Spatially-Aware Comparison and Consensus for Clusterings*

Parasaran Raman Jeff M. Phillips Suresh Venkatasubramanian

School of Computing, University of Utah.
{*praman,jeffp,suresh*}@*cs.utah.edu*

Abstract

This paper proposes a new distance metric between clusterings that incorporates information about the spatial distribution of points and clusters. Our approach builds on the idea of a Hilbert space-based representation of clusters as a combination of the representations of their constituent points. We use this representation and the underlying metric to design a spatially-aware consensus clustering procedure. This consensus procedure is implemented via a novel reduction to Euclidean clustering, and is both simple and efficient. All of our results apply to both soft and hard clusterings. We accompany these algorithms with a detailed experimental evaluation that demonstrates the efficiency and quality of our techniques.

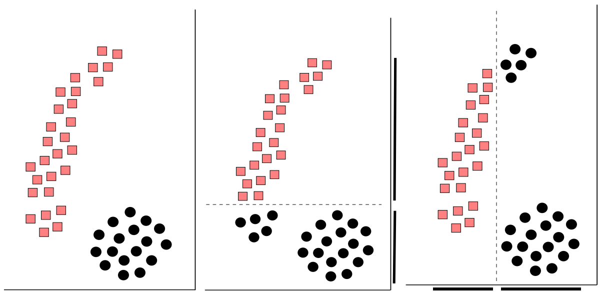
Keywords: Clustering, Ensembles, Consensus, Reproducing Kernel Hilbert Space.

1 Introduction

The problem of *metaclustering* has become important in recent years as researchers have tried to combine the strengths and weaknesses of different clustering algorithms to find patterns in data. A popular metaclustering problem is that of finding a consensus (or ensemble) partition[1] from among a set of candidate partitions. Ensemble-based clustering has been found to be very powerful when different clusters are connected in different ways, each detectable by different classes of clustering algorithms [34]. For instance, no single clustering algorithm can detect clusters of symmetric Gaussian-like distributions of different density and clusters of long thinly-connected paths; but these clusters can be correctly identified by combining multiple techniques (i.e. k-means and single-link) [13].

Other related and important metaclustering problems include finding a different and yet informative partition to a given one, or finding a set of partitions that are mutually diverse (and therefore informative). In all these problems, the key underlying step is comparing two partitions and quantifying the difference between them. Numerous metrics (and similarity measures) have been proposed to compare partitions, and for the most part they are based on comparing the combinatorial structure of the partitions. This is done either by examining pairs of points that are grouped together in one partition and separated in another [29, 4, 25, 10], or by information-theoretic considerations stemming from building a histogram of cluster sizes and normalizing it to form a distribution [24, 34].

These methods ignore the actual spatial description of the data, merely treating the data as atoms in a set and using set information to compare the partitions. As has been observed by many researchers [38, 3, 7], ignoring the spatial relationships in the data can be problematic. Consider the three partitions in Figure 1. The first partition (FP) is obtained by a projection onto the y-axis, and the second (SP) is obtained via a projection onto the x-axis. Partitions (FP) and (SP) are both equidistant from partition (RP) under any of the above mentioned distances, and yet it is clear that



(a) Reference Partition (RP) (b) First Partition (FP) (c) Second Partition (SP)

Figure 1: Why spatially-aware distances are important [7]: first and second partitions are equidistant from the reference partition under a set-based distance. However, *FP* is clearly more similar to *RP* than *SP* (See *2D2C* in Table 1).

*This research was supported by NSF award CCF-0953066 and a subaward to the University of Utah under NSF award 0937060 to the Computing Research Association.

[1]We use the term *partition* instead of *clustering* to represent a set of clusters decomposing a dataset. This avoids confusion between the terms 'cluster', 'clustering' and the procedure used to *compute* a partition, and will help us avoid phrases like, "We compute consensus clusterings by clustering clusters in clusterings!"

(FP) is more similar to the reference partition, based on the spatial distribution of the data.

Some researchers have proposed spatially-aware distances between partitions [38, 3, 7] (as we review below in Section 1.2.3), but they all suffer from various deficiencies. They compromise the spatial information captured by the clusters ([38, 3]), they lack metric properties ([38, 7]) (or have discontinuous ranges of distances to obtain metric properties ([3])), or they are expensive to compute, making them ineffective for large data sets ([7]).

1.1 Our Work. In this paper we exploit a concise, linear reproducing kernel Hilbert space (RKHS) representation of clusters. We use this representation to construct an efficient spatially-aware metric between partitions and an efficient spatially-aware consensus clustering algorithm.

We build on some recent ideas about clusters: (i) that a cluster can be viewed as a sample of data points from a distribution [19], (ii) that through a similarity kernel K, a distribution can be losslessly lifted to a single vector in a RKHS [26] and (iii) that the resulting distance between the representative vectors of two distributions in the RKHS can be used as a metric on the distributions [26, 32, 31].

Representations. We first adapt the representation of the clusters in the RKHS in two ways: approximation and normalization. Typically, vectors in an RKHS are infinite dimensional, but they can be approximated arbitrarily well in a finite-dimensional ℓ_2 space that retains the linear structure of the RKHS [28, 20]. This provides concise and easily manipulated representations for entire clusters. Additionally, we normalize these vectors to focus on the spatial information of the clusters. This turns out to be important in consensus clustering, as illustrated in Figure 4.

Distance Computation. Using this convenient representation (an approximate normalized RKHS vector), we develop a metric between partitions. Since the clusters can now be viewed as points in (scaled) Euclidean space we can apply standard measures for comparing point sets in such spaces. In particular, we define a spatially-aware metric LiftEMD between partitions as the transportation distance [15] between the representatives, weighted by the number of points they represent. While the transportation distance is a standard distance metric on probability distributions, it is expensive to compute (requiring $O(n^3)$ time for n points) [21]. However, since the points here are clusters, and the number of clusters (k) is typically significantly less than the data size (n), this is not a significant bottleneck as we see in Section 6.

Consensus. We exploit the linearity of the RKHS representations of the clusters to design an efficient consensus clustering algorithm. Given several partitions, each represented as a set of vectors in an RKHS, we can find a partition of this data using standard Euclidean clustering algorithms. In particular, we can compute a consensus partition by simply running k-means (or hierarchical agglomerative clustering) on the lifted representations of each cluster from all input partitions. This reduction from consensus to Euclidean clustering is a key contribution: it allows us to utilize the extensive research and fast algorithms for Euclidean clustering, rather than designing complex hypergraph partitioning methods [34].

Evaluation. All of these aspects of our technical contributions are carefully evaluated on real-world and synthetic data. As a result of the convenient isometric representation, the well-founded metric, and reduction to many existing techniques, our methods perform well compared to previous approaches and are much more efficient.

1.2 Background

1.2.1 Clusters as Distributions. The core idea in doing spatially aware comparison of partitions is to treat a cluster as a distribution over the data, for example as a sum of δ-functions at each point of the cluster [7] or as a spatial density over the data [3]. The distance between two clusters can then be defined as a distance between two distributions over a metric space (the underlying spatial domain).

1.2.2 Metrizing Distributions. There are standard constructions for defining such a distance; the most well known metrics are the transportation distance [15] (also known as the Wasserstein distance, the Kantorovich distance, the Mallows distance or the Earth mover's distance), and the Prokhorov metric [27]. Another interesting approach was initiated by Müller [26], and develops a metric between general measures based on integrating test functions over the measure. When the test functions are chosen from a reproducing kernel Hilbert space [2] (RKHS), the resulting metric on distributions has many nice properties [32, 17, 31, 5], most importantly that it can be isometrically embedded into the Hilbert space, yielding a convenient (but infinite dimensional) representation of a measure.

