Mach Learn
DOI 10.1007/s10994-008-5084-4

Semi-supervised graph clustering: a kernel approach

Brian Kulis · Sugato Basu · Inderjit Dhillon · Raymond Mooney

Received: 9 March 2007 / Revised: 17 April 2008 / Accepted: 8 August 2008

Abstract Semi-supervised clustering algorithms aim to improve clustering results using limited supervision. The supervision is generally given as pairwise constraints; such constraints are natural for graphs, yet most semi-supervised clustering algorithms are designed for data represented as vectors. In this paper, we unify vector-based and graph-based approaches. We first show that a recently-proposed objective function for semi-supervised clustering based on Hidden Markov Random Fields, with squared Euclidean distance and a certain class of constraint penalty functions, can be expressed as a special case of the weighted kernel k-means objective (Dhillon et al., in Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining, 2004a). A recent theoretical connection between weighted kernel k-means and several graph clustering objectives enables us to perform semi-supervised clustering of data given either as vectors or as a graph. For graph data, this result leads to algorithms for optimizing several new semi-supervised graph clustering objectives. For vector data, the kernel approach also enables us to find clusters with non-linear boundaries in the input data space. Furthermore, we show that recent work on spectral learning (Kamvar et al., in Proceedings of the 17th International Joint Conference on Artificial Intelligence, 2003) may be viewed as a special case of our formulation. We empirically show that our algorithm is able to outperform current state-of-the-art semi-supervised algorithms on both vector-based and graph-based data sets.

Keywords Semi-supervised clustering · Kernel k-means · Graph clustering · Spectral learning

Editor: Jennifer Dy.

B. Kulis (✉) · I. Dhillon · R. Mooney
Department of Computer Sciences, University of Texas, Austin, TX, USA
e-mail: kulis@cs.utexas.edu

S. Basu
Google, Inc., Mountain View, CA, USA

1 Introduction

Semi-supervised clustering algorithms have recently received a significant amount of attention in the machine learning and data mining communities. In traditional clustering algorithms, only unlabeled data is used to generate clusterings; in semi-supervised clustering, the goal is to incorporate prior information about clusters into the algorithm in order to improve the clustering results. A related but different problem is semi-supervised classification (Chapelle et al. 2006), which considers how unlabeled data can be used to improve the performance of classification on labeled data. A number of recent papers have explored the problem of semi-supervised clustering. Research on semi-supervised clustering has considered supervision in the form of both labeled points (Demiriz et al. 1999; Sinkkonen and Kaski 2002; Basu et al. 2002) or constraints (Wagstaff et al. 2001; Klein et al. 2002; Xing et al. 2003; Bar-Hillel et al. 2003; Bie et al. 2003; Kamvar et al. 2003; Bilenko et al. 2004; Basu et al. 2004a, 2004b; Chang and Yeung 2004; Davidson and Ravi 2005a, 2005b; Law et al. 2005; Lu and Leen 2005; Lange et al. 2005).

As is common for most semi-supervised clustering algorithms, we assume in this paper that we have pairwise must-link constraints (pairs of points that should belong in the same cluster) and cannot-link constraints (pairs of points that should belong in different clusters) provided with the input. Pairwise constraints occur naturally in many domains, e.g., the Database of Interacting Proteins (DIP) data set in biology contains information about proteins co-occurring in processes, which can be viewed as must-link constraints during gene clustering. Constraints of this form are also natural in the context of the graph clustering problem (a.k.a. graph partitioning or vertex partitioning), where edges in the graph encode pairwise relationships. Different application areas of semi-supervised clustering with constraints have been studied recently, including (1) image segmentation for object identification in Aibo robots (Davidson and Ravi 2005a), where cluster-level constraints are used to improve the pixel clustering; (2) object recognition in video sequences (Yan et al. 2004), where must-link constraints are used to group pixels belonging to the same object and cannot-link constraints are used to separate different objects; (3) clustering for lane finding from GPS traces (Wagstaff et al. 2001), where constraints are used to encode lane-contiguity and max-separation criteria; and (4) speaker identification and categorization (Bar-Hillel et al. 2003), where constraints are used to encode whether two speakers are similar or not.

Recently, a probabilistic framework for semi-supervised clustering with pairwise constraints was proposed based on Hidden Markov Random Fields (Basu et al. 2004b). The HMRF framework proposed a general semi-supervised clustering objective based on maximizing the joint likelihood of data and constraints in the HMRF model, as well as a *k*-means-like iterative algorithm for optimizing the objective. However, HMRF-KMEANS can cluster input data only in the form of vectors. While much of the work on semi-supervised clustering has focused on vector-based inputs, very little work has gone into the study of semi-supervised graph clustering algorithms. In this paper, we unify vector-based and graph-based semi-supervised clustering using a kernel approach; in particular, our main contributions in this paper are as follows:

- We show that the HMRF semi-supervised clustering objective with squared Euclidean distance and cluster-size weighted penalties is a special case of the weighted kernel *k*-means objective function (Dhillon et al. 2004a). Given input data in the form of vectors and pairwise constraints, we show how to construct a kernel such that running kernel *k*-means results in a monotonic decrease of this semi-supervised clustering objective function at every iteration of the kernel *k*-means algorithm.

- The weighted kernel k-means algorithm can be used to monotonically optimize a wide class of graph clustering objectives such as minimizing the normalized cut (Dhillon et al. 2007). As a result, our approach can be generalized to optimize a number of different semi-supervised graph clustering objectives for which constraint-based supervision is more natural, including semi-supervised normalized cut, semi-supervised ratio cut, and semi-supervised ratio association objective functions. This equivalence gives us a new semi-supervised clustering algorithm SS-KERNEL-KMEANS that can cluster data given either as vectors or a graph, along with supervision in the form of constraints.
- For vector-based data we have the ability to apply a kernel function, which allows SS-KERNEL-KMEANS to discover clusters with non-linear boundaries in the input space.
- We show how a previously proposed algorithm SPECTRAL-LEARNING (Kamvar et al. 2003) can be viewed as a special case of our framework. The SPECTRAL-LEARNING algorithm can be viewed as optimizing an underlying semi-supervised clustering objective; specifically, it optimizes a relaxation of semi-supervised ratio cut.
- We empirically demonstrate that SS-KERNEL-KMEANS outperforms HMRF-KMEANS (Basu et al. 2004b) and SPECTRAL-LEARNING on several vector-based and graph-based data sets, including handwritten digits data, protein data, a gene network, and a data set of images.

We note that an earlier version of this paper appears as Kulis et al. (2005). This extended version further develops the mathematical connection between kernel k-means and graph clustering for semi-supervised clustering, and presents several new experimental results.

2 Background and related work

In this section, we provide the necessary background for our proposed kernel-based semi-supervised clustering algorithm SS-KERNEL-KMEANS, and also describe related research.

2.1 Graph clustering and kernel k-means

In graph clustering, the input is assumed to be a graph $G = (\mathcal{V}, \mathcal{E}, A)$, where $\mathcal{V}$ is the set of vertices, $\mathcal{E}$ is the set of edges, and A is the edge adjacency matrix. A_{ij} represents the edge-weight between vertices i and j.

Let $\text{links}(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}, j \in \mathcal{B}} A_{ij}$, where $\mathcal{A}$ and $\mathcal{B}$ are subsets of vertices. Furthermore, let $\text{degree}(\mathcal{A}) = \text{links}(\mathcal{A}, \mathcal{V})$. We seek to find a k-way disjoint partitioning[1] $\{\mathcal{V}_c\}_{c=1}^{k}$ of $\mathcal{V}$ to minimize a particular objective. Table 1 gives a list of some common graph clustering objectives. Ratio association tries to maximize the affinity/similarity of points within a cluster,

Table 1 Examples of graph clustering objectives

Name	Objective function
Ratio association	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{maximize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{\lvert\mathcal{V}_c\rvert}$
Ratio cut	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{minimize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}\backslash\mathcal{V}_c)}{\lvert\mathcal{V}_c\rvert}$
Normalized cut	$\underset{\mathcal{V}_1,\ldots,\mathcal{V}_k}{\text{minimize}} \sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}\backslash\mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)}$

[1] k disjoint data subsets, whose union is the whole data.

normalizing each cluster's contribution to the objective by the size of the cluster in order to balance the size of the clusters. Ratio cut (Chan et al. 1994), on the other hand, tries to minimize the total cost of the edges crossing the cluster boundaries—this is again normalized by the size of the clusters, to encourage balanced cluster sizes. The normalized cut criterion (Shi and Malik 2000) uses the same objective as ratio cut but normalizes by the total degree of each cluster (the sum of the degrees of all nodes in the cluster)—this encourages the clusters to have similar degrees.

