Advances in Data Science and Adaptive Analysis

Kernel Treelets

Hedi Xia
Department of Mathematics, University of California Santa Barbara, 93106, USA

Hector D. Ceniceros
Department of Mathematics, University of California Santa Barbara, 93106, USA

Received Day Month Year
Revised Day Month Year

A new method for hierarchical clustering of data points is presented. It combines treelets, a particular multiresolution decomposition of data, with a mapping on a reproducing kernel Hilbert space. The proposed approach, called kernel treelets (KT), uses this mapping to go from a hierarchical clustering over attributes (the natural output of treelets) to a hierarchical clustering over data. KT effectively substitutes the correlation coefficient matrix used in treelets with a symmetric, positive semi-definite matrix efficiently constructed from a symmetric, positive semi-definite kernel function. Unlike most clustering methods, which require datasets to be numeric, KT can be applied to more general data and yields a multi-resolution sequence of orthonormal bases on the data directly in feature space. The effectiveness and potential of KT in clustering analysis is illustrated with some examples.

1. Introduction

Treelets, introduced by Lee, Nadler, and Wasserman [Lee and Nadler (2007); Lee *et al (2008)*], is a method to produce a multiscale, hierarchical representation of unordered data. The central premise of treelets and the treelet transform is to exploit sparsity and capture intrinsic localized structures with only a few features (attributes), represented in terms of an orthonormal basis. The treelet transform consists of a sequence of two-dimensional principal component analyses implemented efficiently via rotations. The resulting multiresolution representation of the data can be used for dimensionality reduction and for feature selection prior to regression/classification [Lee and Nadler (2007); Lee *et al (2008)*].

Cluster analysis, also called clustering [Tryon (1939)], is one of the basic tasks of unsupervised learning and is concerned with finding a partition of a set so that elements in one cluster are more similar to one another than they are to elements in another cluster, i.e. the corresponding equivalence class captures similarity of its elements. The clustering can be flat, where the partition is a collection of disjoint sets, or hierarchical [Johnson (1967)], where a nested tree of partitions is

produced. The treelet transform produces a hierarchical clustering *over attributes*. In this work, we propose to combine the kernel method [Hastie *et al (2016); Theodoridis (2015)*] with the treelet transform to obtain an efficient tool for hierarchical clustering analysis *over data points*. We call this method kernel treelet (KT). The central idea is to project the data onto a reproducing kernel Hilbert space (RKHS). This effectively substitutes the correlation coefficient matrix, used by the original treelet method as a measure of similarity among attributes, with a symmetric, positive semi-definite matrix that measures similarity among data points. The intuition behind this approach is that inner products provide a measure of data similarity and a projection onto a RKHS, done via the so-called kernel trick [Hastie *et al (2016); Theodoridis (2015)*], is a natural and efficient way to construct appropriate (dis)similarity matrices for a wide variety of datasets. We present some examples that demonstrate the potential of KT as an effective tool for data clustering analysis.

The typical complexity of hierarchical clustering methods is $O(n^3)$ (n denotes the number of data points in the dataset) but KT, like single linkage clustering [Sibson (1973)], and complete linkage clustering [Defays (1977)] can be done in $O(n^2)$ operations. Most clustering methods are only directly applicable to numerical datasets. However, many modern datasets do not have clear representations in $\mathbb{R}^d$ due for instance to missing data, length difference, and non-numeric attributes. A typical solution to this problem usually involves finding a projection from each observation to $\mathbb{R}^d$ as is the case for example in text vectorization [Doermann (1997)], array alignment [Wang *et al (2007)*], and missing-data imputation [Shrive *et al (2006)*]. However, these particular projections pose considerable challenges and might raise the bias of the model if false assumptions are made. KT has less limitations and can be applied to a wider range of datasets (where an appropriate kernel function is defined), including those mentioned above.

The rest of the paper is as follows. In Section 2 gives some basic background for the treelet transform and the kernel method. This is followed by the introduction of the KT model in Section 3. Section 4 presents some theory to help explain the success of the KT approach for clustering. Three examples of clustering analysis are given in section 5. In Section 6 an approach that combines KT with supervised learning is proposed to accelerate data hierarchical clustering. Finally, concluding remarks are given in Section 7.

2. Background Information

We give in this section a brief description of the treelet algorithm [Lee and Nadler (2007); Lee *et al (2008)*] and the Kernel method [Aizerman (1964)] as background for the introduction of the KT model. Treelets are based on the repeated application of two dimensional rotations to a matrix measuring the similarity of attributes. So we start by reviewing Jacobi (also called Givens) rotations first.

A Jacobi rotation matrix J is an orthogonal matrix with at most 4 entries

different from the identity. For a given symmetric matrix M and off-diagonal entry p, q, the Jacobi matrix J is constructed so that

$$(J^T MJ)_{pq} = (J^T MJ)_{qp} = 0.$$

The construction of J is equivalent to finding the cosine (c) and sine (s) of the angle of rotation, which satisfy

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} M_{pp} & M_{pq} \\ M_{qp} & M_{qq} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}$$

subject to the constraint $c^2 + s^2 = 1$. The matrix J is then given $J_{pp} = J_{qq} = c$, $J_{pq} = -J_{qp} = s$, and $J_{ij} = \delta_{ij}$ for all the other entries. The new attributes

$$d_1 = M_{pp}c^2 + M_{qq}s^2 - 2M_{pq}cs, \tag{1}$$

$$d_2 = M_{pp}c^2 + M_{qq}s^2 + 2M_{pq}cs, \tag{2}$$

are referred to as the *sum and difference* variables or as the *scaling and detailing* variables.

