Kernel Treelets

Hedi Xia and Hector D. Ceniceros

Department of Mathematics, University of California Santa Barbara, 93106

Abstract

Most clustering methods require datasets to be numeric: each observation in the datasets is an element of $\mathbb{R}^p$ for some fixed p, while many modern datasets violates this assumption. In this paper we present Kernel Treelets (KT), a hierarchical clustering methods combining Treelets method with the Kernel method. Our method can be applied to datasets with kernel and construct a multi-resolution sequence of basis on the data in the feature space. We illustrates some examples about the use of KT in different clustering tasks.

1 Introduction

Cluster analysis[1][2], also called clustering, is concerned with finding a partition of a set such that its corresponding equivalence class captures similarity of its elements. Hierarchical clustering (HC) [3] is a class of clustering methods that provides a nested and multiscale clustering, with a time complexity of $O(n^3)$ (where n denotes the number of data in the dataset) in general with exceptions of $O(n^2)$ for single linkage HC [4] and complete linkage HC [5]. Most of these clustering methods apply to numerical dataset only; however many modern datasets do not have intuitive representations in $\mathbb{R}^m$ with reasons such as missing data, length difference, non-numeric attributes. Current solutions to these problems usually involves in finding a projection from each observation to $\mathbb{R}^m$, such as text vectorization [6], array alignment [7], missing-data imputation [8]. These tasks are themselves challenging and raise the bias of model if false assumptions are made. In this paper we propose a hierarchical clustering method on these datasets. We call it Kernel Treelets (KT), as it is based on Treelets method by Lee et at [9][10]. KT is a model of complexity $O(n^2)$, and it can apply to datasets by specifying a kernel matrix. Kernels are easier to find than projection to $\mathbb{R}^m$, which grants KT the ability to apply to a greater variety of datasets than other HC methods.

2 Background Information

2.1 Kernel Method

The Kernel methods was proposed by Aizerman on 1964 [11]. A kernel over some set X is defined as a function $K : X \times X \to \mathbb{R}$. A Symmetric and Positive Semi-definite (SPSD) kernel is a kernel with the property that it is symmetric:

$$\forall x_1, x_2 \in X,\ K(x_1, x_2) = K(x_2, x_1)$$

and Positive Semi-definite

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

If X is finite, then K is SPSD if and only if $K(X, X)$ is a SPSD matrix. If $X \subseteq \mathbb{R}^n$, then K is SPSD if and only if there exists a function $\Phi_K : \mathbb{R}^n \to \mathbb{H}$, where $\mathbb{H}$ denotes the Hilbert space, such that for all $x_1, x_2 \in X$,

$$K(x_1, x_2) = \langle \Phi_K(x), \Phi_K(y) \rangle_{\mathbb{H}}$$

Here we call the space $\mathbb{H}$ Reproducing Kernel Hilbert Space (RKHS). Introducing Φ_K in calculation usually increases the complexity. Nevertheless, it gives theoretical backgrounds about the similarity between kernels and inner products. Some SPSD kernels are as follows:

- Radial Basis Function (RBF) kernel

$$K(x_1, x_2) = \exp\{-\frac{||x_1 - x_2||^2}{2\sigma^2}\}.$$

- Polynomial Kernel

$$K(x_1, x_2) = (\alpha \langle x_1, x_2 \rangle + c_0)^r.$$

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

2.2 K Nearest Neighbors (KNN)

K-nearest neighbors algorithm is a multi-class classification algorithm [12]. By specifying $k \in \mathbb{N}$ and a metric, the algorithm can, given a test data, predict its labels by the majority vote of a subset of k closest elements in distance metric from training data. If an inner product is specified instead of distance, we can compute the distance between two point in the following way:

$$\|x_1 - x_2\|^2 = \langle x_1 - x_2, x_1 - x_2 \rangle = \langle x_1, x_1 \rangle + \langle x_2, x_2 \rangle - 2\langle x_1, x_2 \rangle.$$

