Knowl Inf Syst (2011) 28:99–116
DOI 10.1007/s10115-010-0318-8

REGULAR PAPER

Composite kernels for semi-supervised clustering

Carlotta Domeniconi · Jing Peng · Bojun Yan

Received: 9 October 2008 / Revised: 14 April 2010 / Accepted: 12 June 2010 /
Published online: 26 June 2010

Abstract A critical problem related to kernel-based methods is how to select *optimal* kernels. A kernel function must conform to the learning target in order to obtain meaningful results. While solutions to the problem of estimating optimal kernel functions and corresponding parameters have been proposed in a supervised setting, it remains a challenge when no labeled data are available, and all we have is a set of pairwise *must-link* and *cannot-link* constraints. In this paper, we address the problem of optimizing the kernel function using pairwise constraints for semi-supervised clustering. We propose a new optimization criterion for automatically estimating the optimal parameters of composite Gaussian kernels, directly from the data and given constraints. We combine our proposal with a semi-supervised kernel-based algorithm to demonstrate experimentally the effectiveness of our approach. The results show that our method is very effective for kernel-based semi-supervised clustering.

Keywords Clustering · Semi-supervised clustering · Kernel methods

1 Introduction

Kernel-based methods enhance the modeling capability of learning algorithms by mapping data from the input space to a new feature space, usually of higher dimensionality. The key feature associated with kernel-based methods is the avoidance of an explicit knowledge of the mapping function. This is achieved by computing dot products in feature space via a kernel function. Kernel methods have been successfully used to solve classification, clustering, and regression problems for a variety of applications.

Representative work on supervised kernel learning includes kernel alignment [10], semi-definite programming [17], multiple kernel learning [2], and hyperkernels [15,20]. Additional

C. Domeniconi (✉) · B. Yan
Department of Computer Science, George Mason University, Fairfax, VA 22030, USA
e-mail: carlotta@cs.gmu.edu

J. Peng
Department of Computer Science, Montclair State University, Montclair, NJ, USA

work on tuning parameters for SVMs and kernel Fisher discriminant rules can be found in Chapelle and Vapnik [7], Wang et al. [28], and in Huang et al. [14]. Recently, a kernel method for semi-supervised clustering has been introduced [16]. In semi-supervised clustering, limited supervision is provided as input. Compared to traditional clustering algorithms, semi-supervised clustering employs both labeled and unlabeled data to obtain a partitioning that conforms more closely to user's preferences. Several recent papers have discussed this problem [1,3,4,8,9,16,19,27,29,31,32]. Typically, supervision is in the form of labeled data or pairwise constraints. A constraint on a pair of points specifies whether the two points belong to the same cluster (must-link), or not (cannot-link). Clearly, pairwise constraints can be induced by labeled data. In many applications, such as information retrieval, a qualitative measure of similarity between pairs of objects can be made available by the user.

The techniques introduced by Kulis et al. [16] and Yan and Domeniconi [30] extend semi-supervised clustering to a kernel induced space, thereby enabling the discovery of clusters with non-linear boundaries in the input space. In Yan and Domeniconi [30], we derive an optimization criterion to automatically estimate the optimal parameter of a (single) Gaussian kernel, directly from the data and the given constraints. The approach integrates the constraints into the clustering objective function, and optimizes the parameter of a Gaussian kernel iteratively during the clustering process. Another recent technique for learning kernels is based on an information-theoretic metric learning approach [11]. A Mahalanobis distance is learned by leveraging the correspondence between the multivariate Gaussian distribution and the set of Mahalanobis distances. The kernel function learned is a linear transformation of the original feature space. The formulation of the learning algorithm can accommodate a variety of constraints (including must-link and cannot-link), and can be kernelized.

A critical issue related to kernel-based methods is the "optimal" kernel selection. The performance of a kernel-based method depends critically on the selection of the kernel function, and the corresponding parameter values. The kernel function must conform to the learning target in order to obtain meaningful results. While solutions to the problem of estimating the optimal kernel function and its parameters have been proposed in a supervised setting, the problem presents major challenges when no labeled data are available, and all we have is a set of pairwise constraints. This is because the setting of kernel's parameters is often left to manual tuning, and the chosen values can significantly affect the quality of the clustering results [16].

In this paper, we propose a technique for kernel optimization using pairwise constraints to solve clustering problems. Our objective is to learn a kernel function that maps pairs of points subject to must-link constraints close to each other in feature space, while mapping points subject to cannot-link constraints far apart. We propose a new optimization criterion for automatically estimating the optimal parameters of composite Gaussian kernels, directly from the data and given constraints. This paper differs from the method introduced in Yan and Domeniconi [30] in many aspects: the objective function and optimization procedure introduced in this paper are fundamentally different from those used in Yan and Domeniconi [30]. Furthermore, here we estimate optimal parameters for composite Gaussian kernels, while in Yan and Domeniconi [30] a single Gaussian kernel is considered.

Combining multiple kernels is particularly useful for heterogeneous data fusion, as demonstrated in Lanckriet [18] with other kernel-based methods. For example, in genomics, data are available in a variety of formats: mRNA expression levels are vectors or time series; protein sequences are strings from an alphabet of 20 symbols; protein to protein interactions are typically represented as (weighted) graphs; and so on. Similarly, in video retrieval, a shot is represented as an image, and may be accompanied by subtitles (text) and audio information.

In both scenarios, the ability to combine complementary pieces of information can greatly boost the performance of the knowledge discovery process.

The rest of the paper is organized as follows. Section 2 provides the necessary background on kernel-based clustering and semi-supervised clustering. Section 3 discusses the details of our algorithm. Section 4 describes our experimental settings and results, and finally we provide conclusions and future research directions in Sect. 5.

2 Background

2.1 Kernel KMeans

Let $X = \{\mathbf{x}_i\}_{i=1}^{N} \subseteq \Re^D$ be a set of N samples with D dimensions. Let $\phi : \Re^D \rightarrow \Re^{D'}$ be a non-linear mapping function that maps data from the D dimensional input space to a D' dimensional feature space, with $D' > D$. The kernel KMeans algorithm generates a k-partitioning $\{\pi_c\}_{c=1}^{k}$ of X (where π_c represents the cth cluster) so that the objective function

$$\sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\phi(\mathbf{x}_i) - \mathbf{m}_c^{\phi}\|$$

is minimized. Here $\mathbf{m}_c^{\phi} = \frac{1}{|\pi_c|} \sum_{\mathbf{x}_i \in \pi_c} \phi(\mathbf{x}_i)$ is the centroid of cluster π_c in feature space. The key issue associated with Kernel-KMeans is the computation of distances in feature space. The distance of a point $\mathbf{x}_i$ from $\mathbf{m}_c^{\phi}$ in feature space can be expressed as:

$$\|\phi(\mathbf{x}_i) - \mathbf{m}_c^{\phi}\| = A_{ii} + B_{cc} - D_{ic}$$

where

$$A_{ii} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i)$$
$$D_{ic} = \frac{2}{|\pi_c|} \sum_{\mathbf{x}_j \in \pi_c} \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

and

$$B_{cc} = \frac{1}{|\pi_c|^2} \sum_{\mathbf{x}_j, \mathbf{x}_{j'} \in \pi_c} \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}_{j'})$$

Using the kernel trick, we can represent the dot product of points in kernel space using an appropriate Mercer kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ [26]. Since all computations involving data points are in the form of dot products, the components for distance computation can be re-written using the kernel trick:

$$A_{ii} = K(\mathbf{x}_i, \mathbf{x}_i)$$
$$D_{ic} = \frac{2}{|\pi_c|} \sum_{\mathbf{x}_j \in \pi_c} K(\mathbf{x}_i, \mathbf{x}_j)$$

and

$$B_{cc} = \frac{1}{|\pi_c|^2} \sum_{\mathbf{x}_j, \mathbf{x}_{j'} \in \pi_c} K(\mathbf{x}_j, \mathbf{x}_{j'})$$

We note that A_{ii} is common to every cluster, thus we can avoid calculating it, while B_{cc} must be calculated once during each iteration.