This measure has been applied to the problem of computing a *single* clustering by Jegelka *et. al.* [19]. In their work, each cluster is treated as a distribution and the partition is found by maximizing the inter-cluster distance of the cluster representatives in the RKHS. We modify this distance and its construction in our work.

A parallel line of development generalized this idea independently to measures over higher dimensional objects (lines, surfaces and so on). The resulting metric (the *current distance*) is exactly the above metric when applied to 0-dimensional objects (scalar measures) and has been used extensively [36, 16, 9, 20] to compare shapes. In fact, thinking of a cluster of points as a "shape" was a motivating factor in this work.

1.2.3 Distances Between Partitions.

Section 1 reviews spatially-aware and space-insensitive approaches to comparing partitions. We now describe prior work on spatially-aware distances between partitions in more detail.

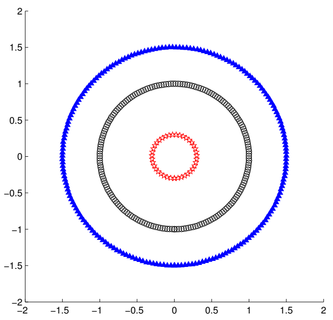


Figure 2: Dataset with 3 concentric circles, each representing a cluster partitioning the data. The CC distance [38] can not distinguish between these clusters.

Zhou *et al* [38] define a distance metric CC by replacing each cluster by its centroid (this of course assumes the data does not lie in an abstract metric space), and computing a weighted transportation distance between the sets of cluster centroids. Technically, their method yields a pseudo-metric, since two different clusters can have the same centroid, for example in the case of concentric ring clusters (Figure 2). It is also oblivious to the distribution of points within a cluster.

Coen *et al* [7] avoid the problem of selecting a cluster center by defining the distance between clusters as the transportation distance between the full sets of points comprising each cluster. This yields a metric on the set of all clusters in both partitions. In a second stage, they define the *similarity distance* CDistance between two partitions as the ratio between the transportation distance between the two partitions (using the metric just constructed as the base metric) and a "non-informative" transportation distance in which each cluster center distributes its mass equally to all cluster centers in the other partition. While this measure is symmetric, it does not satisfy triangle inequality and is therefore not a metric.

Bae *et al* [3] take a slightly different approach. They build a spatial histogram over the points in each cluster, and use the counts as a vector signature for the cluster. Cluster similarity is then computed via a dot product, and the similarity between two partitions is then defined as the sum of cluster similarities in an optimal matching between the clusters of the two partitions, normalized by the self-similarity of the two partitions.

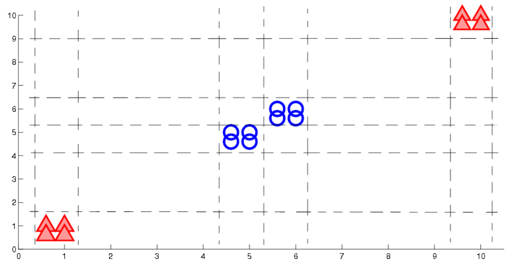


Figure 3: Data set with 4 clusters, each a set grouped in a single grid cell. The two clusters with blue open circles are as close as the two clusters with filled red triangles under the D_{ADCO} distance [3].

In general, such a spatial partitioning would require a number of histogram bins exponential in the dimension; they get around this problem by only retaining information about the marginal distributions along each dimension. One weakness of this approach is that only points that fall into the same bin contribute to the overall similarity. This can lead dissimilar clusters to be viewed as similar; in Figure 3, the two $\triangle$ (red) clusters will be considered as similar as the two $\bigcirc$ (blue) clusters.

Their approach yields a similarity, and not a distance metric. In order to construct a metric, they have to do the usual transformation dist $= 1 -$ sim and then add one to each distance between non-identical items, which yields the somewhat unnatural (and discontinuous) metric D_{ADCO}. Their method also implicitly assumes (like Zhou *et al* [38]) that the data lies in Euclidean space.

Our approach. Our method, centered around the RKHS-based metric between distributions, addresses all of the above problems. It yields a true metric, incorporates the actual distribution of the data correctly, and avoids exponential dependency on the dimension. The price we pay is the requirement that the data lie in a space admitting a positive definite kernel. However, this actually enables us to apply our method to clustering objects like graphs and strings, for which similarity kernels exist [12, 23] but no convenient vector space representation is known.

1.2.4 Consensus Clustering Algorithms.

One of the most popular methods for computing consensus between a collection of partitions is the *majority rule*: for each pair of points, each partition "votes" on whether the pair of points is in the same cluster or not, and the majority vote wins. While this method is simple, it is expensive and is spatially-oblivious; two points might lie in separate clusters that are close to each other.

Alternatively, consensus can be defined via a 1-median formulation: given a distance between partitions, the consensus partition is the partition that minimizes the sum of distances to all partitions. If the distance function is a metric, then the best partition from among the input partitions is guaranteed to be within twice the cost of the optimal solution (via triangle inequality). In general, it is challenging to find an arbitrary partition that minimizes this function. For example, the above majority-based method can be viewed as a heuristic for computing the 1-median under the Rand distance, and algorithms with formal approximations exist for this problem [14].

Recently Ansari *et al* [1] extended these above schemes to be spatially-aware by inserting CDistance in place of Rand distance above. This method is successful in grouping similar clusters from an ensemble of partitions, but it is quite slow on large data sets P since it requires computing d_T (defined in Section 2.1) on the full dataset. Alternatively using representations of each cluster in the ambient space (such as its mean, as in CC [38]) would produce another spatially-aware ensemble clustering variant, but would be less effective because its representation causes unwanted simplification of the clusters; see Figure 2.

2 Preliminaries

2.1 Definitions. Let P be the set of points being clustered, with $|P| = n$. We use the term *cluster* to refer to a subset C of P (i.e an actual cluster of the data), and the term *partition* to refer to a partitioning of P into clusters (i.e what one would usually refer to as a clustering of P). Clusters will always be denoted by the capital letters $A, B, C, \ldots$, and partitions will be denoted by the symbols $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots$. We will also consider *soft partitions* of P, which are fractional assignments $\{p(C|x)\}$ of points x to clusters C such that for any x, the assignment weights $p(C|x)$ sum to one.

We will assume that P is drawn from a space X endowed with a *reproducing kernel* $\kappa : X \times X \to \mathbb{R}$ [2]. The kernel κ induces a Hilbert space $\mathcal{H}_\kappa$ via the *lifting map* $\Phi : X \to \mathcal{H}_\kappa$, with the property that $\kappa(x, y) = \langle \Phi(x), \Phi(y) \rangle_\kappa$, $\langle \cdot, \cdot \rangle_\kappa$ being the inner product that defines $\mathcal{H}_\kappa$.

Let p, q be probability distributions defined over X. Let $\mathcal{F}$ be a set of real-valued bounded measurable functions defined over X. Let $\mathcal{F}_\kappa \triangleq \{f \in \mathcal{F} \mid \|f\|_\kappa \leq 1\}$ denote the *unit ball* in the Hilbert space $\mathcal{H}_\kappa$. The *integral probability metric* [26] γ_κ on distributions p, q is defined as $\gamma_\kappa(p, q) = \sup_{f \in \mathcal{F}_\kappa} |\int_X f dp - \int_X f dq|$. We will make extensive use of the following explicit formula for $\gamma_\kappa(p, q)$:

$$\begin{aligned}(2.1) \qquad \gamma_\kappa^2(p, q) = &\iint_X \kappa(x, y) dp(x) dp(y) \\ &+ \iint_X \kappa(x, y) dq(x) dq(y) \\ &- 2 \iint_X \kappa(x, y) dp(x) dq(y),\end{aligned}$$

which can be derived (via the kernel trick) from the following formula for γ_κ [32]: $\gamma_\kappa(p, q) = \| \int_X \kappa(\cdot, x) dp(x) - \int_X \kappa(\cdot, x) dq(x) \|_{\mathcal{H}_\kappa}$. This formula also gives us the lifting map Φ, since we can write $\Phi(p) = \int_X \kappa(\cdot, x) dp(x)$.