A numerical stable way of computing J is as follows. Assuming $M_{pq} \neq 0$, compute

$$b = \frac{M_{pp} - M_{qq}}{2M_{pq}}$$

and define

$$t = \frac{\text{sgn}(b)}{|b| + \sqrt{b^2 + 1}}.$$

Then $c = \frac{1}{\sqrt{t^2+1}}$ and $s = ct$. A Jacobi rotation over a $n \times n$ matrix uses $O(1)$ space with $O(n)$ operations.

2.1. *Treelets*

The treelets algorithm [Lee and Nadler (2007); Lee *et al (2008)*] constructs a multiresolution basis and a corresponding hierarchical clustering over the attributes of some datasets in $\mathbb{R}^d$, with which sparsity can be exploited. In its most efficient implementation [Lee *et al (2008)*] it is an $O(min(nd^2, n^2d) + Ld)$ algorithm (L is the number of levels in the multiresolution).

The algorithm starts by constructing a $d \times d$ empirical covariance matrix A_0 and an attribute similarity matrix M_0 given by

$$(M_0)_{ij} = \sqrt{\frac{(A_0)_{ij}^2}{(A_0)_{ii}(A_0)_{jj}}} + \lambda \left|(A_0)_{ij}\right|. \tag{3}$$

where λ is a regularization, hyper-parameter.

The initial set S_0 of scaling indices is that of all the variables, i.e. $S_0 = \{1, 2, ...d\}$. Starting with A_0 and S_0, at each tree level $k = 1, \ldots L$ ($L < d$), A_k and S_k are constructed as follows:

1. Compute the $d \times d$ matrix M_k whose entries are given by

$$(M_k)_{ij} = \sqrt{\frac{(A_{k-1})_{ij}^2}{(A_{k-1})_{ii}(A_{k-1})_{jj}}} + \lambda \left|(A_{k-1})_{ij}\right|. \tag{4}$$

2. Find the two indices α_k, β_k such that

$$\alpha_k, \beta_k = \underset{\substack{\alpha,\beta \in S_{k-1} \\ \alpha \neq \beta}}{\operatorname{argmax}} (M_k)_{\alpha\beta}. \tag{5}$$

3. Calculate the Jacobi matrix J_k for $\alpha_k\, \beta_k$ entry of A_{k-1} and set $A_k = J_k^T A_{k-1} J_k$.
4. Set aside the difference variable. Assuming, without loss of generality, that $(A_k)_{\alpha_k\alpha_k} \leq (A_k)_{\beta_k\beta_k}$, set $S_k = S_{k-1} - \{\alpha_k\}$.

The Jacobi rotations in the treelet algorithm produce an orthogonal basis to represent the data for each $k \in \{1, 2, 3, ..., L\}$ ($L = d - 1$ being the maximum level of the tree). Defining

$$B_k = J_k^T J_{k-1}^T \cdots J_2^T J_1^T, \tag{6}$$

then

$$A_k = B_k A_0 B_k^T. \tag{7}$$

Consequently, every vector $v \in \mathbb{R}^d$ has a k-th basis representation $B_k v$. Furthermore, there is a compressed kth basis representation obtained by dropping insignificant ($< \epsilon$) detailing (non-scaling) indices of $B_k v$. That is, if we define e_i to be the ith column of the identity matrix, the compressed kth basis representation is given by

$$\tau_k(v) = B_k v - \sum_{\substack{i \notin S_i \\ |B_k v \cdot e_i| < \epsilon}} (B_k v \cdot e_i) e_i.$$

Treelets can also be viewed as a hierarchical clustering method *over attributes*. The hierarchical clustering structure is stored in α_k, β_k. We start with trivial clustering where each attribute is in its own cluster and labeled by itself. For each k, we merge clusters labeled α_k and β_k and label it as β_k. This is feasible because each step k the set of all cluster labels is exactly S_{k-1}. This operation gives a hierarchical tree for attribute clustering.

2.2. *Kernel Method*

The kernel method [Aizerman (1964)] allow us to map variables into a new feature (Hilbert) space via a kernel function. We now review briefly the basic concepts and ideas of this approach (see for example [Hastie *et al (2016); Theodoridis (2015)*] for a more comprehensive review).

A kernel K over some set X is a function $K : X \times X \to \mathbb{R}$. A symmetric and positive semi-definite (SPSD) kernel K has the properties:

$$K(x_1, x_2) = K(x_2, x_1), \text{ for all } x_1, x_2 \in X, \tag{8}$$

$$\sum_{i=1}^{s}\sum_{j=1}^{s} c_i c_j K(x_i, x_j) \geq 0, \text{ for all } \{x_1, ..., x_s\} \in X \text{ and all } \{c_1, ..., c_s\} \in \mathbb{R}. \tag{9}$$

If X is finite, then K is SPSD if and only if $K(X, X)$ is a SPSD matrix. If $X \subseteq \mathbb{R}^d$, there is some Hilbert space $\mathbb{H}$ and a feature map $\Phi_K : \mathbb{R}^d \to \mathbb{H}$ associated to a SPSD kernel K such that for all $x, y \in X$,

$$K(x, x) = \langle \Phi_K(x), \Phi_K(y) \rangle_{\mathbb{H}}, \tag{10}$$

where $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ stands for the inner product in $\mathbb{H}$.

The space $\mathbb{H}$ here is called a reproducing kernel Hilbert space (RKHS). The following are two common examples of SPSD kernels:

(1) Radial basis function (RBF) kernel

$$K(x, y) = \exp\{-\frac{||x - y||^2}{2\sigma^2}\}. \tag{11}$$

(2) Polynomial kernel

$$K(x, y) = (\alpha\langle x, y \rangle + c_0)^r. \tag{12}$$

A kernel K for a set X can be restricted to a subset $Y \subseteq X$, and the SPSD property is preserved under such restriction. If the task under consideration is clustering over a finite set, the selected kernel needs only be SPSD on the (finite) set of all samples. Thus, we only need to check that the kernel matrix is SPSD. If we need to extend the clustering outcome to other data, e.g. for clustering boosted classification, then X has to include the whole data space as a subset.