2.2 HMRF model and kernel-based semi-supervised clustering

In semi-supervised clustering, we are given two sets of pairwise constraints: a set of must-link constraints $ML = \{(\mathbf{x}_i, \mathbf{x}_j)\}$, and a set of cannot-link constraints $CL = \{(\mathbf{x}_i, \mathbf{x}_j)\}$. The goal is to partition the data into k clusters so that a measure of distortion between each point and the corresponding cluster representative is minimized, while maximally satisfying the set of pairwise constraints. Basu et al. [4] proposed a framework for semi-supervised clustering based on Hidden Markov Random Fields (HMRFs). Considering the squared Euclidean distance as a measure of cluster distortion, and the generalized Potts potential as a constraint violation potential, the semi-supervised clustering objective can be expressed as [4]:

$$J_{\text{obj}}\left(\{\pi_c\}_{c=1}^{k}\right) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\mathbf{x}_i - \mathbf{m}_c\|^2 + \sum_{\mathbf{x}_i, \mathbf{x}_j \in ML, l_i \neq l_j} w_{ij} + \sum_{\mathbf{x}_i, \mathbf{x}_j \in CL, l_i = l_j} \overline{w}_{ij}$$

where $\mathbf{m}_c$ is the centroid of cluster π_c, ML is the set of must-link constraints, CL is the set of cannot-link constraints, w_{ij} and $\overline{w}_{ij}$ are the penalty costs for violating a must-link and a cannot-link constraint, respectively, and l_i represents the cluster label of $\mathbf{x}_i$.

Kulis et al. [16] extended this framework to kernel-based semi-supervised clustering. Instead of adding a penalty term for a must-link violation, a reward is given for the satisfaction of the constraint. This is achieved by subtracting the corresponding penalty term from the objective:

$$J_{\text{obj}}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\mathbf{x}_i \in \pi_c} \|\phi(\mathbf{x}_i) - \mathbf{m}_c^{\phi}\|^2 - \sum_{\mathbf{x}_i, \mathbf{x}_j \in ML, l_i = l_j} w_{ij} + \sum_{\mathbf{x}_i, \mathbf{x}_j \in CL, l_i = l_j} \overline{w}_{ij} \quad (1)$$

The SS-Kernel-KMeans algorithm with a Gaussian kernel [16] is shown to outperform the HMRF-KMeans algorithm [4], and SS-Kernel-KMeans with a linear kernel. However, the setting of kernel's parameters is left to manual tuning, and the chosen value can significantly affect the quality of clustering results. Thus, the selection of kernel parameters remains a critical and open problem when only limited supervision is available. Our approach, discussed in the next section, tackles this problem.

3 Kernel optimization using pairwise constraints

3.1 Kernel construction

Suppose that K_1 and K_2 are known kernel functions. Using the closure properties of kernel functions [23], we can construct a new kernel K as a linear combination of K_1 and K_2: $K = \alpha_1 K_1 + \alpha_2 K_2$, where $\alpha_1, \alpha_2 \in \Re^+$. By generalizing to $m \geq 2$, we consider a linear combination of m Gaussian kernels $K_1, K_2, \ldots, K_m$, with unknown spread parameters

$\sigma_1, \sigma_2, \ldots, \sigma_m$, respectively:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{m} \alpha_l \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_l^2}\right) \tag{2}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_m \in \Re^+$ and $\sum_{l=1}^{m} \alpha_l = 1$. For notational convenience, we group the parameters $\alpha_1, \ldots, \alpha_m, \sigma_1, \ldots, \sigma_m$ into a vector $\theta = (\alpha_1, \ldots, \alpha_{m-1}, \sigma_1, \ldots, \sigma_m)$. We note that $\alpha_m = 1 - \sum_{l=1}^{m-1} \alpha_l$. Our goal is to optimize the kernel K with respect to the parameters θ. We formulate K in terms of Gaussian kernels since they are widely used in the literature, and have shown very good learning properties in a variety of applications. In practice, m may be set to a small positive integer to allow accurate parameter estimation when the number of available pairwise constraints is limited. In our experiments, we set $m = 3$.

3.2 Objective function

We assume that we are given a set of data $X = \{\mathbf{x}_i\}_{i=1}^{N} \subseteq \Re^D$, a set ML of must-link constraints, and a set CL of cannot-link constraints on X. We want to utilize this information to guide the clustering procedure to discover a partition of the data in X that conforms to the given constraints. To this end, we want to compute optimal parameter values for the kernel function given in Eq. (2), using the ML and CL constraints. The idea is then to learn a kernel that maps pairs of points subject to a must-link constraint close to each other in feature space, and maps points subject to a cannot-link constraint far apart in feature space. This goal is achieved by the following objective function:

$$\begin{aligned} F_{\text{kernel}} = &\frac{1}{N_{CL}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in CL} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \\ &- \frac{1}{N_{ML}} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in ML} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 \end{aligned} \tag{3}$$

where N_{CL} and N_{ML} are the number of cannot-link and must-link constraints, respectively.

The maximization of the above function over the hyperparameter θ implicitly defines a feature space in which similarity is measured according to the information provided by the given constraints. Thus, meaningful clustering can be readily achieved in such a feature space. By expanding the distance computation $\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2$, and using the kernel trick $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ and the definition of K given in Eq. (2), we obtain the following:

$$\begin{aligned} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in CL} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 &= \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in CL} \left\{K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)\right\} \\ &= \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in CL} \left\{\sum_{l=1}^{m} \alpha_l + \sum_{l=1}^{m} \alpha_l - 2K(\mathbf{x}_i, \mathbf{x}_j)\right\} \\ &= 2N_{CL} - 2 \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in CL} K(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{4}$$

Similarly, for must-link constraints we obtain the following:

$$\sum_{(\mathbf{x}_i, \mathbf{x}_j) \in ML} \|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 = 2N_{ML} - 2 \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in ML} K(\mathbf{x}_i, \mathbf{x}_j) \tag{5}$$

By substituting the above into (3), we can rewrite the objective function F_{kernel} as follows:

$$F_{\text{kernel}} = 2\frac{1}{N_{ML}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML} K(\mathbf{x}_i,\mathbf{x}_j) - 2\frac{1}{N_{CL}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in CL} K(\mathbf{x}_i,\mathbf{x}_j) \tag{6}$$

3.3 Optimization algorithm

To maximize the objective function F_{kernel} given in Eq. (6), we follow gradient ascent. We consider the definition of K given in (2), and compute the partial derivatives of F_{kernel} with respect to θ:

$$\frac{\partial F_{\text{kernel}}}{\partial \theta_i} = 2\frac{1}{N_{ML}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in ML} \frac{\partial K(\mathbf{x}_i,\mathbf{x}_j)}{\partial \theta_i} - 2\frac{1}{N_{CL}} \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in CL} \frac{\partial K(\mathbf{x}_i,\mathbf{x}_j)}{\partial \theta_i}$$