The transportation metric. Let $D : X \times X \to \mathbb{R}$ be a metric over X. The *transportation distance* between p and q is then defined as $d_T(p, q) = \inf_{f: X \times X \to [0,1]} \int_X \int_X f(x, y) D(x, y) dx dy$, such that $\int_X f(x, y) dx = q(y)$ and $\int_X f(x, y) dy = p(x)$. Intuitively, $f(x, y) D(x, y)$ measures the work in transporting $f(x, y)$ mass from $p(x)$ to $q(y)$.

2.2 An RKHS Distance Between Clusters. We use γ_κ to construct a metric on clusters. Let $C \subset P$ be a cluster. We associate with C the distribution $p(C) = \sum_x p(C|x) w(x) \delta_x(\cdot)$, where $\delta_x(\cdot)$ is the Kronecker δ-function and $w : P \to [0, 1]$ is a weight function. Given two clusters $C, C' \subset P$, we define $d(C, C') = \gamma_\kappa(p(C), p(C'))$.

An example. *A simple example illustrates how this distance generalizes pure partition-based distances between clusters. Suppose we fix the kernel* $\kappa(x, y)$ *to be the discrete kernel:* $\kappa(x, x) = 1$, $\kappa(x, y) = 0 \; \forall x \neq y$. *Then it is easy to verify that* $d(C, C') = \sqrt{|C \Delta C'|}$ *is the square root of the cardinality of the symmetric difference* $C \Delta C'$, *which is a well known set-theoretic measure of dissimilarity between clusters. Since this kernel treats all distinct points as equidistant from each other, the only information remaining is the set-theoretic difference between the clusters. As* κ *acquires more spatial information,* $d(C, C')$ *incorporates this into the distance calculation.*

2.2.1 Representations in $\mathcal{H}_\kappa$. There is an elegant way to represent points, clusters and partitions in the RKHS $\mathcal{H}_\kappa$. Define the lifting map $\Phi(x) = \kappa(\cdot, x)$. This takes a point $x \in P$ to a vector $\Phi(x)$ in $\mathcal{H}_\kappa$. A cluster $C \subset P$ can now be expressed as a weighted sum of these vectors: $\Phi(C) = \sum_{x \in C} w(x) \Phi(x)$. Note that for clarity, in what follows we will assume without loss of generality that all partitions are hard; to construct the corresponding soft partition-based expression, we merely replace terms of the form $\{x \in C\} = \mathbf{1}_{x \in C}$ by the probability $p(C|x)$.

$\Phi(C)$ is also a vector in $\mathcal{H}_\kappa$, and we can now rewrite $d(C, C')$ as $d(C, C') = \|\Phi(C) - \Phi(C')\|_{\mathcal{H}_\kappa}$

Finally, a partition $\mathcal{P} = \{C_1, C_2, \ldots C_k\}$ of P can be represented by the set of vectors $\Phi(\mathcal{P}) = \{\Phi(C_i)\}$ in $\mathcal{H}_\kappa$. We note that as long as the kernel is chosen correctly [32], this mapping is isometric, which implies that the representation $\Phi(C)$ is a *lossless* representation of C.

The linearity of representation is a crucial feature of how clusters are represented. While in the original space, a cluster might describe an unusually shaped collection of points, the same cluster in $\mathcal{H}_\kappa$ is merely the weighted sum of the corresponding vectors $\Phi(x)$. As a consequence, it is easy to represent *soft partitions* as well. A cluster C can be represented by the vector $\Phi(C) = \sum_x w(x) p(C|x) \Phi(x)$.

2.2.2 An RHKS-based Clustering. Jegelka *et al* [19] used the RKHS-based representation of clusters to formulate a new cost function for computing a *single* clustering. In particular, they considered the optimization problem of finding the partition $\mathcal{P} = \{C_1, C_2\}$ of two clusters to maximize

$$\begin{aligned} C(\mathcal{P}) = &|C_1| \cdot |C_2| \cdot \|\Phi(C_1) - \Phi(C_2)\|^2_{\mathcal{H}_K} \\ &+ \lambda_1 \|\Phi(C_1)\|^2_{\mathcal{H}_\kappa} + \lambda_2 \|\Phi(C_2)\|^2_{\mathcal{H}_\kappa}, \end{aligned}$$

for various choices of kernel κ and regularization terms λ_1 and λ_2. They mention that this could then be generalized to find an arbitrary k-partition by introducing more regularizing terms. Their paper focuses primarily on the generality of this approach and how it connects to other clustering frameworks, and they do not discuss algorithmic issues in any great detail.

3 Approximate Normalized Cluster Representation

We adapt the RKHS-based representation of clusters $\Phi(C)$ in two ways to make it more amenable to our meta-clustering goals. First, we approximate $\Phi(C)$ to a finite dimensional (ρ-dimensional) vector. This provides a finite representation of each cluster in $\mathbb{R}^\rho$ (as opposed to a vector in the infinite dimensional $\mathcal{H}_\kappa$), it retains linearity properties, and it allows for fast computation of distance between two clusters. Second, we normalize $\Phi(C)$ to remove any information about the size of the cluster; retaining only spatial information. This property becomes critical for consensus clustering.

3.1 Approximate Lifting Map $\tilde{\Phi}$. The lifted representation $\Phi(x)$ is the key to the representation of clusters and partitions, and its computation plays a critical role in the overall complexity of the distance computation. For kernels of interest (like the Gaussian kernel), $\Phi(x)$ cannot be computed explicitly, since the induced RKHS is an infinite-dimensional function space.

However, we can take advantage of the shift-invariance of commonly occurring kernels[2]. For these kernels a random projection technique in Euclidean space defines an approximate lifting map $\tilde{\Phi} : X \times X \to \mathbb{R}^\rho$ with the property that for any $x, y \in P$,

$$\left| \|\tilde{\Phi}(x) - \tilde{\Phi}(y)\|_2 - \|\Phi(x) - \Phi(y)\|_{\mathcal{H}_\kappa} \right| \le \varepsilon,$$

where $\varepsilon > 0$ is an arbitrary user defined parameter, and $\rho = \rho(\varepsilon)$. Notice that the approximate lifting map takes points to ℓ_2^ρ with the standard inner product, rather than a general Hilbert space. The specific construction is due to Rahimi and Recht [28] and analyzed by Joshi *et al* [20], to yield the following result:

Theorem 3.1. ([20]) *Given a set of n points $P \subset X$, shift-invariant kernel $\kappa : X \times X \to \mathbb{R}$ and any $\varepsilon > 0$, there exists a map $\tilde{\Phi} : X \times X \to \mathbb{R}^\rho$, $\rho = O((1/\varepsilon^2) \log n)$, such that for any $x, y \in P$, $\left| \|\tilde{\Phi}(x) - \tilde{\Phi}(y)\|_2 - \|\Phi(x) - \Phi(y)\|_{\mathcal{H}_\kappa} \right| \le \varepsilon$*

The actual construction is randomized and yields a $\tilde{\Phi}$ as above with probability $1 - \delta$, where $\rho = O((1/\varepsilon^2) \log(n/\delta))$. For any x, constructing the vector $\tilde{\Phi}(x)$ takes $O(\rho)$ time.

