2006 IEEE Conference on
Systems, Man, and Cybernetics
October 8-11, 2006, Taipei, Taiwan

Exploration of Different Constraints and Query Methods with Kernel-based Semi-supervised Clustering

Bojun Yan and Carlotta Domeniconi

***Abstract*—Semi-supervised clustering makes use of a small amount of supervised data to aid unsupervised learning. The method used to obtain the supervised information, and the way such information is integrated within the learning algorithm can greatly affect the final result. This paper introduces two different kernel-based semi-supervised clustering algorithms, and investigates the power of kernel methods in principle. Moreover, driven by practice, two methods to obtain supervised data are considered. We compare our kernel-based semi-supervised clustering approaches with semi-supervised KMeans and unsupervised kernel KMeans. The experimental results show that both our methods can outperform the others, regardless of the technique used to generate the supervised data.**

I. Introduction

WITH the wide and rapid spread of applications such as similar text searching [1, 2], image retrieval [3], and clustering of gene microarray data [4], semi-supervised clustering has become a topic of significant research interest. Semi-supervised clustering uses a small amount of supervised data, under the form of class labels or pairwise constraints on some instances, to aid unsupervised learning [5]. The main approaches for semi-supervised clustering can be categorized into three general methods [6]: constrained-based [7, 8, 9, 10], metric-based [11, 12, 13, 14], and the combination of these two techniques.

The KMeans algorithm is the simplest and most commonly used algorithm for clustering. It minimizes the sum of the squared Euclidean distances between the samples and the corresponding centroids [15]. The assumption behind this measure is the belief that the data space consists of isolated spherical regions. However, this assumption is often violated in practice. To tackle this problem, one can investigate other measures, e.g., the cosine similarity used in information retrieval. An alternative idea is to map the data to a new space that satisfies the requirement of the optimization measure.

A number of kernel-based learning methods have been proposed in recent years [16, 17, 18]. All these methods employ kernel functions to increase the separability of data. Generally speaking, a kernel function implicitly defines a non-linear transformation that maps the data from the original space to a high dimensional space (feature space) where the data are expected to be more separable. Consequently, kernel methods may achieve higher performance by working in the new space.

Kernel KMeans [19, 20, 21] takes advantage of kernel-based learning methods, and finds a smooth surface which separates the points belonging to different clusters in feature space. However, like KMeans and any EM algorithms, Kernel KMeans is sensitive to the initial seeds and to the value of K (number of clusters), and has higher time complexity- $o(n^2)$ (where n is the number of data). Thus, its application is limited in practice.

Recently, applications of kernel methods in the context of semi-supervised clustering have attracted more attention. The work in [22] unifies vector-based and graph-based approaches, and shows the theoretical connection between kernel kmeans and several graph clustering objectives. The proposed method allows to perform semi-supervised clustering of data given either as vectors or as a graph.

In this paper we focus on different constraints and query methods for kernel-based semi-supervised clustering. The contributions of this paper are as follows:

1. We introduce and investigate two different kernel-based semi-supervised clustering algorithms: *Seeded Kernel-KMeans* and *Constrained Kernel-KMeans*. We demonstrate that the two proposed semi-supervised clustering algorithms have the ability to cluster the data with non-linear boundaries in input space.

2. Two methods to obtain supervised data are considered: Random selection and Farthest-first querying [5]. We illustrate, using both simulated and real data, the effect of using supervised data generated by these two techniques.

3. We compare our kernel-based semi-supervised clustering approaches with semi-supervised KMeans and unsupervised kernel KMeans. The experimental results show that both our methods can outperform the others, regardless of the technique used to generate the supervised data.

II. From KMeans to Kernel KMeans

Let X be a dataset of N samples $x_1, x_2, \cdots, x_N$. The KMeans algorithm partitions the N samples into K clusters $C_1, C_2, \cdots, C_K$, in such a way that the objective function $J_{KMeans} = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - m_k\|^2$ is locally minimized, where $m_1, m_2, \cdots, m_K$ are the centroids of the clusters.

The key issue extending traditional KMeans to Kernel

This work was supported in part by NSF CAREER Award IIS-0447814.