Computing the partial derivatives of K with respect to the parameters σ_l, for $l = 1, 2, \ldots, m-1$, gives:

$$\frac{\partial K(\mathbf{x}_i,\mathbf{x}_j)}{\partial \sigma_l} = \alpha_l \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_l^2}\right) \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_l^3}$$

The partial derivative with respect to σ_m is:

$$\frac{\partial K(\mathbf{x}_i,\mathbf{x}_j)}{\partial \sigma_m} = \left(1 - \sum_{l=1}^{m-1} \alpha_l\right) \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_m^2}\right) \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_m^3}$$

and the partial derivatives of K with respect to the parameters α_l, for $l = 1, 2, \ldots, m-1$, are as follows:

$$\frac{\partial K(\mathbf{x}_i,\mathbf{x}_j)}{\partial \alpha_l} = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_l^2}\right) - \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_m^2}\right)$$

We maximize the objective function F_{kernel} by performing gradient ascent in parameter space θ using the above partial derivatives:

$$\theta_i^{(\text{new})} = \theta_i^{(\text{old})} + \rho \frac{\partial F_{\text{kernel}}}{\partial \theta_i}$$

where ρ is a scalar parameter that determines the length of the step at each iteration; ρ is optimized via a line-search method. Line search is performed using a merit function similar to that proposed in Han [12], Powell [21], and Powell [22]. The resulting gradient ascent algorithm is summarized in Algorithm 3.1.

Algorithm 3.1 Optimization algorithm.

Step 1: Initialize the vector of parameters θ to an initial vector value θ_0.
Step 2: Compute the gradients $\frac{\partial F_{\text{kernel}}}{\partial \theta_i}$.
Step 3: Update the components θ_i of the parameter vector θ according to the rule:
$\theta_i^{(\text{new})} = \theta_i^{(\text{old})} + \rho \frac{\partial F_{\text{kernel}}}{\partial \theta_i}$
Step 4: Go to step (2) until the maximal value of F_{kernel} is achieved.
Step 5: Output θ.

The algorithm produces as output the optimal parameters θ that, when coupled with Eq. (2) give rise to the optimal kernel. Such an optimal kernel can then be utilized in conjunction with a kernel-based clustering procedure to determine a partition of the data. In our experiments, resulting kernels from optimization of Eq. (3) are combined with the objective function (Eq. 1) in the SS-Kernel-KMeans algorithm [16] to obtain clustering results reported in Sect. 4.

Our kernel optimization is a constraint optimization problem, where sequential quadratic programming (SQP) is employed to optimize the kernel parameters. SQP coupled with interior point methods has a complexity of $O(n^3)$ [6], where n denotes the number of parameters to be optimized. In our case, n is 2 times the number of kernels (i.e., for each Gaussian kernel we optimize the kernel width and the combination coefficient).

We observe that our optimization procedure can be used to tackle a broad range of problems. In fact, the learned kernel function K induces a notion of similarity in input space and can be used as input to any (unsupervised) clustering algorithm. We verified this conjecture in the second set of experiments we present in this paper.

4 Experimental evaluation

4.1 Datasets

We performed experiments on one simulated dataset and six real datasets.

1. **Two concentric**: The simulated dataset contains two clusters in two dimensions distributed as concentric circles (see Fig. 1). Each cluster contains 200 points.
2. **Digits**: This dataset is the pendigits handwritten character recognition dataset from the UCI repository [5]. 10% of the data was chosen randomly from the four classes {3, 6, 8, 9}. This results in 423 points and 16 dimensions.
3. **Ionosphere**: This dataset is also from the UCI repository [5]. It was collected by a radar system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not. The dataset has 34 features and 351 points.
4. **Vowel**: This dataset concerns the recognition of eleven steady state vowels of British English, using a specified training set of lpc derived log area ratios.[1] Three classes corresponding to the vowels "i," "I," and "E" were chosen, for a total of 126 points in 10 dimensions.
5. **Wine**: This dataset is from the UCI repository [5]. The dataset is based on the chemical analysis of wines grown in the same region in Italy, but derived from three different cultivars. The total number of wines are 178. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
6. **Zip**: The Zip dataset [13] includes normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images have been deslanted and size normalized, resulting in 16×16 grayscale images. The dataset has 256 dimensions, 500 points, and 10 classes.
7. **Modis**: Modis contains remotely sensed data obtained using a global sensor (Moderate Resolution Imaging Spectrometer) with high frequency repeat coverage. This dataset has 112 dimensions, 1989 points, and 10 classes. It can be downloaded from http://mow.ecn.purdue.edu/xz/.

[1] http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/.

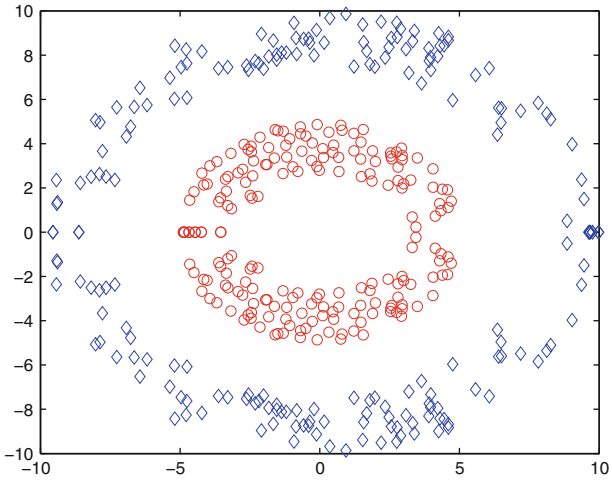


Fig. 1 Two concentric data

4.2 Evaluation criterion

To evaluate clustering performance, we use the Rand Statistic index [25,27,29] and the Normalized Mutual Information (NMI) [24].

The Rand Statistic is an external cluster validity measure that estimates the quality of clustering results with respect to the underlying classes of the data. Let P_1 be the partition of the data X after applying a clustering algorithm, and P_2 be the underlying class structure of the data. We refer to a pair of points $(\mathbf{x}_u, \mathbf{x}_v) \in X \times X$ from the data using the following terms:

- SS: if both points belong to the same cluster of P_1 and to the same group of the underlying class structure P_2.
- SD: if the two points belong to the same cluster of P_1 and to different groups of P_2.
- DS: if the two points belong to different clusters of P_1 and to the same group of P_2.
- DD: if both points belong to different clusters of P_1 and to different groups of P_2.