3.2 Normalizing $\Phi(C)$. The lifting map Φ is linear with respect to the weights of the data points, while being nonlinear in their location. Since $\Phi(C) = \sum_{x \in C} w(x) \Phi(x)$, this means that any scaling of the vectors $\Phi(x)$ translates directly into a uniform scaling of the weights of the data, and does not affect the spatial positioning of the points. This implies that we are free to normalize the cluster vectors $\Phi(C)$, so as to remove the scale information, retaining only, and exactly, the spatial information. In practice, we will normalize the cluster vectors to have unit length; let

$$\bar{\Phi}(C) = \Phi(C) / \|\Phi(C)\|_{\mathcal{H}_\kappa}.$$

Figure 4 shows an example of why it is important to compare RKHS representations of vectors using only their spatial information. In particular, without normalizing, small clusters C will have small norms $\|\Phi(C)\|_{\mathcal{H}_\kappa}$, and the distance between two small vectors $\|\Phi(C_1) - \Phi(C_2)\|_{\mathcal{H}_\kappa}$ is at most $\|\Phi(C_2)\|_{\mathcal{H}_\kappa} + \|\Phi(C_2)\|_{\mathcal{H}_\kappa}$. Thus all small clusters will likely have similar unnormalized RKHS vectors, irrespective of spatial location.

[2] A kernel $\kappa(x, y)$ defined on a vector space is shift-invariant if it can be written as $\kappa(x, y) = g(x - y)$.

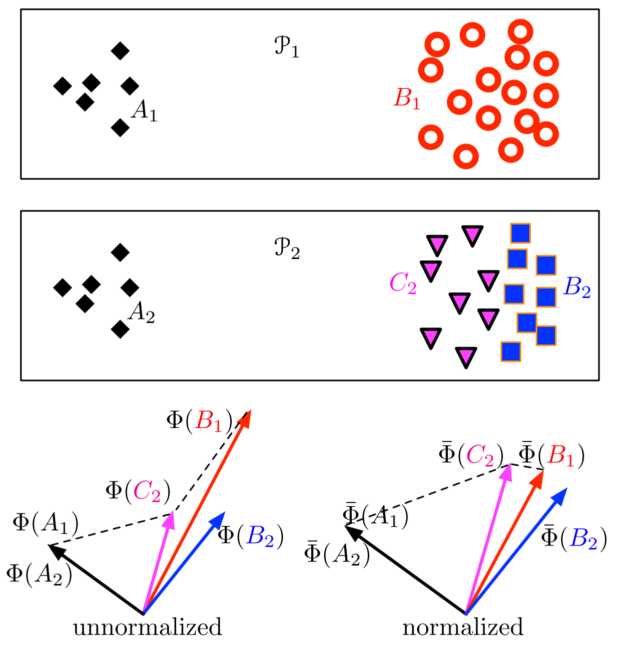


Figure 4: Two partitions $\mathcal{P}_1 = \{A_1, B_1\}$ and $\mathcal{P}_2 = \{A_2, B_2, C_2\}$ of the same dataset, and a 2-d visualization of all of their representations in an RKHS. Note that the unnormalized vectors (on the left) have $\Phi(B_1)$ far from $\Phi(B_2)$ and $\Phi(C_2)$ even though the second two are subsets of the first. The normalized vectors (on the right) have $\bar{\Phi}(B_1)$ close to both $\bar{\Phi}(B_2)$ and $\bar{\Phi}(C_2)$. In particular, $\Phi(C_2)$ is closer to $\Phi(A_1)$ than $\Phi(B_1)$, but $\bar{\Phi}(C_2)$ is much closer to $\bar{\Phi}(B_1)$ than $\bar{\Phi}(A_1)$.

3.3 Computing the Distance Between Clusters.

For two clusters C, C', we defined the distance between them as $d(C, C') = \gamma_\kappa(p(C), p(C'))$. Since the two distributions $p(C)$ and $p(C')$ are discrete (defined over $|C|$ and $|C'|$ elements respectively), we can use (2.1) to compute $d(C, C')$ in time $O(|C| \cdot |C'|)$. While this may be suitable for small clusters, it rapidly becomes expensive as the cluster sizes increase.

If we are willing to approximate $d(C, C')$, we can use Theorem 3.1 combined with the implicit definition of $d(C, C')$ as $\|\Phi(C) - \Phi(C')\|_{\mathcal{H}_\kappa}$. Each cluster C is represented as a sum of $|C|$ ρ-dimensional vectors, and then the ℓ_2 distance between the resulting vectors can be computed in $O(\rho)$ time. The following approximation guarantee on $d(C, C')$ then follows from the triangle inequality and an appropriate choice of ε.

Theorem 3.2. *For any two clusters C, C' and any $\varepsilon > 0$, $d(C, C')$ can be approximated to within an additive error ε in time $O((|C| + |C'|)\rho)$ time, where $\rho = O((1/\varepsilon^2)\log n)$.*

4 New Distances between Partitions

Let $\mathcal{P} = \{C_1, C_2, \ldots\}$ and $\mathcal{P}' = \{C'_1, C'_2, \ldots\}$ be two different partitions of P with associated representations $\Phi(\mathcal{P}) = \{\Phi(C_1), \Phi(C_2), \ldots\}$ and $\Phi(\mathcal{P}') = \{\Phi(C'_1), \Phi(C'_2), \ldots\}$. Similarly, $\bar{\Phi}(\mathcal{P}) = \{\bar{\Phi}(C_1), \bar{\Phi}(C_2), \ldots\}$ and $\bar{\Phi}(\mathcal{P}') = \{\bar{\Phi}(C'_1), \bar{\Phi}(C'_2), \ldots\}$. Since the two representations are sets of points in a Hilbert space, we can draw on a number of techniques for comparing point sets from the world of shape matching and pattern analysis.

We can apply the transportation distance d_T on these vectors to compare the partitions, treating the partitions as distributions. In particular, the partition $\mathcal{P}$ is represented by the distribution

$$\sum_{\bar{\Phi}(C) \in \bar{\Phi}(\mathcal{P})} \frac{|C|}{|P|} \cdot \delta_{\bar{\Phi}(C)} \tag{4.1}$$

where $\delta_{\bar{\Phi}(C)}$ is a Dirac delta function at $\bar{\Phi}(C) \in \mathcal{H}_\kappa$, with $\mathcal{H}_\kappa$ as the underlying metric. We will refer to this metric on partitions as

- LiftEMD$(\mathcal{P}, \mathcal{P}') = d_T(\bar{\Phi}(\mathcal{P}), \bar{\Phi}(\mathcal{P}'))$.

