Kernelized Non-Euclidean Relational c-Means Algorithms

Richard J. Hathaway¹, James C. Bezdek² and Jacalyn M. Huband²
¹Department of Mathematical Sciences, Georgia Southern University
Statesboro, GA  30460
²Computer Science Department, University of West Florida
Pensacola, FL  32514

Abstract
Successes with kernel-based classification methods have spawned many recent efforts to kernelize clustering algorithms for object data. We extend this idea to the relational data case by proposing kernelized forms of the non-Euclidean relational fuzzy (NERF) and hard (NERH) c-means algorithms. We show that these relational forms are dual to kernelized forms of fuzzy and hard c-means (FCM, HCM) that are already in the literature. Moreover, our construction can be done for pure relational data, i.e., even when there is no corresponding set of object data, provided the right type of kernel is chosen. We give an example of this type, as well as three other examples of clustering with the kernelized version of NERF to illustrate the utility of this approach. Two examples show how a visual assessment technique can be used to choose a most useful value for the Gaussian kernel parameter.

Keywords – clustering, relational data, kernel-based learning, Gaussian kernel, non-Euclidean relational fuzzy c-means

1. INTRODUCTION

The use of kernels has greatly enhanced certain pattern recognition methods - e.g., two-subset classifier design with the support vector machine (SVM) (Burges, 1998; Muller et al., 2001). In a nutshell, kernels offer a computationally inexpensive way to simulate a transformation of the original data \( X = \{x_1, \ldots, x_n\} \) in domain space \( \mathbb{R}^s \) to a higher dimensional feature space \( \mathbb{R}^q \) (usually \( s << q \)), where the structure of the transformed
data is simpler. In the case of support vector machines, the underlying desire is to find a function \( \varphi: \mathbb{R}^s \mapsto \mathbb{R}^q \) that maps a pair of crisply labeled, disjoint sets \( X_1, X_2 \subset \mathbb{R}^s \), which are not linearly separable in \( \mathbb{R}^s \), to the sets \( \varphi[X_1], \varphi[X_2] \), which are linearly separable in \( \mathbb{R}^q \). The mapping \( \varphi \) is often called a feature extraction map; and (the coordinates of) the vectors \( \{\varphi(x_i) | x_i \in X_1 \cup X_2\} \) are called the features extracted by the transformation. For clustering algorithms, the expectation is that clustering in the feature space \( \mathbb{R}^q \) will be easier than in the input space \( \mathbb{R}^s \).

How does a kernel facilitate these helpful data transformations? First, kernels can be thought of as similarity measures or as generalizations of the Euclidean dot product

\[
\langle x_i, x_j \rangle = \sum_{k=1}^{s} x_{ik} x_{jk} .
\]

Next, according to Mercer’s Theorem, (see Theorem 2.10 and Proposition 2.11 in Scholkopf & Smola, 2002; or the original statement in Mercer, 1909), under mild regularity assumptions on the kernel, there is a mapping \( \varphi \) from the domain space \( \mathbb{R}^s \) to a finite or infinite dimensional inner product feature space \( \mathcal{H} \) (the default case for finite dimensional feature spaces is that \( \mathcal{H} = \mathbb{R}^q \)) such that

\[
\langle \varphi(x_i), \varphi(x_j) \rangle = k(x_i, x_j) .
\]

The apparent power of this result lies in the fact that we can simulate the computation of the dot product in \( \mathcal{H} \) by simply evaluating a kernel function in \( \mathbb{R}^s \). But the real, hidden power of Mercer’s theorem is that we don’t even need to find the function \( \varphi \). Thus, any algorithm that uses dot product calculations on pairs of its input data (e.g., SVM) can be simulated as if we had first mapped the input data to \( \mathcal{H} \) with \( \varphi \), and then computed the dot products in \( \mathcal{H} \). This technique of replacing dot products in an imagined, higher dimensional feature space with kernel calculations in the input space (or more generally, one kernel with another kernel) is referred to as the "kernel trick".

Four commonly used kernels defined on \( \mathbb{R}^s \times \mathbb{R}^s \) are:

**Gaussian** kernels;

\[
k(x_i, x_j) = \exp[-\|x_i - x_j\|^2 / \sigma^2], \quad (\sigma > 0) \quad , \quad (1a)
\]

**Radial Basis Function (RBF)** kernels;

\[
k(x_i, x_j) = f(d(x_i, x_j)), \text{ where } d(x_i, x_j) \text{ is a metric on } \mathbb{R}^s \times \mathbb{R}^s, \quad f: [0, \infty) \mapsto \mathbb{R} \quad , \quad (1b)
\]

**Hyperbolic tangent** function kernels;

\[
k(x_i, x_j) = 1 - \tanh[-\|x_i - x_j\|^2 / \sigma^2], \quad (\sigma > 0) \quad , \quad \text{and} \quad (1c)
\]

(Homogeneous) **Polynomial** kernels;

\[
k(x_i, x_j) = k(x_i, x_j) = \langle x_i, x_j \rangle^d, \text{ where } d > 0 \text{ is an integer} \quad . \quad (1d)
\]
The Gaussian kernel is a special case of the RBF kernel, and is sometimes referred to as the Gaussian RBF kernel or simply the RBF kernel. Our focus is on the Gaussian kernel, but we mention at places in the following how some other kernels could be used in the proposed relational clustering schemes.