Assume now that N_{SS}, N_{SD}, N_{DS} and N_{DD} are the number of SS, SD, DS and DD pairs, respectively. Then $N_{SS} + N_{SD} + N_{DS} + N_{DD} = N_{\text{Pair}}$, which is the maximum number of all pairs in the data set ($N_{\text{Pair}} = N(N-1)/2$. Here N is the total number of points in the data set). The Rand Statistic index measures the degree of similarity between P_1 and P_2 as follows:

$$\text{Rand Statistic} = (N_{SS} + N_{DD})/N_{\text{Pair}}$$

The Normalized Mutual Information (NMI) is a pairwise similarity measure that quantifies the information shared between two partitions. Let P_1 be the partition of the data X into c_1 clusters after applying a clustering algorithm, and P_2 be the underlying class structure of the data comprising c_2 classes. Let's assume $n_i^{(1)}$ represent the number of points in cluster i of P_1, $n_j^{(2)}$ represent the number of points in cluster j of P_2, and n_{ij} is the number of points shared by cluster i of P_1 and cluster j of P_2. The NMI between P_1 and P_2 is a value in [0, 1], and is computed according to the average mutual information between every pair of clusters

in the two given partitions [24]:

$$\mathrm{NMI}(P_1, P_2) = \frac{\sum_{i=1}^{c_1}\sum_{j=1}^{c_2} n_{ij}\log\frac{n_{ij}n}{n_i^{(1)}n_j^{(2)}}}{\sqrt{\sum_{i=1}^{c_1} n_i^{(1)}\log\frac{n_i^{(1)}}{n}\sum_{j=1}^{c_2} n_j^{(2)}\log\frac{n_j^{(2)}}{n}}} \tag{7}$$

4.3 Results and discussion

To evaluate the effectiveness of the proposed optimal kernel algorithm, we combine it with the SS-Kernel-KMeans technique [16]. This technique is described in Sect. 2. We call the combined approach "SS-Optimal-Kernel-KMeans." In the experiments, we consider the linear combination of three Gaussian kernels, i.e. we set $m = 3$ in Eq. (2). We found that three kernel functions provide enough flexibility and modeling capability, while keeping the number of parameters to be estimated low. We also vary the number of Gaussian kernels, and analyze the quality of the clustering as a function of the number of kernels used. We initialize the weights α_i to equal values, i.e. $\alpha_i = \frac{1}{3}$ for $i = 1, 2, 3$. The spread parameters σ_i are initialized to the values $r_i S$, where r_i is a random number between 0 and 1, and S is the average of standard deviations of the data computed over each dimension.

We compare our method SS-Optimal-Kernel-KMeans with SS-Kernel-KMeans itself. SS-Kernel-KMeans requires as input a predefined value for the Gaussian kernel parameter σ [16]. In the absence of labeled data, parameters cannot be cross-validated; thus, we estimate the expected accuracy of SS-Kernel-KMeans by averaging the resulting clustering quality over multiple runs for different values of σ. Specifically, we test the SS-Kernel-KMeans algorithm with the values of σ^2: 0.1, 1, 10, 100, 1,000, 10,000. We report the average Rand Statistic and NMI achieved over the six σ values, as well as the average over the best three performances achieved (SS-Kernel-KMeans-Best3), in order to show the advantage of our technique. The violation costs w_{ij} and $\overline{w}_{ij}$ in SS-Kernel-Kmeans are set to $\frac{N}{kC}$, as in [16], where N is the number of data points, k is the number of clusters, and C is the total number of constraints. The value of k is set to the actual number of classes in the data. For both SS-Kernel-KMeans and SS-Optimal-Kernel-KMeans, the clusters are initialized using the approach presented in Kulis et al. [4,16]: we take the transitive closure of the constraints to form neighborhoods, and then perform a farthest-first traversal on these neighborhoods to obtain the k initial clusters. We ensure that the same constraint information is given to each competitive algorithm.

Figures 2, 3, 4, 5, 6, 7 and 8 show the learning curves using 20 runs of 2-fold cross-validation for each data set (30% for training and 70% for testing). These plots show the improvement in clustering quality on the test set as a function of an increasing amount of pairwise constraints. To study the effect of constraints in clustering, 30% of the data were randomly drawn from the training set at any particular fold, and the constraints are generated only using the training set. The clustering algorithm was run on the whole data set, but we calculated the Rand Statistic and the NMI only on the test set. Each point on the learning curve is an average of results over 20 runs.

These results clearly demonstrate the effectiveness of the SS-Optimal-Kernel-KMeans approach. For all seven datasets, the clustering quality achieved by the SS-Optimal-Kernel-KMeans is significantly better that the results computed by SS-Kernel-KMeans averaged over the σ values tested. The results provided by the Rand Statistic and the NMI are consistent. In addition, the SS-Kernel-KMeans algorithm with the optimal kernel also outperforms the average top three performances of SS-Kernel-KMeans. These results show that

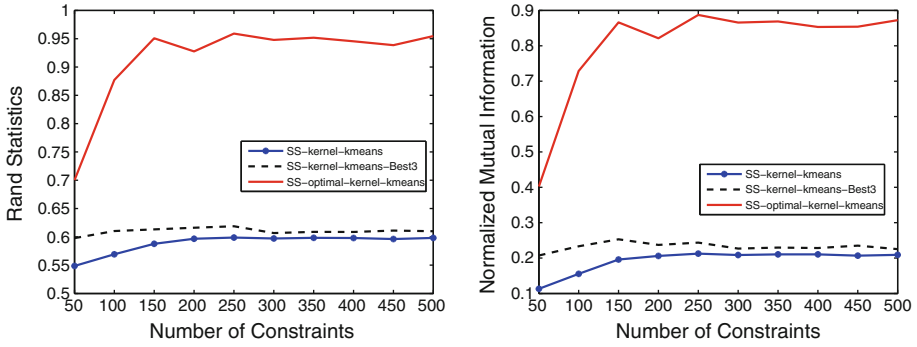


Fig. 2 Clustering results on Two Concentric data: *left* Rand Statistic, *right* Normalized Mutual Information

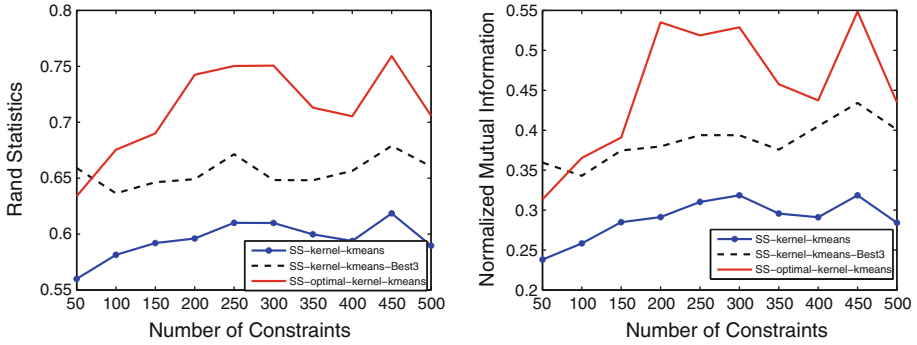


Fig. 3 Clustering results on Vowel data: *left* Rand Statistic, *right* Normalized Mutual Information

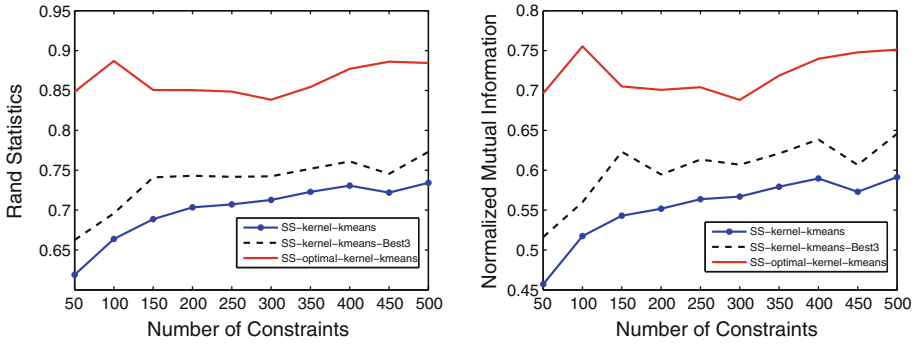


Fig. 4 Clustering results on Digits data: *left* Rand Statistic, *right* Normalized Mutual Information

our technique is capable of estimating the optimal kernel parameter values from the given constraints. In particular, for the Two Concentric data (see Fig. 2), the SS-Kernel-KMeans with the optimal kernel effectively uses the increased amount of constraints to learn an almost perfect separation of the two clusters. For the Digits, Ionosphere, Wine, and Zip data, the SS-Kernel-KMeans with the optimal kernel provides a clustering quality that is significantly higher than the one computed by SS-Kernel-KMeans, even when a small amount of constraints is available. This behavior is very desirable since in practice only a limited amount of supervision might be available.