If the metric is kernelized,

$$\begin{aligned}\|x_1 - x_2\|^2 &= \langle x_1, x_1 \rangle_{\mathbb{H}} + \langle x_2, x_2 \rangle_{\mathbb{H}} - 2\langle x_1, x_2 \rangle_{\mathbb{H}} \\ &= K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2).\end{aligned}$$

2.3 Kernel Support Vector Machine (SVM)

Support Vector Machine (SVM) is a classification method by finding optimal hyper-planes. Kernel SVM [13] is a classification method towards nonlinear problems that performs SVM in RKHS generated by the kernel. When we only apply KT to a small sample, we may use kernel SVM with the same kernel to assign labels for data outside of this sample. This can be viewed as clustering attributes with treelets and using SVM to assign labels to other attributes in RKHS.

2.4 Jacobi Rotation / Given's Rotation

Given's rotation matrix is an orthogonal matrix with at most 4 entries different from the identity, or more generally viewed, a rotation operator on a 2 dimensional subspace generated by two coordinate axes. Jacobi rotation is a process to find a Given's rotation matrix J for a symmetric and SPSD matrix M and entries $p \neq q$, such that

$$[J^T M J]_{pq} = [J^T M J]_{qp} = 0.$$

A way to find it is to solve the following equation subject to $c^2 + s^2 = 1$:

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

and define matrix J such that

- $J_{pp} = J_{qq} = c$
- $J_{pq} = -J_{qp} = s$
- For other entries ij, $J_{ij} = I_{ij}$.

A numerical stable way of computing this problem is as follows:

- Assume $M_{pq} \neq 0$, and compute

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

- Let $sgn(b)$ be 1 if $b \geq 0$ and -1 otherwise, then we define

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

- From which we can calculate $c = \frac{1}{\sqrt{t^2+1}}$ and $s = ct$.

The complexity of storing a Given's rotation matrix is $O(1)$, and Jacobi rotation over a $n \times n$ matrix uses $O(1)$ space with time complexity $O(n)$.

2.5 Treelets

Treelets is an $O(np^2)$ algorithm proposed by Lee et al [9]. It was designed to construct multiscale basis and hierarchical clustering over the attributes of some datasets with the assumptions that attributes are sparce in the sense that they are well-clustered in some subspaces. The algorithm start with a hyper-parameter λ and computing a $p \times p$ covariance matrix A_0. The initial scaling indices are defined as the set $S_0 = \{1, 2, ...p\}$. With base case A_0 and S_0, each step A_k and S_k for $k \in \{1, 2, 3, ..., p-1\}$ can be constructed inductively as follows:

1. Construct matrix M_k of the same shape as A_0 entry-wise:

$$[M_k]_{ij} = \sqrt{\frac{[A_{k-1}]_{ij}^2}{[A_{k-1}]_{ii}[A_{k-1}]_{jj}}} + \lambda|[A_{k-1}]_{ij}|.$$

2. Find the two indices α_k, β_k such that

$$\alpha_k, \beta_k = \underset{\alpha,\beta \in S_{k-1}}{\operatorname{argmax}} [M_k]_{\alpha\beta}.$$

.

3. Calculate Jacobi rotation matrix J_k for α_k, β_k and matrix $A_k = J_k^T A_{k-1} J_k$.

4. Without loss of generality, α_k and β_k is interchangeable, so we require that $[A_k]_{\alpha_k\alpha_k} \leq [A_k]_{\beta_k\beta_k}$, and record α_k and β_k.

5. Define $S_k = S_{k-1} - \{\alpha_k\}$.

2.5.1 Treelets Transform and Treelets Basis

The Treelets rotation produces a Treelets basis for each $k \in \{1, 2, 3, ..., p-1\}$. Recall that in Treelets rotation, the sequence of matrices $\{J_k\}$ can provide a sequence of basis for $\mathbb{R}^p$, defined as

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

it has a property that

$$A_k = B_k A_0 B_k^T.$$

So for every vector $v \in \mathbb{R}^p$, there is a kth basis representation $B_k v$. Furthermore, there is a compressed kth basis representation defined as dropping insignificant ($< \epsilon$) 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

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

2.5.2 Treelets Hierarchical Clustering

Treelets is also a hierarchical clustering method over the attributes. The hierarchical clustering structure is stored in α_k, β_k. We start with trivial clustering where each element is in its own cluster and labeled by itself. For each k, we merge clusters labeled α_k and β_k and label it 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 clustering use on the attributes.

