Semi-Supervised Kernel Mean Shift Clustering

Saket Anand, *Student Member, IEEE,* Sushil Mittal, *Member, IEEE,* Oncel Tuzel, *Member, IEEE,* and Peter Meer, *Fellow, IEEE*

Abstract—Mean shift clustering is a powerful nonparametric technique that does not require prior knowledge of the number of clusters and does not constrain the shape of the clusters. However, being completely unsupervised, its performance suffers when the original distance metric fails to capture the underlying cluster structure. Despite recent advances in semi-supervised clustering methods, there has been little effort towards incorporating supervision into mean shift. We propose a semi-supervised framework for kernel mean shift clustering (SKMS) that uses only pairwise constraints to guide the clustering procedure. The points are first mapped to a high-dimensional kernel space where the constraints are imposed by a linear transformation of the mapped points. This is achieved by modifying the initial kernel matrix by minimizing a $\mathbf{log\,det}$ divergence-based objective function. We show the advantages of SKMS by evaluating its performance on various synthetic and real datasets while comparing with state-of-the-art semi-supervised clustering algorithms.

Index Terms—semi-supervised kernel clustering; $\log\det$ Bregman divergence; mean shift clustering;

1 Introduction

Mean shift is a popular mode seeking algorithm, which iteratively locates the modes in the data by maximizing the kernel density estimate (KDE). Although the procedure was initially described decades ago [17], it was not popular in the vision community until its potential uses for feature space analysis and optimization were understood [8], [12]. The nonparametric nature of mean shift makes it a powerful tool to discover arbitrarily shaped clusters present in the data. Additionally, the number of clusters is automatically determined by the number of discovered modes. Due to these properties, mean shift has been applied to solve several computer vision problems, e.g., image smoothing and segmentation [11], [41], visual tracking [9], [19] and information fusion [7], [10]. Mean shift clustering was also extended to nonlinear spaces, for example, to perform clustering on analytic manifolds [36], [38] and kernel induced feature space [34], [39] under an unsupervised setting.

In many clustering problems, in addition to the unlabeled data, often some additional information is also easily available. Depending on a particular problem, this additional information could be available in different forms. For example, the number of clusters or a few *must-link* (similarity) and *cannot-link* (dissimilarity) pairs could be known a-priori. In the last decade, semi-supervised clustering methods that aim to incorporate prior information into the clustering algorithm as a guide, have received considerable attention in machine learning and computer vision [2], [6], [18]. Increasing interest in this area has triggered the adaptation of several popular unsupervised clustering algorithms into a semi-supervised framework, e.g., background constrained k-means [40], constrained spectral clustering [23], [28] and kernel graph clustering [24]. It is shown that unlabeled data, when used in conjunction with a small amount of labeled data, can produce significant improvement in clustering accuracy. However, despite these advances, mean shift clustering has largely been ignored under the semi-supervised learning framework. To leverage all the useful properties of mean shift, we adapt the original unsupervised algorithm into a semi-supervised clustering technique.

The work in [37] introduced weak supervision into the kernel mean shift algorithm where the additional information was provided through a few pairwise must-link constraints. In that framework, each pair of must-link points was collapsed to a single point through a linear projection operation, guaranteeing their association with the same cluster. In this paper, we extend that work by generalizing the linear projection operation to a linear *transformation* of the kernel space that permits us to scale the distance between the constraint points. Using this transformation, the must-link points are moved closer to each other, while the cannot-link points are moved farther apart. We show that this transformation can be achieved implicitly by modifying the kernel matrix. We also show that given one constraint pair, the corresponding kernel update is equivalent to minimizing the $\log\det$ divergence between the updated and the initial kernel matrix. For multiple constraints, this result proves to be very useful since we can learn the kernel matrix by solving a constrained $\log\det$ minimization problem [22].

Fig. 1 illustrates the basic approach. The original data

- *S. Anand and P. Meer are with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854. E-mail: {anands@eden, meer@cronos}.rutgers.edu.*
- *S. Mittal is with Scibler Technologies, Bangalore, India. E-mail: mittal@scibler.com.*
- *O. Tuzel is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139. E-mail: oncel@merl.com.*

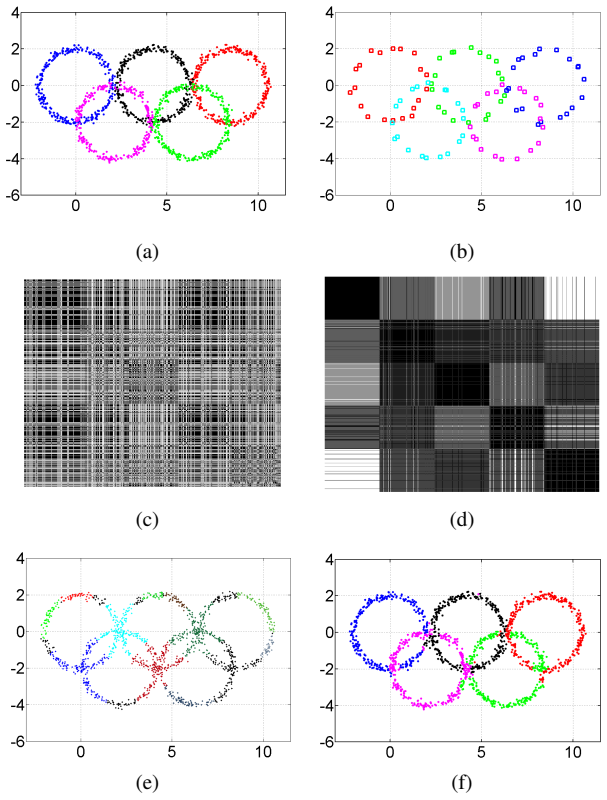


Fig. 1: Olympic Circles. (a) Original data in the input space with five different clusters. (b) Labeled points used to generate pairwise constraints. (c) Pairwise distance matrix (PDM) using the modes discovered by unsupervised mean shift clustering performed in the kernel space. The points are ordered according to class. (d) PDM using the modes found after supervision was added. Sample clustering results using (e) unsupervised and (f) semi-supervised kernel mean shift clustering.

with five circles, each containing 300 points, is shown in Fig. 1a. We assume that 20 labeled points from each cluster are selected at random (only 6.7% of the data) to generate pairwise must-link and cannot-link constraints (Fig. 1b). We use all the $5 \times \binom{20}{2}$ unique intra-class pairs as must-link constraints and a same number of cannot-link constraints. Note that, although we generate pairwise constraints using a small fraction of labeled points, the algorithm *does not* require the explicit knowledge of class labels. The data is first mapped to a kernel space using a Gaussian kernel ($\sigma = 0.5$). In the first case, kernel mean shift is directly applied to the data points in the absence of any supervision. Fig. 1c shows the corresponding 1500×1500 pairwise distance matrix (PDM) obtained using modes recovered by mean-shift. The lack of a block diagonal structure implies the absence of meaningful clusters discovered in the data. In the second case, the constraint pairs are used to transform the initial kernel space by learning an appropriate kernel matrix via $\log\det$ divergence minimization. Fig. 1d shows the PDM between the modes obtained when kernel mean shift is applied to the transformed feature points. In this case, the five-cluster structure is clearly visible in the PDM. Finally, Fig. 1e and 1f show the corresponding clustering labels mapped back to the input space. Through this example, it becomes evident that a small amount of supervision can improve the clustering performance significantly.

In the following, we list our key contributions (•) and present the overall organization of the paper.

- ○ In Section 2, we present an overview of the past work in semi-supervised clustering.
- • In Section 3.1, we briefly review the Euclidean mean shift algorithm and in Section 3.2 we discuss its extension to high-dimensional kernel spaces. By employing the kernel trick, the explicit representation of the kernel mean shift algorithm in terms of the mapping function is transformed into an implicit representation described in terms of the kernel function inducing that mapping.
- • In Section 4.1, we derive the expression for performing kernel updates using orthogonal projection of the must-link constraint points. With little manipulation of the distances between the constraint points, in Section 4.2, we extend this projection operation to the more general linear transformation, that can also utilize cannot-link constraints. By allowing relaxation on the constraints, we motivate the formulation of learning the kernel matrix as a Bregman divergence minimization problem.
- ○ In Section 5, we review the optimization algorithm of [22], [25] for learning the kernel matrix using $\log\det$ Bregman divergences. More details about this algorithm are also furnished in the supplementary material.
- • In Section 6, we describe the complete semi-supervised kernel mean shift (SKMS) algorithm. In addition, several practical enhancements like automatically estimating various algorithmic parameters and low-rank kernel learning for significant computation gains are also discussed in detail.
- ○ In Section 7, we show experimental results on several synthetic and real data sets that encompass a wide spectrum of challenges encountered in practical machine learning problems. We also present detailed comparison with state-of-the-art semi-supervised clustering techniques.
- ○ Finally, in Section 8, we conclude with a discussion around the key strengths and limitations of our work.

2 Related Work

Semi-supervised clustering has received a lot of attention in the past few years due to its highly improved performance over traditional unsupervised clustering methods [3], [6]. As compared to fully-supervised learning algorithms, these methods require a weaker form of supervision in terms of both the amount and the form of labeled data used. In clustering, this is usually achieved by using only a few pairwise must-link and cannot-link constraints. Since generating pairwise constraints does not require the knowledge of class labels or the number of clusters, they can easily be generated using additional information. For example, while

clustering images of objects, two images with similar text annotations could be used as a must-link pair. We discuss some of the traditional unsupervised clustering methods and their semi-supervised variants below.

Partitioning based clustering: k-means is one of the oldest and most popular clustering algorithm. See [21] for an extensive review of all the variants of k-means algorithm. One of its popular semi-supervised extensions using pairwise constraints was proposed in [40]. The method partitions the data into k non-overlapping regions such that must-link pairs are enforced to lie in the same cluster while the cannot-link pairs are enforced to lie in different clusters. However, since the method enforces all constraints rigidly, it is sensitive to labeling errors in pairwise constraints. Basu et al. [2] proposed a more robust, probabilistic model by explicitly allowing relaxation for a few constraints in k-means clustering. Similarly, Bilenko et al. [4] proposed a metric learning based approach to incorporate constraints into k-means clustering framework. Since the clustering in both these methods is performed in the input space, these methods fail to handle non-linear cluster boundaries. More recently, Kulis et al. proposed semi-supervised kernel k-means (SSKK) algorithm [24], that constructs a kernel matrix by adding the input similarity matrix to the constraint penalty matrix. This kernel matrix is then used to perform k-means clustering.