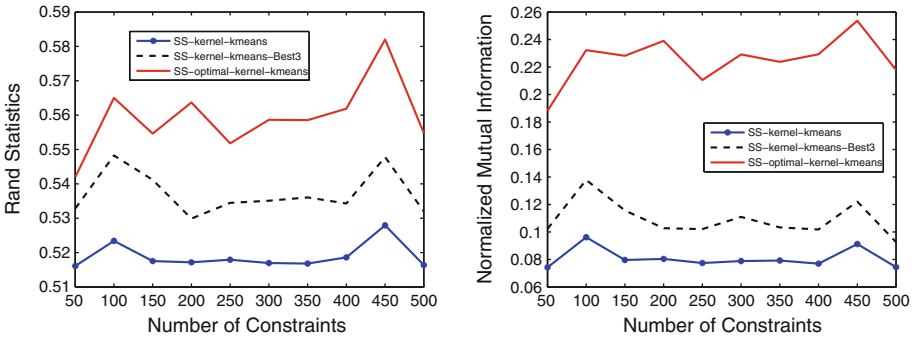


Fig. 5 Clustering results on Ionosphere data: *left* Rand Statistic, *right* Normalized Mutual Information

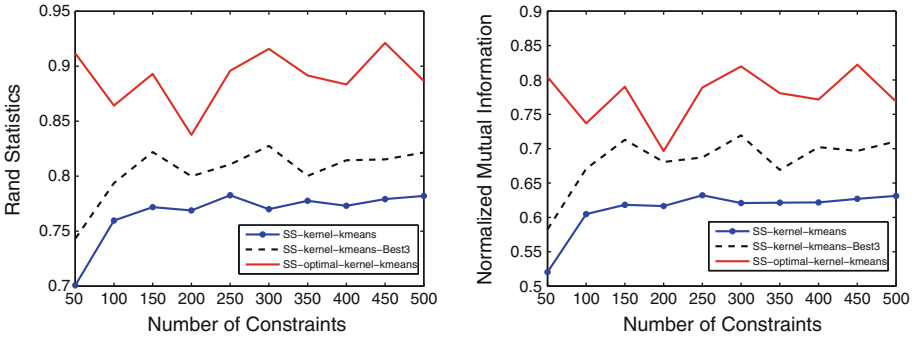


Fig. 6 Clustering results on Wine data: *left* Rand Statistic, *right* Normalized Mutual Information

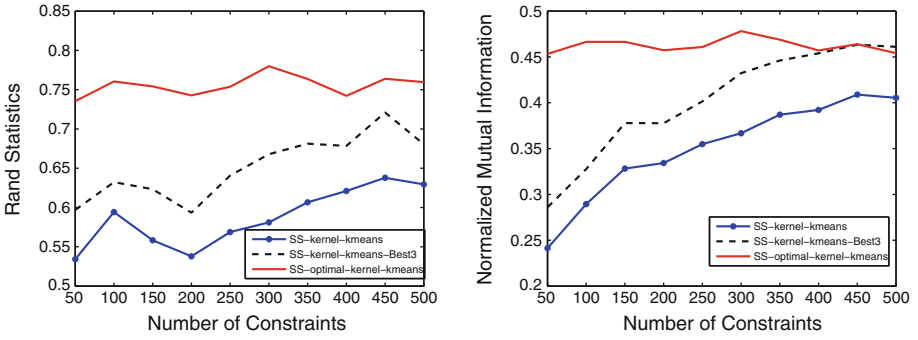


Fig. 7 Clustering results on Zip data: *left* Rand Statistic, *right* Normalized Mutual Information

For some datasets, the Rand statistic (NMI) fluctuates. As mentioned earlier, each point on the learning curves (corresponding to a given number of constraints) is an average of results over 20 runs. For each run, constraints are generated from scratch. Thus, the observed fluctuations may be due to a different degree of significance of the chosen constraints for the problem of kernel learning. Furthermore, the optimization procedure only guarantees local optima solutions, which depend on the initialization of the vector of parameters [this is the reason for the low performance achieved by our method on the Modis data when only 50

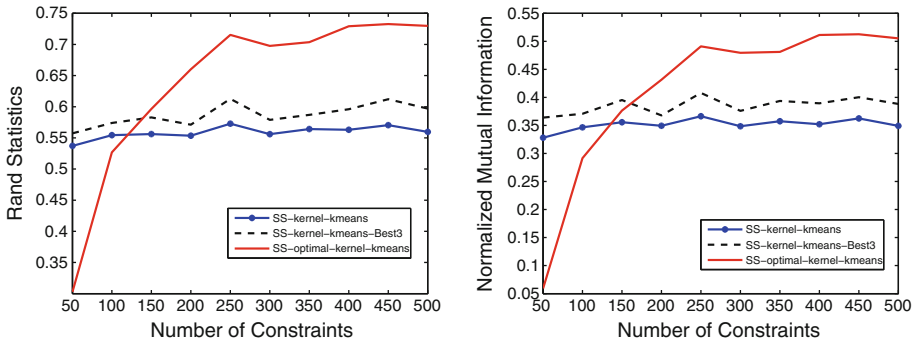


Fig. 8 Clustering results on Modis data: *left* Rand Statistic, *right* Normalized Mutual Information

constraints are available (see Fig. 8)]. It is possible to mitigate this sensitivity to initialization by following stochastic gradient ascent.

We observe that our optimization procedure can be used to tackle a broader range of problems. In fact, the learned kernel function K induces a notion of similarity in input space: for each pair of points $\mathbf{x}_i$ and $\mathbf{x}_j$, it provides a similarity measure that embeds the given constraints. As a result, the learned kernel function can be used as input to any (unsupervised) clustering algorithm. In particular, it can be used in combination with unsupervised Kernel-KMeans.

To verify this conjecture, we ran Kernel-KMeans with the optimal kernel learned through our optimization algorithm and without enforcing the constraints during the clustering process. The results are shown in Figs. 9, 10, 11, 12, 13, 14 and 15. In each case, we plot the learning curves of Kernel-KMeans with the optimal kernel (without enforcing the constraints), SS-Kernel-KMeans with the optimal kernel, and (unsupervised) Kernel-KMeans. For Kernel-KMeans with the optimal kernel, we initialize the clusters using the given constraints, as is the case for SS-Kernel-KMeans. For (unsupervised) Kernel-KMeans, the clusters are randomly initialized. In addition, we tested the (unsupervised) Kernel-KMeans algorithm with the following values of σ^2: 0.1, 1, 10, 100, 1,000, 10,000. The average Rand Statistic and NMI achieved over the six σ values are reported. Each learning curve was obtained as in the previous experiments using 20 runs of 2-fold-cross-validation for each data set (30% for training and 70% for testing).