Kernelizing a particular clustering method with a chosen kernel does not guarantee that it will produce better clustering results. However, classification applications have demonstrated the ability of various kernels to create good non-linear decision surfaces in \( \mathbb{R}^s \) for \( c = 2 \) class problems (Scholkopf & Smola, 2002) by solving a simpler (often linear, in principle, via Mercer's theorem) classification problem in the space \( \mathcal{H} \). Additionally, as pointed out by Girolami (2002), continuity and smoothness of the implicitly represented mapping \( \varphi \) should preserve cluster structure, as nearby points in \( \mathbb{R}^s \) would be mapped to nearby points in \( \mathcal{H} \). So, for clustering algorithms, we anticipate that the data, if actually mapped to \( \mathcal{H} \), would have a simpler, more easily separable form that nonetheless preserves the essential nature of the original cluster structure. Before surveying how kernelization has been applied to cluster algorithms, we present a brief discussion of the major types of data used in cluster analysis.

The clustering of a set of \( n \) objects \( O = \{o_1, \ldots, o_n\} \) is typically done on the basis of one of two types of numerical data. Most commonly the available data are a set of numerical object data \( X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^s \), where for each \( i = 1, \ldots, n \), the \( s \) components of the \( i^{th} \) object datum \( x_i \) (aka feature vector \( x_i \)) give measurements of \( s \) features (such as length, density, etc.) of object \( o_i \). In some cases only numerical relational data are available, which typically consist of a set of \( n^2 \) pair-wise similarities or dissimilarities between pairs of objects in \( O \). Relational data are typically arrayed as an \( n \times n \) matrix \( R \), where element \( r_{ij} = \text{dissimilarity}(o_i, o_j) \) (or similarity\((o_i, o_j)) \). Relational data clustering has been of interest to numerical taxonomists for decades (Sneath & Sokal, 1973), and more recently some important types of bioinformatics problems have led naturally to this case (Pal et al., 2005).

Of particular interest to us is prior work on the kernelization of the c-means clustering algorithms, which has so far been done only for the case of object data. The kernelization work is motivated by the fact that these methods can experience difficulties when the clusters are not hyperspherical, or more generally, ellipsoidal in shape. The first kernelization occurred for the hard c-means (HCM) algorithm (aka K-means; MacQueen (sequential K-means), 1967; Ball & Hall (batch K-means), 1967) and is attributed to an unpublished version by Scholkopf et al. (1998) that is available at http://www.kernel-machines.org/. Kernelized K-means is also discussed in Zhang & Rudnicky (2002) and Girolami (2002). Girolami (2002) gave interesting probabilistic interpretations of the clustering criterion when the Gaussian kernel is used. He offered two important conclusions, both of which apply to kernelizing all c-means algorithms:
(1) "employing a kernel representation of the data removes the implicit assumption of hyperspherical or ellipsoidal clusters in data space" (page 781); and (2) "in the specific case of data partitioning then a kernel which will have universal approximation qualities such as the RBF is most appropriate" (page 784). (RBF in the second quote actually refers to the Gaussian kernel.) The simulation results given in Girolami (2002) using the Gaussian kernel were impressive, and included 100% accuracy in clustering a 2-dimensional "ring" data set consisting of a center cluster surrounded completely by a second cluster consisting of an annular ring of data (cf. Figure 2, Section 3, where we include a similar example for the relational case).

Our emphasis is on the fuzzy case, and the fuzzy c-means (FCM) algorithm (Bezdek, 1981) for object data, which has been kernelized in at least 3 different ways. We point out that different kernelizations are possible, in part, because there are multiple places in the derivation or statement of a c-means algorithm that the kernelization trick can be applied. The first kernelization approach of Zhang & Chen (2002) is completely general regarding the type of kernel, but unfortunately, requires the inversion of the potentially large kernel matrix $K = [k_{ij}] = [k(x_i,x_j)]$. A second kernel-based approach due to Zhang & Chen (2003a) is more attractive computationally and produces an iteration very similar to the original FCM iteration, including cluster "means" in the original domain space $\mathbb{R}^S$. This algorithm was derived specifically for the case of the Gaussian kernel, but the authors suggest it can be effectively used with other kernels satisfying $k(x_i,x_i) = 1$. Their test results using incomplete data sets give evidence that the Gaussian and other RBF kernels work well. The second approach is extended in Zhang & Chen (2003b) to include a kernelized version of the possibilistic c-means approach found in Krishnapuram & Keller (1993). As with all the earlier experiments, good results were obtained using the Gaussian kernel.

The third kernelization of FCM was done by Wu et al. (2003) and further tested by Kim et al. (2005). This kernelization is valid for any choice of kernel and can be efficiently computed without the inversion of the kernel matrix $K = [k_{ij}] = [k(x_i,x_j)]$. We do not include a proof here, but this kernelization of FCM produces the identical terminal data partition (and partition iteration sequence) as the relational fuzzy c-means (RFCM) algorithm in Hathaway et al. (1989) if the kernel $k(x_i,x_j) = \langle x_i, x_j \rangle$ is used. However, this choice of kernel corresponds to performing the identity transformation on the domain data, and thus, is uninteresting from the standpoint of improving FCM clustering performance by kernelization. Kim et al. (2005) reported very good results for this third kernelization of FCM using the Gaussian kernel. We expect similar good results for the relational version given later - and, we get them.

In the next section we show how relational data can be kernelized directly using the Gaussian kernel, whether or not any underlying object data is available. Then a detailed
statement of the new fuzzy relational algorithm is given and its duality with the object data approach of Wu et al. (2003) is established. The hard clustering case is also addressed, but in less detail. Section 3 briefly reviews the applicable (through duality) computational results from Kim et al. (2005) and then gives several new examples. The final section consists of a discussion and concluding remarks.