Graph based clustering: Spectral clustering [31] is another very popular technique that can also cluster data points with non-linear cluster boundaries. In [23], this method was extended to incorporate weak supervision by updating the computed affinity matrix for the specified constraint points. Later, Lu et al. [28] further modified the algorithm by propagating the specified constraints to the remaining points using a Gaussian process. More recently, Lu and Ip [29] showed improved clustering performances by applying exhaustive constraint propagation and handling soft constraints (E2CP). One of the fundamental problems with this method is that it can be sensitive to labeling noise since the effect of a mislabeled data point pair could easily get propagated throughout the affinity matrix. Moreover, in general, spectral clustering methods suffer when the clusters have very different scales and densities [30].

Density based clustering: Clustering methods in this category make use of the estimated local density of the data to discover clusters. Gaussian Mixture Models (GMM) are often used for soft clustering where each mixture represents a cluster distribution [27]. Mean shift [11], [12], [37] was employed in computer vision for clustering data in the feature space by locating the modes of the nonparametric estimate of the underlying density. There exist other density based clustering methods that are less known in the vision community, but have been applied to data mining applications. For example, DBSCAN [15] groups together points that are connected through a chain of high-density neighbors that are determined by the two parameters: neighborhood size and minimum allowable density. SSDBSCAN [26] is a semi-supervised variant of DBSCAN that explicitly uses the labels of a few points to determine the neighborhood parameters. C-DBSCAN [33] performs a hierarchical density based clustering while enforcing the must-link and cannot-link constraints.

3 Kernel Mean Shift Clustering

First, we briefly describe the Euclidean mean shift clustering algorithm in Section 3.1 and then derive the kernel mean shift algorithm in Section 3.2.

3.1 Euclidean Mean Shift Clustering

Given n data points $\mathbf{x}_i$ on a d-dimensional space $\mathbb{R}^d$ and the associated diagonal bandwidth matrices $h_i\mathbf{I}_{d\times d}$, $i = 1, ..., n$, the sample point density estimator obtained with the kernel profile $k(\mathbf{x})$ is given by

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}\,k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right). \tag{1}$$

We utilize multivariate normal profile

$$k(x) = e^{-\frac{1}{2}x} \qquad x \geq 0. \tag{2}$$

Taking the gradient of (1), we observe that the stationary points of the density function satisfy

$$\frac{2}{n}\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}(\mathbf{x}_i-\mathbf{x})\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right) = 0 \tag{3}$$

where $g(x) = -k'(x)$. The solution is a local maximum of the density function which can be iteratively reached using mean shift procedure

$$\delta\mathbf{x} = \frac{\sum_{i=1}^{n}\frac{\mathbf{x}_i}{h_i^{d+2}}\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)} - \mathbf{x} \tag{4}$$

where $\mathbf{x}$ is the current mean and $\delta\mathbf{x}$ is the mean shift vector. To recover from saddle points adding a small perturbation to the current mean vector is usually sufficient. Comaniciu and Meer [11] showed that the convergence to a local mode of the distribution is guaranteed.

3.2 Mean Shift Clustering in Kernel Spaces

We extend the original mean shift algorithm from the Euclidean space to a general inner product space. This makes it possible to apply the algorithm to a larger class of nonlinear problems, such as clustering on manifolds [38]. We also note that a similar derivation for fixed bandwidth mean shift algorithm was given in [39].

Let $\mathcal{X}$ be the input space such that the data points $\mathbf{x}_i \in \mathcal{X}$, $i = 1, ..., n$. Although, in general, $\mathcal{X}$ may not necessarily be a Euclidean space, for sake of simplicity, we assume $\mathcal{X}$ corresponds to $\mathbb{R}^d$. Every point $\mathbf{x}$ is then mapped to a $d_{\boldsymbol{\phi}}$-dimensional feature space $\mathcal{H}$ by applying the mapping functions $\phi_l, l = 1, \ldots, d_{\boldsymbol{\phi}}$, where

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}) \quad \phi_2(\mathbf{x}) \quad \ldots \quad \phi_{d_{\boldsymbol{\phi}}}(\mathbf{x})]^\top. \tag{5}$$

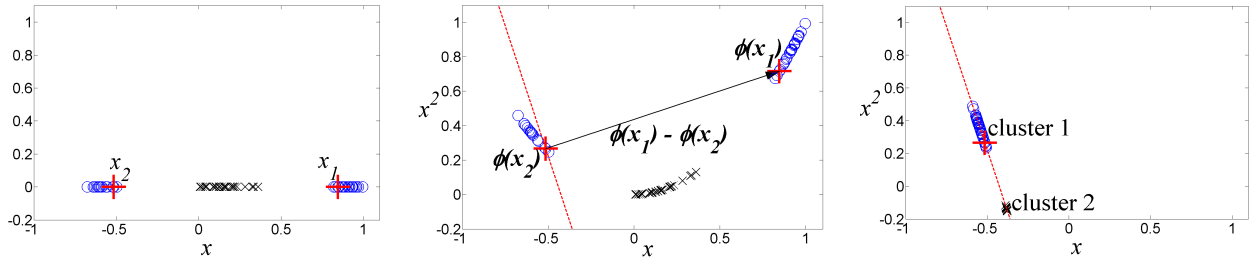


Fig. 2: Clustering with must-link constraints. (a) Input space. Red markers are the constraint pair (x_1, x_2). (b) The input space is mapped to the feature space via quadratic mapping $\phi(x) = [x \;\; x^2]^\top$. The black arrow is the constraint vector $(\phi(x_1) - \phi(x_2))^\top$, and the red dashed line is its null space. (c) The feature space is projected to the null space of the constraint vector. Constraint points collapse to a single point and are guaranteed to be clustered together. Two clusters can be easily identified.

Note that in many problems, this mapping is sufficient to achieve the desired separability between different clusters.

We first derive the mean shift procedure on the feature space $\mathcal{H}$ in terms of the explicit representation of the mapping ϕ. The point sample density estimator at $\mathbf{y} \in \mathcal{H}$, with the diagonal bandwidth matrices $h_i \mathbf{I}_{d_\phi \times d_\phi}$ is

$$f_{\mathcal{H}}(\mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^{d_\phi}} k\left(\left\| \frac{\mathbf{y} - \phi(\mathbf{x}_i)}{h_i} \right\|^2 \right). \tag{6}$$

Taking the gradient of (6) w.r.t. ϕ, like (4), the solution can be found iteratively using the mean shift procedure

$$\delta \mathbf{y} = \frac{\sum_{i=1}^{n} \frac{\phi(\mathbf{x}_i)}{h_i^{d_\phi+2}} g\left(\left\| \frac{\mathbf{y} - \phi(\mathbf{x}_i)}{h_i} \right\|^2 \right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d_\phi+2}} g\left(\left\| \frac{\mathbf{y} - \phi(\mathbf{x}_i)}{h_i} \right\|^2 \right)} - \mathbf{y}. \tag{7}$$

By employing the kernel trick, we now derive the implicit formulation of the kernel mean shift algorithm. We define $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, a positive semidefinite, scalar kernel function satisfying for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}'). \tag{8}$$

$K(\cdot)$ defines an inner product which makes it possible to map the data implicitly to a high-dimensional kernel space. Let

$$\mathbf{\Phi} = [\phi(\mathbf{x}_1) \;\; \phi(\mathbf{x}_2) \;\; \ldots \;\; \phi(\mathbf{x}_n)] \tag{9}$$

be the $d_\phi \times n$ matrix of the mapped points and $\mathbf{K} = \mathbf{\Phi}^T \mathbf{\Phi}$ be the $n \times n$ kernel (Gram) matrix. We observe that at each iteration of the mean shift procedure (7), the estimate $\bar{\mathbf{y}} = \mathbf{y} + \delta \mathbf{y}$ always lies in the column space of $\mathbf{\Phi}$. Any such point $\bar{\mathbf{y}}$ can be written as

$$\bar{\mathbf{y}} = \mathbf{\Phi} \boldsymbol{\alpha}_{\bar{\mathbf{y}}} \tag{10}$$

where $\boldsymbol{\alpha}_{\bar{\mathbf{y}}}$ is an n-dimensional weighting vector. The distance between two points $\mathbf{y}$ and $\mathbf{y}'$ in this space can be expressed in terms of their inner product and their respective weighting vectors

$$\begin{aligned} \|\mathbf{y} - \mathbf{y}'\|^2 &= \|\mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}} - \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}'}\|^2 \\ &= \boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{\Phi}^\top \mathbf{\Phi} \boldsymbol{\alpha}_{\mathbf{y}} + \boldsymbol{\alpha}_{\mathbf{y}'}^\top \mathbf{\Phi}^\top \mathbf{\Phi} \boldsymbol{\alpha}_{\mathbf{y}'} - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{\Phi}^\top \mathbf{\Phi} \boldsymbol{\alpha}_{\mathbf{y}'} \\ &= \boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K} \boldsymbol{\alpha}_{\mathbf{y}} + \boldsymbol{\alpha}_{\mathbf{y}'}^\top \mathbf{K} \boldsymbol{\alpha}_{\mathbf{y}'} - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K} \boldsymbol{\alpha}_{\mathbf{y}'}. \end{aligned} \tag{11}$$

Let $\mathbf{e}_i$ denote the i-th canonical basis vector for $\mathbb{R}^n$. Applying (11) to compute distances in (7) by using the equivalence $\phi(\mathbf{x}_i) = \mathbf{\Phi}\mathbf{e}_i$, the mean shift algorithm iteratively updates the weighting vector $\boldsymbol{\alpha}_{\mathbf{y}}$

$$\boldsymbol{\alpha}_{\bar{\mathbf{y}}} = \frac{\sum_{i=1}^{n} \frac{\mathbf{e}_i}{h_i^{d_\phi+2}} g\left(\frac{\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K} \boldsymbol{\alpha}_{\mathbf{y}} + \mathbf{e}_i^\top \mathbf{K} \mathbf{e}_i - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K} \mathbf{e}_i}{h_i^2} \right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d_\phi+2}} g\left(\frac{\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K} \boldsymbol{\alpha}_{\mathbf{y}} + \mathbf{e}_i^\top \mathbf{K} \mathbf{e}_i - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K} \mathbf{e}_i}{h_i^2} \right)}. \tag{12}$$

The clustering starts on the data points on $\mathcal{H}$, therefore the initial weighting vectors are given by $\boldsymbol{\alpha}_{\mathbf{y}_i} = \mathbf{e}_i$, such that, $\mathbf{y}_i = \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}_i} = \phi(\mathbf{x}_i)$, $i = 1 \ldots n$. At convergence, the mode $\bar{\mathbf{y}}$ can be recovered using (10) as $\mathbf{\Phi}\bar{\boldsymbol{\alpha}}_{\mathbf{y}}$. The points converging close to the same mode are clustered together, following the original proof [11]. Since any positive semidefinite matrix $\mathbf{K}$ is a kernel for some feature space [13], the derived method implicitly applies mean shift on the feature space induced by $\mathbf{K}$. Note that under this formulation, mean shift in the input space can be implemented by simply choosing the mapping function $\phi(\cdot)$ to be identity, i.e. $\phi(\mathbf{x}) = \mathbf{x}$.