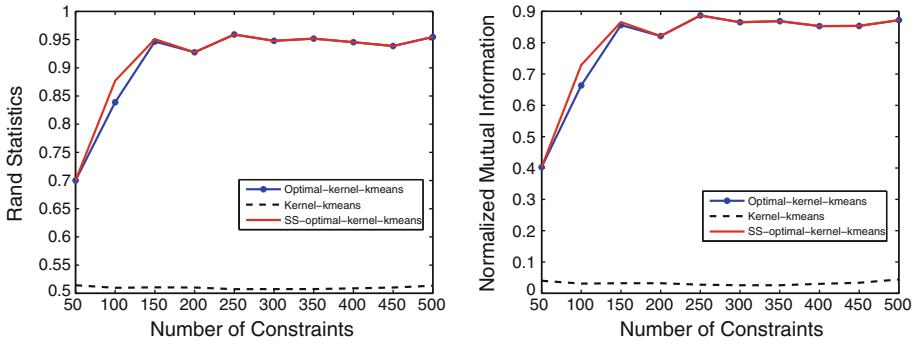


Fig. 9 Clustering results without enforcement of constraints on Two Concentric data: *left* Rand Statistic, *right* Normalized Mutual Information

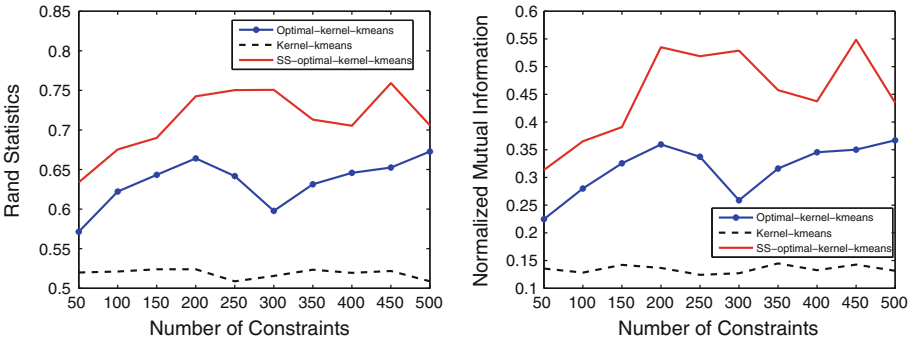


Fig. 10 Clustering results without enforcement of constraints on Vowel data: *left* Rand Statistic, *right* Normalized Mutual Information

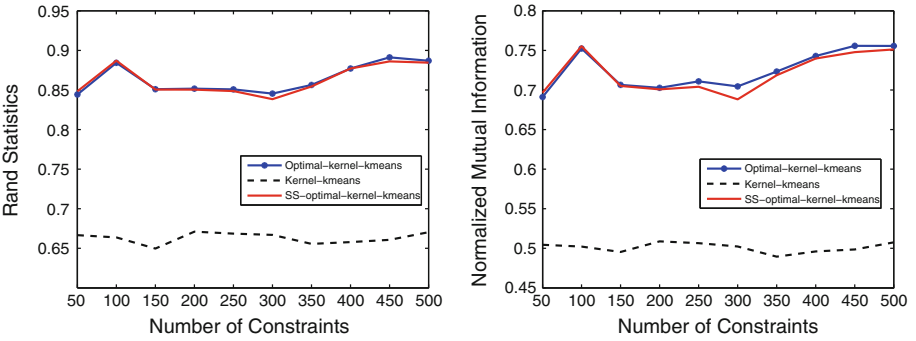


Fig. 11 Clustering results without enforcement of constraints on Digits data: *left* Rand Statistic, *right* Normalized Mutual Information

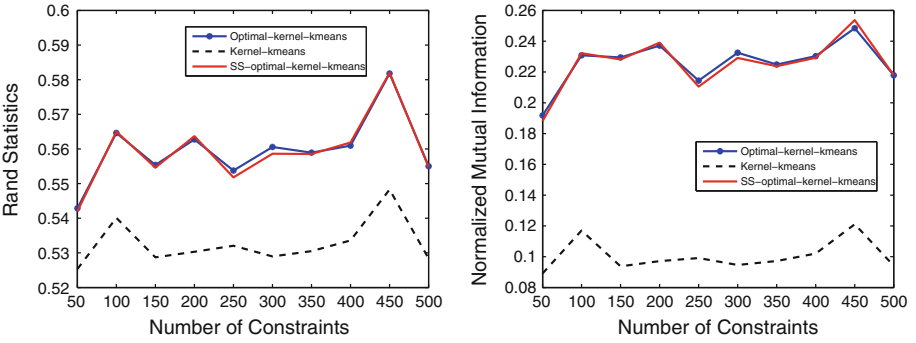


Fig. 12 Clustering results without enforcement of constraints on Ionosphere data: *left* Rand Statistic, *right* Normalized Mutual Information

Figures 9, 10, 11, 12, 13, 14 and 15 show that both Kernel-KMeans with the optimal kernel and SS-Kernel-KMeans greatly outperform unsupervised Kernel-KMeans on every single data set. Furthermore, in support of our conjecture, Kernel-KMeans performs quite well, and its learning curve is very close to the learning curve of SS-Kernel-KMeans for the Two Concentric, Digits, Ionosphere, Wine, Zip, and Modis data sets. For the Vowel data

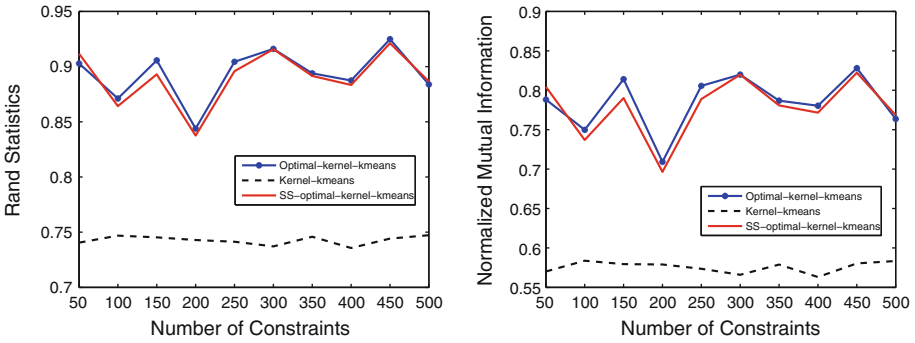


Fig. 13 Clustering results without enforcement of constraints on Wine data: *left* Rand statistic, *right* Normalized Mutual Information

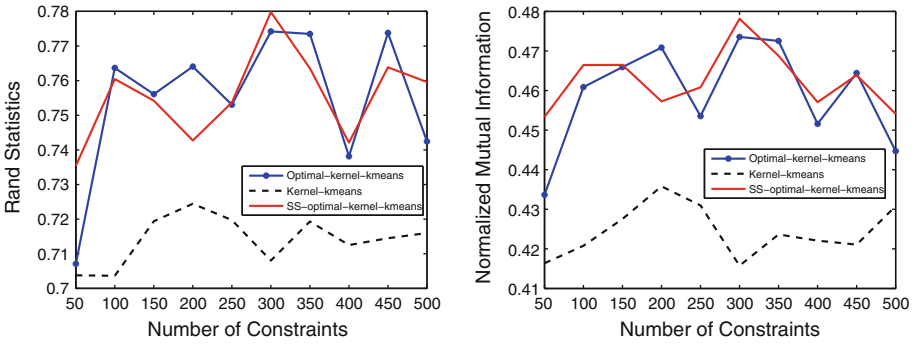


Fig. 14 Clustering results without enforcement of constraints on Zip data: *left* Rand statistic, *right* Normalized Mutual Information