3. The KT Model

The objective of KT is to produce a hierarchical data clustering for some set $\mathcal{D} = \{d_1, \ldots, d_n\}$ given a SPSD kernel $K : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ measuring the data similarity. Instead of using the $d \times d$ empirical covariance matrix A_0 in the initial step (3) of the treelet transform, we replace A_0 with the $n \times n$ kernel matrix and apply the rest of the steps of treelets algorithm. In more detail:

(1) First calculate the $n \times n$ kernel matrix $(A_0)_{ij} = K(d_i, d_j)$. A_0 is a SPSD matrix because K is SPSD kernel, and each column (or row) corresponds to a data point in $\mathcal{D}$.
(2) Apply the treelet algorithm with hyper-parameter λ and $L = n - 1$ using the A_0 of step 1. In our experiments, λ is set to 0 but it can also be tuned as in treelets.

(3) The hierarchical clustering produced by the treelet transform can now be viewed as a clustering of columns (or rows) of A_0 and consequently as a clustering of the data points of the set $\mathcal{D}$.

3.1. *Illustration*

To illustrate how the KT method work we use the following 5 point two-dimensional dataset:

$$\mathcal{D} = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\}. \tag{13}$$

The data points are displayed in Figure 1. If we choose the RBF kernel (11) with

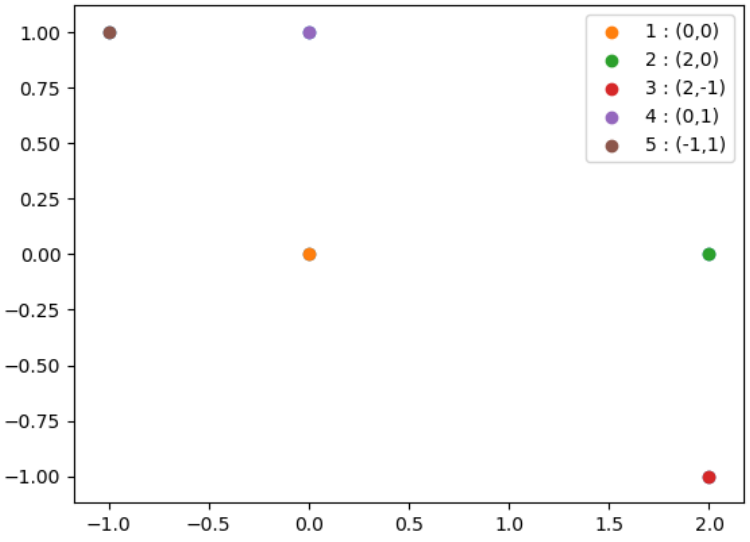

Fig. 1. Distribution of dataset $\mathcal{D} = \{(0,0), (2,0), (2,-1), (0,1), (-1,1)\}$

$\sigma = 0.5$, then the kernel matrix becomes

$$A_0 = \begin{bmatrix} 1 & e^{-8} & e^{-10} & e^{-2} & e^{-4} \\ e^{-8} & 1 & e^{-2} & e^{-10} & e^{-20} \\ e^{-10} & e^{-2} & 1 & e^{-16} & e^{-24} \\ e^{-2} & e^{-10} & e^{-16} & 1 & e^{-2} \\ e^{-4} & e^{-20} & e^{-24} & e^{-2} & 1 \end{bmatrix} \approx \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.135 & 0.018 \\ 0.000 & 1.000 & 0.135 & 0.000 & 0.000 \\ 0.000 & 0.135 & 1.000 & 0.000 & 0.000 \\ 0.135 & 0.000 & 0.000 & 1.000 & 0.135 \\ 0.018 & 0.000 & 0.000 & 0.135 & 1.000 \end{bmatrix}.$$

Applying treelets to it gives a hierarchical tree for columns (or rows) of A_0 and as remarked above, for the data points themselves as Figure 2 illustrates.

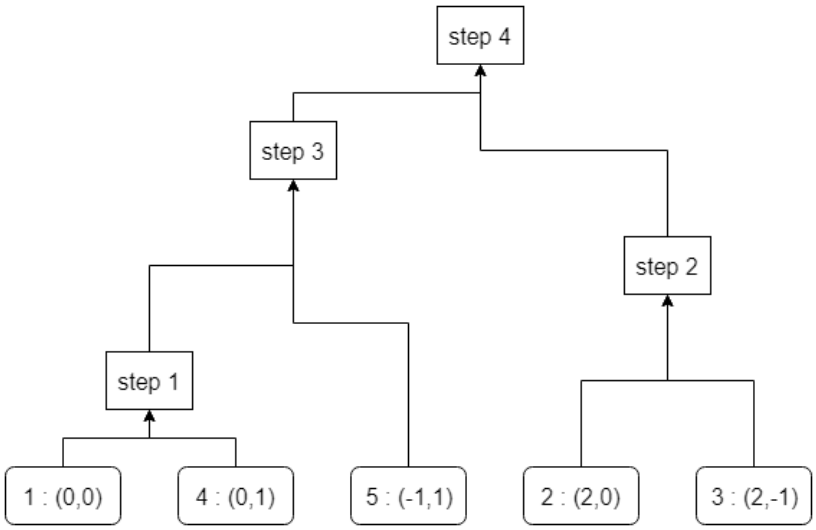

Fig. 2. Clustering tree for $\mathcal{D} = \{(0,0),(2,0),(2,-1),(0,1),(-1,1)\}$.