Bojun Yan is with the Department of Information and Software Engineering, George Mason University, Fairfax VA 22030 USA (e-mail: byan@ gmu.edu).

Carlotta Domeniconi is with the Department of Information and Software Engineering, George Mason University, Fairfax VA 22030 USA. (phone: 703-993-1697; fax: 703-993-1638; e-mail: carlotta@ise.gmu.edu).

KMeans is the computation of distances in a new space. Let $\phi : R^d \rightarrow R^D$ be a non-linear mapping function which maps data from the input (d dimensional) space to a kernel space (D dimensional), with $D > d$. We now follow the standard SVM method, and represent the dot product of points in kernel space using an appropriate Mercer kernel $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$. We use the Gaussian kernel [24] with width parameter q throughout this paper: $K(x_i, x_j) = \exp(-q\|x_i - x_j\|^2)$.

The (squared) Euclidean distance between points x_i and x_j in feature space is written as:

$$D^2(\phi(x_i), \phi(x_j)) = \|\phi(x_i) - \phi(x_j)\|^2$$
$$= \phi(x_i) \cdot \phi(x_i) - 2\phi(x_i) \cdot \phi(x_j) + \phi(x_j) \cdot \phi(x_j)$$
$$= K(x_i, x_i) - 2K(x_i, x_j) + K(x_j, x_j)$$

Let M_k^ϕ be the centroid of cluster C_k in feature space:

$$M_k^\phi = \frac{1}{|C_k|} \sum_{i=1}^{N} \delta(\phi(x_i), C_k)\, \phi(x_i)$$

where

$$\delta(\phi(x_i), C_k) = \begin{cases} 1 & D(\phi(x_i), M_k^\phi) < D(\phi(x_i), M_j^\phi)\ \forall j \neq k \\ 0 & otherwise \end{cases}$$

and $|C_k|$ is the number of samples in C_k .

Then the distance of a point x_i from M_k^ϕ in feature space is expressed as:

$$\left\| \phi(x_i) - \frac{1}{|C_k|} \sum_{j=1}^{N} \delta(\phi(x_j), C_k)\, \phi(x_j) \right\|^2 = A_{ii} + B_{kk} - D_{ik}$$

where: $A_{ii} = \|\phi(x_i)\|^2$

$$B_{kk} = \frac{1}{|C_k|^2} \sum_{j=1}^{N} \sum_{j'=1}^{N} \delta(\phi(x_j), C_k) \delta(\phi(x_{j'}), C_k) \phi(x_j) \cdot \phi(x_{j'})$$

$$D_{ik} = \frac{2}{|C_k|} \sum_{j=1}^{N} \delta(\phi(x_j), C_k)\, \phi(x_i) \cdot \phi(x_j)$$

The above three equations can be rewritten using the kernel trick as:

$$A_{ii} = K(x_i, x_i)$$
$$B_{kk} = \frac{1}{|C_k|^2} \sum_{j=1}^{N} \sum_{j'=1}^{N} \delta(\phi(x_j), C_k) \delta(\phi(x_{j'}), C_k) K(x_j, x_{j'})$$
$$D_{ik} = \frac{2}{|C_k|} \sum_{j=1}^{N} \delta(\phi(x_j), C_k) K(x_i, x_j)$$

The Kernel KMeans locally minimizes the objective function:

$$J_{KKMeans} = \sum_{k=1}^{K} \sum_{x_i \in C_k} \left\| \phi(x_i) - \frac{1}{|C_k|} \sum_{j=1}^{N} \delta(\phi(x_j), C_k)\, \phi(x_j) \right\|^2$$
$$= \sum_{k=1}^{K} \sum_{x_i \in C_k} (A_{ii} + B_{kk} - D_{ik})$$

Since A_{ii} is common to every cluster, we can avoid calculating it, while B_{kk} must be calculated once for each iteration.

