A Kernel-Learning Approach to Semi-supervised Clustering with Relative Distance Comparisons

Ehsan Amid[1(✉)], Aristides Gionis[1], and Antti Ukkonen[2]

[1] Helsinki Institute for Information Technology,
and Department of Computer Science, Aalto University, Espoo, Finland
{ehsan.amid,aristides.gionis}@aalto.fi
[2] Finnish Institute of Occupational Health, Helsinki, Finland
antti.ukkonen@ttl.fi

Abstract. We consider the problem of clustering a given dataset into k clusters subject to an additional set of constraints on relative distance comparisons between the data items. The additional constraints are meant to reflect side-information that is not expressed in the feature vectors, directly. Relative comparisons can express structures at finer level of detail than must-link (ML) and cannot-link (CL) constraints that are commonly used for semi-supervised clustering. Relative comparisons are particularly useful in settings where giving an ML or a CL constraint is difficult because the granularity of the true clustering is unknown.

Our main contribution is an efficient algorithm for learning a kernel matrix using the log determinant divergence (a variant of the Bregman divergence) subject to a set of relative distance constraints. Given the learned kernel matrix, a clustering can be obtained by any suitable algorithm, such as kernel k-means. We show empirically that kernels found by our algorithm yield clusterings of higher quality than existing approaches that either use ML/CL constraints or a different means to implement the supervision using relative comparisons.

1 Introduction

Clustering is the task of partitioning a set of data items into groups, or clusters. However, the desired grouping of the data may not be sufficiently expressed by the features that are used to describe the data items. For instance, when clustering images it may be necessary to make use of semantic information about the image contents in addition to some standard image features. *Semi-supervised clustering* is a principled framework for combining such external information with features. This information is usually given as labels about the *pair-wise distances* between a few data items. Such labels may be provided by the data analyst, and reflect properties of the data that are hard to express as an easily computable function over the data features.

There are two commonly used ways to formalize such side information. The first are *must-link* (ML) and *cannot-link* (CL) constraints. An ML (CL) constraint between data items i and j suggests that the two items are similar (dissimilar), and should thus be assigned to the same cluster (different clusters).

A. Appice et al. (Eds.): ECML PKDD 2015, Part I, LNAI 9284, pp. 219–234, 2015.
DOI: 10.1007/978-3-319-23528-8_14

The second way to express pair-wise similarities are *relative distance comparisons*. These are statements that specify how the distances between some data items relate to each other. The most common relative distance comparison task asks the data analyst to specify which of the items i and j is closer to a third item k. Note that unlike the ML/CL constraints, the relative comparisons do not as such say anything about the clustering structure.

Given a number of similarity constraints, an efficient technique to implement semi-supervised clustering is *metric learning*. The objective of metric learning is to find a new distance function between the data items that takes both the supplied features as well as the additional distance constraints into account. Metric learning can be based on either ML/CL constraints or relative distance comparisons. Both approaches have been studied extensively in literature, and a lot is known about the problem.

The method we discuss in this paper is *a combination of metric-learning and relative distance comparisons*. We deviate from existing literature by eliciting every constraint with the question

"Which one of the items i, j, and k is the least similar to the other two?"

The labeler should thus select one of the items as an *outlier*. Notably, we also allow the labeler to leave the answer as *unspecified*. The main practical novelty of this approach is in the *capability to gain information also from comparisons where the labeler has not been able to give a concrete solution*. Some sets of three items can be all very similar (or dissimilar) to each other, so that picking one item as an obvious outlier is difficult. In those cases that the labeler gives a "don't know" answer, it is beneficial to use this answer in the metric-learning process as it provides a valuable cue, namely, that the three displayed data items are roughly equidistant.

We cast the metric-learning problem as a *kernel-learning problem*. The learned kernel can be used to easily compute distances between data items, even between data items that did not participate in the metric-learning training phase, and only their feature vectors are available. The use of relative comparisons, instead of hard ML/CL constraints, leads to learning a more accurate metric that captures relations between data items at different scales. The learned metric can be used for multi-level clustering, as well as other data-analysis tasks.

On the technical side, we start with an initial kernel $\mathbf{K}_0$, computed using only the feature vectors of the data items. We then formulate the kernel-learning task as an optimization problem: the goal is to find the kernel matrix $\mathbf{K}$ that is the closest to $\mathbf{K}_0$ and satisfies the constraints induced by the relative-comparison labellings. To solve this optimization task we use known efficient techniques, which we adapt for the case of relative comparisons.

More concretely, we make the following contributions:

1. We design a kernel-learning method that can also use unspecified relative distance comparisons. This is done by extending the method of Anand et al. [1], which works with ML and CL constraints.

2. We perform an extensive experimental validation of our approach and show that the proposed labeling is indeed more flexible, and it can lead to a substantial improvement in the clustering accuracy.

The rest of this paper is organized as follows. We start by reviewing the related literature in Section 2. In Section 3 we introduce our setting and formally define our problem, and in Section 4 we present our solution. In Section 5 we discuss our empirical evaluation, and Section 6 is a short conclusion.

2 Related Work

The idea of semi-supervised clustering was initially introduced by Wagstaff and Cardie [2], and since then a large number of different problem variants and methods have been proposed, the first ones being COP-Kmeans [3] and CCL [4]. Some of the later methods handle the constraints in a probabilistic framework. For instance, the ML and CL constraints can be imposed in the form of a Markov random field prior over the data items [5–7]. Alternatively, Lu [8] generalizes the standard Gaussian process to include the preferences imposed by the ML and CL constraints. Recently, Pei et al. [9] propose a discriminative clustering model that uses relative comparisons and, like our method, can also make use of unspecified comparisons.

The semi-supervising clustering setting has also been studied in the context of spectral clustering, and many spectral clustering algorithms have been extended to incorporate pairwise constraints [10,11]. More generally, these methods employ techniques for semi-supervised graph partitioning and kernel k-means algorithms [12]. For instance, Kulis et al. [13] present a unified framework for semi-supervised vector and graph clustering using spectral clustering and kernel learning.

