongs to different clusters} \end{cases}$$

Calculate error rate ϵ_t

$$\epsilon_t = \frac{\rho}{2} \cdot \frac{\sum_{n=1}^{|S|} w_n^t (1 - y_n K^t(i_n, j_n))}{\sum_{n=1}^{|S|} w_n^t} \tag{1}$$

Using ϵ_t, calculate importance value α_t for K^t

$$\alpha_t = \ln \left\{ \frac{1 - \epsilon_t}{\epsilon_t} \right\} \tag{2}$$

Update weight w_n^{t+1} for each constraint

$$w_n^{t+1} = w_n^t \exp \left\{ \frac{-\alpha_t (y_n K^t(i_n, j_n) - \xi)}{\rho} \right\} \tag{3}$$

end for

Calculate final kernel matrix K

$$K = \sum_{t=1}^{T} \alpha_t K^t \tag{4}$$

Run kernel K-means algorithm with K, and return final set of clusters C

We modify the original COP-Kmeans to use as a weak learner for the boosting. This modified COP-Kmeans produces a constrained clustering according to the constraints priorities given by this boosting process. We will describe how we change the algorithm in the next Section. A constrained clustering produced in each boosting step is represented as a kernel matrix whose element indicates whether or not the corresponding data pair belongs to the same cluster. From the point of a weak learner, the modified COP-Kmeans learns this kernel matrix using the constraints as a set of training data. These kernel matrices are summed up with their importance values that are calculated by using the error rate of the constraints satisfaction in each boosting round. The final clustering result is generated using this final kernel matrix.

Another important role of the boosting framework is to give a priority to each constraint. The constraints that are not satisfied by the weak learner will be given higher priorities in order to be satisfied in the next round. These priorities

are used to control the data assignment order in modified COP-Kmeans. The constraints with higher priorities will be satisfied earlier than those with low priorities. The priority calculation can be dealt with by changing the coefficients of the loss function in the boosting. We will describe how our modified COP-Kmeans works and the relation between the data assignment order and the loss function in boosting later in this paper.

IV. The Weak Learner in boosting

The boosting framework described in the previous section is used as the framework of the cluster ensemble. In the cluster ensemble, clustering results to be integrated should be as diverse as possible. We create this diversity by changing the constraints priorities. In this section, we explain how the modified COP-Kmeans uses the priorities and how the loss function in the boosting affects the calculation of the priorities.

A. Modified COP-Kmeans

As described in Section II, we modified the COP-Kmeans algorithm so that it is suitable for use as a weak learner in the boosting. Our modified COP-Kmeans algorithm has the following refinements compared to the original one.

- It produces a clustering result regardless if it fails to satisfy some constraints.
- It utilizes the constraints priorities that are continuously controlled by the boosting process in order to decide the data assignment order in the clustering.

The order of the data assignment in COP-Kmeans is usually decided at random. Our modified algorithm decides the order by sorting with constraints priorities so that a constraint with higher priority is more likely to be satisfied than with lower priority since the likelihood of constraint satisfaction generally increases if it is considered earlier than the other constraints. Our algorithm uses this empirical but practically useful heuristic to decide the order.

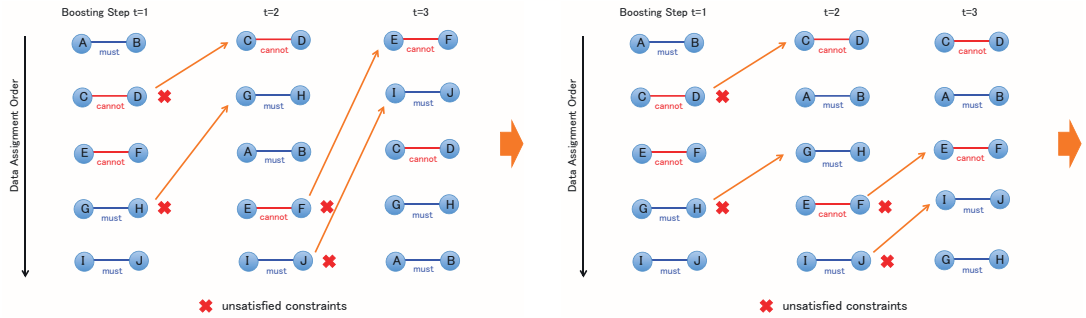
Algorithm 2 lays out the entire procedure of our modified COP-Kmeans. Although it follows the basic procedure of the standard K-means algorithm, which assigns data to its nearest cluster center, its assignment process is rather complicated. There are mainly two parts to the assignment processes, which consists of the procedure for the constrained and unconstrained data, respectively. The latter one (for the unconstrained data) remains the same as that in a normal K-means process. What we need to consider is the process for the previous one (for constrained data). Since the objective of this modification is to use the constraints priorities and they are given for data pairs, our algorithm first assigns constrained data pairs. The assignment procedure for a data pair is more complicated than a single data assignment. We must take several cases like those listed below into consideration.