III. Semi-supervised Framework

Suppose the dataset X is structured into K clusters $\{C_l\}_{l=1}^{K}$. In a semi-supervised setting, usually there are two ways to obtain a small amount of labeled data S , called the seed set $S \subseteq X$ [5]. In one case, the seed set S is given as prior knowledge, and we assume that for each cluster C_l of X , there is at least one seed point $x_i \subseteq S$. Note that we get a disjoint K partitioning $\{S_l\}_{l=1}^{K}$ of the seed set S . In the second case, we can access a noiseless oracle, and ask queries regarding pairwise points in the dataset. The oracle assigns constraints of the kind “must-link” or “cannot-link” on a given pair of points. A fixed number of queries can be posed to the oracle, depending on the resources available. In order to improve the clustering performance with as few queries as possible, we wish to consult the oracle on the most informative pairwise constraints. In order to achieve this goal, we use an active learning scheme, the Explore algorithm [5], to select the pairwise constraints. The Explore algorithm uses a farthest-first traversal scheme to ask more informative queries, and keeps discovering new clusters as fast as possible to form the seed set $S \subseteq X$ when the number of clusters K is unknown. The details of the Explore algorithm can be found in [5].

IV. Our Approach

A. Seeded-Kernel-KMeans

In seeded-Kernel-KMeans, the seed set is used to initialize Kernel KMeans. Unlike the Seeded-KMeans algorithm [5], in which the centroid of the lth cluster is initialized with the centroid of the lth partition S_l of the seed set, the points of the lth partition S_l of the seed set are now directly used as the initial points for the lth cluster. The seed set is only used for initialization, and it is not used in the following steps of the algorithm. Therefore the labels of the points in the seed set may change during the execution of the algorithm. If the seed set is noisy, or if the user allows the seed set to change labels during the execution of the algorithm, Seeded-Kernel-KMeans is more appropriate than the constrained approach (see below). The details of the Seeded-Kernel-KMeans algorithm are given in Figure 1. Compared with Seeded-KMeans, where only the centroids of different partitions of the seed set are used, Seeded-Kernel-KMeans utilizes the whole seed set, and thus leverages more information from the prior knowledge.

```
Algorithm: Seeded-Kernel-KMeans
Input: Set of data points $X = \{x_i\}_{i=1}^{N}$, number of
    clusters $K$, set $S = \cup_{l=1}^{K}\{S_l\}$ of initial seeds
Output: Disjoint $K$ partitioning $C_1, C_2, \cdots, C_K$ of $X$
    such that Kernel KMeans's objective function is minimized.
Method:
1. Initialization: Use the points $x_i \in S_l$ $(l = 1, \ldots, K)$ as
   the initial points for each cluster, forming $k$ initial
   clusters $C_1, C_2, \cdots, C_k$.
2. For each cluster $C_k$, compute $|C_k|$ and $B_{kk}$
3. For each data point $x_i$ and cluster $C_k$, compute $D_{ik}$.
   Then assign $x_i$ to the closest cluster:
   $\delta(\phi(x_i), C_k) = \begin{cases} 1 & B_{kk} + D_{ik} < B_{jj} + D_{ij} \\ & for\ all\ j \neq k \\ 0 & otherwise \end{cases}$
4. Repeat 2, 3 until convergence.
```

Fig. 1. Algorithm Seeded-Kernel-KMeans

B. Constrained-Kernel-KMeans

In Constrained-Kernel-KMeans, the seed set is used to initialize the Kernel-KMeans algorithm as described for the Seeded-Kernel-KMeans algorithm. However, in the subsequent steps, the cluster membership of the data points in the seed set is not re-computed – the cluster labels of the seed set are kept unchanged, and only the labels of the non-seed data are re-estimated. When the seed set is noise-free, Constrained-Kernel-kMeans is the appropriate choice. In this case, in fact, Constrained-Kernel-KMeans can leverage more information from the seed set. Since the kernel function increases the separability of data, and the cluster membership of the seeds does not change, the algorithm can find a smooth surface in feature space that properly separates the supervised data. The algorithm is given in details in Figure 2.