2. KERNELIZED NON-EUCLIDEAN RELATIONAL c-MEANS

All of the prior work done to kernelize c-means algorithms is for the case of object data. Suppose instead that the only available data describing the set of objects \( O = \{o_1,\ldots, o_n\} \) are relational dissimilarity data represented as a symmetric \( n \times n \) matrix \( D = [d_{ij}] = [\text{dissimilarity}(o_i, o_j)] \) with \( d_{ii} = 0 \) for \( i = 1,\ldots, n \) and \( d_{ij} = d_{ji} \geq 0 \) for \( 1 \leq i \neq j \leq n \). Is it possible to directly transform \( R \) in a way that corresponds to Gaussian kernelization of object data? Yes. First consider the special case where object data \( X = \{x_1,\ldots, x_n\} \) is in \( \Re^s \), with \( d_{ij} = \|x_i - x_j\|^2 \). We will use this special case to guide our strategy for kernelization of \( D \), but the final formula we derive can be used whether or not \( D \) admits a corresponding set of object data this way. Now suppose we want to transform this object data set \( X \) using an (implicit) transformation \( \varphi \) represented by a Gaussian kernel \( k(x_i, x_j) \) defined on \( \Re^s \). The transformed relational data matrix \( \hat{D} = [\hat{d}_{ij}] \) will have the form:

\[
\hat{d}_{ij} = \|\varphi(x_i) - \varphi(x_j)\|^2 = \left(\varphi(x_i) - \varphi(x_j)\right)^T \left(\varphi(x_i) - \varphi(x_j)\right) = \varphi(x_i)^T \varphi(x_i) - 2 \varphi(x_i)^T \varphi(x_j) + \varphi(x_j)^T \varphi(x_j) = k(x_i, x_i) - 2k(x_i, x_j) + k(x_j, x_j) = 1 - 2 \exp[-\|x_i - x_j\|^2/ \sigma^2] + 1 = 2 - 2 \exp[-d_{ij}/ \sigma^2].
\]

Equation (2) enables us to form the Gaussian kernelized relational matrix \( \hat{D} = [\hat{d}_{ij}] = [2 - 2 \exp[-d_{ij}/ \sigma^2]] \) directly from the original relational data matrix \( D \). More generally, if we have any RBF in (1b) with metric \( d \) satisfying \( d(x_i, x_j) = g(\|x_i - x_j\|^2) \) for a function \( g: [0,\infty) \rightarrow [0,\infty) \), then this approach could be used to give (RBF) kernelized data of the form \( 2g(d(x_i, x_i)) = 2g(0) \). Note that this derivational approach can also be used for the hyperbolic tangent kernel in (1c), but will not work with (1d).

Now that we have a kernelized form of (arbitrary) relational data, we can kernelize the relational fuzzy and hard c-means algorithms by performing the clustering analysis.
directly on the kernelized distance matrix \( \hat{D} = [\hat{d}_{ij}] = 2 - 2 \exp[-d_{ij}/\sigma^2] \). We give a
detailed discussion of the fuzzy case, based on kernelizing the non-Euclidean relational fuzzy (NERF) c-means algorithm from Hathaway & Bezdek (1994), and then note the
adjustments required to obtain the hard version. A partitioning of the data (or objects)
into c clusters is represented by a c×n partition matrix U. The sets of (nondegenerate)
fuzzy and hard c-partitions of n objects are denoted by \( M_{fcn} \) and \( M_{hcn} \):

\[
M_{fcn} = \left\{ U \in \mathbb{R}^{c \times n} | u_{ik} \in [0,1]; \sum_{i=1}^{c} u_{ik} = 1; \forall i; \sum_{k=1}^{n} u_{ik} > 0; \forall k; \right\}
\]

and

\[
M_{hcn} = \{ U \in M_{fcn} | u_{ik} \in \{0,1\} \}.
\]

The element \( u_{ik} \) of a partition matrix U represents the degree or extent to which
object \( o_k \) (or datum \( x_k \)) belongs to cluster i. The crucial difference between the two sets
in (1) is that fuzzy partitions, which were first used by Ruspini (1969), allow
memberships in \([0,1]\), so that (partial) membership of a datum can be shared between
clusters, while hard partitions require membership values to be 0 or 1, so each datum is
unequivocally placed into one and only one of the c clusters. In the statement of the
kernelized NERF (kNERF) algorithm that is given next, \( e_k \) denotes the \( k \)th unit vector in
\( \mathbb{R}^n \), and M is the \( n \times n \) matrix with 0's on the main diagonal and 1's elsewhere.

**Algorithm kNERF**: (Fuzzy clusters in kernelized dissimilarity matrix \( \hat{D} \))

**Inputs**: An \( n \times n \) dissimilarity matrix D

**Constraints**: D satisfies \( d_{ij} \geq 0; d_{ij} = d_{ji}; d_{ii} = 0, \alpha \in \mathbb{R}, M_{ij} = 1 \ \forall i, j \)

**Choose**: \( c = \# \) of clusters, \( 2 \leq c < n \)

\( m = \) fuzzy weighting exponent, \( m > 1 \)

\( \varepsilon_L = \) termination criterion

\( \|U^{(q)} - U^{(q-1)}\| = \) termination norm on successive estimates of matrix U

\( Q_M = \) maximum number of iterations allowed

Gaussian parameter \( \sigma > 0 \)

**Kernelize D**: \( \hat{D} = [\hat{d}_{ij}] = 2 - 2 \exp[-d_{ij}/\sigma^2] \)

**Initialize**: \( q = 0; \beta = 0; U^{(0)} \in M_{fcn} ; U_{difference} = 2*\varepsilon_L \)

**WHILE** (\( U_{difference} > \varepsilon_L \) AND \( q < Q_M \))

**kNERF-1** Calculate c "mean" vectors \( v_i^{(r)} \)
\[ v_i^{(q)} = ((u_{i1}^{(q)})^m, (u_{i2}^{(q)})^m, \ldots, (u_{in}^{(q)})^m)^T / \sum_{j=1}^{n} (u_{ij}^{(q)})^m \], \quad 1 \leq i \leq c \quad (4) \]