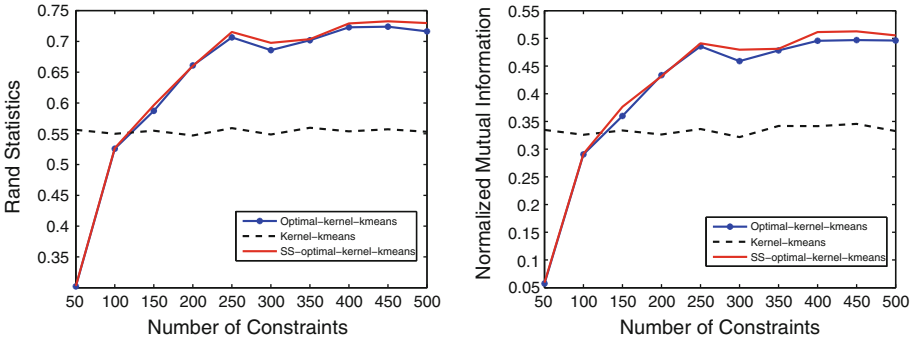


Fig. 15 Clustering results without enforcement of constraints on Modis data: *left* Rand statistic, *right* Normalized Mutual Information

set, the gap between the two curves is narrow for a small and large number of constraints. These results support the feasibility of using our optimization technique as a mechanism that provides an adaptive similarity measure for clustering in general.

Figure 16 shows the Rand Statistic and the NMI as a function of the number of Gaussian kernels used (from one to 10) for four representative datasets: Digits, Wine, Zip, and

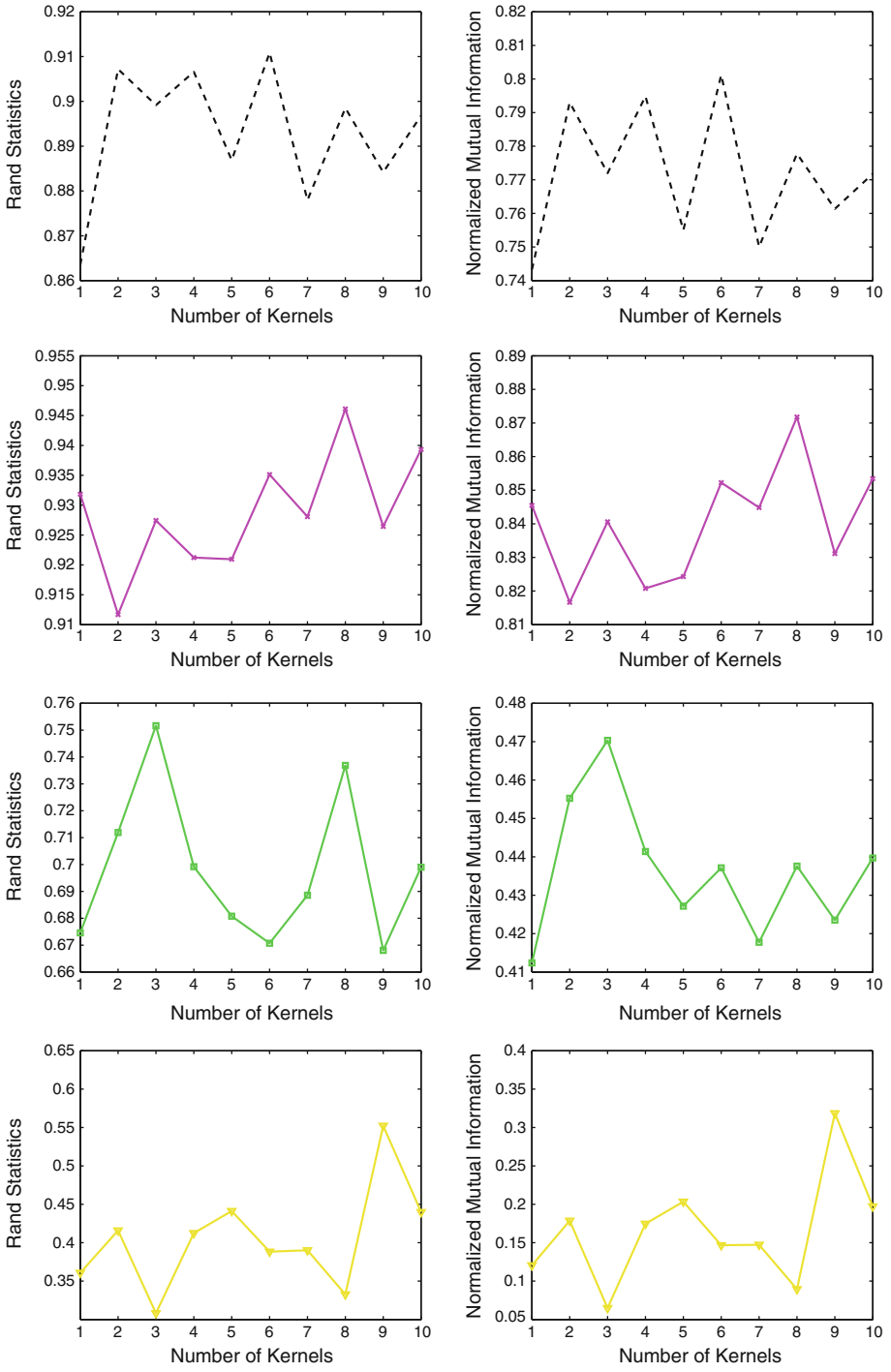


Fig. 16 *Left* Rand Statistic and *right* Normalized Mutual Information versus the number of kernels. *Top to bottom*: Digits, Wine, Zip, and Modis datasets

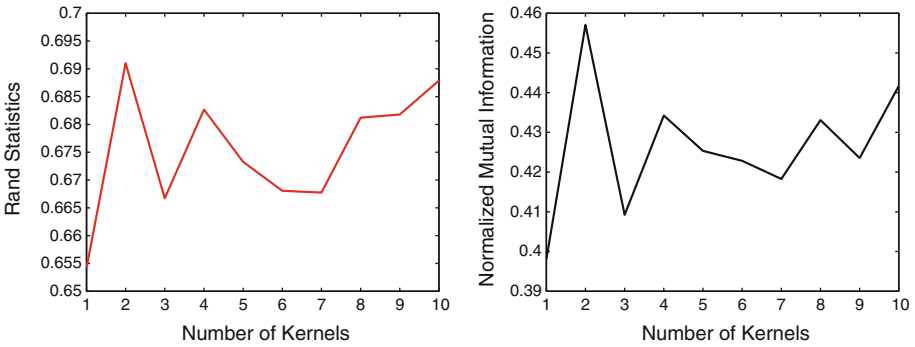


Fig. 17 *Left* Rand Statistic and *right* Normalized Mutual Information versus the number of kernels. Results are averaged across all datasets

Modis. For each dataset, we performed 20 runs and used 100 constraints. The trend fluctuates since the optimization process depends on the initial conditions. For this reason, we also averaged the results across all datasets (see Fig. 17). On average performance can benefit from increased number of kernels. However, the optimal number of kernels depends on the specific problem. Furthermore, results may also depend on initial conditions.