As stated in the Introduction, our work is based on *metric learning*. Most of the metric-learning literature, starting by the work of Xing et al. [14], aims at finding a Mahalanobis matrix subject to either ML/CL or relative distance constraints. Xing et al. [14] use ML/CL constraints, while Schultz and Joachims [15] present a similar approach to handle relative comparisons. Metric learning often requires solving a semidefinite optimization problem. This becomes easier if Bregman divergences, in particular the log det divergence, is used to formulate the optimization problem. Such an approach was first used for metric learning by Davis et al. [16] with ML/CL constraints, and subsequently by Liu et al. [17] likewise with ML/CL, as well as by Liu et al. [18] with relative comparisons. Our algorithm also uses the log det divergence, and we extend the technique of Davis et al. [16] to handle relative comparisons.

The metric-learning approaches can also be more directly combined with a clustering algorithm. The MPCK-Means algorithm by Bilenko et al. [19] is one of the first to combine metric learning with semi-supervised clustering and ML/CL constraints. Xiang et al. [20] use metric learning, as well, to implement ML/CL constraints in a clustering and classification framework, while Kumar et al. [21] follow a similar approach using relative comparisons. Recently, Anand et al. [1]

use a kernel-transformation approach to adopt the mean-shift algorithm [22] to incorporate ML and CL constraints. This algorithm, called semi-supervised kernel mean shift clustering (SKMS), starts with an initial kernel matrix of the data points and generates a transformed matrix by minimizing the log det divergence using an approach based on the work by Kulis et al. [23]. Our paper is largely inspired by the SKMS algorithm. Our main contribution is to extend the SKMS algorithm so that it handles relative distance comparisons.

3 Kernel Learning with Relative Distances

In this section we introduce the notation used throughout the paper and formally define the problem we address.

3.1 Basic Definitions

Let $\mathcal{D} = \{1, \ldots, n\}$ denote a set of *data items*. These are the data we want to cluster. Let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$, with $\mathbf{x}_i \in \mathbb{R}^d$, denote a set of vectors in a d dimensional Euclidean space; one vector for every item in $\mathcal{D}$. The vector set $\mathcal{X}$ is the *feature representation* of the items in $\mathcal{D}$. We are also given the set $\mathcal{C}$ of *relative distance comparisons* between data items in $\mathcal{D}$. These distance comparisons are given in terms of some *unknown distance function* $\delta : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$. We assume that δ reflects certain domain knowledge, which is difficult to quantify precisely, and cannot be computed using only the features in $\mathcal{X}$. Thus, the set of distance comparisons $\mathcal{C}$ *augments our knowledge* about the data items in $\mathcal{D}$, in addition to the feature vectors in $\mathcal{X}$. The comparisons in $\mathcal{C}$ are given by human evaluators, or they may come from some other source. We assume that this information is not directly captured by the features.

Given $\mathcal{X}$ and $\mathcal{C}$, our objective is to find a kernel matrix $\mathbf{K}$ that captures more accurately the distance between data items. Such a kernel matrix can be used for a number of different purposes. In this paper, we focus on using the kernel matrix for clustering the data in $\mathcal{D}$. The kernel matrix $\mathbf{K}$ is computed by considering both the similarities between the points in $\mathcal{X}$ as well as the user-supplied constraints induced by the comparisons in $\mathcal{C}$.

In a nutshell, we compute the kernel matrix $\mathbf{K}$ by first computing an initial kernel matrix $\mathbf{K}_0$ using only the vectors in $\mathcal{X}$. The matrix $\mathbf{K}_0$ is computed by applying a Gaussian kernel on the vectors in $\mathcal{X}$. We then solve an optimization problem in order to find the kernel matrix $\mathbf{K}$ that is the closest to $\mathbf{K}_0$ and satisfies the constraints in $\mathcal{C}$.

3.2 Relative Distance Constraints

The constraints in $\mathcal{C}$ express information about distances between items in $\mathcal{D}$ in terms of the distance function δ. However, we do not need to know the absolute distances between any two items $i, j \in \mathcal{D}$. Instead we consider constraints that express information of the type $\delta(i, j) < \delta(i, k)$ for some $i, j, k \in \mathcal{D}$.

In particular, every constraint $C_i \in \mathcal{C}$ is a statement about the relative distances between *three* items in $\mathcal{D}$. We consider two types of constraints, i.e., $\mathcal{C}$ can be partitioned into two sets $\mathcal{C}_{\text{neq}}$ and $\mathcal{C}_{\text{eq}}$. The set $\mathcal{C}_{\text{neq}}$ contains constraints where one of the three items has been singled out as an "outlier." That is, the distance of the outlying item to the two others is clearly larger than the distance between the two other items. The set $\mathcal{C}_{\text{eq}}$ contains constraints where no item appears to be an obvious outlier. The distances between all three items are then assumed to be approximately the same.

More formally, we define $\mathcal{C}_{\text{neq}}$ to be a set of tuples of the form $(i, j \mid k)$, where every tuple is interpreted as "item k is an outlier among the three items i, j and k." We assume that the item k is an outlier if its distance from i and j is at least γ times larger than the distance $\delta(i, j)$, for some $\gamma > 1$. This is because we assume small differences in the distances to be indistinguishable by the evaluators, and only such cases end up in $\mathcal{C}_{\text{neq}}$ where there is no ambiguity between the distances. As a result each triple $(i, j \mid k)$ in $\mathcal{C}_{\text{neq}}$ implies the following two inequalities

$$(i \leftarrow j \mid k): \quad \gamma\delta(i, j) \leq \delta(i, k) \text{ and} \tag{1}$$

$$(j \leftarrow i \mid k): \quad \gamma\delta(j, i) \leq \delta(j, k), \tag{2}$$

where γ is a parameter that must be set in advance.

Likewise, we define $\mathcal{C}_{\text{eq}}$ to be a set of tuples of the form (i, j, k) that translates to "the distances between items i, j and k are equal." In terms of the distance function δ, each triple (i, j, k) in $\mathcal{C}_{\text{eq}}$ implies

$$\delta(i, j) = \delta(j, k) = \delta(i, k). \tag{3}$$

3.3 Extension to a Kernel Space

As mentioned above, the first step of our approach is forming the initial kernel $\mathbf{K}_0$ using the feature vectors $\mathcal{X}$. We do this using a standard Gaussian kernel. Details are provided in Section 4.2.