An example, continued. *Once again, we can simulate the loss of spatial information by using the discrete kernel as in Section 2.2. The transportation metric is computed (see Section 2.1) by minimizing a functional over all partial assignments $f(x, y)$. If we set $f(C, C') = |C \cap C'|/n$ to be the fraction of points overlapping between clusters, then the resulting transportation cost is precisely the Rand distance [29] between the two partitions! This observation has two implications. First, that standard distance measures between partitions appear as special cases of this general framework. Second,* LiftEMD$(\mathcal{P}, \mathcal{P}')$ *will always be at most the Rand distance between $\mathcal{P}$ and $\mathcal{P}'$.*

We can also use other measures. Let

$$\vec{d_H}(\Phi(\mathcal{P}), \Phi(\mathcal{P}')) = \max_{v \in \Phi(\mathcal{P})} \min_{w \in \Phi(\mathcal{P}')} \|v - w\|_{\mathcal{H}_\kappa}.$$

Then the Hausdorff distance [6] is defined as

$$d_H(\Phi(\mathcal{P}), \Phi(\mathcal{P}')) = \max\left(\vec{d_H}\left(\Phi(\mathcal{P}), \Phi(\mathcal{P}')\right), \vec{d_H}\left(\Phi(\mathcal{P}'), \Phi(\mathcal{P})\right)\right).$$

We refer to this application of the Hausdorff distance to partitions as

- LiftH$(\mathcal{P}, \mathcal{P}') = d_H(\bar{\Phi}(\mathcal{P}), \bar{\Phi}(\mathcal{P}'))$.

We could also use our lifting map again. Since we can view the collection of points $\Phi(\mathcal{P})$ as a spatial distribution in $\mathcal{H}_\kappa$ (see (4.1)), we can define $\gamma_{\kappa'}$ in this space as well, with κ' again given by any appropriate kernel (for example, $\kappa'(v, w) = \exp(-\|v - w\|^2_{\mathcal{H}_{\kappa'}})$. We refer to this metric as

- $\textsc{Lift}\mathrm{KD}(\mathcal{P}, \mathcal{P}') = \gamma_{\kappa'}(\bar{\Phi}(\mathcal{P}), \bar{\Phi}(\mathcal{P}'))$.

4.1 Computing the distance between partitions

The lifting map Φ (and its approximation $\tilde{\Phi}$) create efficiency in two ways. First, it is fast to generate a representation of a cluster C ($O(|C|\rho)$ time), and second, it is easy to estimate the distance between two clusters ($O(\rho)$ time). This implies that after a linear amount of processing, all distance computations between partitions depend only on the number of clusters in each partition, rather than the size of the input data. Since the number of clusters is usually orders of magnitude smaller than the size of the input, this allows us to use asymptotically inefficient algorithms on $\tilde{\Phi}(\mathcal{P})$ and $\tilde{\Phi}(\mathcal{P}')$ that have small overhead, rather than requiring more expensive (but asymptotically cheaper in k) procedures. Assume that we are comparing two partitions $\mathcal{P}, \mathcal{P}'$ with k and k' clusters respectively. LiftEMD is computed in general using a min-cost flow formulation of the problem, which is then solved using the Hungarian algorithm. This algorithm takes time $O((k+k')^3)$. While various approximations of d_T exist [30, 18], the exact method suffices for our setting for the reasons mentioned above.

It is immediate from the definition of LiftH that it can be computed in time $O(k \cdot k')$ by a brute force calculation. A similar bound holds for exact computation of LiftKD. While approximations exist for both of these distances, they incur overhead that makes them inefficient for small k.

5 Computing Consensus Partitions

As an application of our proposed distance between partitions, we describe how to construct a spatially-aware consensus from a collection of partitions. This method reduces the consensus problem to a standard clustering problem, allowing us to leverage the extensive body of work on standard clustering techniques. Furthermore, the representations of clusters as vectors in $\mathbb{R}^\rho$ allows for very concise representation of the data, making our algorithms extremely fast and scalable.

5.1 A Reduction from Consensus Finding to Clustering. Our approach exploits the linearity of cluster representations in $\mathcal{H}_\kappa$, and works as follows. Let $\mathcal{P}_1, \mathcal{P}_2, \dots \mathcal{P}_m$ be the input (hard or soft) partitions of P. Under the lifting map Φ, each partition can be represented by a set of points $\{\Phi(\mathcal{P}_i)\}$ in $\mathcal{H}_\kappa$. Let $Q = \bigcup_i \Phi(\mathcal{P}_i)$ be the collection of these points.

Definition 5.1. *A (soft) consensus k-partition of $\mathcal{P}_1, \mathcal{P}_2, \dots \mathcal{P}_m$ is a partition $\mathcal{P}_{con}$ of $\bigcup_i \mathcal{P}_i$ into k clusters $\{C_1^*, \dots, C_k^*\}$ that minimizes the sum of squared distances from each $\bar{\Phi}(C_{i,j}) \in \bar{\Phi}(\mathcal{P}_i)$ to its associated $\bar{\Phi}(C_l^*) \in \mathcal{P}_{con}$. Formally, for a set of k vectors $V = \{v_1, \dots, v_k\} \subset \mathcal{H}_\kappa$ define*

$$\textsc{lift}\mathrm{SSD}(\{\mathcal{P}_i\}, V) = \sum_{C_{i,j} \in \cup_i \mathcal{P}_i} \frac{|C_{i,j}|}{n} \min_{v \in V} \left\|\bar{\Phi}(C_{i,j}) - v\right\|^2_{\mathcal{H}_\kappa}$$

and then define $\mathcal{P}_{con}$ as the minimum such set

$$\mathcal{P}_{con} = \underset{V^* = \{v_1^*, \dots, v_k^*\} \in \mathcal{H}_\kappa}{\operatorname{argmin}} \textsc{lift}\mathrm{SSD}(\{\mathcal{P}_i\}_i, V^*).$$

How do we interpret $\mathcal{P}_{\mathrm{con}}$? Observe that each element in Q is the lifted representation of some cluster $C_{i,j}$ in some partition $\mathcal{P}_i$, and therefore corresponds to some subset of P. Consider now a single cluster in $\mathcal{P}_{\mathrm{con}}$. Since $\mathcal{H}_\kappa$ is linear and $\mathcal{P}_{\mathrm{con}}$ minimizes distance to some set of cluster representatives, it must be in their linear combination. Hence it can be associated with a weighted subset of elements of $\Phi(P)$, and is hence a soft partition. It can be made hard by voting each point $x \in P$ to the representative $C_l^* \in \mathcal{P}_{\mathrm{con}}$ for which it has the largest weight.

Figure 5.1 shows an example of three partitions, their normalized RKHS representative vectors, and two clusters of those vectors.

5.2 Algorithm. We will use the approximate lifting map $\tilde{\Phi}$ in our procedure. This allows us to operate in a ρ-dimensional Euclidean space, in which there are many clustering procedures we can use. For our experiments, we will use both k-means and hierarchical agglomerative clustering (HAC). That is, let LiftKm be the algorithm of running k-means on $\bigcup_i \tilde{\Phi}(\mathcal{P}_i)$, and let LiftHAC be the algorithm of running HAC on $\bigcup_i \tilde{\Phi}(\mathcal{P}_i)$. For both algorithms, the output is the (soft) clusters represented by the vectors in $\Phi(\mathcal{P}_{\mathrm{con}})$. Our results will show that the particular choice of clustering algorithm (e.g. LiftKm or LiftHAC) is not crucial. There are multiple methods to choose the right number of clusters and we can employ one of them to fix k for our consensus technique. Algorithm 1 summarizes our consensus procedure.

Cost analysis. Computing Q takes $O(mn\rho) = O(mn \log n)$ time. Let $|Q| = s$. Computing $\mathcal{P}_{\mathrm{con}}$ is a single call to any standard Euclidean algorithm like k-means that takes time $O(sk\rho)$ per iteration, and computing the final soft partition takes time linear in $n(\rho + k) + s$. Note that in general we expect that $k, s \ll n$. In particular, when $s < n$ and m is assumed constant, then the runtime is $O(n(k + \log n))$.