5 Summary

We have proposed a new method for optimizing the parameters of composite kernels for semi-supervised clustering. Our optimization strategy can be applied to any kernel-based clustering techniques. We have tested the optimal kernel function computed by our technique in combination with a semi-supervised kernel-Kmeans algorithm. The experimental results demonstrate that our technique makes powerful kernel-based semi-supervised clustering approaches practical by providing a mechanism to automatically select critical parameters. In our future work, we plan on investigating the use of our technique in combination with other clustering techniques, such as spectral clustering. We will also study regularized versions of the proposed objective function to avoid overfitting when a large number of constraints is available.

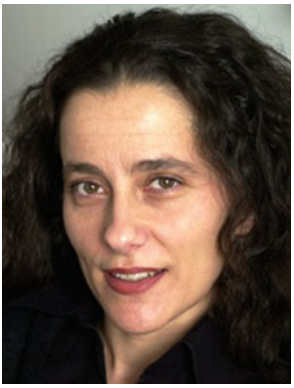
Acknowledgments This work was in part supported by NSF CAREER Award IIS-0447814.

References

1. Amini MR, Gallinari P (2005) Semi-supervised learning with an imperfect supervisor. Knowl Inf Syst 8:385–413
2. Bach F, Lanckriet GRG, Jordan MI (2004) Multiple kernel learning, conic duality, and the SMO algorithm. In: International conference on machine learning, pp 41–48
3. Bar-Hillel A, Hertz T, Shental N, Weinshall D (2003) Learning distance functions using equivalence relations. In: International conference on machine learning, pp 11–18
4. Basu S, Bilenko M, Mooney RJ (2004) A probabilistic framework for semi-supervised clustering. In: International conference on knowledge discovery and data mining, pp 59–68
5. Blake CL, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html

6. Blaszczyk J, Karbowski A, Malinowski K (2007) Object library of algorithms for dynamic optimization problems: benchmaring SQP and nonlinear interior point methods. Int J Appl Math Comput Sci 17(4):515–537
7. Chapelle O, Vapnik V (2002) Choosing mutiple parameters for support vector machines. Machine Learn 46(1):131–159
8. Chen Y, Rege M, Dong M, Hua J (2008) Non-negative matrix factorization for semi-supervised data clustering. Knowl Inf Syst 17:355–379
9. Cohn D, Caruana R, McCallum A (2003) Semi-supervised clustering with user feedback. TR2003-1892, Cornell University
10. Cristianini N, Shawe-Taylor J, Elisseeff A (2001) On kernel-target alignment, neural information processing systems 14. MIT Press, Cambridge 367–373
11. Davis J, Kulis B, Jain P, Sra S, Dhillon IS (2007) Information-theoretic metric learning. In: International conference on machine learning, pp 209–216
12. Han SP (2007) A globally convergent method for nonlinear programming. J Optim Theory Appl 22(3):297–309
13. Hastie T, Tibshirani R, Friedman JH (2001) The elements of statistical learning. Springer, Berlin
14. Huang J, Yuen PC, Chen WS, Lai JH (2004) Kernel subspace LDA with optimized kernel parameters on face recognition. In: The sixth IEEE international conference on automatic face and gesture recognition, pp 327–332
15. Kondor R, Jebara T (2007) Gaussian and Wishart hyperkernels. In: Advances in neural information processing systems vol 19. MIT Press, Cambridge, pp 729–736
16. Kulis B, Basu S, Dhillon I, Moony R (2005) Semi-supervised graph clustering: a kernel approach. In: International conference on machine learning, pp 457–464
17. Lanckriet GRG, Cristianini N, Bartlett P, El Ghaoui L, Jordan MI (2004) Learning the kernel matrix with semidefinite programming. J Machine Learn Res 5:27–72
18. Lanckriet GRG, De Bie T, Cristianini N, Jordan MI, Noble WS (2004) A statistical framework for genomic data fusion. Bioinformatics 20(16):2626–2635
19. Li T, Ogihara M (2004) Semisupervised learning from different information sources. Knowl Inf Syst 7:289–309
20. Ong CS, Smola AJ, Williamson RC (2003) Hyperkernels. In: Advances in neural information processing systems vol 15. MIT Press, Cambridge, pp 478–485
21. Powell MJD (1978) A fast algorithm for nonlinearly constrained optimization calculations. In: Watson GA (ed) Numerical analysis. Lecture notes in mathematics, Springer, Berlin 630:144–157
22. Powell MJD (1978) The convergence of variable metric methods for nonlinearly constrained optimization calculations. In: Mangasarian OL, Meyer RR, Robinson SM (eds) Nonlinear programming 3. Academic Press, New York
23. Shawe-Tayor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambrige University, Cambrige
24. Strehl A, Ghosh J (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J Machine Learn Res 3(3):583–617
25. Theodoridis S, Koutroubas K (1999) Pattern recognition. Academic Press, New York
26. Vapnik V (1995) The nature of statistical learning theory. Wiley, New York
27. Wagstaff K, Cardie C, Rogers S, Schroedl S (2001) Constrained K-Means clustering with background knowledge. In: International conference on machine learning, pp 577–584
28. Wang W, Xu Z, Lu W, Zhang X (2002) Determination of the spread parameter in the Gaussian kernel for classification and regression. Neurocomputing 55(3–4):643–663
29. Xing EP, Ng AY, Jordan MI, Russell S (2003) Distance metric learning, with application to clustering with side-information. In: Advances in neural information processing systems vol 15. MIT Press, Cambridge, pp 505–512
30. Yan B, Domeniconi C (2006) An adaptive kernel method for semi-supervised clustering. In: European conference on machine learning, pp 521–532
31. Ye J, Chen K, Wu T, Li J, Zhao Z, Patel R, Bae M, Janardan R, Liu H, Alexander G, Reiman E (2008) Heterogeneous data fusion for Alzheimer's disease study. In: The 4th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1025–1033
32. Ye J, Ji S, Chen J (2008) Multi-class discriminant kernel learning via convex programming. J Machine Learn Res 9:719–758

Author Biographies



Carlotta Domeniconi is an Associate Professor in the Department of Computer Science at George Mason University. Her research interests include machine learning, pattern recognition, data mining, and feature relevance estimation, with applications in text mining and bioinformatics. She has published extensively in premier journals and conferences in Machine Learning and Data Mining. Dr. Domeniconi is a recipient of an ORAU Ralph E. Powe Junior Faculty Enhancement Award and an NSF CAREER Award.

Jing Peng is an Associate Professor of Computer Science at Montclair State University. His research interests include data mining, information retrieval, and computer science education. He has served as PI and Co-PI of several projects from NSF, ARO, NASA, and ONR. Currently he serves as Editor in Chief of the Open Journal of Artificial Intelligence, Bentham Science Publishers. He has authored or co-authored over 100 technical publications in the areas of his interest.

Bojun Yan received his Masters in Computer Science from Beijing Institute of Technology in 2003. He was a graduate student at George Mason University during 2003–2007. He has authored papers in data mining and statistical learning. He is currently working in the financial industry; his responsibilities include risk management, loss mitigation, and score model development and validation.