Next we show how the constraints implied by the distance comparison sets $\mathcal{C}_{\text{neq}}$ and $\mathcal{C}_{\text{eq}}$ extend to a kernel space, obtained by a mapping $\Phi : \mathcal{D} \to \mathbb{R}^m$. As usual, we assume that an inner product $\Phi(i)^\top \Phi(j)$ between items i and j in $\mathcal{D}$ can be expressed by a symmetric kernel matrix $\mathbf{K}$, that is, $\mathbf{K}_{ij} = \Phi(i)^\top \Phi(j)$. Moreover, we assume that the kernel $\mathbf{K}$ (and the mapping Φ) is connected to the unknown distance function δ via the equation

$$\delta(i, j) = \|\Phi(i) - \Phi(j)\|^2 = \mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{jj}. \tag{4}$$

In other words, we explicitly assume that the distance function δ is in fact the Euclidean distance in some unknown vector space. This is equivalent to assume that the evaluators base their distance-comparison decisions on some implicit features, even if they might not be able to quantify these explicitly.

Next, we discuss the constraint inequalities (Equations (1), (2), and (3)) in the kernel space. Let $\mathbf{e}_i$ denote the vector of all zeros with the value 1 at position i. Equation (4) above can be expressed in matrix form as follows:

$$\mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{jj} = (\mathbf{e}_i - \mathbf{e}_j)^\top \mathbf{K}(\mathbf{e}_i - \mathbf{e}_j) = \mathrm{tr}(\mathbf{K}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top), \quad (5)$$

where $\mathrm{tr}(\mathbf{A})$ denotes the trace of the matrix $\mathbf{A}$ and we use the fact that $\mathbf{K} = \mathbf{K}^\top$. Using the previous equation we can write Equation (1) as

$$\begin{aligned}
\gamma\,\mathrm{tr}\left(\mathbf{K}(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top\right) - \mathrm{tr}\left(\mathbf{K}(\mathbf{e}_i - \mathbf{e}_k)(\mathbf{e}_i - \mathbf{e}_k)^\top\right) &\leq 0 \\
\mathrm{tr}\left(\mathbf{K}\gamma(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top - \mathbf{K}(\mathbf{e}_i - \mathbf{e}_k)(\mathbf{e}_i - \mathbf{e}_k)^\top\right) &\leq 0 \\
\mathrm{tr}\left(\mathbf{K}(\gamma(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top - (\mathbf{e}_i - \mathbf{e}_k)(\mathbf{e}_i - \mathbf{e}_k)^\top)\right) &\leq 0 \\
\mathrm{tr}\left(\mathbf{K}\,\mathbf{C}_{(i \leftarrow j|k)}\right) &\leq 0,
\end{aligned}$$

where $\mathbf{C}_{(i \leftarrow j|k)} = \gamma(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top - (\mathbf{e}_i - \mathbf{e}_k)(\mathbf{e}_i - \mathbf{e}_k)^\top$ is a matrix that represents the corresponding constraint. The constraint matrix $\mathbf{C}_{(j \leftarrow i|k)}$ for Equation (2) can be formed in exactly the same manner. Note that unless we set $\gamma > 1$, the Equations (1) and (2) can be satisfied trivially for a small difference between the longer and the shorter distance and thus, the constraint becomes inactive. Setting $\gamma > 1$ helps avoiding such solutions.

We use a similar technique to represent the constraints in the set $\mathcal{C}_{\mathrm{eq}}$. Recall that the constraint $(i, j, k) \in \mathcal{C}_{\mathrm{eq}}$ implies that i, j, and k are equidistant. This yields three equations on the pairwise distances between the items: $(i \leftrightarrow j, k)$: $\delta(i, j) = \delta(i, k)$, $(j \leftrightarrow i, k)$: $\delta(j, i) = \delta(j, k)$, and $(k \leftrightarrow i, j)$: $\delta(k, i) = \delta(k, j)$. Reasoning as above, we let $\mathbf{C}_{(i \leftrightarrow j,k)} = (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^\top - (\mathbf{e}_i - \mathbf{e}_k)(\mathbf{e}_i - \mathbf{e}_k)^\top$, and can thus write the first equation for the constraint $(i, j, k) \in \mathcal{C}_{\mathrm{eq}}$ as

$$\mathrm{tr}(\mathbf{K}\,\mathbf{C}_{(i \leftrightarrow j,k)}) = 0. \quad (6)$$

The two other equations are defined in a similar manner.

3.4 Log Determinant Divergence for Kernel Learning

Recall that our objective is to find the kernel matrix $\mathbf{K}$ that is close to the initial kernel $\mathbf{K}_0$. Assume that $\mathbf{K}$ and $\mathbf{K}_0$ are both positive semidefinite matrices. We will use the so-called *log determinant divergence* to compute the similarity between $\mathbf{K}_0$ and $\mathbf{K}$. This is a variant of the *Bregman divergence* [24].

The Bregman divergence between two matrices $\mathbf{K}$ and $\mathbf{K}_0$ is defined as

$$\mathrm{D}_\phi(\mathbf{K}, \mathbf{K}_0) = \phi(\mathbf{K}) - \phi(\mathbf{K}_0) - \mathrm{tr}(\nabla\phi(\mathbf{K}_0)^\top(\mathbf{K} - \mathbf{K}_0)), \quad (7)$$

where ϕ is a strictly-convex real-valued function, and $\nabla\phi(\mathbf{K}_0)$ denotes the gradient evaluated at $\mathbf{K}_0$. Many well-known distance measures are special cases of the Bregman divergence. These can be instantiated by selecting the function ϕ appropriately. For instance, $\phi(\mathbf{K}) = \sum_{ij} K_{ij}^2$ gives the squared Frobenius norm $\mathrm{D}_\phi(\mathbf{K}, \mathbf{K}_0) = \|\mathbf{K} - \mathbf{K}_0\|_F^2$.