4. Theory

We now prove that the kernel projection is equivalent to working with an $n \times n$ symmetric positive semi-definite matrix, regardless of the dimension of the KRHS $\mathbb{H}$, and that this matrix can be efficiently evaluated through the kernel K. We also suggest a definition of a *clustering frame* and *clustering equivalence* for similarity-based clustering that allows us to connect the results of the clustering analysis for the original dataset with those of the transformed, projected set and thus explain the usefulness of KT approach.

Hereto $\mathcal{D}$ denotes the dataset and D the corresponding matrix whose columns are the data points in $\mathcal{D}$ and similarly for functions of $\mathcal{D}$.

Lemma 1 *Let $X \subseteq \mathbb{R}^d$. For every finite dataset $\mathcal{D} = \{d_1, \ldots, d_n\} \subseteq X$ and an SPSD kernel K, there exists an orthonormal basis B of the RKHS $\mathbb{H}$ such that*

$$[\Phi_K(d_i)]_B = \begin{bmatrix} \delta_i \\ 0 \end{bmatrix} \quad \text{for } i = 1, \ldots, n, \tag{14}$$

where the left hand side stands for the representation of $\Phi_K(d_i)$ in the basis B and $\delta_i \in \mathbb{R}^n$. Moreover, the matrix $\begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix}$ is symmetric, positive semi-definite.

Proof. We apply the Gram-Schmidt orthogonalization process to the maximal linearly independent subset of $\{\Phi_K(d_1), \ldots, \Phi_K(d_n)\}$ and get a set of orthonormal vectors $\{\hat{\beta}_i, \ldots, \hat{\beta}_m\}$, where

$$m = \dim(\text{span}\{\Phi_K(d_1), \ldots, \Phi_K(d_n)\}) \leq n. \tag{15}$$

We extend this set to a orthonormal basis $\hat{B} = \{\hat{\beta}_1, \dots \hat{\beta}_m, \dots\}$ of $\mathbb{H}$. Then, for all $i \in \{1, 2, ..., n\}$, $[\Phi_K(d_i)]_{\hat{B}}$ is 0 after the m-th entry, and consequently after n-th entry, so there exists $\hat{d}_i \in \mathbb{R}^n$ such that

$$[\Phi_K(d_i)]_{\hat{B}} = \begin{bmatrix} \hat{d}_i \\ 0 \end{bmatrix}. \tag{16}$$

The $n \times n$ matrix $\begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \cdots & \hat{d}_n \end{bmatrix}$ can be written in its singular value decomposition

$$\begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \cdots & \hat{d}_n \end{bmatrix} = U\Sigma V^T, \tag{17}$$

where U and V are orthogonal matrices and Σ is a diagonal matrix with non-negative entries.

We can now define a new orthonormal basis $B = \{\beta_1, \dots \beta_m, \dots\}$ through the change of basis matrix $\begin{bmatrix} VU^T & 0 \\ 0 & I \end{bmatrix}$. Let $\delta_i = VU^T\hat{d}_i$ for all $i \in \{1, 2, ..., n\}$, then

$$[\Phi_K(d_i)]_B = \begin{bmatrix} VU^T & 0 \\ 0 & I \end{bmatrix} [\Phi_K(d_i)]_{\hat{B}} = \begin{bmatrix} VU^T\hat{d}_i \\ 0 \end{bmatrix} = \begin{bmatrix} \delta_i \\ 0 \end{bmatrix}. \tag{18}$$

The projected data $\Phi_K(d_i)$ in the basis B is $\begin{bmatrix} \delta_i & 0 \end{bmatrix}^T$ and the matrix

$$\begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix} = VU^T \begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \cdots & \hat{d}_n \end{bmatrix} = V\Sigma V^T \tag{19}$$

is symmetric and positive semi-definite. □

Corollary 1 *Let $X \subseteq \mathbb{R}^d$ and $\mathcal{D} = \{d_1, \dots, d_n\} \subseteq X$ a dataset. For every $x \in X$, define $\Psi(x)$ as the first n components of $\Phi_K(x)$ in the basis B of Lemma 1, i.e.*

$$\begin{bmatrix} \Psi(x) \\ * \end{bmatrix} = [\Phi_K(x)]_B. \tag{20}$$

Then,

$$\langle \Psi(D), \Psi(D) \rangle = \langle \Phi_K(D), \Phi_K(D) \rangle_{\mathbb{H}} = K(D, D). \tag{21}$$

Here, $\langle \Psi(D), \Psi(D) \rangle$ denotes the matrix with entries $\langle \Psi(d_i), \Psi(d_j) \rangle$ and similarly for
$\langle \Phi_K(D), \Phi_K(D) \rangle_{\mathbb{H}}$ and $K(D, D)$.

Proof. For all $d_i \in \mathbb{H}$,

$$\begin{bmatrix} \Psi(d_i) \\ 0 \end{bmatrix} = [\Phi_K(d_i)]_B,$$

that is $\Psi(d_i) = \delta_i$. From Lemma 1, we have that $\Psi(D) = \begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix}$ is symmetric, positive semi-definite and

$$\langle \Psi(D), \Psi(D) \rangle = \begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix}^2 = \langle \Phi_K(D), \Phi_K(D) \rangle_{\mathbb{H}} = K(D, D).$$

□

4.1. *Clustering Equivalences*

Definition 1 *A clustering frame is a pair $(\mathcal{D}, f)$ where $\mathcal{D}$ is a finite, ordered dataset and $f : \mathcal{D}\times\mathcal{D} \to \mathbb{R}$ is a mapping that measures similarity for the points in the dataset $\mathcal{D}$.*

Definition 2 *Two clustering frames $(\mathcal{D}_1, f_1)$ and $(\mathcal{D}_2, f_2)$ are equivalent, and we write $(\mathcal{D}_1, f_1) = (\mathcal{D}_2, f_2)$, if and only if $f_1(\mathcal{D}_1, \mathcal{D}_1) = f_2(\mathcal{D}_2, \mathcal{D}_2)$.*

For any similarity-based clustering method, using two equivalent clustering frames gives the same clustering outcome.