To make the connection between graph clustering and vector-based clustering, we introduce the weighted kernel k-means objective function (Dhillon et al. 2004a). Given a set of data vectors $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^n$, the goal of weighted kernel k-means is to find a k-way disjoint partitioning $\{\pi_c\}_{c=1}^k$ of the data (where π_c represents the c^{th} cluster) such that the following objective is minimized:

$$\mathcal{J}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \alpha_i \|\phi(\mathbf{x}_i) - \mathbf{m}_c\|^2 \quad \text{where } \mathbf{m}_c = \frac{\sum_{\mathbf{x}_i \in \pi_c} \alpha_i \phi(\mathbf{x}_i)}{\sum_{\mathbf{x}_i \in \pi_c} \alpha_i}.$$

Each vector $\mathbf{x}_i$ has a pre-specified non-negative weight α_i associated with it, and ϕ is a function mapping vectors in $\mathcal{X}$ to a (generally) higher-dimensional space. If all weights are set to one and ϕ is the identity function, this reduces to standard k-means.

If we expand the distance computation $\|\phi(\mathbf{x}_i) - \mathbf{m}_c\|^2$ in the weighted kernel k-means objective function, we obtain the following:

$$\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i) - \frac{2\sum_{\mathbf{x}_j \in \pi_c} \alpha_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)}{\sum_{\mathbf{x}_j \in \pi_c} \alpha_j} + \frac{\sum_{\mathbf{x}_j, \mathbf{x}_l \in \pi_c} \alpha_j \alpha_l \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_l)}{(\sum_{\mathbf{x}_j \in \pi_c} \alpha_j)^2}.$$

Notice that all computation involving data points is in the form of inner products. We can therefore use the *kernel trick*: if we can compute the inner product $K_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ efficiently, we are able to compute distances between points in this mapped space without explicitly knowing the mapping of $\mathbf{x}_i$ and $\mathbf{x}_j$ to $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$. We assume that the input is the kernel matrix K of inner products; it can easily be shown that *any* positive semidefinite matrix K can be thought of as a kernel matrix (Cristianini and Shawe-Taylor 2000).

Using the kernel matrix K, the distance computation $d(\mathbf{x}_i, \mathbf{m}_c) = \|\phi(\mathbf{x}_i) - \mathbf{m}_c\|^2$ is rewritten as:

$$K_{ii} - \frac{2\sum_{\mathbf{x}_j \in \pi_c} \alpha_j K_{ij}}{\sum_{\mathbf{x}_j \in \pi_c} \alpha_j} + \frac{\sum_{\mathbf{x}_j, \mathbf{x}_l \in \pi_c} \alpha_j \alpha_l K_{jl}}{(\sum_{\mathbf{x}_j \in \pi_c} \alpha_j)^2}. \tag{1}$$

As a result, we may derive an algorithm analogous to k-means to monotonically decrease the weighted kernel k-means objective function without knowledge of the map ϕ. Algorithm 1 shows the pseudo-code for the basic weighted kernel k-means algorithm, which is identical to standard k-means except for the fact that distances are computed using the kernel matrix. Note that as opposed to regular k-means, this algorithm does not update the cluster centroids $\mathbf{m}_c$ explicitly.

Using the weighted form of the kernel k-means objective function is important because it is equivalent to a weighted graph-cut objective function, of which many existing graph clustering objectives are special cases (Dhillon et al. 2007). For example, the normalized cut criterion for graph clustering can be written exactly as the weighted kernel k-means objective function, where the kernel matrix K and weights α are derived from the input adjacency

ALGORITHM 1: Basic weighted kernel k-means.

KERNEL-KMEANS(K, k, t_{max}, $\boldsymbol{\alpha}$, $\{\pi_c^{(0)}\}_{c=1}^k$)
Input: K: kernel matrix, k: number of clusters, t_{max}: optional maximum number of iterations, $\boldsymbol{\alpha}$: weight vector, $\{\pi_c^{(0)}\}_{c=1}^k$: optional initial clusters
Output: $\{\pi_c\}_{c=1}^k$: final partitioning of the points
1. Initialize the k clusters from $\{\pi_c^{(0)}\}_{c=1}^k$, if provided as input, else randomly.
2. Set $t = 0$.
3. For each point $\mathbf{x}_i$ and every cluster c, compute $d(\mathbf{x}_i, \mathbf{m}_c)$ as in (1).
4. Find $c^*(\mathbf{x}_i) = \text{argmin}_c\, d(\mathbf{x}_i, \mathbf{m}_c)$, resolving ties arbitrarily.
5. Compute the updated clusters as

$$\pi_c^{(t+1)} = \{\mathbf{x}_i \ : \ c^*(\mathbf{x}_i) = c\}.$$

6. If not converged or $t_{max} > t$, set $t = t + 1$ and go to Step 3; Otherwise, stop and output final clusters $\{\pi_c^{(t+1)}\}_{c=1}^k$.

Table 2 Popular graph clustering objectives and corresponding weights and kernels given affinity matrix A

Objective	Node weights	Kernel
Ratio association	1 for all nodes	$K = \sigma I + A$
Ratio cut	1 for all nodes	$K = \sigma I - L$
Normalized cut	Degree of the node	$K = \sigma D^{-1} + D^{-1} A D^{-1}$

matrix A. Other objective functions for graph clustering can similarly be expressed as special cases of the weighted kernel k-means objective functions. This allows us the flexibility of having as input to the algorithm either a graph or data vectors that have been mapped to a kernel space.

Table 2 shows how to set the weights and kernel for three popular graph clustering objectives to be *equivalent* to the weighted kernel k-means objective function. The node weights in this table are weights on the nodes in the graph that correspond to the weights of the data vectors in k-means, signifying the importance of the data vectors (i.e., the degree of their contribution to the clustering objective function). Here, D is a diagonal matrix whose entries correspond to the sum of the rows of the input affinity matrix A (which is the input similarity matrix or kernel matrix), L is the Laplacian matrix ($D - A$), and σ is a positive real number chosen to be sufficiently large such that K is positive definite. Using simple eigenvalue bounds, one can appropriately compute σ without resorting to expensive eigenvector computation. We note that, in practice, one can typically set $\sigma = 0$ and still observe monotonic convergence despite the matrix K not being positive definite. See Dhillon et al. (2004b) for a more complete discussion on the choice and computation of σ.

While not equivalent, the kernels formed over these graphs bear resemblance to other graph kernels, such as those discussed by Smola and Kondor (2003); for example, the normalized cut criterion can be interpreted in terms of random walks. See Meila and Shi (2001) for a detailed analysis of this connection.

2.2 Semi-supervised clustering

Often when clustering, we have some background knowledge about the cluster structure. As mentioned previously, in this work we assume that this knowledge comes in the form

of pairwise must-link or cannot-link constraints. Such constraints are natural for graphs, as pairwise relationships are explicitly captured via edges in a graph. However, most semi-supervised clustering with pairwise constraints assumes that the input is in the form of data vectors (Wagstaff et al. 2001; Sinkkonen and Kaski 2002; Klein et al. 2002; Xing et al. 2003; Bar-Hillel et al. 2003; Basu et al. 2004b; Bilenko et al. 2004). In contrast, recent work by Bansal et al. (2002), Charikar et al. (2003), Demaine and Immorlica (2003) in semi-supervised clustering considers inputs only in the form of constraints, and does not consider an underlying distance function between the points.