For our application in kernel learning, we are interested in one particular case; setting $\phi(\mathbf{K}) = -\log\det(\mathbf{K})$. This yields the so-called log determinant (log det) matrix divergence:

$$\mathrm{D}_{\mathrm{ld}}(\mathbf{K}, \mathbf{K}_0) = \mathrm{tr}(\mathbf{K}\,\mathbf{K}_0^{-1}) - \log\det(\mathbf{K}\,\mathbf{K}_0^{-1}) - n, \tag{8}$$

The log det divergence has many interesting properties, which make it ideal for kernel learning. As a general result of Bregman divergences, log det divergence is *convex* with respect to the first argument. Moreover, it can be evaluated using the eigenvalues and eigenvectors of the matrices $\mathbf{K}$ and $\mathbf{K}_0$. This property can be used to extend log det divergence to handle rank-deficient matrices [23], and we will make use of this in our algorithm described in Section 4.

3.5 Problem Definition

We now have the necessary ingredients to formulate our semi-supervised kernel learning problem. Given the set of constraints $\mathcal{C} = \mathcal{C}_{\mathrm{neq}} \cup \mathcal{C}_{\mathrm{eq}}$, the parameter γ, and the initial kernel matrix $\mathbf{K}_0$, we aim to find a new kernel matrix $\mathbf{K}$, which is as close as possible to $\mathbf{K}_0$ while satisfying the constraints in $\mathcal{C}$. This objective can be formulated as the following constrained minimization problem:

$$\begin{aligned}
&\underset{\mathbf{K}}{\text{minimize}} \quad \mathrm{D}_{\mathrm{ld}}(\mathbf{K}, \mathbf{K}_0) \\
&\text{subject to} \\
&\quad \mathrm{tr}\left(\mathbf{K}\,\mathbf{C}_{(i\leftarrow j|k)}\right) \le 0,\ \mathrm{tr}\left(\mathbf{K}\,\mathbf{C}_{(j\leftarrow i|k)}\right) \le 0, \quad \forall (i,j \mid k) \in \mathcal{C}_{\mathrm{neq}} \\
&\quad \mathrm{tr}(\mathbf{K}\,\mathbf{C}_{(i\leftrightarrow j,k)}) = 0,\ \mathrm{tr}(\mathbf{K}\,\mathbf{C}_{(j\leftrightarrow i,k)}) = 0,\ \mathrm{tr}(\mathbf{K}\,\mathbf{C}_{(k\leftrightarrow i,j)}) = 0, \quad \forall (i,j,k) \in \mathcal{C}_{\mathrm{eq}} \\
&\quad \mathbf{K} \succeq 0,
\end{aligned} \tag{9}$$

where $\mathbf{K} \succeq 0$ constrains $\mathbf{K}$ to be a positive semidefinite matrix.

4 Semi-supervised Kernel Learning

We now focus on the optimization problem defined above, Problem (9). It can be shown that in order to have a finite value for the log det divergence, the rank of the matrices must remain equal [23]. This property along with the fact that the domain of the log det divergence is the positive-semidefinite matrices, allow us to perform the optimization without explicitly restraining the solution to the positive-semidefinite cone nor checking for the rank of the solution. This is in contrast with performing the optimization using, say, the Frobenius norm, where the projection to the positive semidefinite cone must be explicitly imposed.

4.1 Bregman Projections for Constrained Optimization

In solving the optimization Problem (9), the aim is to minimize the divergence while satisfying the set of constraints imposed by $\mathcal{C} = \mathcal{C}_{\mathrm{neq}} \cup \mathcal{C}_{\mathrm{eq}}$. In other words,

we seek for a kernel matrix $\mathbf{K}$ by projecting the initial kernel matrix $\mathbf{K}_0$ onto the convex set obtained from the intersection of the set of constraints. The optimization Problem (9) can be solved using the method of *Bregman projections* [23–25]. The idea is to consider one unsatisfied constraint at a time and project the matrix so that the constraint gets satisfied. Note that the projections are not orthogonal and thus, a previously satisfied constraint might become unsatisfied. However, as stated before, the objective function in Problem (9) is convex and the method is guaranteed to converge to the global minimum if all the constraints are met infinitely often (randomly or following a more structured procedure).

Let us consider the update rule for an unsatisfied constraint from $\mathcal{C}_{\text{neq}}$. The procedure for dealing with constraints from $\mathcal{C}_{\text{eq}}$ is similar. We first consider the case of full-rank symmetric positive semidefinite matrices. Let $\mathbf{K}_t$ be the value of the kernel matrix at step t. For an unsatisfied inequality constraint $\mathbf{C}$, the optimization problem becomes[1]

$$\begin{aligned} \mathbf{K}_{t+1} &= \arg\min_{\mathbf{K}} \ \mathrm{D}_{\mathrm{ld}}(\mathbf{K}, \mathbf{K}_t), \\ \text{subject to } \ \langle \mathbf{K}, \mathbf{C} \rangle &= \mathrm{tr}(\mathbf{K}\mathbf{C}) \leq 0. \end{aligned} \tag{10}$$

Using a Lagrange multiplier $\alpha \geq 0$, we can write

$$\mathbf{K}_{t+1} = \arg\min_{\mathbf{K}} \ \mathrm{D}_{\mathrm{ld}}(\mathbf{K}, \mathbf{K}_t) + \alpha \, \mathrm{tr}(\mathbf{K}\mathbf{C}). \tag{11}$$

Following standard derivations for computing gradient updates for Bregman projection [25], the solution of Equation (11) can be written as

$$\mathbf{K}_{t+1} = (\mathbf{K}_t^{-1} + \alpha \mathbf{C})^{-1}. \tag{12}$$

Substituting Equation (12) into (10) gives

$$\mathrm{tr}((\mathbf{K}_t^{-1} + \alpha \mathbf{C})^{-1}\mathbf{C}) = 0. \tag{13}$$

Equation (13) does not have a closed form solution for α, in general. However, we exploit the fact that both types of our constraints, the matrix $\mathbf{C}$ has rank 2, i.e., $\mathrm{rank}(\mathbf{C}) = 2$. Let $\mathbf{K}_t = \mathbf{G}\,\mathbf{G}^\top$ and $\mathbf{W} = \mathbf{G}^\top \mathbf{C}\,\mathbf{G}$ and therefore $\mathrm{rank}(\mathbf{W}) = 2$, with eigenvalues $\eta_2 \leq 0 \leq \eta_1$ and $|\eta_2| \leq |\eta_1|$. Solving Equation (13) for α gives

$$\frac{\eta_1}{1 + \alpha\eta_1} + \frac{\eta_2}{1 + \alpha\eta_2} = 0, \tag{14}$$

and

$$\alpha^* = -\frac{1}{2}\frac{\eta_1 + \eta_2}{\eta_1 \eta_2} \geq 0. \tag{15}$$

Substituting Equation (15) into (12), gives the following update equation for the kernel matrix

$$\mathbf{K}_{t+1} = (\mathbf{K}_t^{-1} + \alpha^* \mathbf{C})^{-1} \, . \tag{16}$$

[1] We skip the subscript for notational simplicity.