An important point is that the dimensionality of the feature space d_ϕ can be very large, for example it is infinite in case of the Gaussian kernel function. In such cases it may not be possible to explicitly compute the point sample density estimator (6) and consequently the mean shift vectors (7). Since the dimensionality of the subspace spanned by the feature points is equal to $\operatorname{rank} \mathbf{K} \leq n$, it is sufficient to use the implicit form of the mean shift procedure using (12).

4 Kernel Learning Using Linear Transformations

A nonlinear mapping of the input data to a higher-dimensional kernel space often improves cluster separability. By effectively enforcing the available constraints, it is possible to transform the entire space and guide the clustering to discover the desired structure in the data.

To illustrate this intuition, we present a simple two class example from [37] in Fig. 2. The original data lies along the x-axis, with the blue points associated with one class and the black points with the second class (Fig. 2a). This data

appears to have originated from three clusters. Let (x_1, x_2) be the pair of points marked in red which are constrained to be clustered together. We map the data explicitly to a two-dimensional feature space via a simple quadratic mapping $\phi(x) = [x \quad x^2]^\top$ (Fig. 2b). Although the data is linearly separable, it still appears to form three clusters. Using the must-link constraint pair, we enforce the two red points to be clustered together. The black arrow denotes the constraint vector $\phi(x_1) - \phi(x_2)$ and the dashed red line is its null space. By projecting the feature points to the null space of the constraint vector, the points $\phi(x_1)$ and $\phi(x_2)$ are collapsed to the same point, guaranteeing their association with the same mode. From Fig.2c, we can see that in the projected space, the data has the desired cluster structure which is consistent with its class association.

This approach, although simple, does not scale well with increasing number of constraints for a simple nonlinear mapping like above. Given m linearly independent constraint vectors in a d_ϕ-dimensional space, the dimensionality of the null space of the constraint matrix is $d_\phi - m$. This implies that if d_ϕ or more constraints are specified, *all the points* collapse to a single point and are therefore grouped together in one cluster. This problem can be alleviated if a mapping function $\phi(\cdot)$ is chosen such that d_ϕ is very large. Since explicitly designing the mapping function $\phi(\mathbf{x})$ is not always practical, we use a kernel function $K(\mathbf{x}, \mathbf{x}')$ to implicitly map the input data to a very high-dimensional kernel space. As we show in Section 4.1, the subsequent projection to the null-space of the constraint vectors can also be achieved implicitly by appropriately updating the kernel matrix. In Section 4.2, we further generalize the projection operation to a linear transformation that also utilizes cannot-link constraints.

4.1 Kernel Updates Using Orthogonal Projection

Recall the matrix $\mathbf{\Phi}$ in (9), obtained by mapping the input points to $\mathcal{H}$ via the nonlinear function ϕ. Let (j_1, j_2) be a must-link constraint pair such that $\phi(\mathbf{x}_{j_1}) = \mathbf{\Phi}\mathbf{e}_{j_1}$ and $\phi(\mathbf{x}_{j_2}) = \mathbf{\Phi}\mathbf{e}_{j_2}$ are to be clustered together. Given a set of m such must-link constraint pairs $\mathcal{M}$, for every $(j_1, j_2) \in \mathcal{M}$, the d_ϕ-dimensional constraint vector can be written as $\mathbf{a}_j = \mathbf{\Phi}(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}) = \mathbf{\Phi}\mathbf{z}_j$. We refer to the n-dimensional vector $\mathbf{z}_j$ as the *indicator* vector for the j^{th} constraint. The $d_\phi \times m$ dimensional constraint matrix $\mathbf{A}$ can be obtained by column stacking all the m constraint vectors, i.e., $\mathbf{A} = \mathbf{\Phi}\mathbf{Z}$, where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m]$ is the $n \times m$ matrix of indicator vectors. Similar to the example in Fig. 2, we impose the constraints by projecting the matrix $\mathbf{\Phi}$ to the null space of $\mathbf{A}$ using the the projection matrix

$$\mathbf{P} = \mathbf{I}_{d_\phi} - \mathbf{A}\left(\mathbf{A}^\top\mathbf{A}\right)^+\mathbf{A}^\top \tag{13}$$

where $\mathbf{I}_{d_\phi}$ is the d_ϕ-dimensional identity matrix and '$+$' denotes the matrix pseudoinverse operation. Let $\mathbf{S} = \mathbf{A}^\top\mathbf{A}$ be the $m \times m$ scaling matrix. The matrix $\mathbf{S}$ can be computed using the indicator vectors and the initial kernel matrix $\mathbf{K}$ without knowing the mapping ϕ as

$$\mathbf{S} = \mathbf{A}^\top\mathbf{A} = \mathbf{Z}^\top\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{Z} = \mathbf{Z}^\top\mathbf{K}\mathbf{Z}. \tag{14}$$

Given the constraint set, the new mapping function $\widehat{\phi}(\mathbf{x})$ is computed as $\widehat{\phi}(\mathbf{x}) = \mathbf{P}\phi(\mathbf{x})$. Since all the constraints in $\mathcal{M}$ are satisfied in the projected subspace, we have

$$||\mathbf{P}\mathbf{\Phi}\mathbf{Z}||_F^2 = 0 \tag{15}$$

where $||\cdot||_F$ denotes the Frobenius norm. The initial kernel function (8) correspondingly transforms to the projected kernel function $\widehat{K}(\mathbf{x}, \mathbf{x}') = \widehat{\phi}(\mathbf{x})^\top\widehat{\phi}(\mathbf{x}')$. Using the projection matrix $\mathbf{P}$, it can be rewritten in terms of the initial kernel function $K(\mathbf{x}, \mathbf{x}')$ and the constraint matrix $\mathbf{A}$

$$\begin{aligned}\widehat{K}(\mathbf{x}, \mathbf{x}') &= \phi(\mathbf{x})^\top\mathbf{P}^\top\mathbf{P}\phi(\mathbf{x}') = \phi(\mathbf{x})^\top\mathbf{P}\phi(\mathbf{x}') \\ &= \phi(\mathbf{x})^\top(\mathbf{I}_{d_\phi} - \mathbf{A}\mathbf{S}^+\mathbf{A}^\top)\phi(\mathbf{x}') \\ &= K(\mathbf{x}, \mathbf{x}') - \phi(\mathbf{x})^\top\mathbf{A}\mathbf{S}^+\mathbf{A}^\top\phi(\mathbf{x}').\end{aligned} \tag{16}$$

Note that the identity $\mathbf{P}^\top\mathbf{P} = \mathbf{P}$ follows from the fact that $\mathbf{P}$ is a symmetric projection matrix. The m-dimensional vector $\mathbf{A}^\top\phi(\mathbf{x})$ can also be written in terms of the initial kernel function as

$$\begin{aligned}\mathbf{A}^\top\phi(\mathbf{x}) &= \mathbf{Z}^\top\mathbf{\Phi}^\top\phi(\mathbf{x}) \\ &= \mathbf{Z}^\top\left[K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_n, \mathbf{x})\right]^\top.\end{aligned} \tag{17}$$

Let the vector $\mathbf{k}_\mathbf{x} = \left[K(\mathbf{x}_1, \mathbf{x}), \ldots, K(\mathbf{x}_n, \mathbf{x})\right]^\top$. From (16), the projected kernel function can be written as

$$\widehat{K}(\mathbf{x}, \mathbf{x}') = K(\mathbf{x}, \mathbf{x}') - \mathbf{k}_\mathbf{x}^\top\mathbf{Z}\mathbf{S}^+\mathbf{Z}^\top\mathbf{k}_{\mathbf{x}'} \tag{18}$$

The projected kernel matrix can be directly computed as

$$\widehat{\mathbf{K}} = \mathbf{K} - \mathbf{K}\mathbf{Z}\mathbf{S}^+\mathbf{Z}^\top\mathbf{K} \tag{19}$$

where the symmetric $n \times n$ initial kernel matrix $\mathbf{K}$ has rank $r \leq n$. The rank of the matrix $\mathbf{S}$ is equal to the number of linearly independent constraints, and the rank of the projected kernel matrix $\widehat{\mathbf{K}}$ is $r - \text{rank } \mathbf{S}$.

4.2 Kernel Updates Using Linear Transformation

By projecting the feature points to the null space of the constraint vector $\mathbf{a}$, their components along the $\mathbf{a}$ are fully eliminated. This operation guarantees that the two points belong to the same cluster. As proposed earlier, by appropriately choosing the kernel function $K(\cdot)$ (such that d_ϕ is very large), we can make the initial kernel matrix $\mathbf{K}$ full-rank. However, a sufficiently large number of such linearly independent constraints could still result in a projected space with dimensionality that is too small for meaningful clustering.

From a clustering perspective, it might suffice to bring the two must-link constraint points sufficiently close to each other. This can be achieved through a linear transformation of the kernel space that *only* scales down the component along the constraint vector. Such a linear transformation would preserve the rank of the kernel matrix and is potentially capable of handling a very large number of must-link constraints. A similar transformation that *increases* the distance between two points also enables us to incorporate cannot-link constraints.

Given a constraint vector $\mathbf{a} = \mathbf{\Phi z}$, a symmetric transformation matrix of the form

$$\mathbf{T} = \mathbf{I}_{d_\phi} - s\left(\mathbf{a}\mathbf{a}^\top\right) \tag{20}$$

allows us to control the scaling along the vector $\mathbf{a}$ using the scaling factor s. When $s = \frac{1}{\mathbf{a}^\top\mathbf{a}}$ the transformation becomes a projection to the null space of $\mathbf{a}$. The transformation decreases distances along $\mathbf{a}$ for $0 < s < \frac{2}{\mathbf{a}^\top\mathbf{a}}$, while it is increased for $s < 0$ or $s > \frac{2}{\mathbf{a}^\top\mathbf{a}}$.