**kNERF-2** Calculate object to cluster "distances"

\[ \delta_{ik} = (\hat{D} + \beta M)v_i^{(q)})_k - (v_i^{(q)})^T (\hat{D} + \beta M)v_i^{(q)}) / 2, \quad 1 \leq i \leq c; \quad 1 \leq k \leq n \quad (5) \]

**IF** \( \delta_{ik} < 0 \) **THEN**

Calculate \( \Delta \beta = \max_{i,k} \left\{ -2\delta_{ik} \left\| v_i^{(q)} - e_k \right\|^2 \right\} \)

Update \( \delta_{ik} \leftarrow \delta_{ik} + (\Delta \beta / 2) \cdot \left\| v_i^{(q)} - e_k \right\|^2, \quad 1 \leq i \leq c; \quad 1 \leq k \leq n \)

**ENDIF**

**kNERF-3** Update \( U^{(q)} \) to \( U^{(q+1)} \) \( \in M_{fcn} \) to satisfy for all \( k = 1, \ldots, n \):

**IF** \( \delta_{ik} > 0 \), \( i = 1 \) to \( c \) **THEN** \( u_{ik}^{(q+1)} = 1 / \left[ \sum_{j=1}^{c} (\delta_{ik} / \delta_{jk})^{(m-1)} \right] \quad (6a) \)

**ELSE**

Set \( u_{ik}^{(q+1)} = 0 \) for all \( k \) with \( \delta_{ik} > 0 \), and \( u_{ik}^{(q+1)} \in [0,1] \) such that \( \sum_{j=1}^{c} u_{jk}^{(q+1)} = 1 \quad (6b) \)

**ENDIF**

Update \( q \leftarrow q + 1 \)

Calculate \( U_{\text{difference}} = \left\| U^{(q)} - U^{(q-1)} \right\| \)

**END WHILE**

**Outputs**: Membership matrix \( U_{kNERF} \in M_{fcn} \); "prototype" vectors \( \{v_1, \ldots, v_c \} \subset \mathbb{R}^n \)

A much less imposing specification for kNERF would be: step (i): create \( \hat{D} = [\hat{d}_{ij}] = 2 - 2 \exp[-d_{ij}/\sigma^2] \) from the original dissimilarity data \( D \); step (ii): apply NERF to \( \hat{D} = [\hat{d}_{ij}] \). This is a good place to remark that, when we take the limit of \( 2 - 2 \exp[-d_{ij}/\sigma^2] \) as \( \sigma \to \infty \), \( \hat{D} \to 0 \). However, the Taylor series expansion of \( 2 - 2 \exp[-d_{ij}/\sigma^2] \) is \( (2d_{ij}/\sigma^2) - (2d_{ij}^2/\sigma^4) + (2d_{ij}^3/\sigma^6) - \cdots \). For large values of \( \sigma \), the first term of this series dominates all other terms, and so \( \hat{d}_{ij} \approx 2d_{ij}/\sigma^2 \). Thus, for
large enough \( \sigma, D \) and \( \hat{D} \) are proportional, and consequently, \( k\text{NERF} \) will produce exactly the same clusters as NERF.

The theory of the clustering steps for NERF (\( k\text{NERF}-1 \) through \( k\text{NERF}-3 \)) is covered in detail in Hathaway & Bezdek (1994). The main result is that the sequence of partition matrices produced by FCM on a set of object data \( X \) is identical to the sequence of partition matrices produced by NERF on the corresponding relational matrix of pairwise squared Euclidean distances derived from \( X \), i.e., \( [d_{ij}] = \left\| x_i - x_j \right\|_2 \). The NERF algorithm is called non-Euclidean, however, because the use of the spread parameter \( \beta \) enables the algorithm to continue calculations when the data are non-Euclidean. This is accomplished by adding the dynamically calculated spread term \( \beta M \), as needed, to the off diagonal elements of \( D \) (\( \hat{D} \) for \( k\text{NERF} \)). The addition \( \beta M \) in (5), when triggered at \( (5^* ) \), produces adjusted relational data that is very nearly Euclidean, while preserving most of the cluster structure of the original \( D \) (\( \hat{D} \) for \( k\text{NERF} \)).

Just as NERF has a duality relationship with the FCM algorithm, \( k\text{NERF} \) has the same type of duality relationship with the kernelized form of FCM devised by Wu et al. (2003) and tested by Kim et al. (2005). We will refer to the particular kernelizations of FCM and HCM tested in Kim et al. (2005) throughout the remainder of this article as \( k\text{FCM} \) and \( k\text{HCM} \). This means that the terminal partition produced by \( k\text{FCM} \) on \( X = \{x_1,...,x_n\} \) for a particular Gaussian kernel \( k(x_i,x_j) \) is identical to the terminal partition produced by \( k\text{NERF} \) using the same kernel and original relational data \( D = [d_{ij}] = \left\| x_i - x_j \right\|_2 \). We show this by demonstrating that the calculated object-to-cluster distances for the two algorithms are the same, which will force the partition matrix iterates to be identical. In the following let \( E \) denote the matrix whose entries all equal 1 and \( K = [k(x_i,x_k)] \) denote a Gaussian kernel matrix. Note that \( \beta = 0 \) for this relational data since it is Euclidean, \( k(x_i,x_j) = 1 \), and \( \sum_{k=1}^{n} (v_{ik}) = 1 \) because of the normalization in (4). Suppressing the iterate number notation, we substitute (4) into (5) and rewrite it as:

\[
\delta_{ik} = (\hat{D}v_i)_k - v_i^T \hat{D}v_i / 2 = ((2E - 2K)v_i)_k - v_i^T (E - K)v_i
\]