Let $\mathbf{C} = \mathbf{U}\mathbf{V}^\top$ where $\mathbf{U}, \mathbf{V}$ are $n \times 2$ matrices of rank 2. Using Sherman-Morrison-Woodbury formula, we can write (16) as

$$\begin{aligned} \mathbf{K}_{t+1} &= \mathbf{K}_t - \mathbf{K}_t\, \alpha^* \mathbf{U}\, (\mathbf{I} + \alpha^* \mathbf{V}^\top \mathbf{K}_t\, \mathbf{U})^{-1}\, \mathbf{V}^\top \mathbf{K}_t \\ &= \mathbf{K}_t - \triangle \mathbf{K}_t \end{aligned} \tag{17}$$

in which, $\triangle \mathbf{K}_t$ is the correction term on the current kernel matrix $\mathbf{K}_t$. Calculation of the update rule (17) is simpler since it only involves inverse of a 2×2 matrix, rather than the $n \times n$ matrix in (16).

For a rank-deficient kernel matrix $\mathbf{K}_0$ with $\text{rank}(\mathbf{K}_0) = r$, we employ the results of Kulis et al. [23], which state that for any column-orthogonal matrix $\mathbf{Q}$ with $\text{range}(\mathbf{K}_0) \subseteq \text{range}(\mathbf{Q})$ (e.g., obtained by singular value decomposition of $\mathbf{K}_0$), we first apply the transformation

$$\mathbf{M} \rightarrow \hat{\mathbf{M}} = \mathbf{Q}^\top\, \mathbf{M}\, \mathbf{Q},$$

on all the matrices, and after finding the kernel matrix $\hat{\mathbf{K}}$ satisfying all the transformed constraints, we can obtain the final kernel matrix using the inverse transformation

$$\mathbf{K} = \mathbf{Q}\, \hat{\mathbf{K}}\, \mathbf{Q}^\top.$$

Note that since log det preserves the matrix rank, the mapping is one-to-one and invertible.

As the final remark, the kernel matrix learned by minimizing the log det divergence subject to the set of constraints $\mathcal{C}_{\text{neq}} \cup \mathcal{C}_{\text{eq}}$ can be also extended to handle *out of sample* data points, i.e., data points that were not present when learning the kernel matrix. The inner product between a pair of out of sample data points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ in the transformed kernel space can written as

$$k(\mathbf{x}, \mathbf{y}) = k_0(\mathbf{x}, \mathbf{y}) + \mathbf{k_x}^\top (\mathbf{K}_0^\dagger\, (\mathbf{K} - \mathbf{K}_0)\, \mathbf{K}_0^\dagger)\, \mathbf{k_y} \tag{18}$$

where, $k_0(\mathbf{x}, \mathbf{y})$ and the vectors $\mathbf{k_x} = [k_0(\mathbf{x}, \mathbf{x}_1), \ldots, k_0(\mathbf{x}, \mathbf{x}_n)]^\top$ and $\mathbf{k_y} = [k_0(\mathbf{y}, \mathbf{x}_1), \ldots, k_0(\mathbf{y}, \mathbf{x}_n)]^\top$ are formed using the initial kernel function.

4.2 Semi-supervised Kernel Learning with Relative Comparisons

In this section, we summarize the proposed approach, which we name SKLR, for Semi-supervised Kernel-Learning with Relative comparisons. The pseudo-code of the SKLR method is shown in Algorithm 1. As already discussed, the main ingredients of the method are the following.

Selecting the Bandwidth Parameter. We consider an adaptive approach to select the bandwidth parameter of the Gaussian kernel function. First, we set σ_i equal to the distance between point $\mathbf{x}_i$ and its ℓ-th nearest neighbor. Next, we set the kernel between $\mathbf{x}_i$ and $\mathbf{x}_j$ to

$$k_0(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_{ij}^2}\right), \tag{19}$$

Algorithm 1. (SKLR) Semi-supervised kernel learning with relative comparisons

Input: initial $n \times n$ kernel matrix $\mathbf{K}_0$, set of relative comparisons $\mathcal{C}_{\text{neq}}$ and $\mathcal{C}_{\text{eq}}$, constant distance factor γ
Output: kernel matrix $\mathbf{K}$

- **Find low-rank representation:**
  - Compute the $n \times n$ low-rank kernel matrix $\hat{\mathbf{K}}_0$ such that $\text{rank}(\hat{\mathbf{K}}_0) = r \leq n$ using incomplete Cholesky decomposition such that $\frac{\|\hat{\mathbf{K}}_0\|_F}{\|\mathbf{K}_0\|_F} \geq 0.99$
  - Find $n \times r$ column orthogonal matrix $\mathbf{Q}$ such that $\text{range}(\mathbf{K}_0) \subseteq \text{range}(\mathbf{Q})$
  - Apply the transformation $\hat{\mathbf{M}} \leftarrow \mathbf{Q}^\top \mathbf{M}\, \mathbf{Q}$ on all matrices
- **Initialize the kernel matrix**
  - Set $\hat{\mathbf{K}} \leftarrow \hat{\mathbf{K}}_0$
- **Repeat**

  (1) Select an unsatisfied constraint $\hat{\mathbf{C}} \in \mathcal{C}_{\text{neq}} \cup \mathcal{C}_{\text{eq}}$
  (2) Apply Bregman projection (17)

  Until all the constraints are satisfied
- **Return** $\mathbf{K} \leftarrow \mathbf{Q}\, \hat{\mathbf{K}}\, \mathbf{Q}^\top$

where, $\sigma_{ij}^2 = \sigma_i \sigma_j$. This process ensures a large bandwidth for sparse regions and a small bandwidth for dense regions.