We can set a target distance $d > 0$ for the constraint point pair by applying an appropriate transformation $\mathbf{T}$. The constraint equation can be written as

$$||\mathbf{T\Phi z}||_F^2 = \mathbf{z}^\top\widehat{\mathbf{K}}\mathbf{z} = d \tag{21}$$

where $\widehat{\mathbf{K}} = \mathbf{\Phi}^\top\mathbf{T}^\top\mathbf{T\Phi} = \mathbf{\Phi}^\top\mathbf{T}^2\mathbf{\Phi}$ is the corresponding transformed kernel matrix. To handle must-link constraints, d should be small, while it should be large for cannot-link constraint pairs. Using the specified d, we can compute s and therefore the transformed kernel matrix

$$\widehat{\mathbf{K}} = \mathbf{\Phi}^\top\left(\mathbf{I}_{d_\phi} - s\,\mathbf{a}\mathbf{a}^\top\right)^2\mathbf{\Phi}. \tag{22}$$

Substituting $\mathbf{a} = \mathbf{\Phi z}$ the expression for $\widehat{\mathbf{K}}$ is

$$\widehat{\mathbf{K}} = \mathbf{K} - 2s\,\mathbf{K}\mathbf{z}\mathbf{z}^\top\mathbf{K} + s^2\left(\mathbf{z}^\top\mathbf{K}\mathbf{z}\right)\mathbf{K}\mathbf{z}\mathbf{z}^\top\mathbf{K}. \tag{23}$$

From (21), we can solve for s

$$s = \frac{1}{p}\left(1 \pm \sqrt{\frac{d}{p}}\right) \quad \text{where} \quad p = \mathbf{z}^\top\mathbf{K}\mathbf{z} > 0 \tag{24}$$

and the choice of s does not affect the following kernel update† such that the constraint (21) is satisfied

$$\widehat{\mathbf{K}} = \mathbf{K} + \beta\mathbf{K}\mathbf{z}\mathbf{z}^\top\mathbf{K}, \qquad \beta = \left(\frac{d}{p^2} - \frac{1}{p}\right). \tag{25}$$

The case when $p = 0$ implies that the constraint vector is a zero vector and $\mathbf{T} = \mathbf{I}_{d_\phi}$ and a value of $\beta = 0$ should be used.

When multiple must-link and cannot-link constraints are available, the kernel matrix can be updated iteratively for each constraint by computing the update parameter β_j using corresponding d_j and p_j. However, by enforcing the distance between constraint pairs to be *exactly* d_j, the linear transformation imposes *hard* constraints for learning the kernel matrix which could potentially have two adverse consequences:

1) The hard constraints could make the algorithm difficult (or even impossible) to converge, since previously satisfied constraints could easily get violated during subsequent updates of the kernel matrix.
2) Even when the constraints are non-conflicting in nature, in the absence of any relaxation, the method becomes sensitive to labeling errors. This is illustrated through the following example.

†. This update rule is equivalent to minimizing the $\log\det$ divergence (29) for a single equality constraint using Bregman projections (31).

The input data consists of points along five concentric circles as shown in Fig.3a. Each cluster comprises of 150 noisy points along a circle. Four labeled points per class (shown by square markers) are used to generate a set of $\binom{4}{2} \times 5 = 30$ must-link constraints. The initial kernel matrix $\mathbf{K}$ is computed using a Gaussian kernel with $\sigma = 5$, and updated by imposing the provided constraints (19). The updated kernel matrix $\widehat{\mathbf{K}}$ is used for kernel mean shift clustering (Section 3.2) and the corresponding results are shown in Fig. 3b. To test the performance of the method under labeling errors, we also add one mislabeled must-link constraint (shown in black line). The clustering performance deteriorates drastically with just one mislabeled constraint as shown in Fig. 3c.

In order to overcome these limitations, the learning algorithm must accommodate for systematic relaxation of constraints. This can be achieved by observing that the kernel update in (25) is equivalent to minimizing the $\log\det$ divergence between $\widehat{\mathbf{K}}$ and $\mathbf{K}$ subject to constraint (21) [25, Sec. 5.1.1]. In the following section, we formulate the kernel learning algorithm into a $\log\det$ minimization problem with *soft* constraints.

5 Kernel Learning Using Bregman Divergence

Bregman divergences have been studied in the context of clustering, matrix nearness and metric and kernel learning [22], [25], [1]. We briefly discuss the $\log\det$ Bregman divergence and its properties in Section 5.1. We summarize the kernel learning problem using Bregman divergence which was introduced in [25], [22] in Section 5.2.

5.1 The LogDet Bregman Divergence

The Bregman divergence [5] between real, symmetric $n \times n$ matrices $\mathbf{X}$ and $\mathbf{Y}$ is a scalar function given by

$$D_\varphi(\mathbf{X}, \mathbf{Y}) = \varphi(\mathbf{X}) - \varphi(\mathbf{Y}) - \mathrm{tr}\Big(\nabla\varphi\left(\mathbf{Y}\right)^\top(\mathbf{X} - \mathbf{Y})\Big) \tag{26}$$

where φ is a strictly convex function and ∇ denotes the gradient operator. For $\varphi(\mathbf{X}) = -\log\left(\det(\mathbf{X})\right)$, the resulting divergence is called the $\log\det$ divergence

$$D_{ld}\left(\mathbf{X}, \mathbf{Y}\right) = \mathrm{tr}\left(\mathbf{X}\mathbf{Y}^{-1}\right) - \log\det\left(\mathbf{X}\mathbf{Y}^{-1}\right) - n \tag{27}$$

and is defined when $\mathbf{X}, \mathbf{Y}$ are positive definite. In [25], this definition was extended to rank deficient (positive semidefinite) matrices by restricting the convex function to the range spaces of the matrices. For positive semidefinite matrices $\mathbf{X}$ and $\mathbf{Y}$, both having rank $r \leq n$ and singular value decomposition $\mathbf{X} = \mathbf{V\Lambda V}^\top$ and $\mathbf{Y} = \mathbf{U\Theta U}^\top$, the $\log\det$ divergence is defined as

$$D_{ld}\left(\mathbf{X}, \mathbf{Y}\right) = \sum_{i,j \leq r}\left(\mathbf{v}_i^\top\mathbf{u}_j\right)^2\left(\frac{\lambda_i}{\theta_j} - \log\frac{\lambda_i}{\theta_j} - 1\right). \tag{28}$$

Moreover, the $\log\det$ divergence, like any Bregman divergence, is convex with respect to its first argument $\mathbf{X}$ [1]. This property is useful in formulating the kernel learning as a convex minimization problem in the presence of multiple constraints.

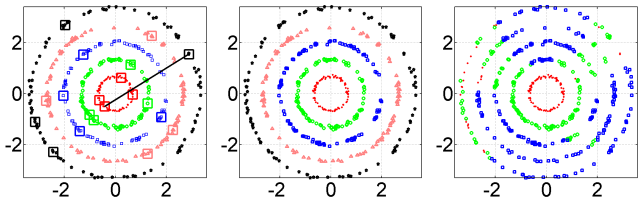


Fig. 3: Five concentric circles. Kernel learning without relaxation (a) Input data. The square markers indicate the data points used to generate pairwise constraints. The black line shows the only mislabeled similarity constraint used. (b) Clustering results when only the correctly labeled similarity constraints were used. (c) Clustering results when the mislabeled constraint was also used.

5.2 Kernel Learning with LogDet Divergences

Let $\mathcal{M}$ and $\mathcal{C}$ denote the sets of m must-link and c cannot-link pairs respectively, such that $m + c = n_c$. Let d_m and d_c be the target squared distance thresholds for must-link and cannot-link constraints respectively. The problem of learning a kernel matrix using linear transformations for multiple constraints, discussed in Section 4.2, can be equivalently formulated as the following constrained $\log\det$ minimization problem

$$\begin{aligned} \min_{\widehat{\mathbf{K}}} \quad & D_{ld}\left(\widehat{\mathbf{K}}, \mathbf{K}\right) \\ \text{s.t.} \quad & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) = d_m \quad \forall (j_1, j_2) \in \mathcal{M} \\ & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) = d_c \quad \forall (j_1, j_2) \in \mathcal{C}. \end{aligned} \tag{29}$$

The final kernel matrix $\widehat{\mathbf{K}}$ is obtained by iteratively updating the initial kernel matrix $\mathbf{K}$.

In order to permit relaxation of constraints, we rewrite the learning problem using a *soft margin* formulation similar to [22], where each constraint pair (j_1, j_2) is associated with a slack variable $\widehat{\xi}_j,\ j = 1, \ldots, n_c$

$$\begin{aligned} \min_{\widehat{\mathbf{K}}, \widehat{\boldsymbol{\xi}}} \quad & D_{ld}\left(\widehat{\mathbf{K}}, \mathbf{K}\right) + \gamma D_{ld}\left(\operatorname{diag}\left(\widehat{\boldsymbol{\xi}}\right), \operatorname{diag}\left(\boldsymbol{\xi}\right)\right) \\ \text{s.t.} \quad & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) \leq \widehat{\xi}_j \quad \forall (j_1, j_2) \in \mathcal{M} \\ & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) \geq \widehat{\xi}_j \quad \forall (j_1, j_2) \in \mathcal{C}. \end{aligned} \tag{30}$$

The n_c-dimensional vector $\widehat{\boldsymbol{\xi}}$ is the vector of slack variables and $\boldsymbol{\xi}$ is the vector of target distance thresholds d_m and d_c. The regularization parameter γ controls the trade-off between fidelity to the original kernel and the training error. Note that by changing the equality constraints to inequality constraints, the algorithm allows must-link pairs to be closer and cannot-link pairs to be farther than their corresponding distance thresholds.

The optimization problem in (30) is solved using the method of Bregman projections [5]. A Bregman projection (not necessarily orthogonal) is performed to update the current matrix, such that the updated matrix satisfies that constraint. In each iteration, it is possible that the current update violates a previously satisfied constraint. Since the problem in (30) is convex [25], the algorithm converges to the global minimum after repeatedly updating the kernel matrix for each constraint in the set $\mathcal{M} \cup \mathcal{C}$.

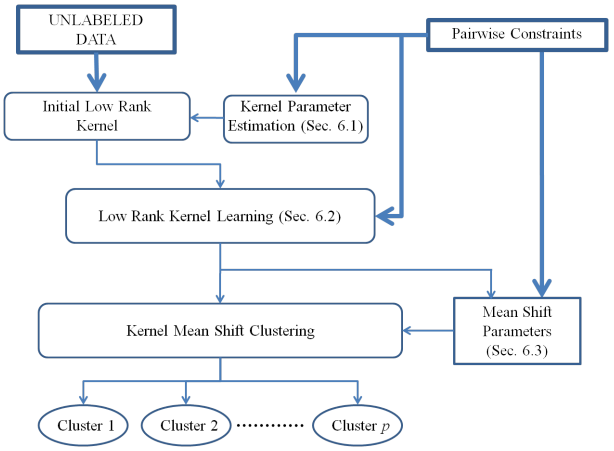


Fig. 4: Block diagram describing the semi-supervised kernel mean shift clustering algorithm. The bold boxes indicate the user input to the algorithm.

For the $\log\det$ divergence, the Bregman projection that minimizes the objective function in (30) for a given constraint $(j_1, j_2) \in \mathcal{M} \cup \mathcal{C}$, can be written as derived in [25]

$$\widehat{\mathbf{K}}_{t+1} = \widehat{\mathbf{K}}_t + \beta_t \widehat{\mathbf{K}}_t \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}}_t. \tag{31}$$

For the t^{th} iteration the parameter β_t is computed in closed form as explained in the supplementary material. The algorithm converges when β_t approaches zero for all $(j_1, j_2) \in \mathcal{M} \cup \mathcal{C}$ with the final learned kernel matrix $\widehat{\mathbf{K}} = \widehat{\boldsymbol{\Phi}}^\top \widehat{\boldsymbol{\Phi}}$.