6 Experimental Evaluation

In this section we empirically show the effectiveness of our distance between partitions, LiftEMD and

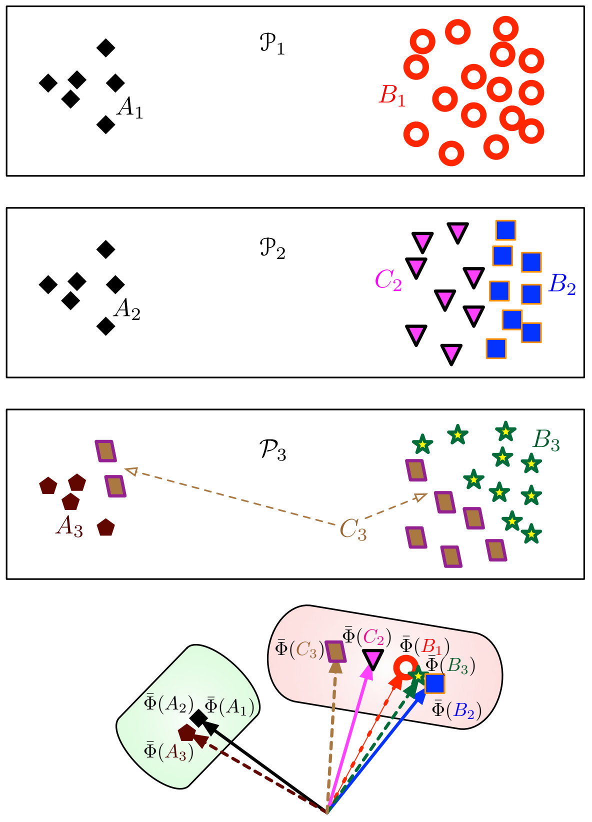


Figure 5: Three partitions $\mathcal{P}_1 = \{A_1, B_1\}$, $\mathcal{P}_2 = \{A_2, B_2, C_2\}$, $\mathcal{P}_3 = \{A_3, B_3, C_3\}$ of the same dataset, and a 2-d visualization of all of their representations in an RKHS. These vectors are then clustered into $k = 2$ consensus clusters consisting of $\{\bar\Phi(A_1), \bar\Phi(A_2), \bar\Phi(A_3)\}$ and $\{\bar\Phi(B_1), \bar\Phi(B_2), \bar\Phi(C_2), \bar\Phi(B_3), \bar\Phi(C_3)\}$.

Algorithm 1 Consensus finding

```
Input: (soft) Partitions $\mathcal{P}_1, \mathcal{P}_2, \dots \mathcal{P}_m$ of $P$, kernel function $\kappa$
Output: Consensus (soft) partition $\mathcal{P}_{\text{con}}$
1: Set $Q = \cup_i \tilde\Phi(\mathcal{P}_i)$
2: Compute $V^* = \{v_1^*, \dots v_k^*\} \subset \mathcal{H}_\kappa$ to minimizes LiftSSD$(Q, V^*)$.
   (Via $k$-means for LiftKm or HAC for LiftHAC)
3: Assign each $p \in P$ to the cluster $C_i \in \mathcal{P}_{\text{con}}$ associated with the vector $v_i^* \in V^*$
     • for which $\langle \tilde\Phi(p), v_i \rangle$ is maximized for a hard partition, or
     • with weight proportional to $\langle \tilde\Phi(p), v_i \rangle$ for a soft partition.
4: Output $\mathcal{P}_{\text{con}}$.
```

LiftKD, and the consensus clustering algorithms that conceptually follow, LiftKm and LiftHAC.

Data. We created two synthetic datasets in $\mathbb{R}^2$ namely, *2D2C* for which data is drawn from 2 Gaussians to produce 2 visibly separate clusters and *2D3C* for which the points are arbitrarily chosen to produce 3 visibly separate clusters. We also use 5 different datasets from the UCI repository [11] (Wine, Ionosphere, Glass, Iris, Soybean) with various numbers of dimensions and labeled data classes. To show the ability of our consensus procedure and the distance metric to handle large data, we use both the training and test data of MNIST [22] database of handwritten digits which has 60,000 and 10,000 examples respectively in $\mathbb{R}^{784}$.

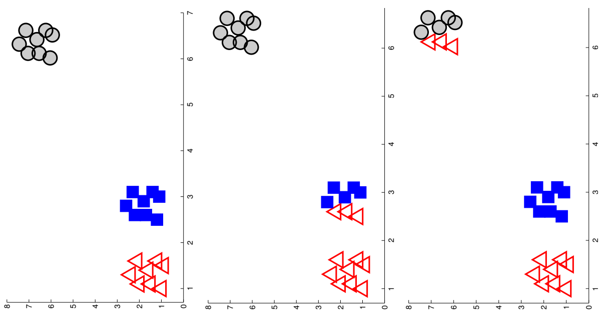
Methodology. We will compare our approach with the partition-based measures namely, Rand Distance and Jaccard distance and information-theoretic measures namely, Normalized Mutual Information and Normalized Variation of Information [37], as well as the spatially-aware measures D_{ADCO} [3] and CDistance [7]. We ran k-means, single-linkage, average-linkage, complete-linkage and Ward's method [35] on the datasets to generate input partitions to the consensus clustering methods. We use accuracy [8] and Rand distance [29] to measure the effectiveness of the consensus clustering algorithms by comparing the returned consensus partitions to the original class labeling. We compare our consensus technique against few hypergraph partitioning based consensus methods CSPA, HGPA and MCLA [34]. For the MNIST data we also visualize the cluster centroids of each of the input and consensus partitions in a 28x28 grayscale image.

Accuracy studies the one-to-one relationship between clusters and classes; it measures the extent to which each cluster contains data points from the corresponding class. Given a set P of n elements, consider a set of k clusters $\mathcal{P} = \{C_1, \dots, C_k\}$ and $m \geq k$ classes $\mathcal{L} = \{L_1, \dots, L_m\}$ denoting the ground truth partition. Accuracy is expressed as

$$\mathsf{A}(\mathcal{P}, \mathcal{L}) = \max_{\mu:[1:k]\to[1:m]} \sum_{i=1}^{k} \frac{|C_i \cap L_{\mu(i)}|}{n},$$

where μ assigns each cluster to a distinct class. The Rand distance counts the fraction of pairs which are assigned consistently in both partitions as in the same or in different classes. Let $R_S(\mathcal{P}, \mathcal{L})$ be the number of pairs of points that are in the same cluster in both $\mathcal{P}$ and $\mathcal{L}$, and let $R_D(\mathcal{P}, \mathcal{L})$ be the number of pairs of points that are in different clusters in both $\mathcal{P}$ and $\mathcal{L}$. Now we can define the Rand distance as

$$\mathsf{R}(\mathcal{P}, \mathcal{L}) = 1 - \frac{R_S(\mathcal{P}, \mathcal{L}) + R_D(\mathcal{P}, \mathcal{L})}{\binom{n}{2}}.$$



(a) Reference Partition (*RP*) (b) First Partition (*FP*) (c) Second Partition (*SP*)

Figure 6: Different partitions of *2D3C* dataset.

Code. We implement $\tilde{\Phi}$ as the *Random Projection* feature map [20] in C to lift each data point into $\mathbb{R}^\rho$. We set $\rho = 200$ for the two synthetic datasets *2D2C* and *2D3C* and all the UCI datasets. We set $\rho = 4000$ for the larger datasets, MNIST training and MNIST test. The same lifting is applied to all data points, and thus all clusters.

The LiftEMD, LiftKD, and LiftH distances between two partitions $\mathcal{P}$ and $\mathcal{P}'$ are computed by invoking brute-force transportation distance, kernel distance, and Hausdorff distance on $\tilde{\Phi}(\mathcal{P}), \tilde{\Phi}(\mathcal{P}') \subset \mathbb{R}^\rho$ representing the lifted clusters.