V. Experimental evaluation

In our experiments we use one synthetic datasets and four real datasets. The synthetic dataset has two uniformly distributed concentric clusters: the radius of the innermost cluster is 1.0 (100 points); the radius of the outer ring is 4.0 (150 points). Three of four real datasets are from the UCI (Iris, Liver, Spectf); the Vowel dataset is available at www-stat-class.stanford.edu/~tibs/ElemStatLearn/ (for our experiments three clusters are used, corresponding to the vowels “i”, “I” and “E”.). Their characteristics are as follows. Iris: 150 points, 4 features and 3 clusters; Liver: 345 points, 6 features, and 2 clusters; Spectf: 267 points, 44 features, and 2 clusters; Vowel: 126 points, 10 features, and 3 clusters. All real data are normalized in our experiments. For both the synthetic and real datasets, we used the class labels to compute error rates according to the confusion matrices. That is: each cluster is assigned the label corresponding to the class which has the largest number of points among those in the cluster. The number of non-majority-class points in all clusters (normalized by the total number of data points) gives the error rate. This computation is carried out for all algorithms being compared in our experiments.

We compare the following clustering algorithms: (1) Seeded-Kernel-KMeans, (2) Seeded-KMeans, (3) Constrained-Kernel-KMeans, (4) Constrained-KMeans and (5) Kernel-KMeans without any supervised information. For the first four algorithms, the seed set is obtained either via prior knowledge, or by means of an oracle. We simulate the prior knowledge approach by randomly selecting points from the dataset, according to the corresponding seed set percentage. The generation of queries by means of an oracle is achieved via the Explore algorithm [2]. The unsupervised Kernel-KMeans algorithm is initialized by choosing well-scattered points among the given data.

```
Algorithm: Constrained-Kernel-KMeans
Input: Set of data points $X = \{x_i\}_{i=1}^{N}$, number of
    clusters $K$, set $S = \cup_{l=1}^{K}\{S_l\}_{l=1}^{K}$ of initial seeds
Output: Disjoint $K$ partitioning $C_1, C_2, \cdots, C_K$ of $X$
    such that Kernel KMeans's objective function is minimized.
Method:
1. Initialize: Use the points $x_i \in S_l$ $(l = 1, \ldots, K)$ as the
   initial points for each cluster, forming $k$ initial
   clusters $C_1, C_2, \cdots, C_k$.
2. For each cluster $C_k$, computer $|C_k|$ and $B_{kk}$
3. For a point $x_i \in S$, if $x_i \in S_l$, assign $x_i$ to the
   cluster $l$. For a point $x_i \notin S$, compute $D_{ik}$. Then
   assign $x_i$ to the closest cluster:
   $\delta(x_i, C_k) = \begin{cases} 1 & B_{kk} + D_{ik} < B_{jj} + D_{ij} \\ & for\ all\ j \neq k \\ 0 & otherwise \end{cases}$
4. Repeat 2, 3 until convergence.
```

Fig. 2. Algorithm Constrained-Kernel-KMeans

For each dataset, and for each method of producing the seed set, we run each algorithm five times, and report the average error rates and standard deviations. Seeds are generated either randomly or using the Explore algorithm. Clearly, when seeds are generated randomly, each of the five runs in our experiments will have (in general) a different seed set. Likewise, the Explore algorithm has a stochastic component, since the first seed point is chosen randomly. Therefore, different runs of the Explore algorithm will generate different seed sets as well.

The Explore algorithm is used to generate both seed sets and pairwise constraints. The procedure is the same in the two cases: at each step, the farthest point from the previously chosen ones is selected. At completion, the resulting set of points, with the corresponding labels, gives a seed set. To obtain pairwise constraints, all pairs of points are considered and translated to must-link or cannot link constraints, depending on whether their class labels match or not.

We set the kernel width parameter value to $q = 0.1$ for the synthetic dataset. For the real datasets, we fix the width parameter to $q = 5.0$.

Figures 3 and 4 show the results for the synthetic dataset. Seeded-Kernel-KMeans, Constrained-Kernel-KMeans, and unsupervised Kernel- KMeans give 0.0% error rate in all cases for $q = 0.1$. For lack of space, we omit these plots. Figures 3(a)-(b) show the results of Seeded- and Constrained-KMeans, respectively, using 20 queries obtained via the Explore algorithm. Similarly, Figures 4(a)-(b) show the corresponding results when the seed set is randomly chosen. In all four cases, the error rate of both Seeded- and Constrained- KMeans is 40%. Clearly, KMeans (both Seeded and Constrained versions) fails to detect the underlying structure of the clusters for this dataset.