A pertinent example of clustering frame equivalence is the following. If an SPSD kernel K corresponds to the projection Φ_K, then there is a RKHS $\mathbb{H}$ such that

$$K(D, D) = \langle \Phi_K(D), \Phi_K(D) \rangle_{\mathbb{H}}. \tag{22}$$

Therefore, the clustering on $\mathcal{D}$ using similarity mapping K corresponds to the clustering on $\Phi_K(\mathcal{D})$ using the inner product in $\mathbb{H}$, i.e.

$$(\mathcal{D}, K) = (\Phi_K(\mathcal{D}), \langle \cdot, \cdot \rangle_{\mathbb{H}}). \tag{23}$$

4.2. *Kernel Treelets Clustering Equivalences*

Recall that the treelet transform applied to some dataset $\hat{\mathcal{D}}$ produces a hierarchical clustering *over its attributes*, or in other words, a clustering *over the data* of $\hat{D}^T$. If we apply the treelet transform to $\hat{D} = \Psi^T(D)$, then the outcome would be a hierarchical clustering over the columns of $\hat{D}^T = (\Psi^T(D))^T = \Psi(D)$. Therefore, the clustering frame for the KT is $(\Psi(\mathcal{D}), \langle \cdot, \cdot \rangle)$ and because

$$\langle \Phi_K(D), \Phi_K(D) \rangle_{\mathbb{H}} = \langle \Psi(D), \Psi(D) \rangle,$$

we have $(\Psi(\mathcal{D}), \langle \cdot, \cdot \rangle) = (\Phi_K(\mathcal{D}), \langle \cdot, \cdot \rangle_{\mathbb{H}})$. Finally, from (23) we get the clustering frame equivalence

$$(\mathcal{D}, K) = (\Phi_K(\mathcal{D}), \langle \cdot, \cdot \rangle_{\mathbb{H}}) = (\Psi(\mathcal{D}), \langle \cdot, \cdot \rangle), \tag{24}$$

which implies that applying the treelet transform to the matrix $\hat{D} = \Psi^T(D)$ provides the same hierarchical clustering as that produced by the clustering frame $(\mathcal{D}, K)$. Also, rather than computing A_0 directly through $\hat{D}$, one computes this matrix efficiently with the *kernel trick* (22) : $A_0 = K(D, D)$.

4.3. *Complexity*

The complexity of computing kernel matrix is $O(\xi n^2)$, where ξ is the complexity of applying kernel function to a pair of data and $\xi = d$ if the data is numeric. Computing the rotation steps can be seen as applying treelets to a $d = n$ matrix, and thus it complexity is $O(Ld) = O((n-1)n) = O(n^2)$ if properly optimized as in treelets. So the total complexity of KT is $O(\xi n^2)$.

5. Examples

We implemented KT and the following examples in Python with the packages Numpy [Oliphant (2006)] and Scikit-learn [Pedregosa *et al (2011)*]. Plots were generated with Matplotlib [Hunter (2007)]. The treelets part of our implementation is not optimized, so its cost is $O(n^3)$ operations (an $O(n^2)$ implementations is also possible [Lee and Nadler (2007)]) . The hyperparameter λ is set to 0 in all of the experiments below.

5.1. *Clustering for Six Datasets*

To illustrate how KT works as a hierarchical clustering method over data, we use first an example from Scikit-learn [Pedregosa *et al (2011)*] which consists of 6 datasets, each of which has 1500 two-dimensional data points (i.e. $n = 1500$ and $d = 2$). We can visualize each dataset and each cluster by plotting each observation as a point in the plane. Each of the first five datasets consists of data drawn from multiple shapes with an error in distance. The sixth dataset is a uniform random sample from $[0, 1]^2$ to show how clustering methods work for uniform distributed data and specially how smooth the boundaries of their partitions are.

Figure 3 compares the performance of KT with different kernels with that of some other clustering methods for the six aforementioned datasets. The number of clusters and hyper-parameters are tuned for each method and the sample sizes are set to 1000 for each instance of the KT method. Each row in Figure 3 represents a dataset and each column represents a clustering method. In this experiment, KT with RBF kernel is the method that performs clustering closest to human intuition for all first five datasets. Only spectral clustering (column 5) has a similar performance but, unlike KT, is flat clustering method; KT produces a hierarchy of clusterings not just one. The sixth dataset shows that KT is affected by the relative density deficiency in some area due to sampling and shows porous boundaries. The excellent performance of the RBF KT on the first five datasets can be traced to the fact that these datasets are to some extent Euclidean distance-based, which corresponds to the assumptions for RBF kernel.

5.2. *Clustering for a Social Network Dataset*

We now consider an example of network analysis from the Stanford Network Analysis Project [Leskovec and Mcauley (2012)]. This is a dataset consisting of 'circles of friends' (or 'friends lists') from Facebook. It has $n_V = 4039$ surveyed individuals, which can be viewed as vertices of an undirected graph. Each two vertices are connected with an edge if the corresponding individuals are friends and not otherwise. The edges are not weighted and the total number is 88234. We use KT to obtain a hierarchical clustering on this dataset.