\[
= 2 - 2 \sum_{j=1}^{n} u_{ij}^m k(x_k,x_j) / \sum_{j=1}^{n} u_{ij}^m - 1 + \sum_{h=1}^{n} \sum_{j=1}^{n} u_{ih}^m u_{ij}^m k(x_h,x_j) / \left( \sum_{j=1}^{n} u_{ij}^m \right)^2
\]

\[
= k(x_k,x_k) - 2 \sum_{j=1}^{n} u_{ij}^m k(x_k,x_j) / \sum_{j=1}^{n} u_{ij}^m + \sum_{h=1}^{n} \sum_{j=1}^{n} u_{ih}^m u_{ij}^m k(x_h,x_j) / \left( \sum_{j=1}^{n} u_{ij}^m \right)^2. \quad (7)
\]
The expression in (7) is identical to expression (6) in Kim et al. (2005), which gives the calculated object-to-cluster distances for kFCM, thus providing the duality result that we now formally state.

**Duality of kFCM and kNERF.** If relational and object data $D$ and $X$ satisfy $D = [d_{ij}] = [\|x_i - x_j\|_2^2]$, then the terminal partitions produced by kFCM using $X$ and kNERF using $D$ are identical when both algorithms are applied using the same initialization $U^{(0)}$ and parameters $\sigma, c, m, \varepsilon$, and $Q_M$. Since our emphasis is on kernelization of the fuzzy approach, we give only a very brief description of the hard (crisp) case by pointing out how it differs from kNERF. In contrast to kNERF, *kernelized non-Euclidean relational hard* (kNERH) $c$-means algorithm attempts to find a hard partition $U$ in $M_{hc}$ that accurately represents cluster structure in the data. An important practical difference is that hard $c$-means algorithms are often much more sensitive to initialization than their fuzzy counterparts (i.e., they may terminate at a larger number of different partitions depending on the initial guess of $U$). The kNERH algorithm is obtained by making the following 2 changes to the kNERF algorithm given above: (1) use $m = 1$ in the $v_i$ update equations of (4) and replace the fuzzy $U$ update in (6) with the hard $U$ update:

$$u_{ik}^q = \begin{cases} 1 & \text{if } \delta_{ik} < \delta_{jk} \forall j \neq i \forall i, k, \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Using the same type of manipulation that established (7), it can be shown that kNERH shares a duality relationship with the kHCM algorithm tested in Kim et al. (2005). This is easily established showing that $\delta_{ik}$ in (5) above is equal to expression (2) in Kim et al. 2005. The duality relationships for both kNERF and kNERH are exploited in the next section to take advantage of some previous simulations of kFCM and kHCM which appear in the literature.

### 3. NUMERICAL EXPERIMENTS

In this section we review some previous results from the literature applicable to kNERF and kNERH, and then add new numerical experiments for kNERF to further demonstrate that kernelization is beneficial to the relational $c$-means algorithms for certain types of problems. The new numerical experiments are discussed in three parts. First we report simulation on a small artificial data set to verify that kernels give us a simple way to improve the behavior of NERF clustering in some cases when the object dissimilarities vary greatly within a given cluster. The corresponding object data form clusters that are highly elongated and not hyperspherical. Second, we will closely examine two examples
– one difficult and one easy – derived from object data that enable us to study the
dependence of kNERF on the "variance" parameter $\sigma$ used when kernelizing $D$ with
(1a). The fourth example demonstrates the kNERF approach on relational data which is
not derived from object data at all (and so, relational clustering is the only choice). This
data represents the driving times between two sets of cities that are (roughly) annular, in
the sense that one subset of the cities is in a tight central cluster (on the map), while the
second subset of cities surround this inner cluster in a circular fashion.

We do not attempt to devise methods to automatically select the number of clusters $c$
and kernel parameter $\sigma$, but these are certainly important, difficult problems. Selection
of $\sigma$ is discussed in Muller et al. (2001) and Girolami (2002), but there is apparently not
yet an accepted way to select good parameter values for general unlabeled data (and this
is one objective of our Examples 2 and 3). The problem of identifying the number of
clusters $c$ (cluster validity) has been widely studied. See Bezdek & Hathaway (2002) for
a recent image-based approach somewhat related to that given in Girolami (2002) and
Zhang & Chen (2003a).

Because of the duality relationship between the pairs (kHCM/kNERH) and
(kFCM/kNERF), we can infer some behavioral traits of kNERH and kNERF from
examples which already exist in the literature that use their duals - specifically, in Kim
et al. (2005). Kim et al. tested (among others) HCM, FCM, kHCM and kFCM on a set of
10 problems. Four of the problems involved experimental object data sets (e.g., Fisher's
Iris data) while the others used artificially generated data sets in $\mathbb{R}^2$ or $\mathbb{R}^3$. Each of the
10 data sets was labeled so that the algorithms' terminal fuzzy partition matrices could be
used to assess "clustering accuracy", which was defined as the percentage of the data set
that is correctly clustered. In this assessment of accuracy, the terminal fuzzy partitions
(for FCM and kFCM) are first "hardened", based on maximum membership, to provide a
crisp $c$-partition of the data. For example, if $[.35 .45 .20]^T$ is the $k^{th}$ column of the fuzzy
terminal fuzzy partition $U$, then its hardened form is $[0 \ 1 \ 0]^T$ and $x_k$ (or object $o_k$) is
assigned to cluster 2. The hardened partition is then compared with the known
(physically collected or mathematically observed) labels to compute the percentage of
data correctly clustered. This same method of assessing accuracy is used in the new
experiments involving kNERF reported later. Terminal partition matrices produced by
HCM and kHCM do not require the hardening step.