Algorithm 2 Modified COP-Kmeans

```
INPUT: Dataset X, Constraints S, No. of clusters k,
       Constraints Priorities w_n(n = 1 ~ |S|)
OUTPUT: Clusters C

Select initial cluster centers
for r = 1 to r_max do
  Sort constraints in descending order of w_n^t and assign each corresponding data pair (x_i, x_j) to cluster centers considering the following cases.
  if Both of (x_i, x_j) is not assigned then
    Let c_i, c_j be nearest cluster centers for x_i, x_j respectively, then let d(x_i, c_i), d(x_j, c_j) be distances between each data and its nearest cluster center.
    if (x_i, x_j) is constrained by must-link then
      if d(x_i, c_i) < d(x_j, c_j) then
        Assign x_i and x_j to c_i
      else
        Assign them to c_j
      end if
    else if (x_i, x_j) is constrained by cannot-link then
      if c_i ≠ c_j then
        Assign x_i to c_i, and x_j to c_j, respectively
      else
        if d(x_i, c_i) < d(x_j, c_j) then
          Assign x_i to c_i, x_j to second nearest center
        else
          Assign x_j to c_j, x_i to second nearest center
        end if
      end if
    end if
  else if x_i is assigned and x_j is not assigned then
    Let c_i be cluster center to which x_i is assigned
    if x_i and x_j are constrained by must-link then
      Assign x_j to c_i
    else if x_i and x_j are constrained by cannot-link then
      Assign x_j to the nearest cluster center but c_i
    end if
  else if x_i is not assigned and x_j is assigned then
    Let c_j be cluster center to which x_j is assigned
    if x_i and x_j are constrained by must-link then
      Assign x_i to c_j
    else if x_i and x_j are constrained by cannot-link then
      Assign x_i to the nearest cluster center but c_j
    end if
  end if
  Assign unconstrained data to their nearest cluster centers
  if Clustering result does not change from previous one then
    Return result and exit
  else
    Update cluster centers and go to next step
  end if
end for
```

- **Whether or not one of the data pairs has already been assigned?**
  Our algorithm assigns a constrained data pair at the same time. Since some data contain several constraints, one (or both) of the data may have already been assigned in some cases. We must prepare procedures for such situations.



(a) Radical change (b) Gradual change

Figure 1. Behavior of data assignment order

- **Which constraint the data pair has? - must-link or cannot-link?**
  Depending on the situation described above, we must prepare different procedures according to which constraint the data pair has.

We describe the concrete procedure considering the above cases in Algorithm 2 (l.9 $\sim$ 42).

B. Loss Function and Priority Calculation

One of key roles of the weak learner in the boosting is to create diverse hypotheses. In our case, a hypothesis corresponds to a set of clusters or a kernel matrix that is created by our modified COP-Kmeans, and the diversity can be created by changing the data assignment order. If the order changes, the subset of satisfied constraints will change, which indirectly produces diverse clusters.

The data assignment order is controlled by constraints priorities, which are calculated by using formula (3) in Algorithm 1 (l.8). We briefly explain the introduction of the formula according to the minimization of the expected loss in the boosting, and then, describe how the parameters introduced by us affect the calculation of the priorities and the data assignment order.