To compute the consensus clustering in the lifted space, we apply k-means (for LiftKm) or HAC (for LiftHAC) (with the appropriate numbers of clusters) on the set $Q \subset \mathbb{R}^\rho$ of all lifted clusters from all partitions. The only parameters required by the procedure is the error term ε (needed in our choice of ρ) associated with $\tilde{\Phi}$, and any clustering-related parameters.

We used the cluster analysis functions in MATLAB with the default settings to generate the input partitions to the consensus methods and the given number of classes as the number of clusters. We implemented the algorithm provided by the authors [3] in MATLAB to compute D_{ADCO}. To compute CDistance, we used the code provided by the authors [7]. We used the ClusterPack MATLAB toolbox [33] to run the hypergraph partitioning based consensus methods CSPA, HGPA and MCLA.

6.1 Spatial Sensitivity. We start by evaluating the sensitivity of our method. We consider three partitions– the ground truth (or) reference partition (*RP*), and manually constructed first and second partitions (*FP* and *SP*) for the datasets *2D2C* (see Figure 1) and *2D3C* (see Figure 6). For both the datasets the reference partition is by construction spatially closer to the first partition than the second partition, but each of the two partitions are equidistant from the reference under any partition-based and information-theoretic measures. Table 1 shows that in each example, our measures correctly conclude that *RP* is closer to *FP* than it is to *SP*.

6.2 Efficiency. We compare our distance computation procedure to CDistance. We do not compare against D_{ADCO} and CC because they are not well-founded. Both LiftEMD and CDistance compute d_T between clusters after an initial step of either lifting to a feature space or computing d_T between all pairs of clusters. Thus the proper comparison, and runtime bottleneck, is the initial phase of the algorithms; LiftEMD takes $O(n \log n)$ time whereas CDistance takes $O(n^3)$ time. Table 2 summarizes our results. For instance, on the *2D3C* data set with $n = 24$, our initial phase takes 1.02 milliseconds, and CDistance's initial phase takes 2.03 milliseconds. On the Wine data set with $n = 178$, our initial phase takes 6.9 milliseconds, while CDistance's initial phase takes 18.8 milliseconds. As the dataset size increases, the advantage of LiftEMD over CDistance becomes even larger. On the MNIST training data with $n = 60,000$, our initial phase takes a little less than 30 minutes, while CDistance's initial phase takes more than 56 hours.

6.3 Consensus Clustering. We now evaluate our spatially-aware consensus clustering method. We do this first by comparing our consensus partition to the reference solution based on using the Rand distance (i.e a partition-based measure) in Table 3. Note that for all data sets, our consensus clustering methods (LiftKm and LiftHAC) return answers that are almost always as close as the best answer returned by any of the hypergraph partitioning based consensus methods CSPA, HGPA, or MCLA. We get very similar results using the accuracy [8] measure in place of Rand.

In Table 4, we then run the same comparisons, but this time using LiftEMD (i.e a spatially-aware measure). Here, it is interesting to note that in all cases (with the slight exception of Ionosphere) the distance we get is smaller than the distance reported by the hypergraph partitioning based consensus methods, indicating that our method is returning a consensus partition that is spatially closer to the true answer. The two tables also illustrate the flexibility of our framework, since the results using LiftKm and LiftHAC are mostly identical (with one exception being IRIS under LiftEMD).

To summarize, our method provides results that are comparable or better on partition-based measures of consensus, and are superior using spatially-aware measures. The running time of our approach is comparable to the best hypergraph partitioning based approaches,

Technique	Dataset *2D2C*		Dataset *2D3C*	
	d(*RP*,*FP*)	d(*RP*,*SP*)	d(*RP*,*FP*)	d(*RP*,*SP*)
D_{ADCO}	1.710	1.780	1.790	1.820
CDistance	0.240	0.350	0.092	0.407
LiftEMD	0.430	0.512	0.256	0.310
LiftKD	0.290	0.325	0.243	0.325
LiftH	0.410	0.490	1.227	1.291

Table 1: Comparing Partitions. Each cell indicates the distance returned under the methods along the rows for the dataset in the column. Spatially, the left column of each data set (*2D2C* or *2D3C*) should be smaller than the right column; this holds for all 5 spatial measures/algorithms tested. In all cases, the two partition-based measures and the two information-theoretic measures yield the same values for d(*RP*, *FP*) and d(*RP*, *SP*), but are not shown.

Dataset	Number of points	Number of dimensions	CDistance	LiftEMD
2D3C	24	2	2.03 ms	1.02 ms
2D2C	45	2	4.10 ms	1.95 ms
Wine	178	13	18.80 ms	6.90 ms
MNIST test data	10,000	784	1360.20 s	303.90 s
MNIST training data	60,000	784	202681 s	1774.20 s

Table 2: Comparison of runtimes: Distance between true partition and partition generated by k-means)

so using our consensus procedure yields the best overall result.

We also run consensus experiments on the MNIST test data and compare against CSPA and MCLA. We do not compare against HGPA since it runs very slow for the large MNIST datasets ($n = 10,000$); it has quadratic complexity in the input size, and in fact, the authors do not recommend this for large data. Figure 7 provides a visualization of the cluster centroids of input partitions generated using k-means, complete linkage HAC and average linkage HAC and the consensus partitions generated by CSPA and LiftKm. From the k-means input, only 5 clusters can be easily associated with digits (*0, 3, 6, 8, 9*); from the complete linkage HAC input, only 7 clusters can be easily associated with digits (*0, 1 , 3, 6, 7, 8, 9*); and from the average linkage HAC output, only 6 clusters can be easily associated with digits (*0, 1, 2, 3, 7, 9*). The partition that we obtain from running CSPA also lets us identify up to 6 digits (*0, 1, 2, 3, 8, 9*). In all the above three partitions, there occurs cases where two clusters seem to represent the same digit. In contrast, we can identify 9 digits (*0, 1, 2, 3, 4, 5, 7, 8, 9*) with only the digit *6* being noisy from our LiftKm output.

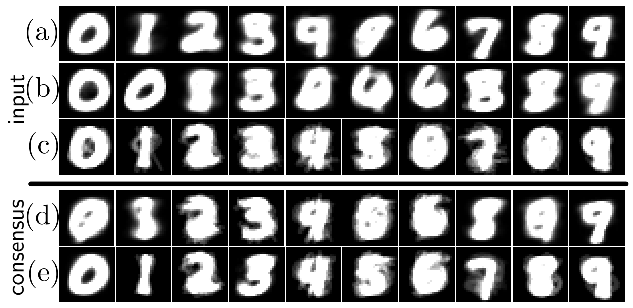


Figure 7: 28x28 pixel representation of the cluster centroids for MNIST test input partitions generated using (a) k-means, (b) complete linkage HAC, and (c) average linkage HAC, and the consensus partitions generated by (d) CSPA and (e) LiftKm.

6.4 Error in $\tilde{\Phi}$.

There is a tradeoff between the desired error ε in computing LiftEMD and the number of dimensions ρ needed for $\tilde{\Phi}$. Figure 8 shows the error as a function of ρ on the *2D2C* dataset ($n = 45$). From the chart, we can see that $\rho = 100$ dimensions suffice to yield a very accurate approximation for the distances. Figure 9 shows the error as a function of ρ on the MNIST training dataset that has $n = 60,000$ points. From the chart, we can see that $\rho = 4,000$ dimensions suffice to yield a very accurate approximation for the distances.