The results on the real datasets are shown in Figures 5-8, in which we plot the average error rates (and standard deviations) as a function of the number of queries (or percentage size of the seed set). The semi-supervised kernel methods provide higher performance with respect to the corresponding seeded- and constrained- KMeans methods, and with respect to Kernel-KMeans without any supervised information, for all the four datasets considered. The constrained methods perform better than (or equally to) the seeded methods. For the Iris data, we observed a steep decrease in error (for an increasing number of queries) when the Explore algorithm is used. As expected, in general, the performance of Seeded-KMeans does not scale as more supervised data become available.

We observe that kernel methods require a more costly training phase. While an iteration of KMeans has a complexity of *O(Knd)* (where *K* is the number of clusters, *n* is the number of data, and *d* is the number of dimensions), kernel-KMeans has a complexity of *O(nnd)* (squared in the number of data). Efficient implementations of kernel-KMeans exist to reduce computation and storage costs when dealing with large corpuses of data [25].

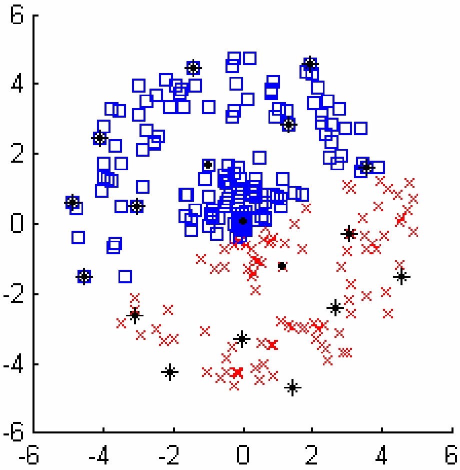

Fig. 3(a). Seeded-KMeans with the seed set obtained via the Explore algorithm. 20 queries; Error rate = 40.0%.

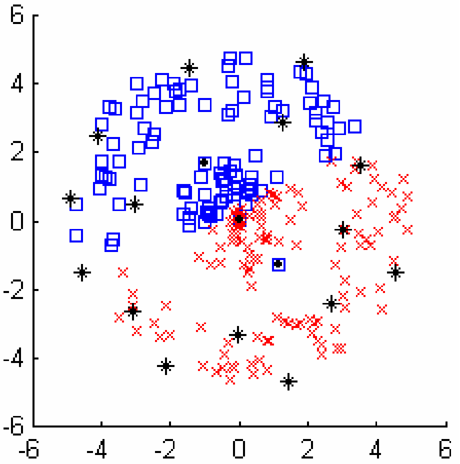

Fig. 3(b). Constrained-KMeans with the seed set obtained via the Explore algorithm. 20 queries, Error rate = 40.0%.

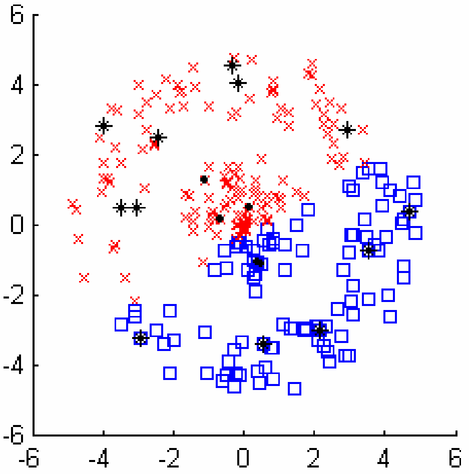

Fig. 4(a). Seeded-KMeans with the seed set randomly chosen (size= 8%). Error rate = 40.0%.