Semi-Supervised Kernel Learning with Relative Comparisons. After finding the low-rank approximation of the initial kernel matrix $\mathbf{K}_0$ and transforming all the matrices by a proper matrix $\mathbf{Q}$, as discussed in Section 4.1, the algorithm proceeds by randomly considering one unsatisfied constrained at a time and performing the Bregman projections (17) until all the constraints are satisfied.

Clustering Method. After obtaining the kernel matrix $\mathbf{K}$ satisfying the set of all relative and undetermined constraints, we can obtain the final clustering of the points by applying any standard kernelized clustering method. In this paper, we consider the kernel k-means because of its simplicity and good performance. Generalization of the method to other clustering techniques such as kernel mean-shift is straightforward.

5 Experimental Results

In this section, we evaluate the performance of the proposed kernel-learning method, SKLR. As the under-the-hood clustering method required by SKLR, we use the standard kernel k-means with Gaussian kernel and without any supervision (Equation (19) and $\ell = 100$). We compare SKLR to three different semi-supervised metric-learning algorithms, namely, ITML [16], SKkm [1] (a variant

of SKMS with kernel k-means in the final stage), and LSML [18]. We select the SKkm variant as Anand et al. [1] have shown that SKkm tends to produce more accurate results than other semi-supervised clustering methods. Two of the baselines, ITML and SKkm, are based on pairwise ML/CL constraints, while LSML uses relative comparisons. For ITML and LSML we apply k-means on the transformed feature vectors to find the final clustering, while for SKkm and SKLR we apply kernel k-means on the transformed kernel matrices.

To assess the quality of the resulting clusterings, we use the Adjusted Rand (AR) index [26]. Each experiment is repeated 20 times and the average over all executions is reported. For the parameter γ required by SKLR we use $\gamma = 2$. Our implementation of SKLR is in MATLAB and the code is publicly available.[2] For the other three methods we use publicly available implementations.[3][4][5]

Finally, we note that in this paper we do not report running-time results, but all tested methods have comparable running times. In particular, the computational overhead of our method can be limited by leveraging the fact that the algorithm has to perform rank-2 matrix updates.

5.1 Datasets

We conduct the experiments on three different real-world datasets.

Vehicle:[6] The dataset contains 846 instances from 4 different classes and is available on the LIBSVM repository.

MIT Scene:[7] The dataset contains 2688 outdoor images, each sized 256×256, from 8 different categories: 4 natural and 4 man-made. We use the GIST descriptors [27] as the feature vectors.

USPS Digits:[8] The dataset contains 16×16 grayscale images of handwritten digits. It contains 1100 instances from each class. The columns of each images are concatenated to form a 256 dimensional feature vector.

5.2 Relative Constraints vs. Pairwise Constraints

We first demonstrate the performance of the different methods using relative and pairwise constraints. For each dataset, we consider two different experiments: (*i*) *binary* in which each dataset is clustered into *two groups*, based on some predefined criterion, and (*ii*) *multi-class* where for each dataset the clustering is performed with number of clusters being equal to *number of classes*. In the binary experiment, we aim to find a crude partitioning of the data, while in the multi-class experiment we seek a clustering at a finer granularity.

[2] https://github.com/eamid/sklr
[3] http://www.cs.utexas.edu/~pjain/itml
[4] https://github.com/all-umass/metric_learn
[5] https://www.iiitd.edu.in/~anands/files/code/skms.zip
[6] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[7] http://people.csail.mit.edu/torralba/code/spatialenvelope/
[8] http://cs.nyu.edu/~roweis/data.html

The 2-class partitionings of our datasets required for the binary experiment are defined as follows: For the **vehicle** dataset, we consider class 4 as one group and the rest of the classes as the second group (an arbitrary choice). For the **MIT Scene** dataset, we perform a partitioning of the data into natural vs. man-made scenes. Finally, for the **USPS Digits**, we divide the data instances into even vs. odd digits.

To generate the pairwise constraints for each dataset, we vary the number of labeled instances from each class (from 5 to 19 with step-size of 2) and form all possible ML constraints. We then consider the same number of CL constraints. Note that for the binary case, we only have two classes for each dataset. To compare with the methods that use relative comparisons, we consider an equal number of relative comparisons and generate them by sampling two random points from the same class and one point (outlier) from one of the other classes. Note that for the relative comparisons, there is no need to restrict the points to the labeled samples, as the comparisons are made in a relative manner.

Finally, in these experiments, we consider a subsample of both **MIT Scene** and **USPS Digits** datasets by randomly selecting 100 data points from each class, yielding 800 and 1000 data points, respectively.

The results for the binary and multi-class experiments are shown in Figures 1(a) and 1(b), respectively. We see that all methods perform equally with no constraints. As constraints or relative comparisons are introduced the accuracy of all methods improves very rapidly. The only surprising behavior is the one of ITML in the multi-class setting, whose accuracy drops as the number of constraints increases. From the figures we see that SKLR outperforms all competing methods by a large margin, for all three datasets and in both settings.

5.3 Multi-resolution Analysis

As discussed earlier, one of the main advantages of kernel learning with relative comparisons is the feasibility of multi-resolution clustering using a single kernel matrix. To validate this claim, we repeat the *binary* and *multi-class* experiments described above. However, this time, we mix the binary and multi-class constraints and use the same set of constraints in both experimental conditions. We evaluate the results by performing binary and multi-class clustering, as before.

Figures 1(c) and 1(d) illustrate the performance of different algorithms using the mixed set of constraints. Again, SKLR produces more accurate clusterings, especially in the multi-class setting. In fact, two of the methods, SKkm and ITML, perform worse than the kernel k-means baseline in the multi-class setting. On the other hand all methods outperform the baseline in the binary setting. The reason is that most of the constraints in the multi-class setting are also relevant to the binary setting, but not the other way around.