3 Model

The task of KT is to find a clustering for some set V given a SPSD kernel $K : V \times V \to \mathbb{R}$ measuring their relationships. The way we use Treelets with kernels is to replace the covariance A_0 with kernel matrix, and apply the rest of the steps of Treelets. The exact steps are as follows:

- First we draw a sample S with size n_S from uniform distribution on V and some sample size n_V. If more information about V is given, it may be possible to draw a sample S that better represent V with smaller sample size.

- Then we calculate the kernel matrix $A_0 = K(S, S)$. A_0 is a SPSD matrix because K is SPSD, and thus we can apply Treelets with hyper-parameter λ using A_0 instead of covariance matrix. λ can be set to 0 or tuned experimentally as in Treelets. In this step, Treelet method provides a hierarchical clustering tree of each columns of A_0, which corresponds to each observations in S.

- If $S = V$, we are finished on the step above. Otherwise, we need to cluster the elements in V based on clusters we have from elements in S. We use kernel SVM to complete this task. Given S and its corresponding cluster labels, we train the kernel SVM with the same kernel K, and then apply to predict the cluster labels of V. K-Nearest Neighbors with distance induced by kernel

  $$d(v_1, v_2)^2 = K(v_1, v_1) + K(v_2, v_2) - 2K(v_1, v_2)$$

  is an alternative to kernel SVM.

3.1 Theory

3.1.1 Lemma 1

Proposition: for every finite dataset $D = \{d_i : i = 1, 2, ..., n\} \subseteq V$ and an SPSD kernel K, there exists an orthonormal Hilbert basis B in the RKHS such that

$$[\phi_K(d_i)]_B = \begin{bmatrix} \delta_i \\ 0 \end{bmatrix}$$

where $\delta_i \in \mathbb{R}^n$ and $\begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix}$ is symmetric and positive semi-definite.

3.1.2 Proof of Lemma 1

We apply Gram-schmidt algorithm to the maximal linearly independent subset of the set $\{\phi(d_i) : i = 1, 2, ..., n\}$ and get a set of orthonormal vectors $\{\hat{\beta}_i : i = 1, 2, ..., \eta\}$, where

$$\eta = dim(span\{\phi(d_i) : i = 1, 2, ..., n\}) \leq n.$$

We may extend this set to a orthonormal Hilbert basis $\hat{B} = \{\hat{\beta}_i : i = 1, 2, ...\}$. Then $\forall i \in \{1, 2, ..., n\}$, $[\phi(d_i)]_{\hat{B}}$ is 0 for all entries after η and consequently after n, so there exists $\hat{d}_i \in \mathbb{R}^n$ such that

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

As $\begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \cdots & \hat{d}_n \end{bmatrix}$ is a square matrix, we may compute its Singular Value Decomposition

$$\begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \cdots & \hat{d}_n \end{bmatrix} = U \Sigma Q^T.$$

Then we can define a new orthonormal Hilbert basis $B = \{\beta_i : i = 1, 2, ...\}$ by the change of basis matrix $\begin{bmatrix} QU^T & 0 \\ 0 & I \end{bmatrix}$. Let $\delta_i = QU^T \hat{d}_i$ for all $i \in \{1, 2, ..., n\}$, then

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

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

$$\begin{bmatrix} \delta_1 & \delta_2 & \cdots & \delta_n \end{bmatrix} = QU^T \begin{bmatrix} \hat{d}_1 & \hat{d}_2 & \cdots & \hat{d}_n \end{bmatrix} = Q \Sigma Q^T$$

is symmetric and positive definite.