2.2.1 HMRF model

We now briefly describe a recently proposed objective function for semi-supervised clustering of data vectors, which we will use in our formulation. Basu et al. (2004b) proposed a framework for semi-supervised clustering based on Hidden Markov Random Fields (HMRFs). Choosing squared Euclidean distance as the cluster distortion measure and the generalized Potts potential (Kleinberg and Tardos 1999) as the constraint violation potential, the semi-supervised clustering objective can be expressed as:

$$\mathcal{J}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2 + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ \text{s.t. } l_i \neq l_j}} w_{ij} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ \text{s.t. } l_i = l_j}} w_{ij}, \tag{2}$$

where $\mathcal{M}$ is the set of must-link constraints, $\mathcal{C}$ is the set of cannot-link constraints, w_{ij} is the penalty cost for violating a constraint between $\mathbf{x}_i$ and $\mathbf{x}_j$, and l_i refers to the cluster label of $\mathbf{x}_i$. The first term in this objective function is the standard k-means objective function, the second term is a penalty function for violating must-link constraints, and the third term is a penalty function for violating cannot-link constraints. The algorithm HMRF-KMEANS developed for minimizing this objective (Basu et al. 2004b) uses an iterative relocation approach like k-means. One of the drawbacks to HMRF-KMEANS is that, in the E-step of the algorithm, given the current cluster centroids, the order in which points are greedily reassigned to clusters determines the new clusters; that is, different orderings produce different clusterings. It is computationally intractable to solve for the optimal ordering. Several recent extensions of the HMRF-KMEANS model address this order-dependence issue and propose approximations that must be employed in the E-step, e.g., loopy belief propagation (Bilenko and Basu 2004; Law et al. 2005), Gibbs sampling (Lu and Leen 2005), and mean field approximation (Lange et al. 2005). As we will see in Sect. 3, our proposed algorithm naturally avoids such an order-dependence problem.

2.2.2 Spectral learning

Recently, spectral methods have become increasingly popular for clustering. These algorithms cluster data given in the form of a graph. One spectral approach to semi-supervised clustering is the SPECTRAL-LEARNING algorithm of Kamvar et al. (2003):

- Form the affinity matrix A: the entries of A are assumed to be normalized between 0 and 1, where entry A_{ij} is the similarity between points i and j.
- For all points i, j that have a must-link constraint, set $A_{ij} = 1$ (maximum affinity); for all points i, j that have a cannot-link constraint, set $A_{ij} = 0$ (minimum affinity).
- Re-normalize the matrix using additive normalization (Kamvar et al. 2003): $N = \frac{1}{d_{max}}(A + d_{max} I - D)$. N now represents a normalized Markov transition process.

- Take the top k eigenvectors of A to be the columns of the matrix V, and cluster the rows of V. If N has k piecewise constant eigenvectors, then N has k strong cliques—so clustering the eigenvectors effectively tries to find these k strongly connected components or clusters.

In the above algorithm, d_{max} is the maximum row-sum of A and D is the degree matrix. Notice that the additive normalization is equal to $I - \frac{1}{d_{max}} L$, which is equivalent to $d_{max} I - L$ up to a scalar factor, where $L = D - A$ is the graph Laplacian of A. Furthermore, the top eigenvectors of $d_{max} I - L$ are the same as the bottom eigenvectors of L. In the presentation of Kamvar et al. (2003), the SPECTRAL-LEARNING algorithm does not have an explicit underlying objective function. However in Sect. 4.3, we will show that this algorithm can be viewed as a special case of our unified semi-supervised clustering framework. Note that the SPECTRAL-LEARNING algorithm needs $O(kn)$ memory to store the k eigenvectors for n points, which can be a substantial overhead for large graphs.

Another recent paper (Yu and Shi 2004) considered a semi-supervised formulation of the normalized cut objective and had a spectral algorithm associated with it. The main differences between this work and our algorithm are: (1) only must-link constraints are considered in their formulation, and there is no way to specify penalty weights for constraint violations: SS-KERNEL-KMEANS considers both must-link and cannot-link constraints and allows constraint violation with an associated penalty, thereby making it more robust to constraint noise; (2) their formulation solves an expensive constrained eigen-decomposition problem while SS-KERNEL-KMEANS uses an efficient iterative algorithm, making it easily scalable to large data sets.

3 Kernel-based semi-supervised clustering for HMRF-KMEANS

This section demonstrates the connection between the semi-supervised clustering objective for the HMRF-KMEANS algorithm and the kernel k-means objective function. This leads to a kernel-based algorithm for optimizing the HMRF-KMEANS objective function with squared Euclidean distance and penalties weighted by the cluster size.

3.1 Constructing the kernel

Recall the objective for semi-supervised clustering on a HMRF from (2), using squared Euclidean distance as the clustering distortion measure and the generalized Potts potential for constraint penalty. Let us alter this penalty function: instead of adding a penalty term for a must-link violation if the two points are in different clusters, we give a *reward* for constraint satisfaction if the points are in the same cluster, by subtracting the corresponding penalty term from the objective. If the sum of weights for all must-link constraints is a constant, then this is equivalent to the original objective function up to an additive constant. Therefore minimizing $\mathcal{J}(\{\pi_c\}_{c=1}^k)$ is equivalent to minimizing:

$$\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2 - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ l_i = l_j}} w_{ij} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ l_i = l_j}} w_{ij}.$$

We also introduce the notion of cluster-size weighted penalties: if there are two points that have a cannot-link constraint in the same cluster, we will penalize higher if the corresponding cluster is smaller. Similarly, we will reward higher if two points in a small cluster have

a must-link constraint. Thus, we divide each w_{ij} by the size of the cluster that the points are in. This gives us:

$$\mathcal{J}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2 - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ l_i = l_j}} \frac{w_{ij}}{|\pi_{l_i}|} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ l_i = l_j}} \frac{w_{ij}}{|\pi_{l_i}|}. \tag{3}$$

Using the following simple result (see Duda and Hart 1973):

$$\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} 2\|\mathbf{x}_i - \mathbf{m}_c\|^2 = \sum_{c=1}^{k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \pi_c} \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{|\pi_c|},$$

and re-writing the sums, minimizing the objective in (3) becomes equivalent to minimizing:

$$\mathcal{J}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \pi_c} \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{|\pi_c|} - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ l_i = l_j}} \frac{2w_{ij}}{|\pi_{l_i}|} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ l_i = l_j}} \frac{2w_{ij}}{|\pi_{l_i}|}. \tag{4}$$

Rewriting the sums yields the following:

$$\mathcal{J}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i, \mathbf{x}_j \in \pi_c} \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{|\pi_c|} - \sum_{c=1}^{k} \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \pi_c \\ (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}}} \frac{2w_{ij}}{|\pi_c|}$$

$$+ \sum_{c=1}^{k} \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \pi_c \\ (\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}}} \frac{2w_{ij}}{|\pi_c|}.$$

Let E be the matrix of pairwise squared Euclidean distances among the data points, such that $E_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$ and W be the constraint matrix such that W_{ij} is $-w_{ij}$ for a cannot link, w_{ij} for a must link, and 0 otherwise. Furthermore, we introduce an indicator vector $\mathbf{z}_c$ for cluster c. This vector is of length n and $\mathbf{z}_c(i) = 0$ if $\mathbf{x}_i$ is not in cluster c, and 1 otherwise. We can now write the above objective in the following way:

$$\mathcal{J}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \frac{\mathbf{z}_c^T (E - 2W)\mathbf{z}_c}{\mathbf{z}_c^T \mathbf{z}_c}, \tag{5}$$

where $\mathbf{z}_c^T \mathbf{z}_c$ is the size of cluster π_c, and $\mathbf{z}_c^T (E - 2W)\mathbf{z}_c$ gives the sum of $E_{ij} - 2W_{ij}$ over all $\mathbf{x}_i$ and $\mathbf{x}_j$ in π_c.

Now define a matrix $\tilde{Z}$ such that $\tilde{Z}_{\cdot c}$, the column c of $\tilde{Z}$, is equal to $\mathbf{z}_c / (\mathbf{z}_c^T \mathbf{z}_c)^{1/2}$. $\tilde{Z}$ is an orthonormal matrix ($\tilde{Z}^T \tilde{Z} = I_k$), and the objective that we want to minimize can be re-written as:

$$\sum_{c=1}^{k} \tilde{Z}_{\cdot c}^T (E - 2W) \tilde{Z}_{\cdot c} = \text{trace}(\tilde{Z}^T (E - 2W) \tilde{Z}). \tag{6}$$

It has been shown that the kernel k-means objective function can also be expressed as a trace optimization problem (Dhillon et al. 2004a). In particular, the kernel k-means objective is

ALGORITHM 2: Semi-supervised kernel k-means for HMRF-KMEANS.

KERNEL-HMRF-KMEANS(S, k, W, t_{max})
Input: S: input similarity matrix, k: number of clusters, W: constraint penalty matrix, t_{max}: optional maximum number of iterations
Output: $\{\pi_c\}_{c=1}^k$: final partitioning of the points
1. Form the matrix $K = S + W$.
2. Diagonal-shift K by adding σI to guarantee positive definiteness.
3. Get initial clusters $\{\pi_c^{(0)}\}_{c=1}^k$ using constraints encoded in W and farthest-first initialization (see Algorithm 3).
4. Return $\{\pi_c\}_{c=1}^k$ = KERNEL-KMEANS(K, k, t_{max}, $\mathbf{1}$, $\{\pi_c^{(0)}\}_{c=1}^k$), where $\mathbf{1}$ is the vector of all ones.