7 Conclusions

We provide a spatially-aware metric between partitions based on a RKHS representation of clusters. We also provide a spatially-aware consensus clustering formulation using this representation that reduces to Euclidean clustering. We demonstrate that our algorithms are efficient and are comparable to or better than prior spatially-aware and non-spatially-aware methods.

Dataset	CSPA	HGPA	MCLA	LiftKm	LiftHAC
IRIS	**0.088**	0.270	0.115	**0.114**	0.125
Glass	**0.277**	**0.305**	0.428	0.425	0.430
Ionosphere	0.422	0.502	**0.410**	0.420	**0.410**
Soybean	0.188	**0.150**	0.163	**0.150**	0.154
Wine	**0.296**	0.374	0.330	0.320	**0.310**
MNIST test data	0.149	-	0.163	**0.091**	**0.110**

Table 3: Comparison of LiftKm and LiftHAC with hypergraph partitioning based consensus methods under the Rand distance (with respect to ground truth). The numbers are comparable across each row corresponding to a different dataset, and smaller numbers indicate better accuracy. The top two methods for each dataset are highlighted.

Dataset	CSPA	HGPA	MCLA	LiftKm	LiftHAC
IRIS	**0.113**	0.295	0.812	**0.106**	0.210
Glass	0.573	**0.519**	0.731	**0.531**	0.540
Ionosphere	**0.729**	0.767	0.993	0.731	**0.720**
Soybean	0.510	0.495	0.951	**0.277**	**0.290**
Wine	0.873	0.875	0.917	**0.831**	**0.842**
MNIST test data	0.182	-	0.344	**0.106**	**0.112**

Table 4: Comparison of LiftKm and LiftHAC with hypergraph partitioning based consensus methods under LiftEMD (with respect to ground truth). The numbers are comparable across each row corresponding to a different dataset, and smaller numbers indicate better accuracy. The top two methods for each dataset are highlighted.

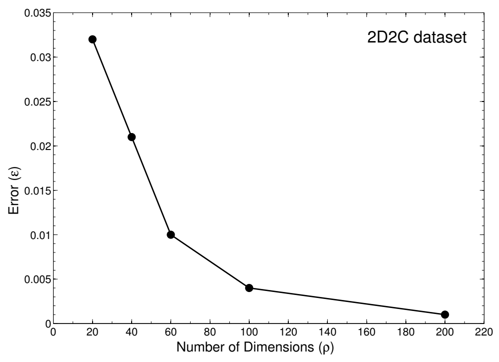


Figure 8: Error in LiftEMD on *2D2C* dataset (45 samples) as a function of ρ.

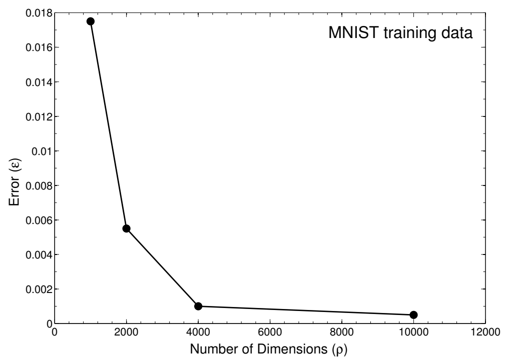


Figure 9: Error in LiftEMD on MNIST training data (60,000 samples) as a function of ρ.

References

[1] M. H. Ansari, N. Fillmore, and M. H. Coen. Incorporating spatial similarity into ensemble clustering. In *1st Workshop on Discovering, Summarizing and Using Multiple Clusterings (MulitClustKDD-2010)*, 2010.

[2] N. Aronszajn. Theory of reproducing kernels. *Trans. American Math. Society*, 68:337 – 404, 1950.

[3] E. Bae, J. Bailey, and G. Dong. A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings. *Data Mining and Knowledge Discovery*, 2010.

[4] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. In *Pacific Symposium on Biocomputing*, 2002.

[5] A. Berlinet and C. Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Netherlands, 2004.

[6] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, and J. M. K. a nd Dina Kravets. Geometric pattern matching under Euclidian motion. *Comp. Geom.: The. and App.*, 7:113–124, 1997.

[7] M. Coen, H. Ansari, and N. Fillmore. Comparing clusterings in space. In *ICML*, 2010.

[8] C. H. Q. Ding and T. Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *ICML*, 2007.

[9] S. Durrleman, X. Pennec, A. Trouvé, and N. Ayache. Sparse approximation of currents for statistics on curves and surfaces. In *MICCAI*, 2008.

[10] E. Fowlkes and C. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[11] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[12] T. Gärtner. Exponential and geometric kernels for graphs. In *NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data*, 2002.

[13] R. Ghaemi, M. N. bin Sulaiman, H. Ibrahim, and N. Mustapha. A survey: clustering ensembles techniques, World Academy of Science, Engineering and Technology 50, 2009.

[14] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Trans. KDD*, 1:4, 2007.

[15] C. R. Givens and R. M. Shortt. A class of wasserstein metrics for probability distributions. *Michigan Math Journal*, 31:231–240, 1984.

[16] J. Glaunès and S. Joshi. Template estimation form unlabeled point set data and surfaces for computational anatomy. In *International Workshop on the Mathematical Foundations of Computational Anatomy*, 2006.

[17] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel method for the two-sample-problem. *JMLR*, 1:1–10, 2008.

[18] P. Indyk and N. Thaper. Fast image retrieval via embeddings. In *Intr. Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003.

[19] S. Jegelka, A. Gretton, B. Schölkopf, B. K. Sriperumbudur, and U. von Luxumberg. Generalized clustering via kernel embeddings. In *32nd Annual German Conference on AI (KI 2009)*, 2009.

[20] S. Joshi, R. V. Kommaraju, J. M. Phillips, and S. Venkatasubramanian. Matching shapes using the current distance. *CoRR*, abs/1001.0591, 2010.

[21] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[22] Y. Lecun and C. Cortes. The MNIST database of handwritten digits.

[23] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *JMLR*, 2:444, 2002.

[24] M. Meilă. Comparing clusterings–an information based distance. *J. Multivar. Anal.*, 98:873–895, 2007.

[25] B. G. Mirkin and L. B. Cherny. Measurement of the distance between distinct partitions of a finite set of objects. *Automation & Remote Control*, 31:786, 1970.

[26] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[27] Y. V. Prokhorov. Convergence of random processes and limit theorems in probability theory. *Theory of Probability and its Applications*, 1(2):157–214, 1956.

[28] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.

[29] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[30] S. Shirdhonkar and D. Jacobs. Approximate Earth mover's distance in linear time. In *CVPR*, 2008.

[31] A. Smola, A. Gretton, L. Song, and B. Schölkopf. A Hilbert space embedding for distributions. In *Algorithmic Learning Theory*, pages 13–31. Springer, 2007.

[32] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. Lanckriet. Hilbert space embeddings and metrics on probability measures. *JMLR*, 11:1517–1561, 2010.

[33] A. Strehl. Clusterpack matlab toolbox. http://www.ideal.ece.utexas.edu/∼strehl/soft.html.

[34] A. Strehl and J. Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2003.

[35] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley Longman, 2005.

[36] M. Vaillant and J. Glaunès. Surface matching via currents. In *Proc. Information processing in medical imaging*, volume 19, pages 381–92, January 2005.

[37] S. Wagner and D. Wagner. Comparing clusterings - an overview. Technical Report 2006-04, ITI Wagner, Informatics, Universität Karlsruhe, 2007.

[38] D. Zhou, J. Li, and H. Zha. A new Mallows distance based metric for comparing clusterings. In *ICML*, 2005.