Using $\widehat{\mathbf{K}}$, the kernel function that defines the inner product in the corresponding transformed kernel space can be written as

$$\widehat{K}\left(\mathbf{x}, \mathbf{y}\right) = K\left(\mathbf{x}, \mathbf{y}\right) + \mathbf{k}_\mathbf{x}^\top \left(\mathbf{K}^+ \left(\widehat{\mathbf{K}} - \mathbf{K}\right) \mathbf{K}^+\right) \mathbf{k}_\mathbf{y} \tag{32}$$

where $K(\cdot)$ is the scalar initial kernel function (8), and the vectors $\mathbf{k}_\mathbf{x} = [K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n)]^\top$ and $\mathbf{k}_\mathbf{y} = [K(\mathbf{y}, \mathbf{x}_1), \ldots, K(\mathbf{y}, \mathbf{x}_n)]^\top$ [22]. The points $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ could be *out of sample* points, i.e., points that are not in the sample set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ used to learn $\widehat{\mathbf{K}}$. Note that (18) also generalizes the inner product in the projected kernel space to out of sample points.

6 Semi-supervised Kernel Mean-Shift Clustering Algorithm

We present the complete algorithm for semi-supervised kernel mean shift clustering (SKMS) in this section. Fig. 4 shows a block diagram for the overall algorithm. We explain each of the modules using two examples: *Olympic circles*, a synthetic data set and a 1000 sample subset of *USPS digits*, a real data set with images of handwritten digits. In Section 6.1, we propose a method to select the scale parameter σ for the initial Gaussian kernel function using the sets $\mathcal{M}, \mathcal{C}$ and target distances d_m, d_c. We show the speed-up achieved by performing the low-rank kernel matrix updates (as opposed to updating the full kernel matrix) in Section 6.2. For mean shift clustering, we present a strategy to automatically select the bandwidth parameter using the pairwise must-link constraints in Section 6.3. Finally, in Section 6.4, we discuss the selection of the trade-off parameter γ.

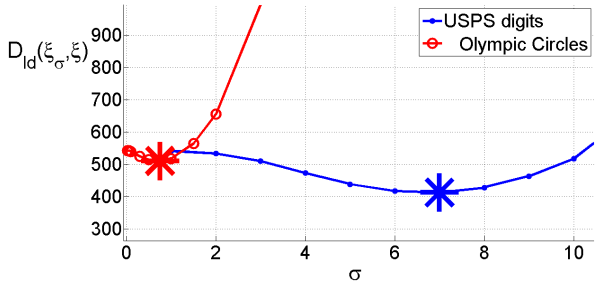


Fig. 5: Selecting the scale parameter $\hat{\sigma}$ that minimizes $D_{ld}(\boldsymbol{\xi}, \boldsymbol{\xi}_\sigma)$ using grid search. Selected σ values: Olympic circles, $\hat{\sigma} = 0.75$; USPS digits, $\hat{\sigma} = 7$.

6.1 Initial Parameter Selection

Given two input sample points $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, the Gaussian kernel function is given by

$$K_\sigma(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \in [0, 1] \qquad (33)$$

where σ is the scale parameter. From (33) and (11) it is evident that pairwise distances between sample points in the feature space induced by $K_\sigma(\cdot)$ lies in the interval $[0, 2]$. This provides us with an effective way of setting up the target distances $d_m = \min(d_1, 0.05)$ and $d_c = \max(d_{99}, 1.95)$, where d_1 and d_{99} are the 1^{st} and 99^{th} percentile of distances between *all* pairs of points in the kernel space. We select the scale parameter σ such that, in the initial kernel space, distances between must-link points are small while those between cannot-link points are large. This results in good regularization and faster convergence of the learning algorithm.

Recall that $\boldsymbol{\xi}$ is the n_c-dimensional vector of target squared distances between the constraint pairs

$$\xi_j = \begin{cases} d_m & \forall (j_1, j_2) \in \mathcal{M} \\ d_c & \forall (j_1, j_2) \in \mathcal{C}. \end{cases} \qquad (34)$$

Let $\boldsymbol{\xi}_\sigma$ be the vector of distances computed for the n_c constraint pairs using the kernel matrix $\mathbf{K}_\sigma$. The scale parameter that minimizes the $\log\det$ divergence between the vectors $\boldsymbol{\xi}$ and $\boldsymbol{\xi}_\sigma$ is

$$\hat{\sigma} = \arg\min_{\sigma \in \mathcal{S}} \; D_{ld}\left(\operatorname{diag}(\boldsymbol{\xi}), \operatorname{diag}(\boldsymbol{\xi}_\sigma)\right) \qquad (35)$$

and the corresponding kernel matrix is $\mathbf{K}_{\hat{\sigma}}$. The kernel learning is insensitive to small variations in $\hat{\sigma}$, thus it is sufficient to do a search over a *discrete* set $\mathcal{S}$. The elements of the set $\mathcal{S}$ are roughly the centers of equal probability bins over the range of all pairwise distances between the input sample points $\mathbf{x}_i, i = 1, \ldots, n$. For Olympic circles, we used $\mathcal{S} = \{0.025, 0.05, 0.1, 0.3, 0.5, 0.75, 1, 1.5, 2, 3, 5\}$ and for USPS digits, $\mathcal{S} = \{1, 2, \ldots, 10, 15, 20, 25\}$. A similar set can be automatically generated by mapping a uniform probability grid (over $0-1$) via the inverse cdf of the pairwise distances. Fig. 5 shows the plot of the objective function in (35) against different values of σ for the *Olympic circles* data set and the *USPS digits* data set.

6.2 Low Rank Kernel Learning

When the initial kernel matrix has rank $r \leq n$, the $n \times n$ matrix updates (31) can be modified to achieve a significant computational speed-up [25] (see supplementary material for the low-rank kernel learning algorithm). We learn a kernel matrix for clustering the two example data sets: Olympic circles (5 classes, $n = 1500$) and USPS digits (10 classes, $n = 1000$). The must-link constraints are generated using 15 labeled points from each class: 525 for Olympic circles and 1050 for USPS digits, while an equal number of cannot-link constraints is used.

The $n \times n$ initial kernel matrix $\mathbf{K}_{\hat{\sigma}}$ is computed as described in the previous section. Using singular value decomposition (SVD), we compute an $n \times n$ low-rank kernel matrix $\mathbf{K}$ such that $\operatorname{rank}\mathbf{K} = r \leq n$ and $\frac{\|\mathbf{K}\|_F}{\|\mathbf{K}_{\hat{\sigma}}\|_F} \geq 0.99$. For Olympic circles, this results in a rank 58 approximation of the 1500×1500 matrix $\mathbf{K}$ leading to a computational speed-up from 323.3 secs. to 0.92 secs. In case of USPS digits, the 1000×1000 matrix $\mathbf{K}$ has rank 499 and the run time reduces from 151.1 secs. to 68.2 secs. We also observed that in all our experiments in Section 7, the clustering performance did not deteriorate significantly when the low-rank approximation of the kernel matrix was used.

6.3 Setting the Mean Shift Parameters

For kernel mean shift clustering, we define the bandwidth for each point as the distance to its k^{th} nearest neighbor. In general, clustering is an interactive process where the bandwidth parameter for mean shift is provided by the user, or is estimated based on the desired number of clusters. In this section we propose an automatic recommendation method for the bandwidth parameter k by using *only* the must-link constraint pairs.

We build upon the intuition that in the transformed kernel space, the neighborhood of a must-link pair comprises of points similar to the constraint points. Given the j^{th} constraint pair $(j_1, j_2) \in \mathcal{M}$, we compute the pairwise distances in the transformed kernel space between the first constraint point $\widehat{\mathbf{\Phi}}\mathbf{e}_{j_1}$ and all other feature points as $d_i = (\mathbf{e}_{j_1} - \mathbf{e}_i)^\top \widehat{\mathbf{K}}(\mathbf{e}_{j_1} - \mathbf{e}_i),\ i = 1, \ldots, n,\ i \neq j_1$. These points are then sorted in the increasing order of d_i. The bandwidth parameter k_j for the j^{th} constraint corresponds to the index of j_2 in this sorted list. Therefore, the point $\widehat{\mathbf{\Phi}}\mathbf{e}_{j_2}$ is the k_j^{th} neighbor of $\widehat{\mathbf{\Phi}}\mathbf{e}_{j_1}$. Finally, the value of k is selected as the median over the set $\{k_j, j = 1, \ldots, m\}$.

For a correct choice of k, we expect the performance of mean shift to be insensitive to small changes in the value of k. In Fig. 6, we plot the number of clusters recovered by kernel mean shift as the bandwidth parameter is varied. For learning the kernel, we used 5 points from each class to generate must-link constraints: 50 for the Olympic circles and 100 for USPS digits. An equal number of cannot-link constraints is used. Fig. 6a shows the plot for Olympic circles (5 classes), where the median based value ($k = 6$) underestimates the bandwidth parameter. A good choice is

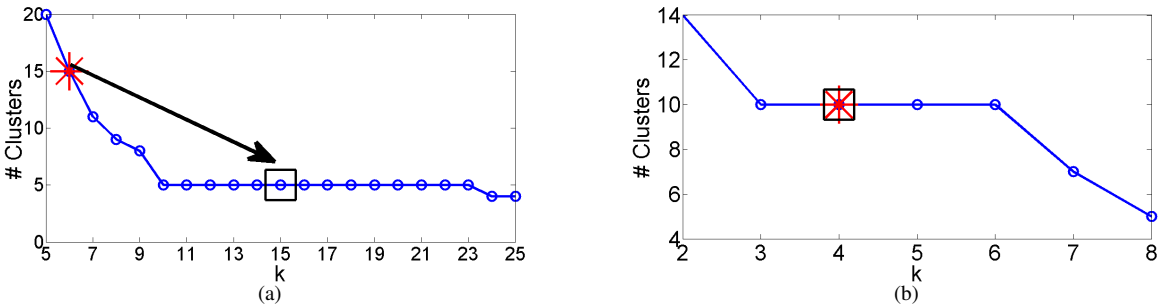


Fig. 6: Selecting the mean shift bandwidth parameter k, given five labeled points per class. (a) Olympic circles. Number of recovered clusters is sensitive at the median based estimate $k = 6$, but not at $k = 15$ (see text). (b) USPS digits. Number of recovered clusters is not sensitive at the median based estimate $k = 4$. The asterisk indicates the median based estimate, while the square marker shows a good estimate of k.

$k = 15$, which lies in the range (10–23) where the clustering output is insensitive to changes in k.