Denote the set of vertices on the graph as V and define a kernel function K :

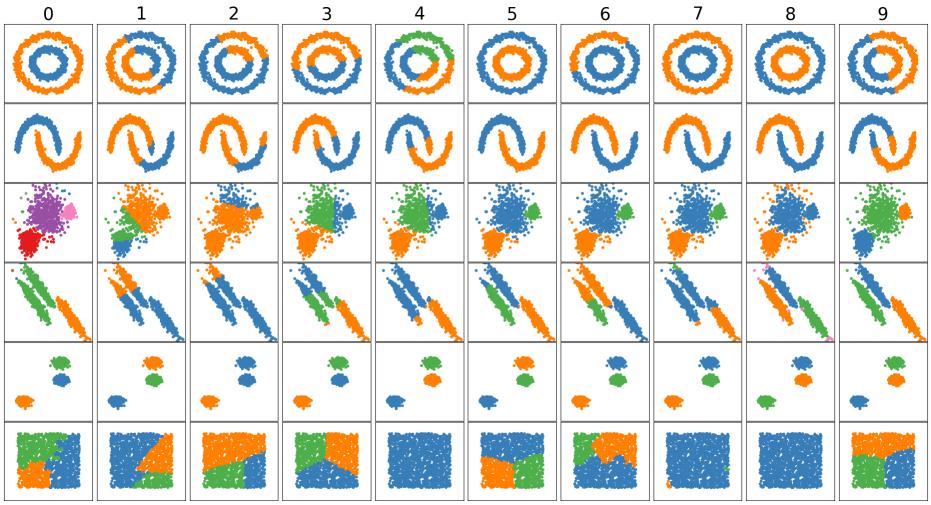

Fig. 3. Comparison of different clustering algorithms on 6 datasets. 0 - KT (RBF), 1 - KT ($r = 1$), 2 - KT ($r = 2$), 3 - Mini batch Kmeans, 4 - Mean shift, 5 - Spectral clustering, 6 - Ward, 7 - Agglomerative clustering, 8 - DBSCAN, 9 - Gaussian mixture.

$V \times V \to \mathbb{R}$ such that for every $v_1, v_2 \in V$

$$K(v_1, v_2) = \begin{cases} 1045 & \text{if } v_1 = v_2. \\ 1 & \text{if } v_1, v_2 \text{ are connected.} \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The number 1045 is the largest degree of all vertices.

Lemma 2 *The matrix $K(V, V)$ defined by (25) is SPSD.*

Proof. Clearly, $K(V, V)$ symmetric. Moreover it is diagonally dominant with a positive diagonal:

$$\sum_{j \neq i} |K(V, V)_{i,j}| = \text{degree}(v_i) \leq \max_j \text{degree}(v_j) = 1045 = K(V, V)_{i,i}, \quad i = 1, 2, ..., n_V.$$

Now, since $K(V, V)$ is symmetric all its eigenvalues are real. Suppose there is a negative eigenvalue $-\lambda^2$. Then $K(V, V) + \lambda^2 I$ is singular but this is impossible because $K(V, V) + \lambda^2 I$ is strictly diagonally dominant. Therefore $K(V, V)$ is SPSD. □

To estimate the performance of KT as a multi-resolution, hierarchical clustering method on this dataset, we do the following evaluation. For each cluster partition in the hierarchy, i.e. for each tree level $k = 1, \ldots L = n - 1$, we compute its confusion matrix and the corresponding true positive rate and false positive rate. The confusion matrix, is a 2×2 array recording the number of true positives (true predicted connections), true negatives (true predicted no-connections), false positives

(false predicted connections), and false negatives (false predicted no-connections) for pairwise associations. The true positive rate (TPR) measures the proportion of two nodes being in the same cluster given that the two nodes are connected. The false positive rate (FPR) measures the proportion of two nodes being in the same cluster given that the two nodes are not connected. For each tree level, $k = 1, \dots n-1$, there corresponds a point in the plane $(\mathrm{TPR}(k), \mathrm{FPR}(k))$ and interpolating these points we obtain the so-called receiver operating characteristic (ROC) curve. The KT ROC is compared in Figure 4 with that of other popular hierarchical clustering methods such as the average, complete, and single linkage methods, and the Ward method. As a reference, the line $y = x$ corresponds to a prediction accuracy of random guessing. KT outperforms the other hierarchical clustering methods on this dataset. The average linkage method has a similar performance as the KT but the later is still slightly better as evidence by the area under the curve (AUC), is the numerical integral of the ROC over $[0, 1]$. For the KT AUC=0.958 whereas for average linkage hierarchical clustering AUC=0.950 and KT's AUC is substantially higher than that of the other hierarchical clustering methods. For the KT clustering, at about 20% FPR we obtain almost 100% TPR, i.e. if we take 20% of nodes that are not connected and place them in the clusters obtained by the KT method then they will be connected to other vertices with almost probability 1.

KT with radial basis and average linkage hierarchical clustering are similar in their construction: at each step they both merge two clusters that are closest and represent the resulting cluster with a weighted average of the representatives of the two merging clusters. The way the representatives and weights are chosen is what differentiates KT and average linkage hierarchical clustering. In average linkage hierarchical clustering, data are initially represented directly $\mathbb{R}^p$ and weights are chosen according to the size of the two clusters. In KT, data are represented by the projected data in the RKHS, and the weight is defined by via Jacobi rotations. This projection enables treelets to produce a clustering that is not only centroid-based but also connectivity-based.

5.3. *Clustering for a Dataset with Missing Information*

Our last example is a dataset with missing information. We use the mice protein expression (MPE) dataset [Higuera *et al (2015)*] from the UCI Machine Learning Repository. This is a dataset consisting of 1080 observations for 8 classes of mice, each of which containing 77 expression levels of different proteins with some of the entries missing.