Figure 2 shows a visualization of the **USPS Digits** dataset using the SNE method [28] in the original space, and the spaces induced by SKkm and SKLR. We see that SKLR provides an excellent separation of the clusters that correspond to even/odd digits as well as the sub-clusters that correspond to individual digits.

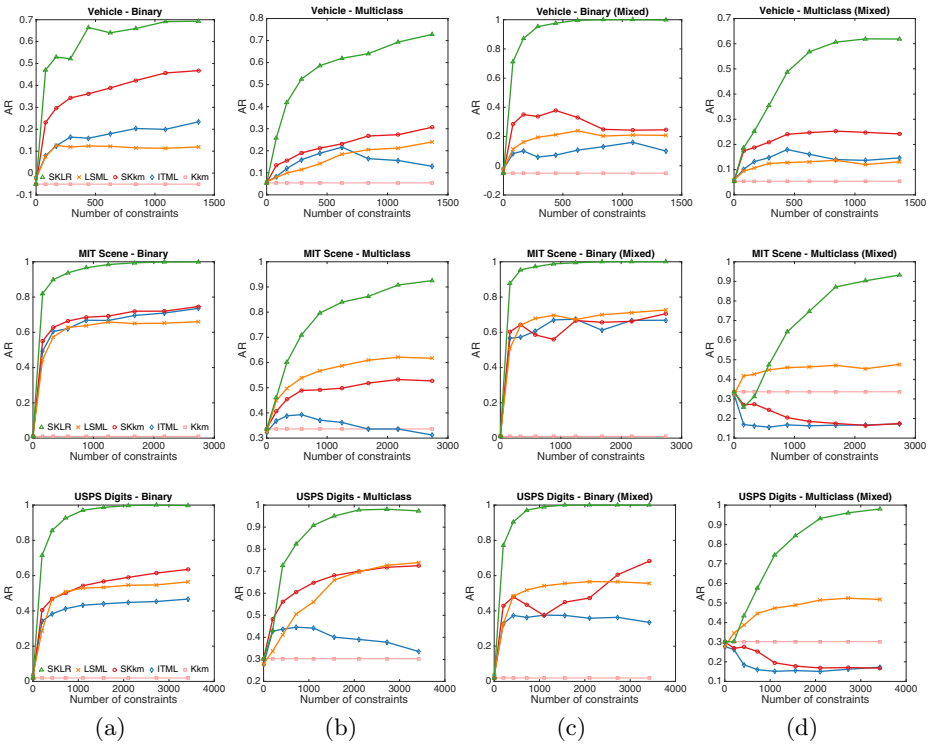

Fig. 1. Clustering accuracy measured with Adjusted Rand index (AR). Rows correspond to different datasets: (1) **Vehicle**; (2) **MIT Scene**; (3) **USPS Digits**. Columns correspond to different experimental settings: (a) binary with separate constraints; (b) multi-class with separate constraints; (c) binary with mixed constraints; (d) multi-class with mixed constraints.

5.4 Generalization Performance

We now evaluate the generalization performance of the different methods to out-of-sample data on the **MIT Scene** and **USPS Digits** datasets (recall that we do not subsample the **Vehicles** dataset). For the baseline kernel k-means algorithm, we run the algorithm on the whole datasets. For ITML and LSML, we apply the learned transformation matrix on the new out-of-sample data points. For SKkm and SKLR, we use Equation (18) to find the transformed kernel matrix of the whole datasets. The results of this experiment are shown in Figure 3. As can be seen from the figure, also in this case, when generalizing to out-of-sample data, SKLR produces significantly more accurate clusterings.

5.5 Effect of Equality Constraints

To evaluate the effect of equality constraints on the clustering, we consider a multi-class clustering scheme. For all datasets, we first generate a fixed number

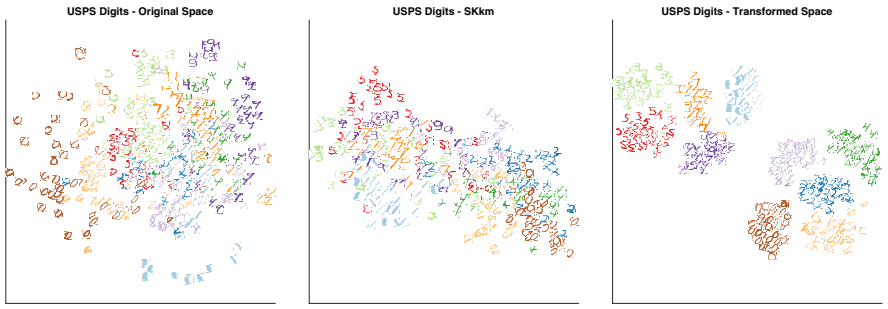

Fig. 2. Visualization of the **USPS Digits** using SNE: (a) original space; (b) space obtained by SK*k*m; (c) space obtained by our method, SKLR.

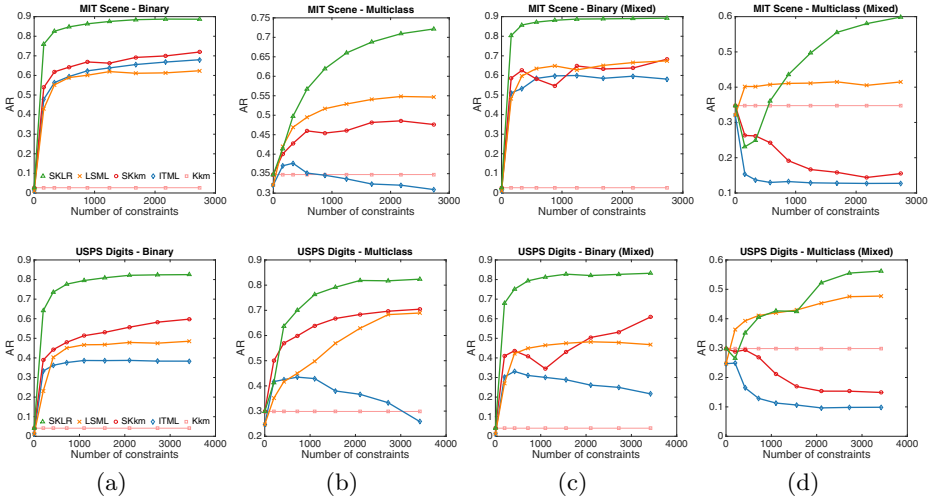

Fig. 3. Clustering accuracy on out-of-sample data (generalization performance). Rows correspond to different datasets: (1) **MIT Scene**; (2) **USPS Digits**. Columns correspond to different experimental settings: (a) binary with separate constraints; (b) multi-class with separate constraints; (c) binary with mixed constraints; (d) multi-class with mixed constraints.