As seen in Fig. 6b, in case of USPS digits (10 classes), the number of clusters recovered by mean shift is insensitive to the median based estimate ($k=4$). For all other data sets we used in our experiments, the median based estimates produced good clustering output. Therefore, with the exception of Olympic circles, we used the bandwidth parameter obtained using the median based approach. However, for completeness, the choice of k should be verified, and corrected if necessary, by analyzing the sensitivity of the clustering output to small perturbations in k.

For computational efficiency, we run the mean shift algorithm in a lower dimensional subspace spanned by the singular vectors corresponding to at most the 25 largest singular values of $\widehat{\mathbf{K}}$. For all the experiments in Section 7, a 25-dimensional representation of the kernel matrix was large enough to represent the data in the kernel space.

6.4 Selecting the Trade-off Parameter

The trade-off parameter γ is used to weight the objective function for the kernel learning. We select γ by performing a two-fold cross-validation over different values of γ and the clustering performance is evaluated using the scalar measure Adjusted Rand (AR) index [20]. The kernel matrix is learned using half of the data with 15 points used to generate the pairwise constraints, 525 must-link constraints for the Olympic circles and 1050 for the USPS digits. An equal number of cannot-link constraint pairs is also generated.

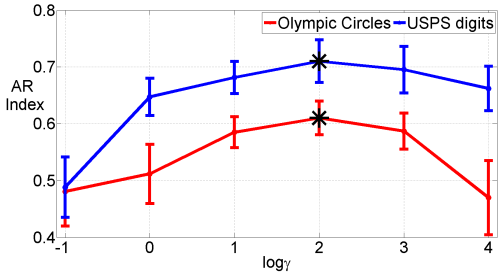


Fig. 7: Selecting the trade-off parameter γ. The AR index vs $\log \gamma$ for 50 cross-validation runs. The asterisks mark the selected value of $\gamma = 100$.

Input:
$\mathbf{X}$ - $n \times d$ data matrix
$\mathcal{M}, \mathcal{C}$ - similarity and dissimilarity constraint sets
$\mathcal{S}$ - set of possible σ values
γ - the trade-off parameter
Procedure:

- Initialize slack variables
  $\hat{\xi}_j = \xi_j = d_m$ for $(j_1, j_2) \in \mathcal{M}$
  $\hat{\xi}_j = \xi_j = d_c$ for $(j_1, j_2) \in \mathcal{C}$
- Select $\hat{\sigma} \in \mathcal{S}$ by minimizing (35) and compute initial Gaussian kernel matrix $\mathbf{K}_{\hat{\sigma}}$
- Compute $n \times n$ low-rank kernel matrix $\mathbf{K}$ such that $\operatorname{rank} \mathbf{K} = r \leq n$ and $\frac{\|\mathbf{K}\|_F}{\|\mathbf{K}_{\hat{\sigma}}\|_F} \geq 0.99$
- Learn $\widehat{\mathbf{K}}$ using $\log\det$ divergences (See suppl. material)
- Using $\widehat{\mathbf{K}}$ and $\mathcal{M}$, estimate bandwidth parameter k
- Compute bandwidths h_i as the k-th smallest distance from the point using $\widehat{\mathbf{K}}$ and (11), $d_{\hat{\Phi}} = \operatorname{rank} \widehat{\mathbf{K}}$
- Repeat for all data points $i = 1, \ldots, n$
  - Initialize $\bar{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_{\mathbf{y}} = \mathbf{e}_i$
  - Update $\bar{\boldsymbol{\alpha}}_i$ using $\widehat{\mathbf{K}}$ in (12) until convergence to local mode
  - Group data points $\mathbf{x}_i$ and $\mathbf{x}_j$ together if $\bar{\boldsymbol{\alpha}}_i \widehat{\mathbf{K}} \bar{\boldsymbol{\alpha}}_i + \bar{\boldsymbol{\alpha}}_j \widehat{\mathbf{K}} \bar{\boldsymbol{\alpha}}_j - 2\bar{\boldsymbol{\alpha}}_i \widehat{\mathbf{K}} \bar{\boldsymbol{\alpha}}_j = 0$
- Return cluster labels

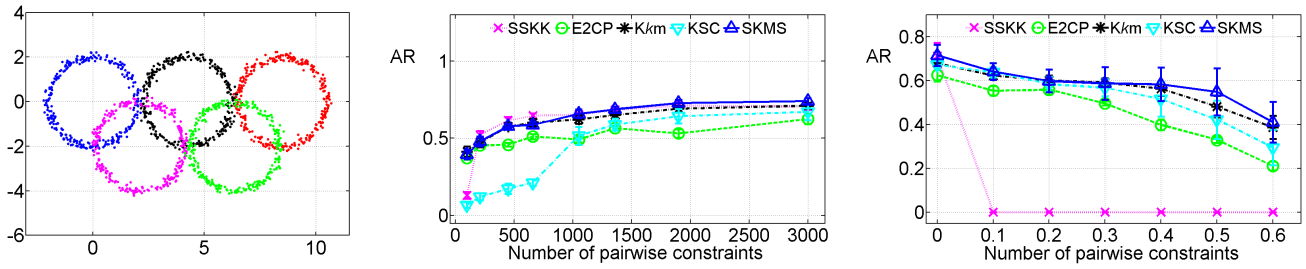
Fig. 8: Semi-supervised kernel mean shift algorithm (SKMS).

Each cross-validation step involves learning the kernel matrix with the specified γ and clustering the testing subset using the kernel mean shift algorithm and the transformed kernel function (32). Fig. 7 shows the average AR index plotted against $\log\ \gamma$ for the two data sets. In both the examples an optimum value of $\gamma = 100$ was obtained. In general, this value may not be optimum for other applications. However, in all our experiments in Section 7, we use $\gamma = 100$, since we obtained similar curves with small variations in the AR index in the vicinity of $\gamma = 100$.

Fig. 8 shows a step-by-step summary of the semi-supervised kernel mean shift (SKMS) algorithm.

7 Experiments

We show the performance of semi-supervised kernel mean shift (SKMS) algorithm on two synthetic examples and four real-world examples. We also compare our method with two state-of-the-art methods: the semi-supervised kernel k-means (SSKK) [24] and the constrained spectral clustering (E2CP) [29]. In the past, the superior performance of



(a) (b) (c)

Fig. 9: Olympic circles. (a) Input data. (b) AR index as the number of pairwise constraints is varied. (c) AR index as the fraction of mislabeled constraints is varied.

E2CP over other recent methods has been demonstrated. Please see [29] for more details. In addition to this, we also compare SKMS with the kernelized k-means (Kkm) and kernelized spectral clustering (KSC) algorithms. The learned kernel matrix is used to compute distances in SKMS and Kkm, while KSC uses it as the affinity matrix. By providing the same learned kernel matrix to the three algorithms, we compare the clustering performance of mean shift with that of k-means and spectral clustering. Note that, unlike Kkm and KSC, the methods SSKK and E2CP do not use the learned kernel matrix and supervision is supplied with alternative methods. With the exception of SKMS, all the methods require the user to provide the correct number of clusters.

Comparison metric. The clustering performance of different algorithms is compared using the Adjusted Rand (AR) index [20]. It is an adaptation of the rand index that penalizes random cluster assignments, while measuring the agreement between the clustering output and the *ground truth* labels. The AR index is a scalar and takes values between zero and one, with perfect clustering yielding a value of one.

Experimental setup. To generate pairwise constraints, we randomly select b labeled points from each class. All possible must-link constraints are generated for each class such that $m = \binom{b}{2}$ and a subset of all cannot-link constraint pairs is selected at random such that $c = m = \frac{n_c}{2}$. For each experimental setting, we average the clustering results over 50 independent runs, each with randomly selected pairwise constraints.

The initial kernel matrix is computed using a Gaussian kernel (33) for all the methods. We hand-picked the scale parameter σ for SSKK and E2CP from a wide range of values such that their final clustering performance on each data set was maximized. For Kkm, KSC and SKMS, the values of σ and the target distances d_m and d_c are estimated as described in Section 6.1. Finally, the mean shift bandwidth parameter k, is estimated as described in Section 6.3.

For each experiment, we specify the scale parameter σ we used for SSKK and E2CP. For Kkm, KSC and SKMS, σ is chosen using (35) and the most frequent value is reported. We also list the range of k, the bandwidth parameter for SKMS for each application. For the $\log\det$ divergence based kernel learning, we set the maximum number of iterations to 100000 and the trade-off parameter γ to 100.

7.1 Synthetic Examples

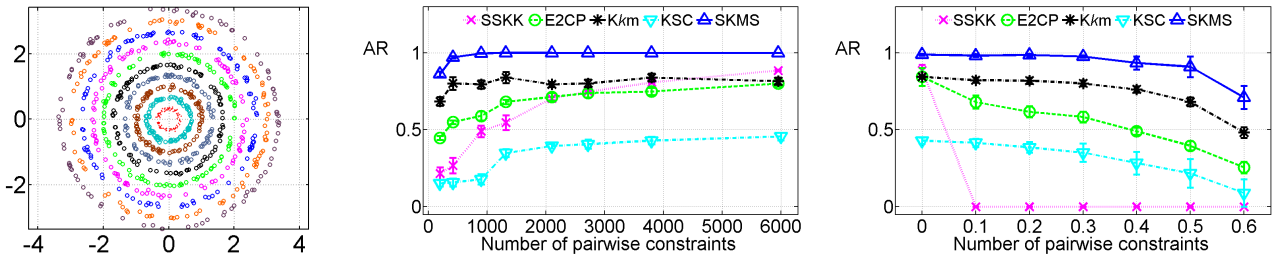
Olympic Circles. As shown in Fig. 9a, the data consists of noisy points along five intersecting circles each comprising 300 points. For SSKK and E2CP algorithms, the value of the initial kernel parameter σ was 0.5. For Kkm, KSC and SKMS the most frequently selected σ was 0.75 and the range of the bandwidth parameter k was $15-35$.

We performed two sets of experiments. In the first experiment, the performance of all the algorithms is compared by varying the total number of pairwise constraints. The number of labeled points per class vary as $\{5, 7, 10, 12, 15, 17, 20, 25\}$ and are used to generate must-link and cannot-link constraints that vary between 100 and 3000. Fig. 9b demonstrates SKMS performs better than E2CP and KSC while its performance is similar to SSKK and Kkm. For 100 constraints, SKMS recovered $5-8$ clusters, making a mistake 22% of the times. For all other settings together, it recovered an incorrect number of clusters $(4-6)$ only 6.3% of the times.

In the second experiment, we introduced labeling errors by randomly swapping the labels of a fraction of the pairwise constraints. We use 20 labeled sample points per class to generate 1900 pairwise constraints and vary the fraction of mislabeled constraints between 0 and 0.6 in steps of 0.1. Fig. 9c shows the clustering performance of all the methods. The performance of SKMS degrades only slightly even when half the constraint points are labeled wrongly, but it starts deteriorating when 60% of the constraint pairs are mislabeled.