We use the following loss function in our boosting Algorithm 1.

$$L(y, f(x_i, x_j)) = \exp(\frac{\xi - yf(x_i, x_j)}{\rho}) \qquad (5)$$

where $f(x_i, x_j)(= \sum_t^T \alpha^t K^t(i,j))$ is a function to predict whether a data pair (x_i, x_j) is a must-link or cannot-link and y indicates -1 or 1 if a data pair (x_i, x_j) belongs to the same cluster or not. This function is slightly different from the one used in Adaboost. The loss function of Adaboost is $\exp(-f(x_i, x_j))$ (stylized in our notation).

The main reason we introduce parameters ρ, ξ is to soften the gap in the priority values between the satisfied and unsatisfied constraints. If we use a loss function used in the standard Adaboost algorithm, the priority calculation is too radical to create variations of the data assignment order. The loss function tends to produce only a radically changed order. However, we find that such radical changes in the order are not suitable for creating clustering variations, and that gradual change is preferable.

Figure 1 illustrates how the data assignment order changes depending on the loss function. Figure 1 (a) shows the behavior of the standard loss function. The order radically changes when the boosting step increases. On the other hand, Figure 1 (b) shows the behavior of our loss function. The order can change rather gradually if we provide appropriate values for parameters ρ and ξ.

Table I
UCI DATASETS

Name	No. of Data	No. of Class	No. of Attribute
iris	150	3	4
wine	178	3	13
parkinsons	195	2	23
sonar	208	2	60
segmentation	210	7	19
glass	214	6	10
ionosphere	351	2	34
wdbc	569	2	30
balance	625	3	4

V. EXPERIMENTS

We evaluated our proposed method on six datasets from the UCI repository[1]. These datasets are summarized in detail in Table I.

We compared the following methods.

- **BCKM**: Boosted Constrained K-means is our proposed method. We set parameters $\rho = 5.0$ and $\xi = 0.5$ respectively. The boosting step T is set to 100.
- **DBOOST**: DistBoost [10] is a method to learn a distance function based on boosting and a Gaussian mixture model. The approach is similar to ours. We set its boosting step $T = 100$, and use default values for other parameters. The learned distance function is used for normal k-means clustering algorithm. We evaluate the performance of the produced clusters.

[1] http://archive.ics.uci.edu/ml/

- **ITML**: Information Theoretic Metric Learning [2] is a state-of-the-art distance metric learning method. We set default values for the parameters. The learned distance metric (Mahalanobis distance metric with learned translation matrix) is used for normal k-means clustering algorithm. We evaluate the performance of the produced clusters.
- **RCKM**: This method uses randomly calculated priorities. The other part is the same as our method. We conduct this method to evaluate the effect of the data assignment order controlled by the constraints priorities calculated through the boosting.
- **CKM**: This is the original COP-Kmeans. Using the same way in our modified COP-Kmeans, we permit constraint violation to produce final clusters in any case.

Distance metrics we used for BCKM, RCKM and CKM in the experiments were all the Euclid distance.

We used normalized mutual information (NMI) [8] to measure the clustering accuracy.

Constraints are randomly selected by first picking a data pair, then give it a label of must-link or cannot-link using true class labels in each dataset. Thus cannot-link is relatively more frequent than must-link in any datasets. We changed the number of constraints to be tested from 100 to 500 in 100 steps and prepared 10 sets of constraints for each number because algorithm performance generally changes well depending on a set of constraints. The performance at n constraints is the average of the results on those 10 sets.

A. Clustering Accuracy

Figure 2 shows the results. In every graph, horizontal axis indicates the number of given constraints, and vertical axis indicates the NMI value. DBOOST failed to produce results in parkinsons, sonar, segmentation ionosphere and balance datasets because it halted by an error when calculating the inverse of covariant matrix for Gaussian distribution.

Our method BCKM showed good performance on most of the datasets. It outperformed others when the number of constraints increased. For example, it showed better or comparable performance at 200 or more constraints on iris, parkinsons and sonar, at 300 or more on glass and ionosphere. In particular, BCKM achieved the maximum performance at 500 constraints on parkinsons and sonar datasets.