of relative comparisons (360, 720, and 900 relative comparisons for **Vehicle**, **MIT Scene**, and **USPS Digits**, respectively) and then we add some additional equality constraints (up to 200). The equality constraints are generated by randomly selecting three data points, all from the same class, or each from a different class. The results are shown in Figure 4. As can be seen, considering the equality constraint also improves the performance, especially on the **MIT Scene** and **USPS Digits** datasets. Note that none of the other methods can handle these type of constraints.

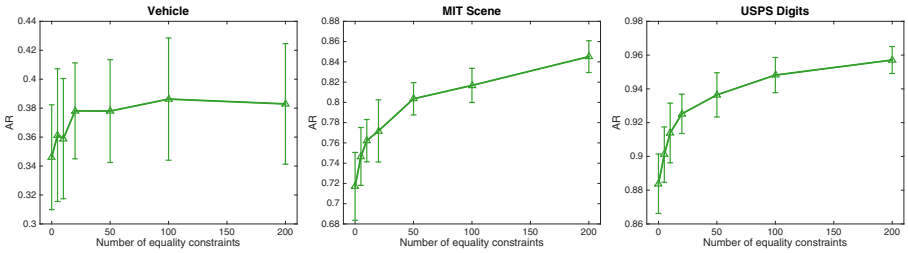

Fig. 4. Effect of equality constraints on the three datasets.

6 Conclusion

We have devised a semi-supervised kernel-learning algorithm that can incorporate various types of relative distance constraints, and used the resulting kernels for clustering. Our experiments show that our method outperforms by a large margin other competing methods, which either use ML/CL constraints or use relative constraints but different metric-learning approaches. Our method is compatible with existing kernel-learning techniques [1] in the sense that if ML and CL constraints are available, they can be used together with relative comparisons. We have also proposed to interpret an "unsolved" distance comparison so that the interpoint distances are roughly equal. Our experiments suggest that incorporating such equality constraints to the kernel learning task can be advantageous, especially in settings where it is costly to collect constraints.

For future work we would like to extend our method to incorporate more robust clustering methods such as spectral clustering and mean-shift. Additionally, the soft formulation of the relative constraints for handling possibly inconsistent constraints is straightforward, however, we leave it for future study.

References

1. Anand, S., Mittal, S., Tuzel, O., Meer, P.: Semi-supervised kernel mean shift clustering. PAMI **36**, 1201–1215 (2014)
2. Wagstaff, K., Cardie, C.: Clustering with instance-level constraints. In: ICML (2000)
3. Wagstaff, K., Cardie, C., Rogers, S., Schroedl, S.: Constrained k-means clustering with background knowledge. In: ICML (2001)
4. Klein, D., Kamvar, S.D., Manning, C.D.: From instance-level constraints to space-level constraints. In: ICML (2002)
5. Basu, S., Bilenko, M., Mooney, R.J.: A probabilistic framework for semi-supervised clustering. In: KDD (2004)
6. Basu, S., Banerjee, A., Mooney, R.J.: Active semi-supervision for pairwise constrained clustering. In: SDM (2004)
7. Lu, Z., Leen, T.K.: Semi-supervised learning with penalized probabilistic clustering. NIPS (2005)
8. Lu, Z.: Semi-supervised clustering with pairwise constraints: A discriminative approach. In: AISTATS (2007)

9. Pei, Y., Fern, X.Z., Rosales, R., Tjahja, T.V.: Discriminative clustering with relative constraints. arXiv:1501.00037 (2014)
10. Lu, Z., Ip, H.H.S.: Constrained Spectral Clustering via Exhaustive and Efficient Constraint Propagation. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 1–14. Springer, Heidelberg (2010)
11. Lu, Z., Carreira-Perpiñán, M.: Constrained spectral clustering through affinity propagation. In: CVPR (2008)
12. Dhillon, I.S., Guan, Y., Kulis, B.: A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, University of Texas (2005)
13. Kulis, B., Basu, S., Dhillon, I., Mooney, R.: Semi-supervised graph clustering: a kernel approach. Machine Learning **74**, 1–22 (2009)
14. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.J.: Distance metric learning with application to clustering with side-information. In: NIPS (2002)
15. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: NIPS (2003)
16. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML (2007)
17. Liu, W., Ma, S., Tao, D., Liu, J., Liu, P.: Semi-supervised sparse metric learning using alternating linearization optimization. In: KDD (2010)
18. Liu, E.Y., Guo, Z., Zhang, X., Jojic, V., Wang, W.: Metric learning from relative comparisons by minimizing squared residual. In: ICDM (2012)
19. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML (2004)
20. Xiang, S., Nie, F., Zhang, C.: Learning a mahalanobis distance metric for data clustering and classification. Pattern Recognition **41**, 3600–3612 (2008)
21. Kumar, N., Kummamuru, K.: Semisupervised clustering with metric learning using relative comparisons. TKDE **20**, 496–503 (2008)
22. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. PAMI **24**, 603–619 (2002)
23. Kulis, B., Sustik, M.A., Dhillon, I.S.: Low-rank kernel learning with Bregman matrix divergences. JMLR **10**, 341–376 (2009)
24. Bregman, L.: 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)
25. Tsuda, K., Rätsch, G., Warmuth, M.: Matrix exponentiated gradient updates for on-line learning and Bregman projection. JMLR **6**, 995–1018 (2005)
26. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2**, 193–218 (1985)
27. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV **42**, 145–175 (2001)
28. Hinton, G., Roweis, S.: Stochastic neighbor embedding. In: NIPS (2003)