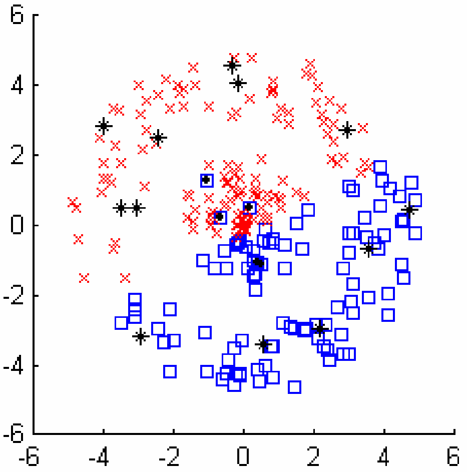

Fig. 4(b). Figure 7(b): Constrained-KMeans with the seed set randomly chosen (size= 8%). Error rate = 40.0%

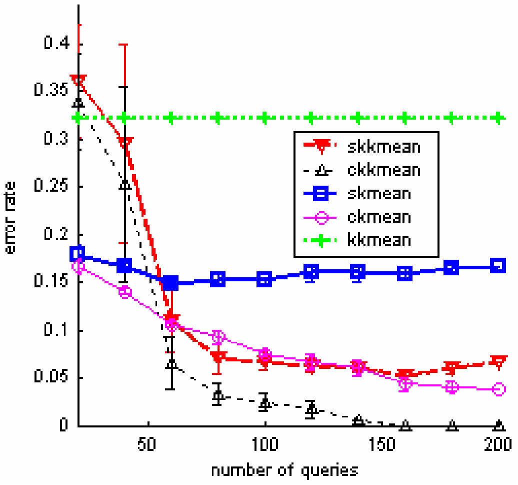

Fig. 5(a). Iris data: Performance comparison ($q = 5.0$). The seed set is obtained via the Explore algorithm

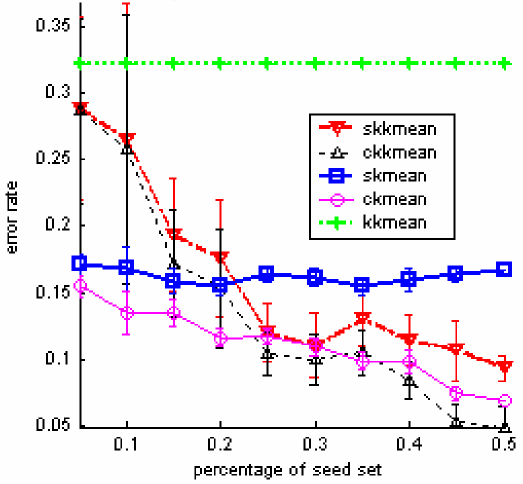

Fig. 5(b). Iris data: Performance comparison ($q = 5.0$). The seed set is randomly selected.

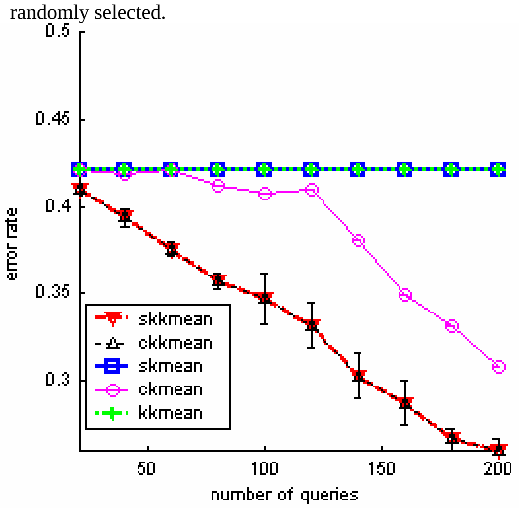

Fig. 6(a). Liver data: Performance comparison ($q = 5.0$). The seed set is obtained via the Explore algorithm

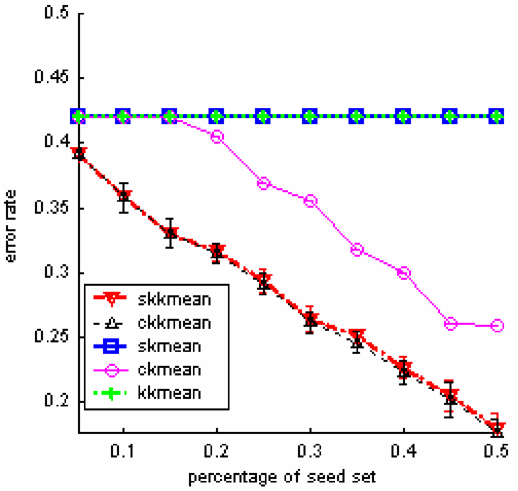

Fig. 6(b). Liver data: Performance comparison ($q = 5.0$). The seed set is randomly selected