On the other hand, BCKM showed worse performance compared with others when the number of constraints is relatively small, such as at 100 or 200 constraints on glass, ionosphere, wdbc and balance. This is because even normal constrained k-means can satisfy most of constraints on such small number of constraints. In such cases, data assignment order in our modified COP-Kmeans becomes almost fixed and it cannot create diverse clustering results. In fact, RCKM and CKM showed almost the same performance in such cases.

Table II
COMPUTATION TIME OF 500 CONSTRAINTS (SEC)

	iris	glass	wdbc
CKM	0.05	0.06	0.18
BCKM	3.10	5.56	18.13
DBOOST	17.29	14.96	131.34
ITML	7.56	17.01	184.34

Although DBOOST and BCKM are based on similar boosting frameworks, difference of weak learner and the usage of constraints priorities produced substantially different results. ITML showed better performance, especially when the number of constraints is relatively small though the performance increase was saturated early and was surpassed by BCKM on several datasets.

We could verify the advantage of data assignment order in our modified COP-Kmeans in the experiments because BCKM showed outperformed or comparable performance compared with RCKM and CKM in all the datasets. The improvement is achieved by the effective usage of constraints priorities.

B. Computational Cost

Table II shows the computation time of each algorithm on several datasets with 500 constraints. We selected iris as a standard dataset, glass and wdbc as datasets that have relatively larger number of classes and attributes, respectively.

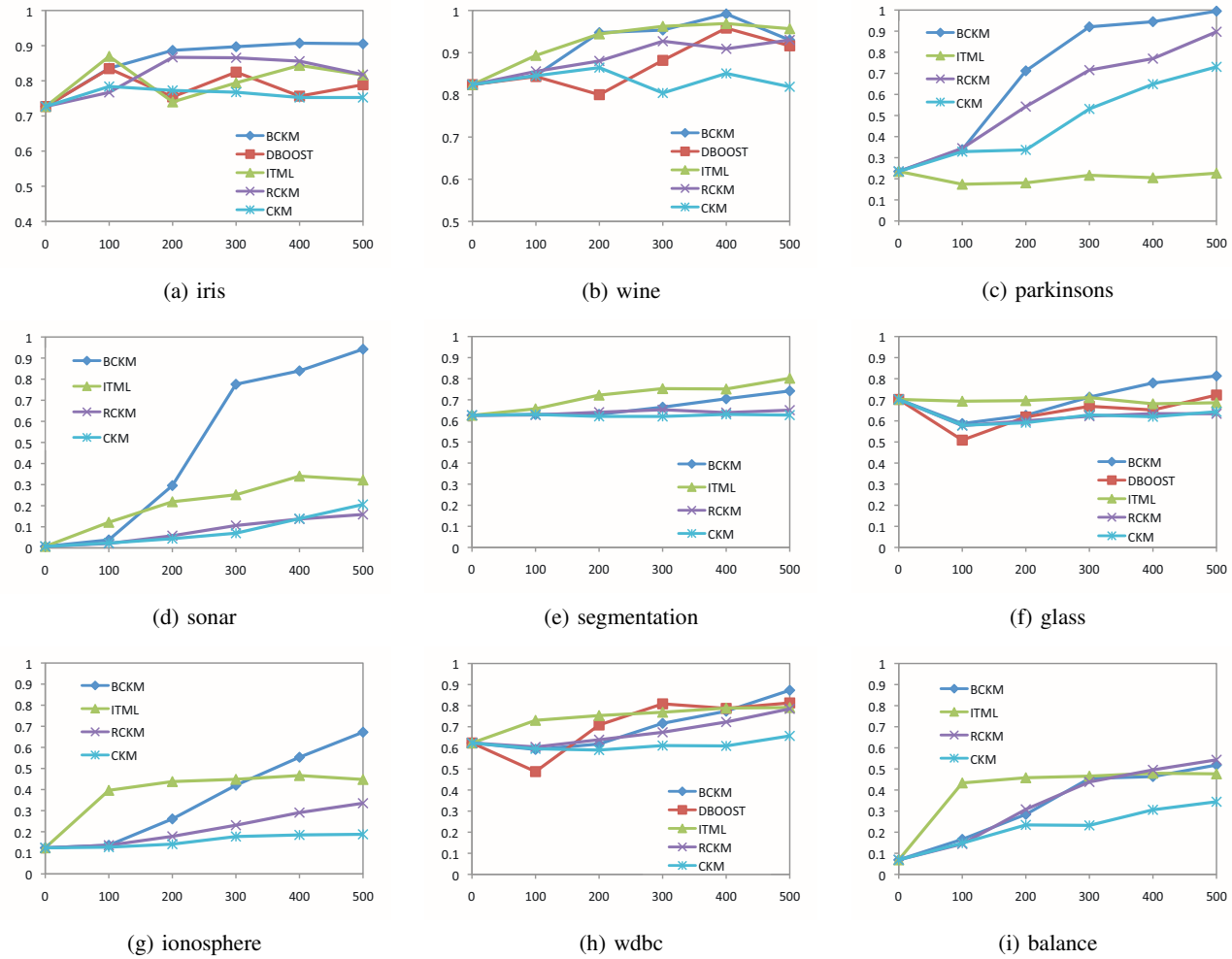
Computation time of CKM is the smallest in every dataset. Though BCKM spent more than CKM since it repeats CKM 100 times (= boosting steps), the cost is less than DBOOST and ITML, especially on wdbc dataset that has a relatively large number of attributes.