We employ KT to obtain a hierarchical clustering on this dataset. First, we standardize the data so that each attribute has empirical mean 0 and standard deviation 1. Then, we define the following RBF kernel. For for all observations u, v

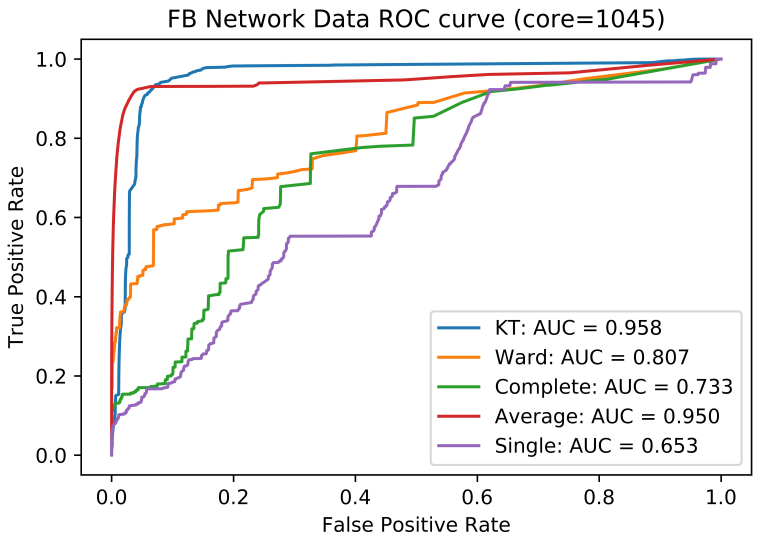

Fig. 4. Receiver operating curve (ROC) for the social network dataset, as the tree level increases, for the KT clustering and Ward method and the average, complete, and single linkage methods.

in the dataset V,

$$K(u,v) = \exp\left\{ -\frac{32}{|E_{uv}|} \sum_{i \in E_{uv}} \|u_i - v_i\|^2 \right\}, \tag{26}$$

where E_{uv} is the set of indices for which the data is available in both u and v. We check that $E_{uv} \neq \emptyset$ so that K is well-defined. The number 32 is a parameter empirically selected. We confirmed numerically that the kernel matrix is SPSD.

We compare the predicted clusters and the true labels according to pairwise scores. Figure 5 shows KT's ROC and that of the other common hierarchical methods mentioned in the previous example. Here, we measure the true positive rate as the proportion of two records being in the same cluster given that they are from mice of the same type, and the false positive rate as the proportion of two records being in the same cluster given that they are from mice of different type. As in the previous example of the social network dataset, we plot the ROC and calculate its AUC. KT's AUC=0.726 is much greater than that of other hierarchical clustering methods, demonstrating KT's superior performance for this dataset. Ward clustering is second best with AUC=0.621.

6. Accelarating Kernel Treelets

The treelet approach is meant for small to moderate size high-dimensional datasets (small n, large d) because of its $O(n^2)$ complexity. In the context of spectral clustering, Yan, Huang and Jordan [Yan *et al (2009)*] proposed a preprocessing method,

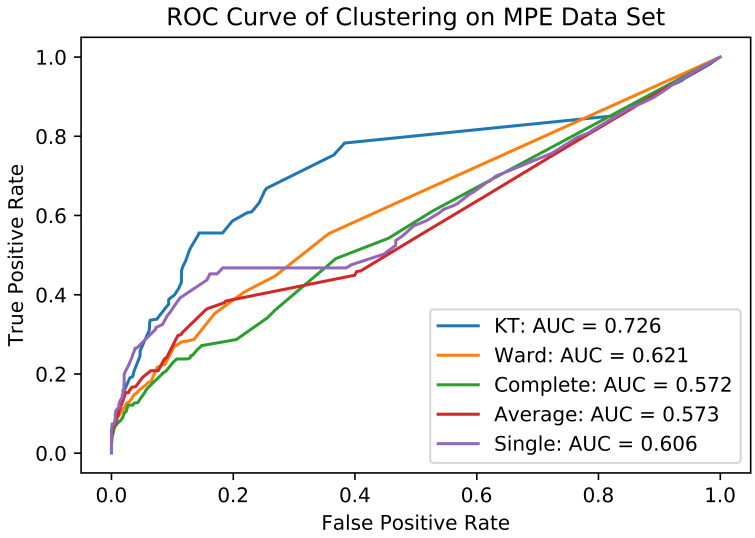

Fig. 5. Receiver operating curve (ROC) for the MPE dataset, as the tree level increases, for the KT clustering and Ward method and the average, complete, and single linkage methods.

using classical k-means or random projections trees, to reduce the size of the input dataset and thus accelerate the spectral analysis of the method.

In a similar spirit, we propose here to apply KT to a moderate size sample $\mathcal{S} \subset \mathcal{D}$ for an initial clustering and then use a kernel support vector machine (SVM) to assign cluster labels to the rest of the data, i.e. the all $d_j \in \mathcal{D} \setminus \mathcal{S}$. To illustrate this approach, we consider again the six datasets of Example 1. Figure 6 demonstrates how the number of sample points n_S affects the clustering result. Each column represents KT using the RBF kernel for different n_S, denoted KTn_s in Table 1. The run time is displayed in Table 1, columns 2-7. The hyper-parameter $\sigma = 0.1$ is tuned towards $n_S = 1000$ case and is used for all other sample sizes. Note that as KT1500 is of full sample size, it does not trigger the kernel SVM whereas KT1499 does. From Figure 6 and Table 1, we observe that the minimum optimal n_s for the first 5 datasets is 1000, 100, 1000, 200, 50, respectively (n_s is dataset dependent, as expected) and thus the overall cost of the clustering analysis could be substantially reduced with this approach. Furthermore, the fourth dataset shows that optimal hyper-parameter σ is n_s-dependent. The RBF kernel can be considered as a weighted average of distance and connectivity, where a larger σ means a higher weight on distance. For the same $\sigma = 0.1$, as sample size n_s increases, the clustering result becomes more distance-based rather than connectivity-based, demonstrating that the optimal σ for those sample sizes is actually smaller.