3.1.3 Corollary of Lemma 1

If we denote $\psi : V \to \mathbb{R}^n$ such that for all $v \in V$,

$$\begin{bmatrix} \psi(v) \\ * \end{bmatrix} = [\phi_K(v)]_B.$$

or in other words, $\psi(v)$ is the first n components of $\phi_K(v)$ in the basis B. Then for all $d_i \in D$,

$$\begin{bmatrix}\psi(d_i)\\0\end{bmatrix} = [\phi_K(d_i)]_B.$$

Which means that $\psi(d_i) = \delta_i$. With lemma 1, we have that $\psi(D) = \begin{bmatrix}\delta_1 & \delta_2 & \cdots & \delta_n\end{bmatrix}$ is symmetric and positive definite. Also,

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

3.1.4 Clustering Equivalences

A clustering setting is a pair (D, f) where D is an finite dataset with ordering $1, 2, ..., n$ and $f : D \times D \to \mathbb{R}$ is a metric on the dataset D. We define an equivalence on the clustering setting that $(D_1, f_1) = (D_2, f_2)$ if and only if $f_1(D_1, D_1) = f_2(D_2, D_2)$. For any measurement based clustering method, using measurement f_1 on D_1 provides the same exact clustering outcome on the labels as using measurement f_2 on D_2. An example of clustering equivalences is that if kernel K corresponds to projection ϕ_K, then there is $K(D, D) = \langle\phi_K(D), \phi_K(D)\rangle_{\mathbb{H}}$, and therefore $(D, K) = (\phi_K(D), \langle\cdot,\cdot\rangle_{\mathbb{H}})$.

3.1.5 Kernel Treelets Equivalences

For a dataset $\{d_i : i = 1, 2, ..., n\}$ and a kernel K, we already know that there is a clustering equivalence $(D, K) = (\phi_K(D), \langle\cdot,\cdot\rangle_{\mathbb{H}})$. From the corollary of lemma 1, there is $\langle\psi(D), \psi(D)\rangle = \langle\phi_K(D), \phi_K(D)\rangle_{\mathbb{H}}$, which provides the equivalence $(\phi_K(D), \langle\cdot,\cdot\rangle_{\mathbb{H}}) = (\psi(D), \langle\cdot,\cdot\rangle)$. As $\psi(D)$ is symmetric, $(\psi(D), \langle\cdot,\cdot\rangle) = (\psi^T(D), \langle\cdot,\cdot\rangle)$. As a conclusion, $(D, K) = (\psi^T(D), \langle\cdot,\cdot\rangle)$, which implies that a clustering method measured with inner product on dataset $\psi^T(D)$ provides a clustering of D measured with kernel K. Therefore, Treelets on $\psi(D)$ without centering provides a hierarchical clustering of attributes of $\psi(D)$ based on attribute inner product (covariance matrix), which is a hierarchical clustering of $\psi^T(D)$ based on inner product. According to clustering setting equivalences, this hierarchical clustering is equivalent to a hierarchical clustering of D based on kernel K. Furthermore, a property of Treelets is that $\psi(D)$ does not necessarily need to be computed. The "covariance matrix" of $\psi(D)$ without centering has a easier computation method:

$$Cov(\psi(D)) = \psi(D)\psi(D)^T = \psi(D)^2 = \langle\psi(D), \psi(D)\rangle = \langle\phi_K(D), \phi_K(D)\rangle_{\mathbb{H}} = K(D, D).$$

So we may avoid the costly spectral decomposition to compute $\psi(D)$ and define A_0 of Treelets as

$$A_0 = Cov(\psi(D)) = K(D, D).$$

3.2 Complexity

The complexity of this algorithm is $O(\xi n_S^2 + n_S n_V)$, where ξ is the complexity of applying kernel function to a pair of data and $\xi = p$ if the data is numeric. In this model, the choice of kernel K determines the expected outcome of the prediction and the choice of sample S determines the variability of the outcome. A small sample size S speeds up the algorithm with the cost of generating false clustering by unrepresentative samples, while large sample size slow down the algorithm and also produces numerical issues because data is more likely to be close to orthogonal as the dimension of projected space grows, and Treelets method would be forced to stop if all remaining components are almost orthogonal. The optimal sample size depends on the floating number accuracy and computation time allowed and should be as large as possible without exceeding the time limit and accuracy limit.