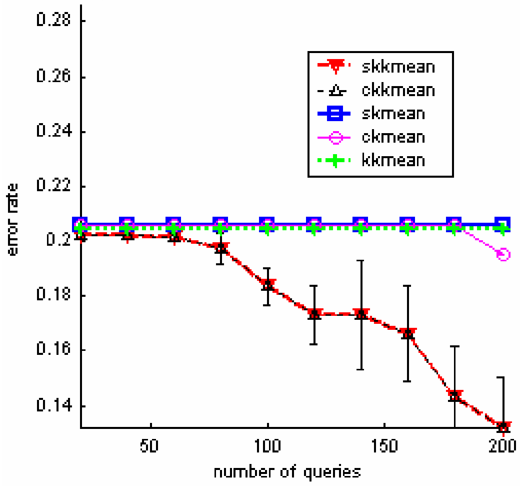

Fig. 7(a). Spectf data: Performance comparison ($q = 5.0$). The seed set is obtained via the Explore algorithm

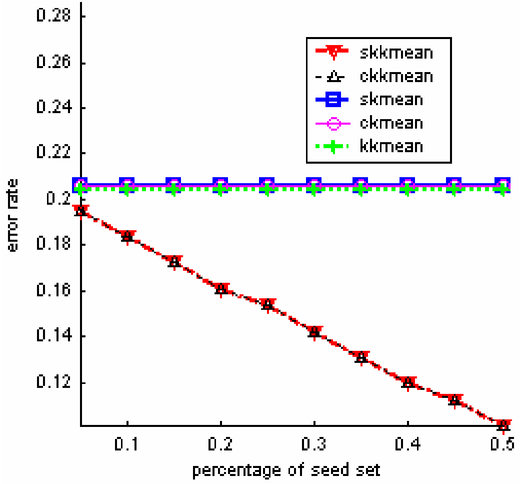

Fig. 7(b). Spectf data: Performance comparison ($q = 5.0$). The seed set is randomly selected

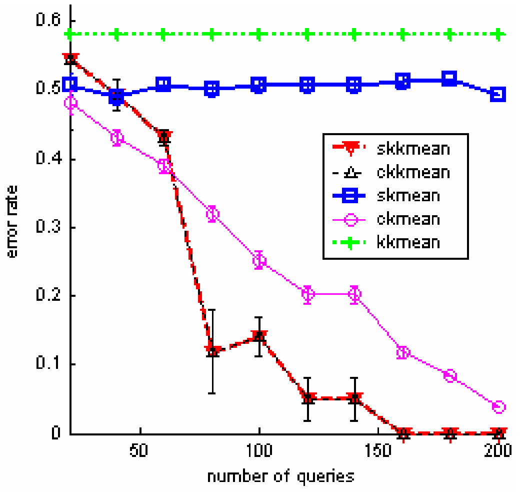

Fig. 8(a). Vowel data: Performance comparison ($q = 5.0$). The seed set is obtained via the Explore algorithm

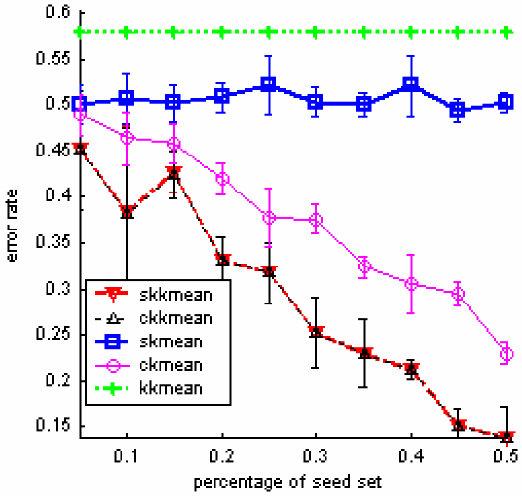

Fig. 8(b). Vowel data: Performance comparison ($q = 5.0$). The seed set is randomly selected