ALGORITHM 3: Farthest-first initialization.

FARTHESTFIRSTINIT(A, k, W)
Input: A: input affinity matrix, k: number of clusters, W: constraint penalty matrix
Output: $\{\pi_c^{(0)}\}_{c=1}^k$: initial partitioning of the points
1. M = transitive closure of must-link constraints in W.
2. C_M = connected components in M.
3. CC = components from C_M disconnected across cannot-link boundaries.
4. Set $\pi_1^{(0)}$ = largest connected component from CC; set $i = 1$.
5. If $k > i$, go to Step 6. Else, go to Step 8.
6. Find component CC_j that is farthest from the current set of selected components $\{\pi_c^{(0)}\}_{c=1}^i$.
7. Set $\pi_{i+1}^{(0)} = CC_j$; set $i = i + 1$. Go to Step 5.
8. Return $\{\pi_c^{(0)}\}_{c=1}^k$.

written as a maximization of trace($Y^T KY$), with Y analogous to $\tilde{Z}$ (an orthonormal indicator matrix). The only difference between these objectives is that our semi-supervised objective is expressed as a trace minimization, while kernel k-means is expressed as a trace maximization. To make our problem into a maximization problem, we note that for squared Euclidean distance, $\|\mathbf{x}_i - \mathbf{x}_j\|^2 = \mathbf{x}_i^T\mathbf{x}_i + \mathbf{x}_j^T\mathbf{x}_j - 2\mathbf{x}_i^T\mathbf{x}_j$. Let S be the similarity matrix ($S_{ij} = \mathbf{x}_i^T\mathbf{x}_j$) and let $\tilde{S}$ be the matrix such that $\tilde{S}_{ij} = S_{ii} + S_{jj}$. Then, $E = \tilde{S} - 2S$.

By replacing E in the trace minimization, the problem is equivalent to the minimization of trace($\tilde{Z}^T(\tilde{S} - 2S - 2W)\tilde{Z}$). We calculate trace($\tilde{Z}^T\tilde{S}\tilde{Z}$) as $2 \cdot$ trace(S), which is a constant and can be ignored in the optimization. This leads to a maximization of trace($\tilde{Z}^T(S + W)\tilde{Z}$). If we define a matrix $K = S + W$, our problem is expressed as a maximization of trace($\tilde{Z}^T K\tilde{Z}$), and is mathematically equivalent to unweighted kernel k-means.

Note that this matrix K may not be positive semidefinite, a requirement for kernel k-means to converge. One can avoid this problem by performing diagonal shifting (Dhillon et al. 2007) for kernelizing K, as shown in Step 2 of Algorithm 2. This is done by adding σI to K. Note that

$$\begin{aligned}\text{trace}(\tilde{Z}^T(\sigma I + K)\tilde{Z}) &= \sigma \cdot \text{trace}(\tilde{Z}^T\tilde{Z}) + \text{trace}(\tilde{Z}^T K\tilde{Z}) \\ &= \sigma k + \text{trace}(\tilde{Z}^T K\tilde{Z}).\end{aligned}$$

A constant is added to the objective function, so the globally optimal solution is the same for both the shifted and unshifted matrices. Thus, shifting the diagonal allows one to construct a positive semi-definite matrix K (since adding σI to K increases the eigenvalues by σ), which guarantees monotonic convergence of the kernel k-means algorithm to a local optimum, while maintaining the globally optimal solution. As mentioned in Sect. 2.1, one can run kernel k-means directly on K—even though there is no theoretical guarantee of convergence in this case, the algorithm generally monotonically converges in practice.

3.2 The algorithm

Summarizing the result from the previous section, we have shown an equivalence between unweighted kernel k-means and HMRF-KMEANS with squared Euclidean distance and cluster-size weighted penalties. We can now discuss a kernel-based algorithm for this HMRF-KMEANS objective by constructing the appropriate kernel matrix K (see Algorithm 2). This algorithm simply constructs the kernel (as derived in the previous section), performs proper cluster initialization and runs kernel k-means.

A key advantage to this approach is that the algorithm assumes a similarity matrix as input. As given in the derivation, the similarity matrix is the matrix of vector inner products (i.e. the Gram matrix), but one can easily generalize this by applying a kernel function on the vector data to form the similarity matrix. Thus, unlike previous approaches to optimizing the HMRF-KMEANS objective function, our approach is easily generalized to handle non-linear cluster boundaries—for example, by applying a Gaussian kernel when forming the similarity matrix.

A step that we have not yet discussed is the initialization of the clusters prior to running kernel k-means. In standard k-means, many different initialization schemes are possible for generating starting clusters. However, in our case, the constraint information gives us important clues about the cluster structure, and this information may be incorporated into the initialization step of the algorithm. For cluster initialization, we follow the approach of Basu et al. (2004b). We first take the transitive closure of the must-link constraints, assuming that all constraints are consistent. This gives us a set of connected components of must-link constraints.

We then infer additional cannot-link constraints in the following way: if there exists a cannot-link constraint between two connected components, we add cannot-link constraints between *all* pairs of points a and b, where a belongs to the first connected component and b belongs to the second connected component.

After this step, we generate initial clusters by using a farthest-first algorithm. We compute the connected components and then choose k of these as the k initial clusters. The farthest first algorithm selects the largest connected component, then iteratively chooses the cluster farthest away from the currently chosen cluster, where the distance between clusters is measured by the total pairwise distance between the points in these clusters. Once these k initial clusters are chosen, we initialize all points by placing them in their closest cluster. Note that all the distance computations in this procedure can be performed efficiently in kernel space.

4 Semi-supervised graph clustering

As discussed earlier, there is a direct mathematical connection between the weighted kernel k-means objective function and a wide class of graph clustering objectives. So far we have

considered the vector case while deriving the connection between HMRF-KMEANS and kernel k-means. In this section, we generalize this result for the case of several semi-supervised graph clustering objective functions.

4.1 Objective functions

The HMRF-KMEANS objective function was motivated as a three-term objective function: one term for the clustering objective, one term for enforcing must-link constraints, and one term for enforcing cannot-link constraints. In the same vein, we now consider three semi-supervised graph clustering objectives, where the clustering objectives are graph-based instead of vector-based.

Semi-supervised normalized cut A semi-supervised version of the normalized graph cut objective seeks to minimize:

$$\sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{\text{degree}(\mathcal{V}_c)} - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ l_i = l_j}} \frac{w_{ij}}{\deg(\mathcal{V}_{l_i})} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ l_i = l_j}} \frac{w_{ij}}{\deg(\mathcal{V}_{l_i})}. \tag{7}$$

Notice that, instead of cluster-size weighted penalties, we use degree-weighted penalties.

Semi-supervised ratio cut Similarly, a semi-supervised ratio cut objective can be written as the minimization of:

$$\sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V} \setminus \mathcal{V}_c)}{|\mathcal{V}_c|} - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ l_i = l_j}} \frac{w_{ij}}{|\mathcal{V}_{l_i}|} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ l_i = l_j}} \frac{w_{ij}}{|\mathcal{V}_{l_i}|}. \tag{8}$$

Semi-supervised ratio association Finally, the semi-supervised ratio association objective can be written as the maximization of:

$$\sum_{c=1}^{k} \frac{\text{links}(\mathcal{V}_c, \mathcal{V}_c)}{|\mathcal{V}_c|} + \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{M} \\ l_i = l_j}} \frac{w_{ij}}{|\mathcal{V}_{l_i}|} - \sum_{\substack{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{C} \\ l_i = l_j}} \frac{w_{ij}}{|\mathcal{V}_{l_i}|}. \tag{9}$$

Note that the second and third terms in this objective function have opposite signs to the corresponding terms in the semi-supervised ratio cut and semi-supervised normalized cut objectives; this is because we aim to maximize the association.

4.2 A kernel-based algorithm for semi-supervised graph clustering

We now provide a general algorithm for semi-supervised graph clustering that can optimize each of the three objectives given above. To do that, we must show that each of these objectives can be written as a special case of the weighted kernel k-means objective function.