For each of the 10 data sets, Kim et al. (2005) conducted experiments using $\sigma = 0.1,$
0.2, 0.3, ..., 2.0, and reported only the best accuracy obtained. The kFCM and kHCM
algorithms averaged respective clustering accuracies of 89.73% and 89.27% over the 10
data sets; while FCM and HCM averaged only 74.39% and 73.60%, respectively. The
kernelized methods did better in each of the 10 examples, with improvements in
clustering accuracy ranging from gains of 1.5% up to 40.61% for kFCM, and from 1.11%
up to 49.24% for kHCM. By the duality discussed earlier, kNERF and kNERH will have
(exactly) the same advantage over NERF and NERH if those experiments are replicated in relational data space using the same initialization and stopping criteria. In other words, the existing study of Kim et al. establishes, via duality, the benefits of kernelizing the relational c-means algorithms for some problems.

Four new experiments for kNERF are described next. All computations were done using version 5.3 of MATLAB on a PC. Random number generators from the statistical toolbox and image display routines from the image processing toolboxes were used. In all experiments we chose NERF parameter values $\varepsilon_L = 0.00001$, $m = 2$ and $Q_M = 1000$. The Gaussian kernel was used in all applications of kNERF with the various values of the parameter $\sigma$ indicated below.

**Example 1. Mixture of Bivariate Normals Simulation**

This first example demonstrates the potential usefulness of kNERF when the relational dissimilarities between objects vary greatly within clusters. We use relational data derived from a mixture of 2 bivariate normal distributions, with component: mixing proportions $\alpha_1 = \alpha_2 = 0.5$; means $\mu_1 = [-3 \ -5]^T$ and $\mu_2 = [3 \ 5]^T$; and covariance matrices

$$
\Sigma_1 = \Sigma_2 = \begin{bmatrix}
100 & 0 \\
0 & 0.10
\end{bmatrix}.
$$

A scatter plot for a typical sample of size $n = 100$ from this normal mixture is shown in Figure 1. The clusters are highly elongated because the variance in the first direction is 1000 times as great as it is in the second direction. Geometric configurations of this type cause difficulty for (unscaled) c-means algorithms using the standard choices for inner product induced norms, because the objective function has a natural affinity for hyperspherical clusters. Once we know that the data has the structure seen in Figure 1, we could use the covariance matrix above to induce a Mahalonobis norm on $\mathbb{R}^2$, and this would equip the c-means objective function so that it had a good chance of finding these clusters. But of course, we might have this type of structure in a higher dimensional space, and would have no a priori means for making this decision.

![Figure 1. A typical sample from the mixture of $c = 2$ bivariate normals](image-url)
The simulation consisted of 100 trials, each using a different normal mixture sample of size \( n = 100 \). For each trial, a corresponding \( 100 \times 100 \) relational data matrix \( D = [d_{ij}] = [\|x_i - x_j\|^2] \) was calculated using the normal mixture sample. Each data matrix \( D \) was clustered using NERF and kNERF for \( \sigma = 1, 1.25, 1.50, 1.75, \ldots, 25 \). Each application of NERF was initialized using the "true" hard 2-partition, which was recorded at the time each random sample was generated. Each application of kNERF was initialized using the terminal fuzzy partition produced by NERF. Thus, any improvement in the results demonstrated by kNERF is due entirely to the kernelization of \( D \); improvements are not artifacts of the initialization at the true hard partition.

The average (over the 100 trials) clustering accuracy of NERF was 67.51%. For some perspective on how poor this is, we note that ignoring the data values and grouping each data set into a single cluster gives an average accuracy of about 50%. The average clustering accuracy of kNERF (using the best-performing \( \sigma \) for each sample) was 89.46%, which is an accuracy gain of about 22% over NERF. The average of the best-performing \( \sigma \) values was 7.3.

**Example 2. A Ring Example (Difficult for NERF)**

This is the first of two examples that studies the quality of kNERF clustering as a function of the Gaussian kernel parameter \( \sigma \). This type of experimentation has not received much attention in the literature, and we think that Examples 2 and 3 bear an important message for kernelization aficionados. The relational data was derived as the pair-wise squared Euclidean distances between the 50 points shown in Figure 2. The data set consists of 2 visually apparent clusters, arranged as an inner core and outer annular ring. The pattern of "+" and "o" symbols shows the clustering obtained by NERF, which achieves a cluster accuracy of 74%. (The 13 "+" points on the outer ring are mislabeled).

![Figure 2. NERF clustering of the ring: Example 2](image)
Using the (74% accurate) terminal partition produced by NERF for initialization, we ran kNERF for \( \sigma = 0.02, 0.04, 0.06, \ldots, 2.0 \), and graphically recorded its accuracy in Figure 3. The horizontal dashed line in Figure 3 indicates the 74% accuracy level of NERF. The circles are the values, fit with a piecewise linear function to make the behavior of kNERF a little easier to see. The accuracy is low for very small values of \( \sigma \). For example, accuracy is 62% at the first plotted value, \( \sigma = 0.02 \). As \( \sigma \) increases towards 0.10 from below, the accuracy of kNERF approaches 100%. For \( 0.10 \leq \sigma \leq 0.46 \), kNERF achieves and maintains 100% accuracy. For \( \sigma > 0.46 \), the accuracy decreases until it matches the NERF accuracy (74%) at \( \sigma = 1.36 \). kNERF and NERF produce the same 13 mistakes for all values of \( \sigma > 1.36 \). Thus, we see there is an interval for \( \sigma \) over which kNERF reproduces the exact "true" 2-partition of the Ring data. For the significant values of \( \sigma \) we include Visual Assessment of (Cluster) Tendency (VAT) images (Bezdek & Hathaway, 2002) of the kernelized relational data matrices \( \hat{D} \) in Figure 4.