VI. Conclusion and Future Work

Two new semi-supervised variants of Kernel KMeans clustering are introduced: Seeded- and Constrained-Kernel-KMeans. We exploit two different methods to generate supervised data (random selection and the Explore algorithm), and perform experiments using both synthetic and real data. The results show the performance improvements achieved by our semi-supervised Kernel KMeans approaches. In our future work, we plan to study mechanisms to automatically set the width parameter of the Gaussian kernel based on the knowledge of the seed sets. In addition, we will combine our semi-supervised approaches with efficient implementations of Kernel KMeans [25] to reduce computation and storage costs when dealing with large corpuses of data.

References

[1] B. Liu, W. S. Lee, P. S. Yu, and X. Li, Partially Supervised Classification of Text Documents, Intl. Conf. on Machine Learning, 2002.

[2] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, Text classification from labeled and unlabeled documents using EM. Machine Learning, 39: 103-134, 2000.

[3] Y. Chen, X. Zhou, and T. Huang, One-class SVM for learning in image retrieval, IEEE Intl. Conf. on Image Processing, 2001.

[4] M. Brown, W. Grundy, D. Lin, N. Cristianini, et al., Knowledge-based Analysis of Microarray Gene Expression Data By Using Support Vector Machines. National Academy of Sciences, 2000.

[5] S. Basu, A. Banerjee, and R. J. Mooney, Active semi-supervision for pairwise constrained clustering, SIAM Interl. Conf. on Data Mining, 2004.

[6] M. Bilenko, S. Basu, and R. J. Mooney, Integrating constraints and metric learning in semi-supervised clustering, Intl. Conf. on Machine Learning, 2004.

[7] A. Demiriz, K. P. Bennett, and M. J. Embrechts, Semi-superwised clustering using genetic algorithms. Artificial Neural Networks in Engineering, 1999.

[8] S. Basu, A. Banerjee and R. J. Mooney, Semi-supervised clustering by seeding, Interl. Conf. on Machine Learning, 2002.

[9] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, Constrained K-Means clustering with background knowledge, Intl. Conf. on Machine Learning, 2001.

[10] K. D. Kamvar and C. Manning, From instance level constraints to space-level constraints: Making the most of prior knowledge in data clustering, Intl. Conf. on Machine Learning, 2002.

[11] M. Bilenko M and R. J. Mooney, Adaptive duplicate detection using learnable string similarity measures, Intl. Conf. on Knowledge Discovery and Data Mining, 2003.

[12] D. Cohn, R. Caruana, and A. McCallum, Semi-supervised clustering with user feedback (Tech. Report TR2003-1892). Cornell University, 2003.

[13] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, Distance metric learning, with application to clustering with side information, Advances in Neural Information Processing System 15, 2003.

[14] A. Bar-Hillel, T. Hetz, N. Shental, and D. Weinshall, Learning distance functions using equivalence relation, Intl. Conf. on Machine Learning, 2003.

[15] J. MacQueen, Some methods for classification and analysis of multivariate observations. Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967.

[16] C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery, 1998.

[17] B. Scholkopf and A. Smola, Learning with Kernels. MIT Press, 2001.

[18] V. N. Vapnik, Statistical learning theory. Wiley Interscience, 1998.

[19] K. R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithm, IEEE Trans. On Neural Network, 12(2):181-201, 2001.

[20] M. Girolani., Mercer Kernel Based Clustering in Feature Space, IEEE Transactions on Neural Networks, 13(4):780-784, 2002.

[21] S. V. N. Vishwanathan and M. N. Murty, Kernel Enabled K-Means Algorithm, Tech. Rep. 2002.

[22] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, Semi-supervised Graph Clustering: A Kernel Approach. ICML 2005.

[23] N. Cristianini and J. S. Taylor, Support vector machines and other kernel based learning methods, Cambridge University Press, 2000.

[24] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, Support vector clustering, Journal of Machine Research, 2, 125-137, 2001.

[25] R. Zhang and A. I. Rudnicky, A large scale clustering scheme for Kernel K-Means. ICPR 2002.