First, consider minimizing semi-supervised normalized cut. Based on the analysis of Yu and Shi (2003), the first term in the semi-supervised normalized cut objective function can be expressed as $k - \text{trace}(\tilde{Y}^T D^{-1/2} A D^{-1/2} \tilde{Y})$, where $\tilde{Y}$ is an orthonormal cluster indicator matrix, D is the diagonal matrix of node degrees, and A is the graph adjacency

Table 3 Semi-supervised graph clustering objectives and corresponding weights and kernels given affinity matrix A and constraint matrix W

Objective	Node weights	Kernel
SS ratio association	1 for all nodes	$K = \sigma I + A + W$
SS ratio cut	1 for all nodes	$K = \sigma I - L + W$
SS normalized cut	Degree of the node	$K = \sigma D^{-1} + D^{-1}(A + W)D^{-1}$

ALGORITHM 4: Semi-supervised graph clustering algorithm.

SS-KERNEL-KMEANS(A, obj, k, W, t_{max})
Input: A: input affinity matrix, obj: clustering objective, k: number of clusters, W: constraint penalty matrix, t_{max}: optional maximum number of iterations
Output: $\{\pi_c\}_{c=1}^{k}$: final partitioning of the points
1. Generate a vector $\boldsymbol{\alpha}$ of node weights based on obj using Table 3.
2. If obj is SS-ratio-cut, reset A as $A - D$.
3. Form $K = \sigma \cdot \text{diag}(\boldsymbol{\alpha})^{-1} + \text{diag}(\boldsymbol{\alpha})^{-1}(A + W)\,\text{diag}(\boldsymbol{\alpha})^{-1}$.
4. Get initial clusters $\{\pi_c^{(0)}\}_{c=1}^{k}$ using the constraints encoded in W and weighted farthest-first initialization (see Algorithm 3).
5. Return $\{\pi_c\}_{c=1}^{k} =$ KERNEL-KMEANS(K, k, t_{max}, $\boldsymbol{\alpha}$, $\{\pi_c^{(0)}\}_{c=1}^{k}$).

matrix. The second and third term of the objective function may be expressed concisely as $-\text{trace}(\tilde{Y}^T D^{-1/2} W D^{-1/2} \tilde{Y})$, using similar analysis as in the derivations of (5) and (6). Combining these two terms, we get that the semi-supervised normalized cut objective is equivalent to minimizing $k - \text{trace}(\tilde{Y}^T D^{-1/2}(A + W)D^{-1/2}\tilde{Y})$ or, equivalently, maximizing $\text{trace}(\tilde{Y}^T D^{-1/2}(A + W)D^{-1/2}\tilde{Y})$. A generalization of the analysis of Dhillon et al. (2007) allows us to express this trace maximization exactly as weighted kernel k-means, as follows. We set $A' = A + W$ as in the unweighted case, and run weighted kernel k-means on $\sigma D^{-1} + D^{-1}A'D^{-1}$ with weights equal to the degree of the node. Running weighted kernel k-means with this kernel and weights will then monotonically decrease the objective in (7) for a graph with adjacency matrix A.

Analogous results may be obtained for semi-supervised ratio cut and semi-supervised ratio association, using similar analysis. For semi-supervised ratio cut, we run weighted kernel k-means on $\sigma I - L + W$ (L is the Laplacian of A) with all weights equal to one. In ratio association, we run weighted kernel k-means on $\sigma I + A + W$ with all weights equal to one. In all cases, the addition of supervision to weighted kernel k-means simply requires adding a constraint matrix W appropriately to the kernel matrix.

Table 3 summarizes the kernel and weight construction for each of these semi-supervised clustering objective functions to be equivalent to the weighted kernel k-means objective, and Algorithm 4 describes the general kernel-based semi-supervised clustering algorithm. Note that further generalizations are possible; one could have a general, weighted semi-supervised graph clustering objective that is normalized by the sum of arbitrary weights instead of the degrees of a cluster or the cluster sizes. In this case $\boldsymbol{\alpha}$ would be an arbitrary weight vector.

Also note that the case of *semi-supervised ratio association is identical to the earlier algorithm for* HMRF-KMEANS*, with a graph adjacency matrix used as the input similarity matrix.*

4.3 Connections to spectral learning

We can view SPECTRAL-LEARNING (Kamvar et al. 2003), described in Sect. 2.2.2, in our framework as follows: for a must-link constraint, if A_{ij} is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, we set $W_{ij} = 1 - A_{ij}$ (and hence the corresponding value in $A + W$ is 1). Similarly, for cannot-links, we set $W_{ij} = -A_{ij}$ (and hence the corresponding value in $A + W$ is 0). With this particular choice of constraint weights, the matrix $A + W$ is identical to the matrix from SPECTRAL-LEARNING before additive normalization. This leaves just the additive normalization step, which is the same normalization required for semi-supervised ratio cut (up to a diagonal shift, which does not change the globally optimal solution).

A well-known relaxation to our trace maximization problem, trace$(Y^T KY)$, is achieved by taking the top k eigenvectors of K. The matrix formed by these eigenvectors is the optimal Y matrix, under the relaxation that Y is an arbitrary orthonormal matrix. Such a relaxation leads to the SPECTRAL-LEARNING algorithm that was detailed in Sect. 2.2.2.

Thus, the SPECTRAL-LEARNING algorithm may be viewed as solving a relaxed semi-supervised ratio cut objective. Moreover, our earlier analysis shows that the fast iterative kernel k-means algorithm can be used to optimize this objective, thus removing any requirement for the expensive operation of eigen-decomposition (for large data sets). Alternatively, this analysis suggests other spectral algorithms based on semi-supervised normalized cut or semi-supervised ratio association.

One could analogously hope to employ kernel k-means to optimize the semi-supervised formulation of Yu and Shi (2004). However, because of the additional constraint introduced for must-link constraints into the formulation, it is not possible to apply kernel k-means directly. However, recent results have shown how to express kernel k-means as a continuous optimization problem (Kulis et al. 2007). The extra must-link constraints may be added naturally in this setting, and the algorithm of Kulis et al. (2007) may then be applied to optimize the semi-supervised clustering objective introduced by Yu and Shi. As a result, no eigenvector decomposition would be required. We leave this as a potential area of future work.

5 Experimental results

In this section, we present experimental results on both vector data and graph data. First, we compare SS-KERNEL-KMEANS to HMRF-KMEANS on vector data, which optimizes an HMRF-based objective function using squared Euclidean distance as the distortion measure. We present results on a synthetic data set, handwritten character recognition data sets, and a protein data set, demonstrating that SS-KERNEL-KMEANS has better performance on these data sets with a proper choice of the kernel. We then present results of SS-KERNEL-KMEANS on graph-based data—a gene network and an image data set, both of which cannot be clustered with HMRF-KMEANS. In this case, we compare results to SPECTRAL-LEARNING.

5.1 Data sets

We performed experiments on the following vector-based datasets:

1. `TwoCircles`: A synthetic data set comprising of 200 data points in 2 dimensions—100 points in an inner circle are labeled to be in one class, and 100 data points in a surrounding outer circle are labeled to be in another class, as shown in Fig. 1.

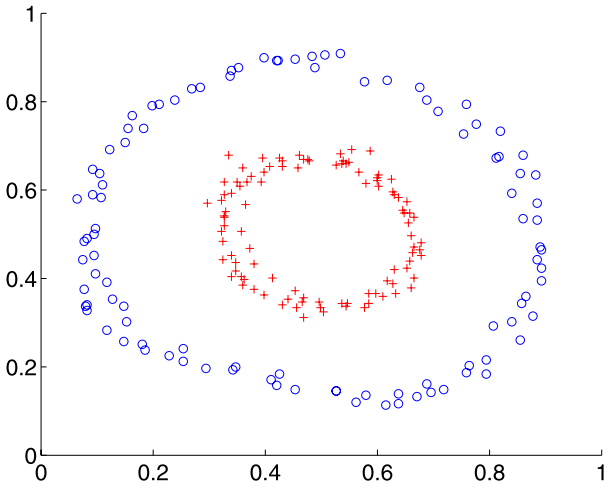

Fig. 1 `TwoCircles` data set with correct clustering

2. `Digits`: A subset of 10% of the data points chosen randomly from three classes {3, 8, 9} of the *Pendigits* handwritten digit recognition data set from the UCI Machine Learning Repository[2]—these three classes were chosen since distinguishing between sample handwritten digits from these classes visually is a difficult task. Each image is a black-and-white rectangular pixel display of a handwritten digit (either 3, 8 or 9). Using features like statistical moments and edge counts, each image was transformed to a 16-dimensional vector. This dataset has 317 points, each in a 16 dimensional-space. Note that this data set has been used for testing other semi-supervised clustering algorithms, such as Bilenko et al. (2004), and that a subset of all the digits are used to make the clustering problem more challenging.
3. `Letters`: A subset of 10% of the data points chosen randomly from three classes $\{I, J, L\}$ of the *Letters* handwritten character recognition data set from the UCI Machine Learning Repository. This particular subset of 227 points in a 16 dimensional-space was chosen since it is a difficult visual problem to discriminate between these 3 types of handwritten characters.
4. `Protein`: This dataset contains 20-dimensional feature vectors representing 116 proteins from 6 functional classes (Xing et al. 2003).