4 Examples

We implemented KT and the following examples in Python (`https://www.python.org/`) with package Numpy [14], Scikit-learn [15], and plots are generated with package Matplotlib [16]. The Treelets part of our implementation is not optimized, so it is $O(n^3)$ runtime in the followings examples rather than $O(n^2)$ as designed by Lee et al [9]. The hyperparameter λ is set to 0 for all experiments below.

4.1 Clustering for 6 Datasets

To illustrate how KT works as a hierarchical clustering method, we use an example from scikit-learn (`http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`) [15]. This is a set of 6 datasets, each of which has 1500 data of 2 dimensions ($n = 1500$ and $p = 2$)and we may visualize the dataset and each clusters 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 consists of a uniform random sample from $[0, 1]^2$ to show how clustering method work for uniform distributed data. Fig. 1 shows how KT with different kernels works on these datasets compared to the performance of other clustering method. The number of clusters and hyper-parameters are tuned for each method and the sample sizes are set to 1000 for each KT method. Each row of this image represents a dataset and each column represents a clustering method. The method each column represents and and its runtime on each dataset is in recorded in Table 1.

In this experiment, KT with RBF kernel is the only method that performs clustering closest to human intuition for all first five datasets. The sixth dataset is a uniform distribution in $[0, 1]^2$ which we may see how KT is affected by the relative density deficiency in some area due to sampling. Its high performance on the first five datasets is expected as these datasets are to some extent Euclidean distance-based, which corresponds to the assumptions for RBF kernels. Fig.2 shows how difference of number of sample points affects the clustering result. Each column represents KT using RBF kernel with different sample sizes. The hyper-parameter $\sigma = 0.1$ is tuned towards $n_S = 1000$ case and is used for all other sample sizes. Notice that as KT1500 is of full sample size, it does not trigger kernel SVM whereas KT1499 do. Their number of clusters and runtime is recorded in Table 2. From here we