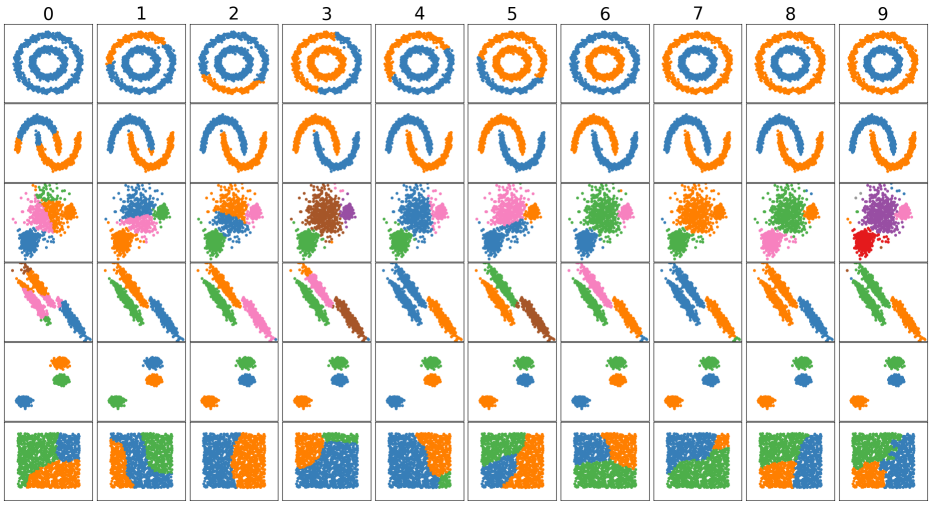

Fig. 6. Comparison of different number-of-cluster estimate on 6 datasets.

KTn_S	1	2	3	4	5	6
0 - KT50	0.011	0.012	0.013	0.011	0.011	0.01
1 - KT100	0.035	0.044	0.039	0.045	0.033	0.028
2 - KT200	0.109	0.099	0.128	0.12	0.121	0.132
3 - KT300	0.225	0.217	0.242	0.269	0.259	0.235
4 - KT500	0.551	0.568	0.62	0.569	0.652	0.536
5 - KT800	1.315	1.513	1.534	1.378	1.699	1.295
6 - KT1000	2.016	2.055	2.336	2.098	2.782	1.941
7 - KT1200	2.88	2.94	3.242	3.004	4.146	2.77
8 - KT1499	4.438	4.532	5.4	4.713	6.788	4.341
9 - KT1500	4.472	4.69	5.398	4.807	6.782	4.274

Table 1. KTn_s denotes the use of KT to a sample of size n_s. The numbers on columns 2-7 are the run times for the clusters in Figure 7.

7. Concluding remarks

In the paper we describe a novel approach, kernel treelets (KT), for hierarchical clustering. The method relies on applying the treelet transform to an $n \times n$ matrix measuring data similarities in a feature, reproducing kernel Hilbert space. We show with some examples that KT can outperform other hierarchical clustering methods and is especially competitive for datasets without numerical matrix representation and or then there is missing data. The KT approach also shows significant potential for semi-supervised learning tasks and as a pre-processing, post-processing step in deep-learning. Work in these directions is underway.

8. Acknowledgements

HDC gratefully acknowledges support from the National Science Foundation through grant DMS 1818821.

References

Ann B Lee and Boaz Nadler. Treelets— a tool for dimensionality reduction and multi-scale analysis of unstructured data. In *Artificial Intelligence and Statistics*, pages 259–266, 2007.

Ann B. Lee, Boaz Nadler, and Larry Wasserman. Treelets: an adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, 2(2):435–471, 2008.

Robert C. Tryon. *Cluster analysis: Correlation profile and orthometric (factor) analysis for the isolation of unities in mind and personality*. Edwards brother, Incorporated, lithoprinters and publishers, 1939.

Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

David S Doermann. An introduction to vectorization and segmentation. In *International Workshop on Graphics Recognition*, pages 1–8. Springer, 1997.

Yongjin Wang, Foteini Agrafioti, Dimitrios Hatzinakos, and Konstantinos N Plataniotis. Analysis of human electrocardiogram for biometric recognition. *EURASIP journal on Advances in Signal Processing*, 2008(1):148658, 2007.

Fiona M Shrive, Heather Stuart, Hude Quan, and William A Ghali. Dealing with missing data in a multi-question depression scale: a comparison of imputation methods. *BMC medical research methodology*, 6(1):57, 2006.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction*. Springer, New York, second edition, 2016.

S. Theodoridis. *Machine Learning. A Bayesian and Optimization Perspective*. Academic Press, London, 2015.

Mark A Aizerman. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

John D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*, pages 539–547, 2012.

Clara Higuera, Katheleen J Gardiner, and Krzysztof J Cios. Self-organizing feature maps

identify proteins critical to learning in a mouse model of Down syndrome. *PloS one*, 10(6):e0129126, 2015.

Donghui Yan, Ling Huang, and Michael I. Jordan. Fast Approximate Spectral Clustering *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 907–916, 2009.