The four views in Figure 4 are VAT images of \( \hat{D} \) (after reordering by VAT) of the four cases corresponding to \( \sigma = 0.02, \sigma = 0.10, \sigma = 0.46 \) and \( \sigma = 1.36 \). In these images black represents small pairwise dissimilarities and white represents large dissimilarities. The VAT algorithm reorders the columns and rows of any dissimilarity matrix using a modification of Prim's algorithm so that similar objects are reordered to be near one another. This causes well separated clusters to appear as dark blocks along the main diagonal in the reordered display. The VAT image for \( \sigma = 0.02 \) (62% accuracy, Figure 4a) indicates that almost every object is isolated, which implies that kNERF is not finding 2 clusters. Figure (4b) is the VAT image of \( \hat{D} \) when kernelized using \( \sigma = 0.10 \),
the lower threshold at which kNERF achieves 100% accuracy, and this is clear from the visual evidence, which indicates two fairly distinct, dark diagonal blocks, corresponding to the 2 clusters.

Figure 4. VAT images of $\hat{D}$ for the ring: Example 2

Notice that the type of cluster can be "seen" in view (4b): the inner core is a spherical cluster with a lot of similar intracluster distances, hence the large square lower block. The upper block corresponds to the points arranged around the core in the outer ring. This long stringy cluster is mirrored by the thin, long, linear block of dark points along the upper diagonal in Figure (4b). Figure (4c) shows the VAT image for the upper end of the 100% regime. The two clusters are still very apparent in this image, as they are for every $\sigma$ in the interval $[0.10, 0.46]$. Finally, Figure (4d) shows the VAT image of the kernelized data at $\sigma = 1.36$, the point at which kNERF has degenerated back to the same level of accuracy that NERF achieves. The two clusters are still evident in this view, but off diagonal entries are much darker, indicating that there is a lot more mixing of individual points, creating confusion about the memberships in the two clusters.
Example 3. A Normal Mixture Example (Easy for NERF)

We use relational data derived from a mixture of $c = 3$ normal distributions on $\mathbb{R}^4$, with mixing proportions $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$; means $\mu_1 = [1 \ 1 \ 1 \ 1]^T$, $\mu_2 = [4 \ 4 \ 4 \ 4]^T$, and $\mu_3 = [7 \ 7 \ 7 \ 7]^T$; and covariance matrices $\Sigma_1 = \Sigma_2 = \Sigma_3 = I$. A random sample of size $n = 300$ was generated, and a scatter plot of the first two features (coordinates) of the sample is shown in Figure 5. The entries of the $300 \times 300$ relational data matrix $D$ were calculated as the pairwise squared Euclidean distances of the points in $\mathbb{R}^4$. The NERF algorithm, using the true class label partition as its initialization, correctly clustered all but one of the objects (success rate of 99.67%).

![Figure 5. Scatter plot of first two features for Example 3](image)

There is very little room for improvement, but kNERF will make the improvement for various experimentally determined values of the kernel parameter $\sigma$. kNERF was initialized using the NERF terminal partition and applied to the kernelized relational data $\hat{D}$ with $\sigma = 0.1, 0.15, 0.20, ..., 4.0$. The results are shown in Figure 6. 100% accuracy is first attained by kNERF at the value $\sigma = 0.35$. Then … disaster strikes! The accuracy of kNERF plummets to 76% in the neighborhood $\sigma = 0.70$, but then slowly recovers to the NERF accuracy of 99.76% at $\sigma = 1.60$. This interesting behavior shows that unnecessary kernelization can turn an easy problem into a difficult one (at least for certain kernel parameter values). As $\sigma$ continues to increase, kNERF enters a second 100% accuracy regime centered near $\sigma = 3.0$.

VAT images for some significant $\sigma$ values are shown in Figure 7. Notice the clarity of the 3 cluster structure for $\sigma = 3.0$ (one of the values at which kNERF enjoys 100% accuracy), and how each point appears to be isolated when $\sigma = 0.7$. The final image in Figure 7 gives the VAT image of the original (unkernelized) relational matrix.
Figure 6. kNERF: labeling accuracy (vertical) vs. $\sigma$ (horizontal) for Example 3

(7a) $\sigma = 0.70 : \text{kNERF} = 76\%$

(7b) $\sigma = 1.60 : \text{kNERF} = 99.67\%$

(7c) $\sigma = 3.00 : \text{kNERF} = 100\%$

(7d) VAT image of unkernelized $\hat{D}$

Figure 7. VAT images of $\hat{D}$ for the normal mixture: Example 3
The VAT images in Figures 4 and 7 give us some insight into what kNERF is able to "see" in (kernelized) relational data space. We imagine that each choice of $\sigma$ is analogous to a different prescription for eyeglasses. A very small value of $\sigma$ corresponds to glasses that focus only on very nearby objects; objects even a moderate distance away are effectively out of sight. As $\sigma$ gets larger, the glasses enable kNERF to see more and more objects at greater distances from each other. The best choice of $\sigma$ is one that allows good vision from each point to other points in its cluster, but poor vision between points in different clusters. In other words, we get the best accuracy by choosing a kernel parameter that effectively enables kNERF to see compact, separated clusters in the sense of Dunn (1973).