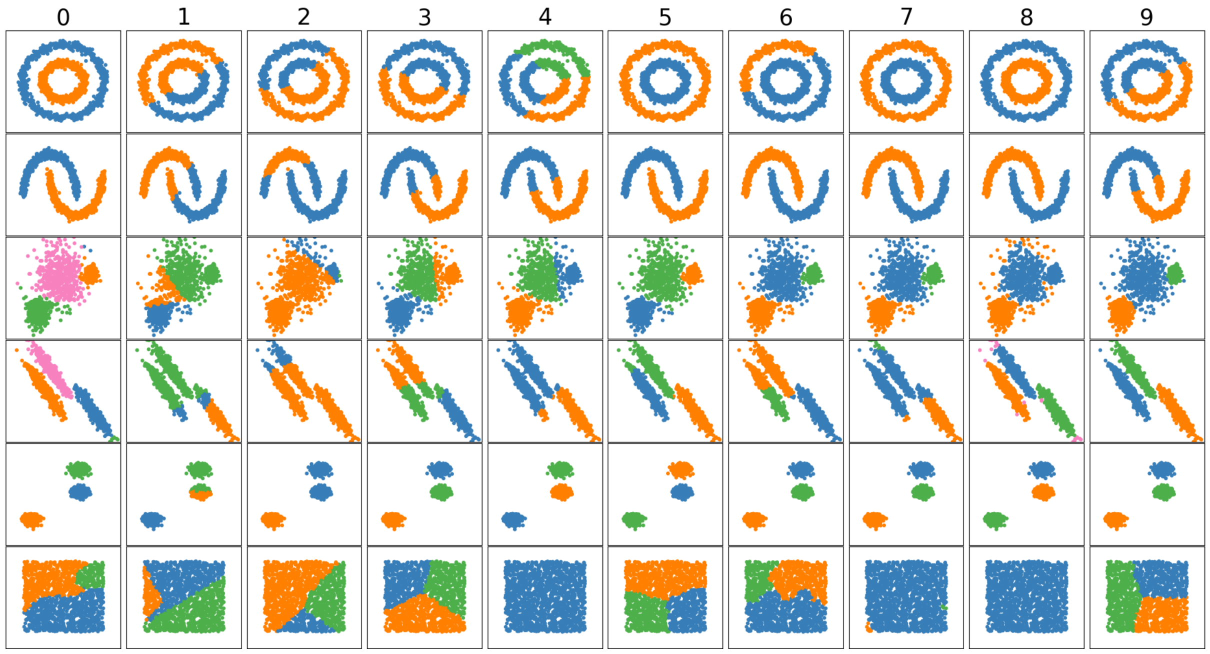

Figure 1: Comparison of Different Clustering Algorithm on Toy Dataset

Method\Dataset	1	2	3	4	5	6
0 - KTrbf	2.003	2.063	2.325	2.094	2.819	1.967
1 - KTlinear	1.585	1.613	1.402	1.73	2.341	1.469
2 - KTpoly	3.956	6.08	6.878	9.582	9.836	4.526
3 - MiniBatchKMeans	0.006	0.018	0.009	0.01	0.007	0.009
4 - MeanShift	0.047	0.032	0.063	0.057	0.032	0.05
5 - SpectralClustering	0.642	1.011	0.13	0.352	0.257	0.208
6 - Ward	0.114	0.098	0.513	0.245	0.111	0.087
7 - AgglomerateClustering	0.085	0.102	0.374	0.196	0.103	0.078
8 - DBSCAN	0.015	0.014	0.015	0.012	0.067	0.012
9 - GaussianMixture	0.005	0.005	0.008	0.012	0.004	0.009

Table 1: Method and Runtime Table for Figure 1

can see that more sample data implies more runtime and more stable outcome. The minimum optimal number of samples required for the first 5 datasets are 1000, 100, 1000, 200, 50, respectively, which shows that different datasets requires different amount of samples to explain its shape. Furthermore, the fourth dataset shows that optimal hyper-parameter σ is number-of-sample dependent. 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 gets larger, the clustering result becomes more distance-based rather than connectivity based, demonstrating that optimal σ for those sample sizes are actually smaller.