Concentric Circles. The data consists of ten concentric circles each comprising 100 noisy points (Fig. 10a). For Kkm, KSC and SKMS the most frequently selected σ was 1 and the range of the bandwidth parameter for SKMS was $25-45$. Both algorithms, SSKK and E2CP have the value of $\sigma = 0.2$. In the first experiment, we vary the number of labeled points per class between 5 and 25 as in the previous example to generate pairwise constraints between 200 and 6000. For 200 constraints, SKMS incorrectly recovered nine clusters 50% of the times, while for all the other settings it correctly detected 10 clusters every time.

In the second experiment, we use 25 labeled points to generate 6000 pairwise constraints and mislabeled a



(a) (b) (c)

Fig. 10: Ten concentric circles. (a) Input data. (b) AR index as the number of pairwise constraints is varied. (c) AR index as the fraction of mislabeled constraints is varied.

fraction of randomly selected constraints. This mislabeled fraction was varied between 0 to 0.6 in steps of 0.1 and the performance of all the algorithms in the two experiments is shown in Fig. 10b-c.

7.2 Real-World Applications

In this section, we demonstrate the performance of our method on two real applications having a small number of classes; USPS digits: 10 classes and MIT scene: 8 classes; and two real applications with a large number of classes; PIE faces: 68 classes and Caltech-101 subset: 50 classes. We also show the performance of SKMS while clustering out of sample points using (32) on the USPS digits and the MIT scene data sets. Since the sample size per class is much smaller for PIE faces and Caltech-101 subset, results for generalization are not shown. We observe that the superiority of SKMS over other competing algorithms is clearer when the number of clusters is large.

USPS Digits. The USPS digits data set is a collection of 16×16 grayscale images of natural handwritten digits and is available from `http://cs.nyu.edu/ roweis/data.html`. Each class contains 1100 images of one of the ten digits. Fig. 11 shows sample images from each class. Each image is then represented with a 256-dimensional vector where the columns of the image are concatenated. We vary the number of labeled points per class between 5 and 25 as in the previous example to generate pairwise constraints from 200 to 6000. The maximum number of labeled points per class used comprises only 2.27% of the total data.

Fig. 11: Sample images from the USPS digits data set.

Since the whole data set has 11000 points, we select 100 points from each class at random to generate a 1000 sample subset. The labeled points are selected at random from this subset for learning the 1000×1000 kernel matrix. The value of σ used was 5 for SSKK and 2 for E2CP. For Kkm, KSC and SKMS the most frequently selected σ was 7 and the range of the bandwidth parameter k was $4-14$.

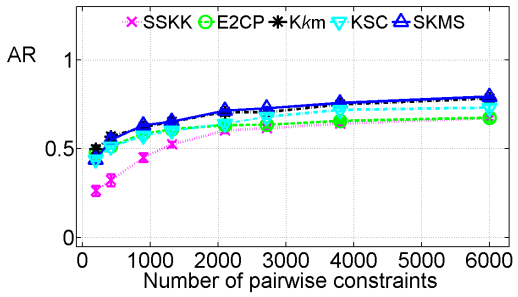
In the first experiment we compare the performance of all the algorithms on this subset of 1000 points. Fig. 12a shows the clustering performance of all the methods. Once the number of constraints were increased beyond 500, SKMS outperformed the other algorithms. For 200 constraints, the SKMS discovered $9-11$ clusters, making a mistake 18% of the times, while it recovered exactly 10 clusters in all the other cases.

In the second experiment, we evaluated the performance of SKMS for the entire data set using 25 labeled points per class. The AR index averaged over 50 runs was 0.7529 ± 0.0510. Note that from Fig. 12a it can be observed that there is only a marginal decrease in clustering accuracy. The pairwise distance matrix (PDM) after performing mean shift clustering is shown in Fig. 12b. For illustration purpose, the data is ordered such that the images belonging to the same class appear together. The block diagonal structure indicates good generalization of SKMS for out of sample points with little confusion between classes. Neither SSKK nor E2CP could be generalized to out of sample points because these methods need to learn a new kernel or affinity matrix (11000×11000).

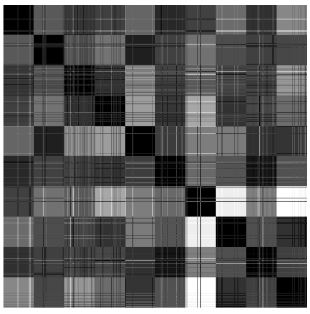
MIT Scene. The data set is available from MIT `http://people.csail.mit.edu/torralba/code/spatialenvelope/` and contains 2688 labeled images. Each image is 256×256 pixels in size and belongs to one of the eight outdoor scene categories, four natural and four man-made. Fig. 13 shows one example image from each of the eight categories. Using the code provided with the data, we extracted the GIST descriptors [32] which were then used to test all the algorithms. We vary the number of labeled points per class like in the previous example, but only between 5 and 20 to generate the pairwise constraints from 160 to 3040. The maximum number of labeled points used comprises only 7.44% of the total data.

We select 100 points at random from each of the eight classes to generate the 800×800 initial kernel matrix. The pairwise constraints are obtained from the randomly chosen labeled points from each class. The value of σ used was 1 for SSKK and 0.5 for E2CP. For Kkm, KSC and SKMS the most frequently selected σ was 1.75 and the range of the bandwidth parameter k was $4-14$.

Fig. 14a shows the clustering performance of all the algorithms as the number of constraint points are varied.

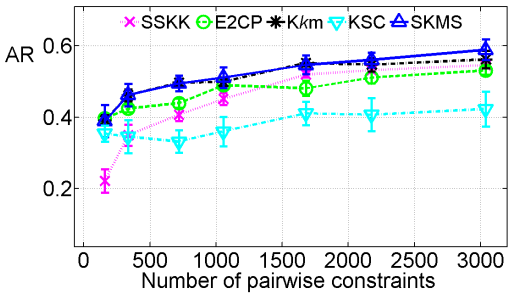


(a)

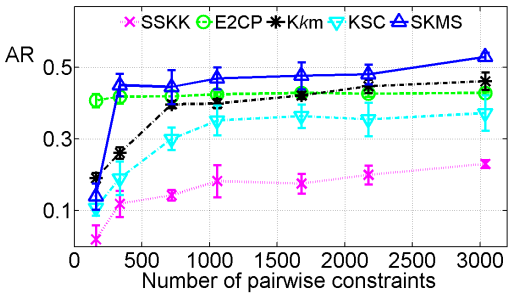


(b)

Fig. 12: USPS digits. (a) AR index as the number of pairwise constraints is varied. (b) The 11000×11000 pairwise distance matrix (PDM) after performing mean shift clustering.



(a)



(b)

Fig. 14: MIT Scene data set. (a) AR index on the 800×800 kernel matrix as the number of pairwise constraints is varied. (b) AR index on *entire data* as the number of pairwise constraints is varied.

For 160 constraint pairs, SKMS incorrectly discovered 7–10 clusters about 22% of the time, while it correctly recovered eight clusters in all other settings.

In the second experiment, the whole data set was clustered using the 800×800 learned kernel matrix and generalizing to the out of sample points (32). Both SSKK and E2CP were used to learn the full 2688×2688 kernel and affinity matrix respectively. Fig. 14b shows the performance of all the algorithms as the number of pairwise constraints was varied. Note that in [29] for similar experiments, the superior performance of E2CP was probably because of the use of spatial Markov kernel instead of the Gaussian kernel.

PIE Faces. From the CMU PIE face data set [35], we use only the frontal pose and neutral expression of all 68 subjects under 21 different lighting conditions. We coarsely aligned the images with respect to eye and mouth locations and resized them to be 128×128. In Fig.15, we show eight illumination conditions for three different subjects. Due to significant illumination variation, interclass variability is very large and some of the samples from different subjects appear to be much closer to each other than within classes.

We convert the images from color to gray scale and normalize the intensities between zero and one. Each image is then represented with a 16384-dimensional vector where the columns of the image are concatenated. We vary the number of labeled points per class as $\{3, 4, 5, 6, 7\}$ to generate pairwise constraints between 408 and 2856. The maximum number of labeled points comprises 30% of the total data.

We generate the 1428×1428 initial kernel matrix using all the data. The value of σ used was 10 for SSKK and 22 for E2CP. For Kkm, KSC and SKMS the most frequently selected σ was 25 and the range of the bandwidth parameter k was $4 - 7$. Fig. 16 shows the performance comparison and it can be observed that SKMS outperforms all the other algorithms. Note that SKMS approaches near perfect clustering for more than 5 labeled points per class. When three labeled points per class were used, the SKMS

Fig. 13: Sample images from each of the eight categories of the MIT scene data set.

Fig. 15: PIE faces data set. Sample images showing eight different illuminations for three subjects.

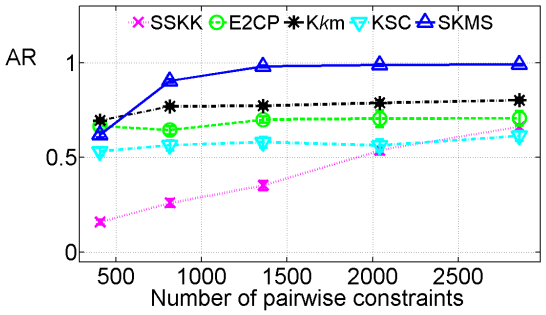


Fig. 16: PIE Faces data set. AR index as the number of pairwise constraints is varied.

discovered $61-71$ clusters, making a mistake about 84% of the time. For all other settings, SKMS correctly recovered 68 clusters about 62% of the times, while it recovered 67 clusters about 32% of the time and between $69-71$ clusters in the remaining runs. The Kkm and KSC methods perform poorly inspite of explicitly using the number of clusters and the same learned kernel matrix as SKMS.

Caltech-101 Objects. The Caltech-101 data set [16] is a collection of variable sized images across 101 object categories. This is a particularly hard data set with large intraclass variability. Fig. 17 shows sample images from eight different categories.

We randomly sampled a subset of 50 categories, as listed in Table 1, with each class containing 31 to 40 samples. For each sample image, we extract GIST descriptors [32] and use them for evaluation of all the competing clustering algorithms. We vary the number of labeled points per class as $\{5, 7, 10, 12, 15\}$ to generate pairwise constraints between 500 and 10500. The maximum number of labeled points comprises 38% of the total data. We use a larger number of constraints in order to overcome the large variability in the data set.