We also performed experiments on the following graph-based datasets:

1. `GeneNetwork`: An interaction network between 216 yeast genes, where each gene is labeled with one of 3 KEGG (Ogata et al. 1999) functional pathway labels. This data is a subgraph of a high-quality probabilistic functional network of yeast genes (Lee et al. 2004): each edge weight in this network represents a probability of linkage between two genes, estimated by integrating diverse functional genomic data sources. The genes do not have an explicit vector representation in this data set.
2. `Caltech Image Data`: We experimented with the Caltech-4 data set which contains 1155 automobile, 800 airplane, 435 face, and 798 motorcycle images (http://www.robots.ox.ac.uk/~vgg/data3.html). The pyramid match kernel (Grauman and Darrell 2005), a kernel function between images, was used to generate a kernel matrix on a subset

[2]http://www.ics.uci.edu/~mlearn/MLRepository.html.

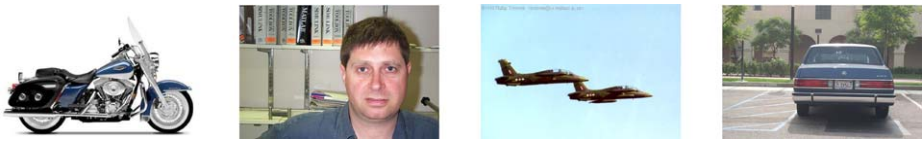

Fig. 2 Example images from the Caltech-4 data set

of 300 images in this data set. This kernel function takes as input sets of image features from two images (the sets may have different cardinality), and does an appropriate partial matching between these sets. As with the gene network, there is no explicit vector representation of this data set—the kernel function defines a matrix of kernel function evaluations, which is then viewed as a dense graph. Figure 2 contains examples from this data set.

Two different kernel matrices were constructed from this data set, generated using different methods. The first matrix, `Caltech1`, was generated using vocabulary-guided bins with SIFT descriptors from Harris and MSER detectors. The second, `Caltech2`, used uniform grid cells and SIFT descriptors extracted from a uniformly spaced grid on the image (Grauman and Darrell 2005).

5.2 Methodology

Figures 3–9 show learning curves with 2-fold cross validation. These plots show the improvement in clustering quality on the test set with increasing amount of pairwise constraints provided as supervision from the training set. These constraints are obtained by randomly selecting pairs of points from the training set, and creating must-link or cannot-link constraints depending on whether the underlying classes of the two points are the same or different. Each point on the learning curve is an average of results over 20 runs. We generate constraints only using the training set, cluster the whole data without the labels, and evaluate the cluster quality only on the test set (Basu et al. 2004b). We set each penalty weight w_{ij} to be equal to $n/(kC)$, where n is the number of data points, k is the number of clusters, and C is the total number of constraints – this was done to make the constraint penalty term approximately of the same scale as the distance penalty term between data points and the cluster centroids.

To evaluate clusters, we use Normalized Mutual Information (NMI): it is a standard technique for determining the quality of clusters, by measuring the amount of statistical information shared by the random variables representing the cluster distribution and the underlying class distribution of the data points (Strehl et al. 2000). If C is the random variable denoting the cluster assignments of the points, and K is the random variable denoting the underlying class labels on the points then the NMI measure is defined as:

$$NMI = \frac{I(C;K)}{(H(C)+H(K))/2}, \tag{10}$$

where $I(X;Y) = H(X) - H(X|Y)$ is the mutual information between the random variables X and Y, $H(X)$ is the Shannon entropy of X, and $H(X|Y)$ is the conditional entropy of X given Y (Cover and Thomas 1991). The normalization by the average entropy of C and K makes the value of NMI stay between 0 and 1.

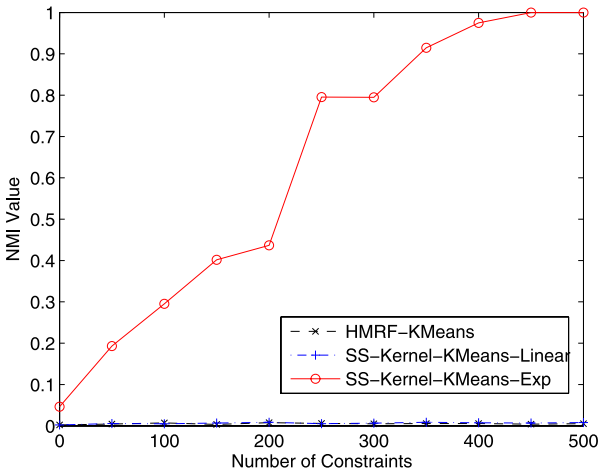

Fig. 3 Results on `TwoCircles` data set

5.3 Results and discussion

Figure 3 shows the results on the `TwoCircle` data set. This synthetic dataset is used to demonstrate the effectiveness of SS-KERNEL-KMEANS with the exponential kernel: this data set is not linearly separable in the original input space, but an exponential kernel $K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/2\rho^2)$ is able to linearly separate the 2 classes in the mapped space. Parameters such as ρ are typically chosen via cross-validation (for this simple example, an arbitrary value was chosen). SS-KERNEL-KMEANS with the exponential kernel (SS-Kernel-KMeans-Exp) gives improved results with increasing number of pairwise constraints, until it reaches a NMI score of 1.0 on the test set. On the other hand, as expected, HMRF-KMEANS and SS-KERNEL-KMEANS with a linear kernel (SS-Kernel-KMeans-Linear) are not able to get any improvement in performance on this dataset (NMI close to 0), since both algorithms split the data linearly into two clusters and are completely unable to re-construct the non-linear class structure, even with as many as 200 constraints. For this figure and the other figures, the NMI score with 0 constraints corresponds to the performance of the unsupervised clustering algorithm.

Figure 4 shows results on the `Digits` dataset. SS-KERNEL-KMEANS gives the best performance when used with an exponential kernel, outperforming both HMRF-KMEANS with squared Euclidean distance and SS-KERNEL-KMEANS with a linear kernel. Note that the SS-KERNEL-KMEANS learning curves display a characteristic "dip", where clustering accuracy decreases as a few initial constraints are provided, but after a certain point starts to increase and eventually rises above the initial point on the learning curve. We conjecture that this phenomenon is due to the fact that the kernel parameters learned using too few constraints are unreliable, and with higher number of constraints the parameter estimation becomes more accurate.

Since the `GeneNetwork` dataset is not vector-based, it cannot be clustered using HMRF-KMEANS. Three graph clustering objectives were used while clustering this dataset using SS-KERNEL-KMEANS — normalized cut (SS-Kernel-KMeans-NormCut), ratio cut (SS-Kernel-KMeans-RatioCut) and ratio association (SS-Kernel-KMeans-RatioAssoc). The fourth plot in the figure (Spectral-Learning) corresponds to the SPECTRAL-LEARNING algorithm (Kamvar et al. 2003). As shown in Fig. 5, all these algorithms have an improvement in performance with increasing supervision, but SS-KERNEL-KMEANS with normalized cut has the best performance for this dataset.