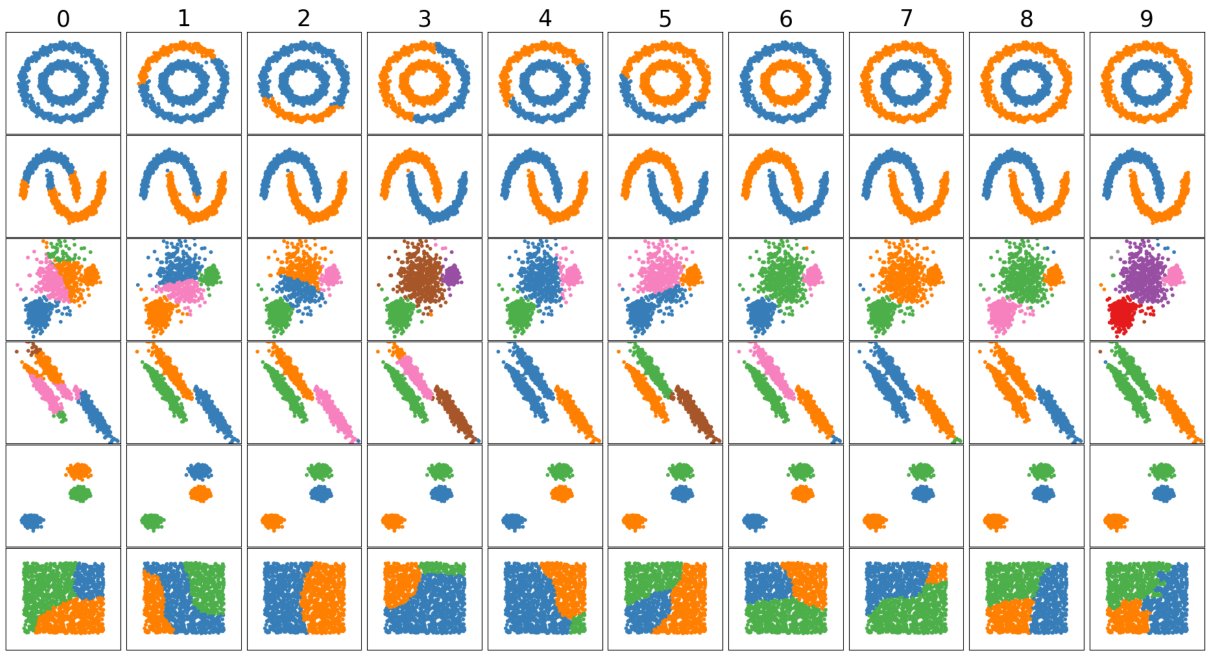

Figure 2: Comparison of Different Number-of-cluster Estimate on Toy Dataset

Method\Dataset	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 2: Method and Runtime Table for Figure 2

4.2 Clustering for Social Network Dataset

To illustrate how KT works in network analysis we use an example from Stanford Network Analysis Project (`http://snap.stanford.edu/data/ego-Facebook.html`) [17]. This is a dataset consisting of 'circles' (or 'friends lists') from Facebook. It has $n_V = 4039$ surveyed individual (vertices) and each two of them is connected with vertices if they are friends and not if they are not friends, which are the edges. The edges are undirected and not weighted, and the total number of edges is 88234. We use KT to do clustering on this dataset with full sample size ($S = V$). Denote the set of vertices on the graph as V, and define a kernel function $K : V \times V \to \mathbb{R}$ such that

$$K(v_1, v_2) = \begin{cases} 1045 & v_1 = v_2 \\ 1 & v_1, v_2 \textit{ are connected} \\ 0 & \textit{otherwise} \end{cases}$$

The number 1045 is computed and chosen as the largest degree of all vertices. Notice that K is a SPSD kernel on V because $K(V, V)$ is a symmetric matrix and is also dominant by the positive diagonal, as $\forall i \in \{1, 2, ..., n\}$

$$\sum_{j \neq i} |K(V, V)_{i,j}| = deg(v_i) \leq \max_{\zeta} deg(v_\zeta) = 1045 = K(V, V)_{i,i}.$$

To estimate the performance of KT as a multi-scale clustering method on this dataset, we use an evaluation as follows. For each cluster partition in the hierarchy, we compute its matching matrix and its corresponding true positive rate as well as false positive rate. Matching matrix, a type of confusion matrix, is a 2 by 2 matrix recording the number of true positives, true negatives, false positives, and false negatives for pairwise associations. True positive rate measure the proportion of two nodes being in the same cluster given the two nodes are connected and false positive rate measures the proportion of two nodes being in the same cluster given the two nodes are not connected. Each pair of true positive rate and false positive rate produces a point on the plane, and interpolating the set of points of all clustering results in the hierarchy (with order) produces the Receiver operating characteristic (ROC) curve, and the numerical integral over $[0, 1]$ interval of this curve is known as Area Under Curve (AUC). Fig.3 demonstrates the performance of KT on the dataset, which provides good clusterings for the dataset because it has an AUC as high as 0.958.