Computational cost of BCKM grows linearly depending on the number of data, class and boosting steps. This is an advantage of our method compared with algorithms that suffer from the exponential growth of computational cost with data or attribute size.

VI. CONCLUSION

We proposed a method for creating a constrained clustering ensemble by learning the priorities of pairwise constraints in this paper. This method integrates multiple clusters produced by using a simple constrained K-means algorithm that we modify to utilize the constraints priorities. The cluster ensemble is executed according to a boosting framework, which adaptively learns the constraints priorities and provides them for the modified constrained K-means to create diverse clusters that finally improve the clustering performance.

The experimental results showed that our proposed method outperforms the original constrained K-means and is comparable to several state-of-the-art constrained clustering methods. In addition to the performance, we clarified that our method has the advantage of the computational cost compared with other metric learning based approach.



(a) iris (b) wine (c) parkinsons

(d) sonar (e) segmentation (f) glass

(g) ionosphere (h) wdbc (i) balance

Figure 2. Results on UCI Datasets

REFERENCES

[1] S. Basu, I. Davidson, and K. Wagstaff, *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC, 2008.

[2] J. V. Davis, B. Julis, P. Jain, A. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *ICML*, 2007, pp. 209–216.

[3] K. Weinberger and L. Saul, "Distance metric learning for large margin nearest neighbor classification," *The Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[4] L. Wu, R. Jin, S. C. H. Hoi, J. Zhu, and N. Yu, "Learning bregman distance functions and its application for semi-supervised clustering," in *NIPS*, 2009, pp. 2089–2097.

[5] I. Davidson, K. L. Wagstaff, and S. Basu, "Measuring constraint-set utility for partitional clustering algorithms," in *PKDD*, 2006, pp. 115–126.

[6] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *ICML*, 2001, pp. 577–584.

[7] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT*, 1995, pp. 23–37.

[8] A. Strehl and J. Ghosh, "Cluster ensembles—a knowledge reuse framework for combining multiple partitions," *JMLR*, vol. 3, pp. 583–617, 2003.

[9] K. Crammer, J. Keshet, and Y. Singer, "Kernel design using boosting," in *NIPS*, 2002, pp. 537–544.

[10] T. Hertz, A. Bar-Hillel, and D. Weinshall, "Boosting margin based distance functions for clustering," in *ICML*, 2004, pp. 393–400.

[11] Y. Liu, R. Jin, and A. K. Jain, "Boostcluster: boosting clustering by pairwise constraints," in *KDD*, 2007, pp. 450–459.

[12] S. Zilberstein, "Using anytime algorithms in intelligent systems," *AI Magazine*, vol. 17, no. 3, pp. 73–83, 1996.