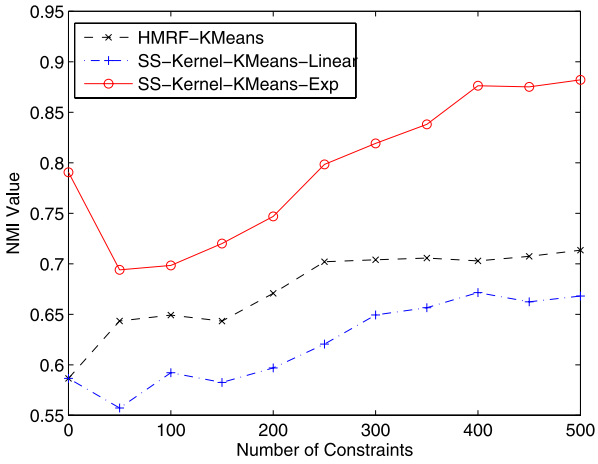

Fig. 4 Results on `Digits` data set

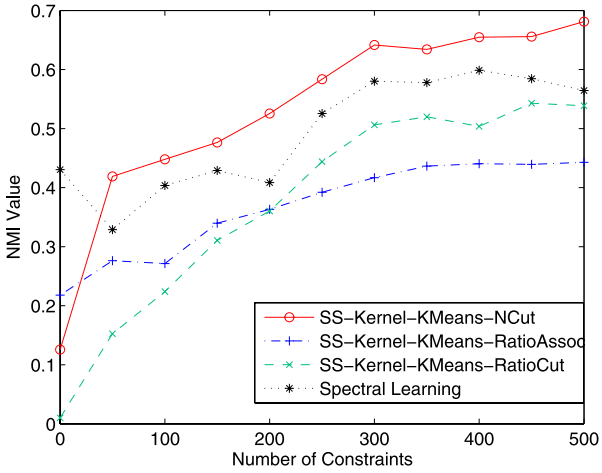

Fig. 5 Results on `GeneNetwork` data set

The benefit of using SS-KERNEL-KMEANS is clearly demonstrated in this experiment. A closer look showed that the `GeneNetwork` data set has 3 clusters of different sizes and different edge densities, which explains why performing degree normalization in the normalized cut objective gives good results. While SPECTRAL-LEARNING can only use the ratio cut objective, SS-KERNEL-KMEANS can work with other graph clustering objectives like normalized cut, therefore making it useful in domains where graph clustering objectives other than ratio cut are more effective.

Figures 6 and 7 show the results on the `Letters` and `Protein` data respectively. As expected, the NMI improves with increasing number of constraints, demonstrating that semi-supervised kernel KMeans performs better than the unsupervised version for these datasets too. The ratio association using an exponential kernel (SS-Kernel-Ratio-Association) outperforms normalized cut and ratio cut on these two datasets. For a detailed analysis of why different graph clustering objectives perform well on different datasets, please see Dhillon et al. (2007). Note that on these datasets, HMRF-KMEANS outperforms

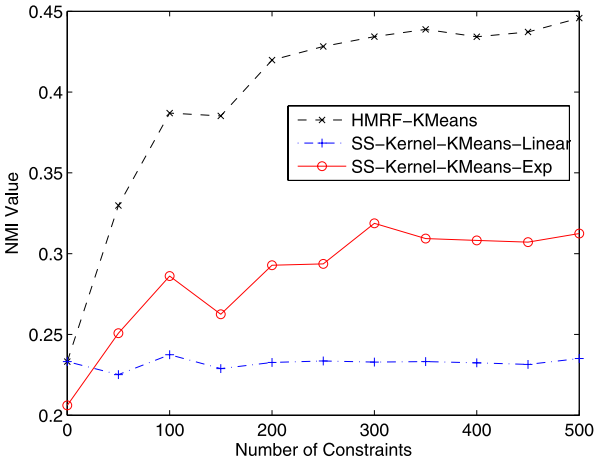

Fig. 6 Results on `Letters` data set

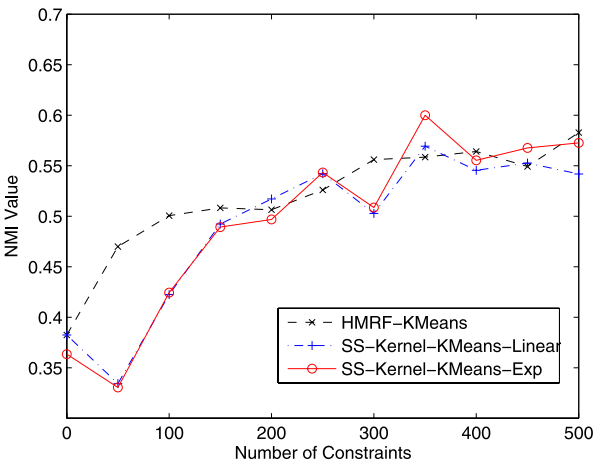

Fig. 7 Results on `Protein` data set

or matches the performance of our methods, implying that it is necessary to try different kernels for these particular datasets.

Figures 8 and 9 show the results on the `Caltech1` and `Caltech2` kernel matrices. Again, we see an increase in the NMI as the number of constraints increases. Moreover, we see that semi-supervised ratio cut performs much worse than the other methods; this may be due to the positive-definite shifting issues inherent in using kernel k-means for ratio cut.

For comparison, we also ran SPECTRAL-LEARNING on the Caltech image data sets. Surprisingly, we found that Spectral Learning demonstrates extremely poor clustering generalization accuracy on this data. The NMI of the clustering results on all the data points improves with increasing supervision, but the NMI on the test data points (which is what is shown in the plots) actually *decreases* as the number of constraints increases. For example, on `Caltech1`, the NMI goes from 0.11 with 50 constraints to 0.01 with 500 constraints. Similar performance is achieved on the `Caltech2` data. Thus, spectral learning appears to be overfitting on this data. We hypothesize that the penalty weights inherent in spectral learning lead to perturbation in the kernel matrix, which ultimately degrades generalization performance. In our kernel-based algorithms, the penalties decrease as the number of con-

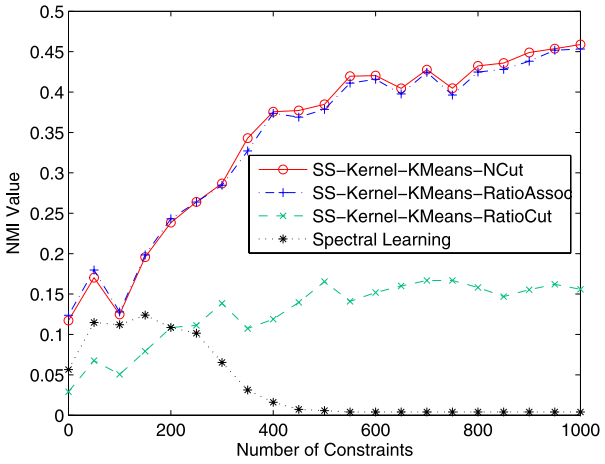

Fig. 8 Results on `Caltech1` data set

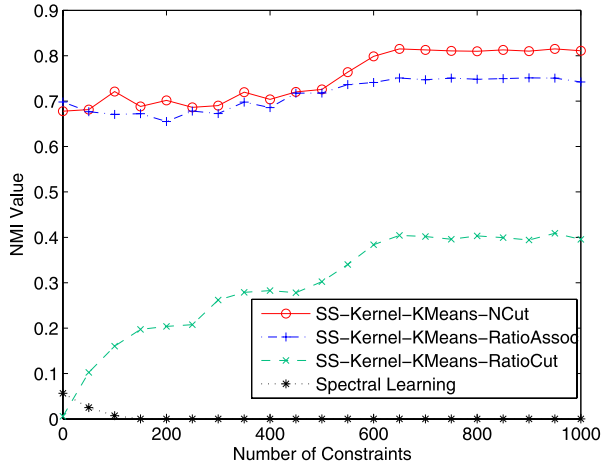

Fig. 9 Results on `Caltech2` data set

straints increase, which empirically appears to be an effective strategy in the presence of many constraints.

Regarding run-time complexity, the different variants of the semi-supervised clustering algorithms took substantially less time than the SPECTRAL-LEARNING algorithm. This is because SPECTRAL-LEARNING requires expensive eigenvalue calculations, while the SS-KERNEL-KMEANS algorithms are pseudo-linear in the size of the input (i.e., linear in the size of the graph $\times$ the number of iterations until convergence). In particular, on the `GeneNetwork` data set, the SS-KERNEL-KMEANS algorithm required 0.307 seconds on average to converge, while SPECTRAL-LEARNING cost 4.787 seconds on average (15.6 times longer). Given that this is a small data set, even more substantial gains will occur for larger data sets. For example, in Dhillon et al. (2007), it was shown that an implementation of kernel k-means is hundreds to thousands of times faster than spectral methods for unsupervised large-scale graph clustering.