**Example 4. Driving Times Example (pure relational data)**

The previous examples all used data derived from object data, and this is certainly a case when kernelization in the relational domain can be used. In fact, it can be computationally efficient in this instance to choose the relational approach if the dimension of the object data $s$ is larger than the number of object data $n$. One advantage of using relational data derived from low dimensional object data in our examples is that the nature of the clusters can be easily understood using an appropriate scatter plot. On the other hand, we would like to know that the good properties of kNERF observed for object-derived relational data also hold for other types of relational data. To address this, we include an example based on relational data collected from http://maps.yahoo.com/ on 2/15/05, consisting of the intercity driving times (in minutes) for 20 Georgia cities: Carrollton, Griffin, Cartersville, Bremen, Canton, Newnan, Covington, Dallas, Buford, Lawrenceville, North Atlanta, Decatur, North Decatur, College Park, Panthersville, Hapeville, East Point, Stratford, Sylvan Hills, and Atlanta. There are $c = 2$ natural clusters for this set of cities which become apparent by looking at a map. The last 10 cities in the list are all inside the I-285 loop around Atlanta, while the first 10 are half an hour or more outside of the loop. Although we do not have any object data corresponding to the relational data, the physical situation for these cities is fundamentally similar to the geometric structure of the ring data in Example 2. Consequently, we expect kNERF to demonstrate some advantage over NERF when processing this data.

Using the same kNERF parameters used in earlier experiments, we applied NERF, initialized with the true cluster labels, and then kNERF, initialized with the terminal fuzzy partition produced by NERF. We ran kNERF for parameter values $\sigma = 1, 1.5, 2.0, \ldots, 30$ and obtained the clustering accuracies depicted in Figure 8. The horizontal line in Figure 8 indicates that NERF performs very respectably with an accuracy of 90%. KNERF starts off at 100%, but again shows a dip in accuracy before it achieves 100% accuracy for kernel parameter values $\sigma = 4.5$ to 14. Then kNERF drops to a second,
lower stable state at 95% for $\sigma = 14$ to 25.5. Finally, kNERF drops to the NERF accuracy level at $\sigma = 25.5$, and there it remains.

![Figure 8. kNERF : labeling accuracy (vertical) vs. $\sigma$ (horizontal) for Example 4](image)

The VAT images (with rows and columns ordered exactly as the cities were listed above) of the kernelized $\hat{D}$'s for several significant $\sigma$ values are shown in Figure 9. The images are somewhat difficult to interpret because of their coarseness (due to the small sample size), but we see the same type of pattern emerge that was seen in the views of Figure 4. The last image in Figure 9 corresponds to the original (unkernelized) set of driving times, and we see it is virtually indistinguishable from the kernelized form with a parameter value of $\sigma = 14$.

![Image of kernelized D for different $\sigma$ values](image)

(9a) $\sigma = 2.5$ : kNERF = 80%   
(9b) $\sigma = 4.5$ : kNERF = 100%
We have presented two new kernelized forms of the non-Euclidean relational c-means algorithms; kNERF and kNERH. The kNERF algorithm complements the 3 existing (object data) kernelizations of fuzzy c-means and has a duality relationship with the kFCM algorithm introduced in Wu et al. (2003) and tested by Kim et al. (2005). One of the most interesting aspects of the kNERF and kNERH algorithms is that they provide (Gaussian) kernelized results directly from relational data. This is so independently of whether or not the relational dissimilarities might be calculated from object data. The potential effectiveness of kNERF was demonstrated using a ring data example, two normal mixtures, and a set of relational data corresponding to intercity driving times amongst 20 cities in Georgia. The most dramatic improvement resulting from kernelization was demonstrated in the ring data example, where the cluster accuracy of 74% for NERF was improved to 100% by kNERF for an experimentally determined range of values of the kernelizing parameter $\sigma$. We believe that there are numerous situations where kNERF can be helpful.

Because $D$ and $\hat{D}$ are proportional for large enough $\sigma$, it is always possible to do at least as well with kNERF as with NERF by simply taking the Gaussian kernelization parameter $\sigma$ to be sufficiently large. Since the only difference between NERF and kNERF is the kernelization of $D$ to $\hat{D}$, this may suggest at first glance that whenever NERF is the algorithm of choice, kNERF is a better choice. However, this is a bit dangerous, for as we have seen in Figures 6 and 8, there are also values for $\sigma$ that result in kNERF being less accurate than NERF. Thus, some experimentation with $\sigma$ may be required to get into a regime where "kNERF vision", as determined by $\sigma$, is better than...
"NERF vision". Since real clustering problems always come with unlabeled data, we will not be able to perform studies like those represented by Figures 3, 6 and 8, which enable us to select an "optimal" value for $\sigma$. But, we think that using VAT images such as those shown in Figures 4, 7 and 9 will point us towards good choices for $\sigma$.

As mentioned earlier, it was not the intent of this work to develop a scheme for accurately selecting the number of clusters $c$ or the kernelization parameter $\sigma$, but our experiments show that VAT images may be useful approaches to both problems. With real, unlabeled data, VAT could be applied to the (unkernelized) relational data matrix (or to a sample of it if the entire matrix is too large). Then, using the reordering obtained from VAT, kernelized forms of the relational matrix could be displayed as a function of $\sigma$ (note especially that we do not need to specify a value for $c$ during these experiments), until one is found that shows a clear cluster structure. When (and if) this happens, we can extract two things: a good guess for $c$, and a good value for $\sigma$. While the earlier work of Girolami (2002) notes that specially ordered images of the kernel matrix can give some information about the number of clusters, we point out that VAT actually includes a mechanism for producing the useful ordering that is required to see the structure in the relational dissimilarity data. We plan to research this topic next, understanding that much more experimentation is needed in order to usefully correlate the appearance of a VAT image to the quality of the results produced by kNERF or other clustering algorithms.

Finally, we note that the non-Euclidean relational possibilistic (NERP) c-means algorithm mentioned in Hathaway et al. (1996) can also be kernelized in exactly the same way as NERH and NERF. One complication (or maybe advantage?) in the NERP case is that application of the possibilistic approach requires the determination of a reference distance, which in turn depends on the scale and dispersion of the data. In this case, the calculations could be simulated in the transformed space via the kernel, if this strategy is advantageous.

REFERENCES