We generate the 1959×1959 initial kernel matrix using all the data. The value of σ used is 0.2 for E2CP and 0.3 for SSKK. For Kkm, KSC and SKMS the most frequently selected σ was 0.5 and the range of the bandwidth parameter k was $4-11$. Fig. 18 shows the comparison of SKMS with the other competing algorithms. It can be seen that SKMS outperforms all the other methods. For five labeled points per class, SKMS detected $50-52$ clusters, making mistakes 75% of the times. For all the other settings together, SKMS recovered the incorrect number $(48-51)$ of clusters only

Fig. 17: Sample images from eight of the 50 classes used from Caltech-101 data set.

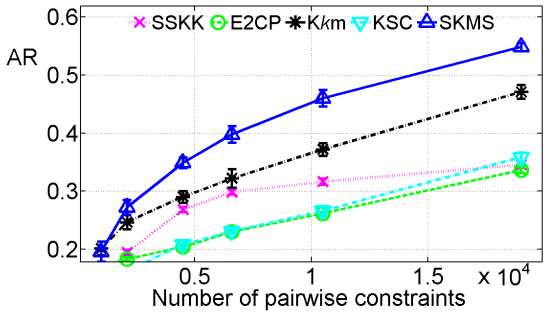


Fig. 18: Caltech-101 data set. AR index as the number of pairwise constraints is varied.

9% of the times.

A tabular summary of all comparisons across different data sets is provided in the supplementary material.

8 Discussion

We presented the semi-supervised kernel mean shift (SKMS) clustering algorithm where the inherent structure of the data points is learned using a few user supplied pairwise constraints. The data is nonlinearly mapped to a higher-dimensional kernel space where the constraints are effectively imposed by applying a linear transformation. This transformation is learned by minimizing a $\log\det$ Bregman divergence between the initial and the learned kernels. The method also estimates the parameters for the mean shift algorithm and recovers an *unknown* number of clusters automatically. We evaluate the performance of SKMS on challenging real and synthetic data sets and compare it with state-of-the-art methods.

We compared the SKMS algorithm with kernelized k-means (Kkm) and kernelized spectral clustering (KSC) algorithms, which used the same learned kernel matrix. The linear transformation applied to the initial kernel space imposes soft distance (inequality) constraints, and may result in clusters that have very different densities in the transformed space. This explains the relatively poor performance of KSC in most experiments because spectral clustering methods perform poorly when the clusters have significantly different densities [30]. The k-means method is less affected by cluster density, but it is sensitive to initialization, shape of the clusters and *outliers*, i.e., sparse points far from the cluster center.

Unlike the other methods, mean shift clustering does not need the number of clusters as input and can identify clusters of different shapes, sizes and density. Since locality is imposed by the bandwidth parameter, mean shift is more robust to outliers. As shown in our experiments, this advantage gets pronounced when the number of clusters in the data is large.

Clustering, in general, becomes very challenging when the data has large intra-class variability. For example, Fig. 19 shows three images from the *highway* category of the MIT scene data set that were misclassified. The misclassification error rate in Caltech-101 data set was even higher. On large scale data sets with over 10000

TABLE 1: Object classes used from Caltech-101 data set.

elephant	flamingo_head	emu	faces	gerenuk	stegosaurus	accordion	ferry	cougar_face	mayfly
chair	scissors	menorah	platypus	butterfly	tick	metronome	inline_skate	bass	pyramid
leopards	sea_horse	cougar_body	stop_sign	lotus	dalmatian	gramophone	camera	trilobite	dragonfly
grand_piano	headphone	sunflower	ketch	wild_cat	crayfish	nautilus	buddha	yin_yang	dolphin
minaret	anchor	car_side	rooster	wheelchair	octopus	joshua_tree	ant	umbrella	crocodile

Fig. 19: Three images from the class *highway* of the MIT Scene data set that were misclassified.

categories [14], all methods typically perform poorly, as an image may qualify for multiple categories. For example, the images in Fig. 19 were classified in the *street* category, which is semantically correct. To deal with a large number of categories, clustering (classification) algorithms should incorporate the ability to use higher level semantic features that connect an image to its possible categories.

The code for SKMS is written in MATLAB and C and is available for download at `http://coewww.rutgers.edu/riul/research/code/SKMS/index.html`

REFERENCES

[1] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[2] S. Basu, M. Bilenko, and R. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. 10th Intl. Conf. on Knowledge Discovery and Data Mining*, 2004.

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

[4] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Intl. Conf. Machine Learning*, pages 81–88, 2004.

[5] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[6] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, 2006.

[7] H. Chen and P. Meer. Robust fusion of uncertain information. *IEEE Sys. Man Cyber. Part B*, 35:578–586, 2005.

[8] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE PAMI*, 17:790–799, 1995.

[9] R. Collins. Mean shift blob tracking through scale space. In *CVPR*, volume 2, pages 234–240, 2003.

[10] D. Comaniciu. Variable bandwidth density-based fusion. In *CVPR*, volume 1, pages 56–66, 2003.

[11] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–619, 2002.

[12] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, volume 1, pages 142–149, 2000.

[13] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge U. Press, 2000.

[14] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, pages 71–84, 2010.

[15] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[16] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*, 2004.

[17] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function with applications in pattern recognition. *IEEE Information Theory*, 21:32–40, 1975.

[18] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *Proc. 19th Intl. Conf. on Machine Learning*, pages 187–194, 2002.

[19] G. Hager, M. Dewan, and C. Stewart. Multiple kernel tracking with SSD. In *CVPR*, volume 1, pages 790–797, 2004.

[20] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[21] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010.

[22] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 2012.

[23] S. D. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *Intl. Conf. on Artificial Intelligence*, pages 561–566, 2003.

[24] B. Kulis, S. Basu, I. S. Dhillon, and R. J. Mooney. Semi-supervised graph clustering: A kernel approach. *Machine Learning*, 74:1–22, 2009.

[25] B. Kulis, M. A. Sustik, and I. S. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

[26] L. Lelis and J. Sander. Semi-supervised density-based clustering. In *IEEE Proc. Int. Conf. of Data Mining*, pages 842–847, 2009.

[27] M. Liu, B. Vemuri, S.-I. Amari, and F. Nielsen. Shape retrieval using hierarchical total bregman soft clustering. *PAMI*, 34:2407 – 2419, 2012.

[28] Lu Zhengdong and M. A. Carreira-Perpiñán. Constrained spectral clustering through affinity propagation. In *CVPR*, 2008.

[29] Lu Zhiwu and H. H.-S. Ip. Constrained spectral clustering via exhaustive and efficient constraint propagation. In *ECCV*, pages 1–14, 2010.

[30] B. Nadler and M. Galun. Fundamental limitations of spectral clustering. In *NIPS*, pages 1017–1024, 2007.

[31] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[32] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001.

[33] C. Ruiz, M. Spiliopoulou, and E. M. Ruiz. Density-based semi-supervised clustering. *Data Min. Knowl. Discov.*, 21:345–370, 2010.

[34] Y. Sheikh, E. Khan, and T. Kanade. Mode-seeking by medoidshifts. In *ICCV*, pages 1–8, 2007.

[35] T. Sim, S. Baker, and M. Bsat. The CMU Pose, Illumination, and Expression (PIE) Database. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002.

[36] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *CVPR*, pages 1168–1175, 2006.

[37] O. Tuzel, F. Porikli, and P. Meer. Kernel methods for weakly supervised mean shift clustering. In *ICCV*, pages 48–55, 2009.

[38] O. Tuzel, R. Subbarao, and P. Meer. Simultaneous multiple 3D motion estimation via mode finding on Lie groups. *ICCV*, pages 18–25, 2005.

[39] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, pages 705–718, 2008.

[40] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Intl. Conf. Machine Learning*, pages 577–584, 2001.

[41] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *ECCV*, volume 2, pages 238–249, 2004.

Saket Anand Saket Anand received his B.E. degree in Electronics Engineering from University of Pune, India, in 2003. He completed his MS in Electrical and Computer Engineering from Rutgers University, New Jersey, in 2006. From 2007 to 2009, he worked in handwriting recognition as a Research Engineer at Read-Ink Technologies, Bangalore, India. He is currently pursuing a PhD degree in Electrical and Computer Engineering at Rutgers University, New Jersey. His research interests include computer vision, statistical pattern recognition and machine learning. He is a student member of the IEEE.

Sushil Mittal Sushil Mittal received his B.Tech degree in Electronics and Communication Engineering from National Institute of Technology, Warangal, India in 2004. From 2004 to 2005, he was in the Department of Electrical Engineering at Indian Institute of Science as a research associate. He completed his MS and PhD degrees in Electrical and Computer Engineering from Rutgers University, New Jersey in 2008 and 2011, respectively. After his PhD, he worked in the Department of Statistics at Columbia University as a postdoctoral research scientist until 2013. He is currently a co-founder at Scibler Technologies, Bangalore. His research interests include computer vision, machine learning and applied statistics. He is a member of the IEEE.

Oncel Tuzel Oncel Tuzel received the BS and the MS degrees in computer engineering from the Middle East Technical University, Ankara, Turkey, in 1999 and 2002, and PhD degree in computer science from the Rutgers University, Piscataway, New Jersey, in 2008. He is a principal member of research staff at Mitsubishi Electric Research Labs (MERL), Cambridge, Massachusetts. His research interests are in computer vision, machine learning, and robotics. He coauthored more than 30 technical publications, and has applied for more than 20 patents. He was on the program committee of several international conferences such as CVPR, ICCV, and ECCV. He received the best paper runner-up award at the IEEE Computer Vision and Pattern Recognition Conference in 2007. He is a member of the IEEE.

Peter Meer Peter Meer received the Dipl. Engn. degree from the Bucharest Polytechnic Institute, Romania, in 1971, and the D.Sc. degree from the Technion, Israel Institute of Technology, Haifa, in 1986, both in electrical engineering. From 1971 to 1979 he was with the Computer Research Institute, Cluj, Romania, working on R&D of digital hardware. Between 1986 and 1990 he was Assistant Research Scientist at the Center for Automation Research, University of Maryland at College Park. In 1991 he joined the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ and is currently a Professor. He has held visiting appointments in Japan, Korea, Sweden, Israel and France, and was on the organizing committees of numerous international workshops and conferences. He was an Associate Editor of the *IEEE Transaction on Pattern Analysis and Machine Intelligence* between 1998 and 2002, a member of the Editorial Board of *Pattern Recognition* between 1990 and 2005, and was a Guest Editor of *Computer Vision and Image Understanding* for a special issue on robustness in computer vision in 2000. He is coauthor of an award winning paper in *Pattern Recognition* in 1989, the best student paper in the 1999, the best paper in the 2000 and the runner-up paper in 2007 of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. With coauthors Dorin Comaniciu and Visvanathan Ramesh he received at the 2010 CVPR the Longuet-Higgins prize for fundamental contributions in computer vision in the past ten years. His research interest is in application of modern statistical methods to image understanding problems. He is an IEEE Fellow.