6 Conclusions and future work

In this paper, a new algorithm SS-KERNEL-KMEANS was developed to optimize a semi-supervised clustering objective that can cluster both vector-based and graph-based data. The analysis for this algorithm used kernel methods: by constructing an appropriate kernel, we were able to prove an equivalence between a special case of the HMRF-based semi-supervised clustering objective and the kernel k-means objective function. The resulting algorithm provided a number of advantages over previously studied methods for semi-supervised clustering: (1) our analysis allows us to (locally) optimize the semi-supervised objective for both vector-based and graph-based inputs; (2) for vector-based inputs, the inputs can be mapped to a higher-dimensional kernel space to get non-linear cluster boundaries; (3) we can easily generalize the result to handle a number of semi-supervised graph clustering objectives; (4) the analysis allows us to link prior work on spectral learning to our semi-supervised clustering framework.

There are a number of interesting potential avenues for future research in kernel methods for semi-supervised clustering. Along with learning the kernel matrix before the clustering, one could additionally incorporate kernel matrix learning into the clustering iteration, as was done in Basu et al. (2004b). One way to incorporate learning into the clustering step is to devise a way to learn the weights in weighted kernel k-means, by using the constraints. Another possibility would be to explore the generalization of techniques in this paper beyond squared Euclidean distance, for unifying semi-supervised graph clustering with kernel-based clustering on an HMRF using other popular clustering distortion measures, e.g., KL-divergence, or cosine distance (Basu et al. 2004b). We would also like to extend the work of Basu et al. (2004a) to explore techniques of active learning and model selection in the context of semi-supervised kernel clustering. To speed up the algorithm further, the multi-level methods of Dhillon et al. (2007) can be employed.

Acknowledgements This research was supported by NSF grant CCF-0431257, NSF-ITR award IIS-0325116, NSF Career Award ACI-0093404, and a Google Research Grant. We thank Kristen Grauman for the pyramid match data used for image clustering. We would also like to thank the anonymous reviewers for their detailed and very useful feedback.

References

Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering. In *Proceedings of the 43rd IEEE symposium on foundations of computer science (FOCS-02)* (pp. 238–247).

Bar-Hillel, A., Hertz, T., Shental, N., & Weinshall, D. (2003). Learning distance functions using equivalence relations. In *Proceedings 20th international conference on machine learning* (pp. 11–18).

Basu, S., Banerjee, A., & Mooney, R. J. (2002). Semi-supervised clustering by seeding. In *Proceedings of 19th international conference on machine learning (ICML-2002)* (pp. 19–26).

Basu, S., Banerjee, A., & Mooney, R. J. (2004a). Active semi-supervision for pairwise constrained clustering. In *Proceedings 4th SIAM international conference on data mining*.

Basu, S., Bilenko, M., & Mooney, R. J. (2004b). A probabilistic framework for semi-supervised clustering In *Proceedings of 10th ACM SIGKDD international conference on knowledge discovery and data mining (KDD-2004)* (pp. 59–68).

Bie, T. D., Momma, M., & Cristianini, N. (2003). Efficiently learning the metric using side-information. In *Lecture notes in artificial intelligence: Vol. 2842. Proceedings of the 14th international conference on algorithmic learning theory (ALT2003)* (pp. 175–189). Berlin: Springer.

Bilenko, M., & Basu, S. (2004). A comparison of inference techniques for semi-supervised clustering with hidden Markov random fields. In *Proceedings of the ICML-2004 workshop on statistical relational learning and its connections to other fields (SRL-2004)*, Banff, Canada.

Bilenko, M., Basu, S., & Mooney, R. (2004). Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st international conference on machine learning*.

Chan, P., Schlag, M., & Zien, J. (1994). Spectral k-way ratio cut partitioning. *IEEE Transactions CAD-Integrated Circuits and Systems*, *13*, 1088–1096.

Chang, H., & Yeung, D. (2004). Locally linear metric adaptation for semi-supervised clustering. In *Proceedings of the twenty-first international conference on machine learning (ICML)* (pp. 153–160).

Chapelle, O., Schölkopf, B., & Zien, A. (Eds.) (2006). *Semi-supervised learning*. Cambridge: MIT Press.

Charikar, M., Guruswami, V., & Wirth, A. (2003). Clustering with qualitative information. In *Proceedings of the 44th annual IEEE symposium on foundations of computer science* (pp. 524–533).

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley-Interscience.

Cristianini, N., & Shawe-Taylor, J. (2000). *Introduction to support vector machines*. Cambridge: Cambridge University Press.

Davidson, I., & Ravi, S. S. (2005a). Clustering with constraints: feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*.

Davidson, I., & Ravi, S. S. (2005b). Hierarchical clustering with constraints: theory and practice. In *Proceedings of the ninth European principles and practice of KDD (PKDD)* (pp. 59–70).

Demaine, E. D., & Immorlica, N. (2003). Correlation clustering with partial information. In *Proceedings of the 6th international workshop on approximation algorithms for combinatorial optimization problems and 7th international workshop on randomization and approximation techniques in computer science (RANDOM-APPROX)*.

Demiriz, A., Bennett, K. P., & Embrechts, M. J. (1999). Semi-supervised clustering using genetic algorithms. In *Artificial neural networks in engineering (ANNIE-99)* (pp. 809–814).

Dhillon, I., Guan, Y., & Kulis, B. (2004a). Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of the 10th international conference on knowledge discovery and data mining* (pp. 551–556).

Dhillon, I., Guan, Y., & Kulis, B. (2004b). *A unified view of kernel k-means, spectral clustering and graph cuts* (Tech. rep. TR-04-25). University of Texas at Austin.

Dhillon, I., Guan, Y., & Kulis, B. (2007). Weighted graph cuts without eigenvectors: a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(11), 1944–1957.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Grauman, K., & Darrell, T. (2005). The pyramid match kernel: discriminative classification with sets of image features. In *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 1458–1465).

Kamvar, S. D., Klein, D., & Manning, C. (2003). Spectral learning. In *Proceedings of the 17th international joint conference on artificial intelligence* (pp. 561–566).

Klein, D., Kamvar, D., & Manning, C. (2002). From instance-level constraints to space-level constraints: making the most of prior knowledge in data clustering. In *Proceedings of the 19th international conference on machine learning* (pp. 307–314).

Kleinberg, J., & Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: metric labeling and Markov random fields. In: *Proceedings of the 40th IEEE symposium on foundations of computer science (FOCS-99)*, (pp. 14–23).

Kulis, B., Basu, S., Dhillon, I., & Mooney, R. (2005). Semi-supervised graph clustering: a kernel approach. In *Proceedings of the 22nd international conference on machine learning* (pp. 457–464).

Kulis, B., Surendran, A., & Platt, J. (2007). Fast low-rank semidefinite programming for embedding and clustering. In *Proceedings 11th international conference on AI and statistics (AISTATS)*.

Lange, T., Law, M. H. C., Jain, A. K., & Buhmann, J. M. (2005). Learning with constrained and unlabelled data. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Law, M. H. C., Topchy, A., & Jain, A. K. (2005). Model-based clustering with probabilistic constraints. In *Proceedings of the 2005 SIAM international conference on data mining* (pp. 641–645).

Lee, I., Date, S. V., Adai, A. T., & Marcotte, E. M. (2004). A probabilistic functional network of yeast genes. *Science*, *306*(5701), 1555–1558.

Lu, Z., & Leen, T. (2005). Semi-supervised learning with penalized probabilistic clustering. In *Advances in neural information processing systems*.

Meila, M., & Shi, J. (2001). A random walks view of spectral segmentation. In *Proceedings of the 8th international workshop on artificial intelligence and statistics (AISTATS)*.

Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., & Kanehisa, M. (1999). KEGG: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Research*, *27*, 29–34.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Sinkkonen, J., & Kaski, S. (2002). Clustering based on conditional distributions in an auxiliary space. *Neural Computation*, *14*(1), 217–239.

Smola, A. J., & Kondor, R. (2003). Kernels and regularization on computational graphs. In *Proceedings conference on computational learning theory (COLT)* (pp. 144–158).

Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on web-page clustering. In *Workshop on artificial intelligence for web search (AAAI)*.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the 18th international conference on machine learning* (pp. 577–584).

Xing, E. P., Ng, A. Y., Jordan, M. I., & Russell, S. (2003). Distance metric learning, with application to clustering with side-information. In *Advances in neural information processing systems 15*.

Yan, R., Zhang, J., Yang, J., & Hauptmann, A. G. (2004). A discriminative learning framework with pairwise constraints for video object classification. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition (CVPR)* (Vol. 2, pp. 284–291).

Yu, S., & Shi, J. (2003). Multiclass spectral clustering. In *International conference on computer vision* (pp. 313–319).

Yu, S., & Shi, J. (2004). Segmentation given partial grouping constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(2), 173–183.