4.3 Clustering for Dataset with Missing Infomation

To illustrate how KT works on dataset with missing information, we use Mice Protein Expression (MPE) dataset [18] from UCI Machine Learning Repository as an example (`https://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression`). 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 are not avalible. We use KT to do clustering on this dataset. First we normalize these attributes so that each of them has empirical mean 0 and standard deviation 1. Then we define a RBF kernel for dataset with missing data such that for all observation u, v,

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

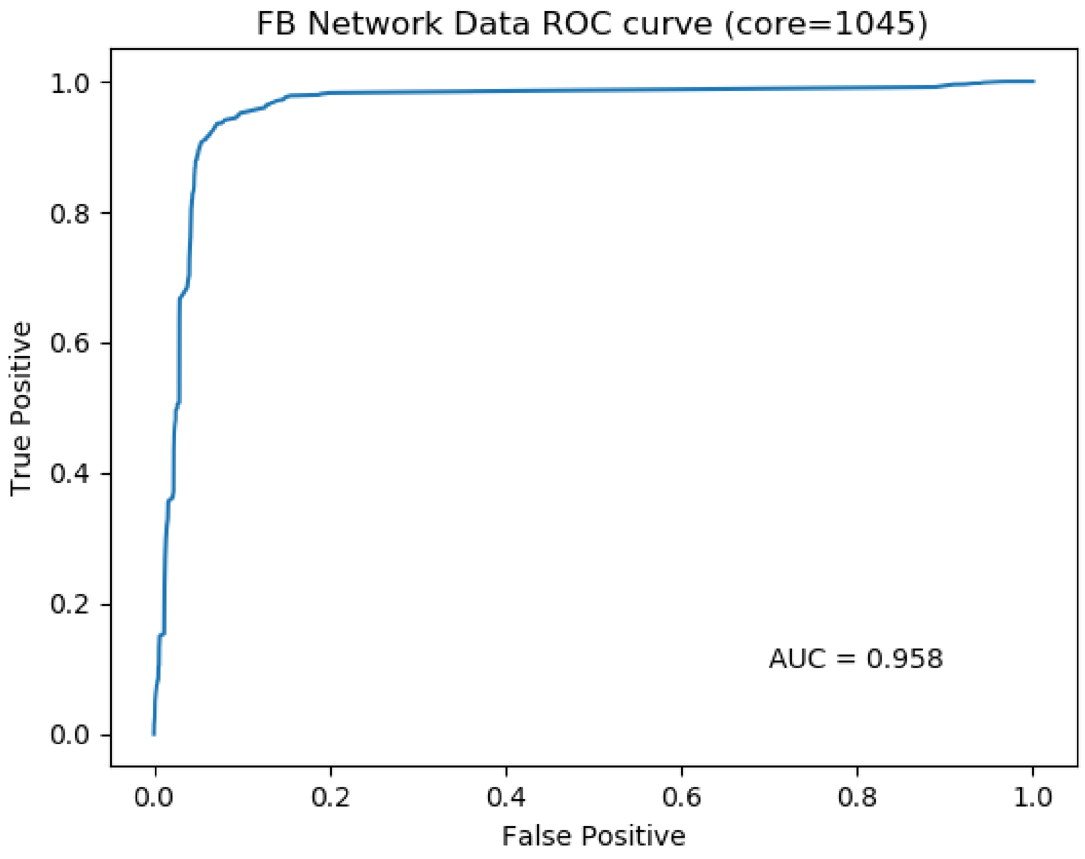

Figure 3: Clustering of Facebook Network Dataset Result

Where E_{uv} is the set of indices that is avalible (not missing) in both u and v. We check that $E_{uv} \neq \emptyset$ so that it is well-defined. The number 32 is a parameter tuned with experiments. We compare the predicted clusters and the true labels according to pairwise scores. Fig.4 shows how KT performs compared to KMeans clustering. We measure the true positive rate as the proportion of two record 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 record being in the same cluster given that they are from mice of different type. Similar as the example of network dataset, we draw its ROC curve and calculate its AUC. Also, we use KMeans with multiple number of clusters for comparison. The AUC of KT is much higher than the AUC of KMeans $(0.726 > 0.579)$, demonstrating KT is a much better clustering method for this dataset than KMeans.

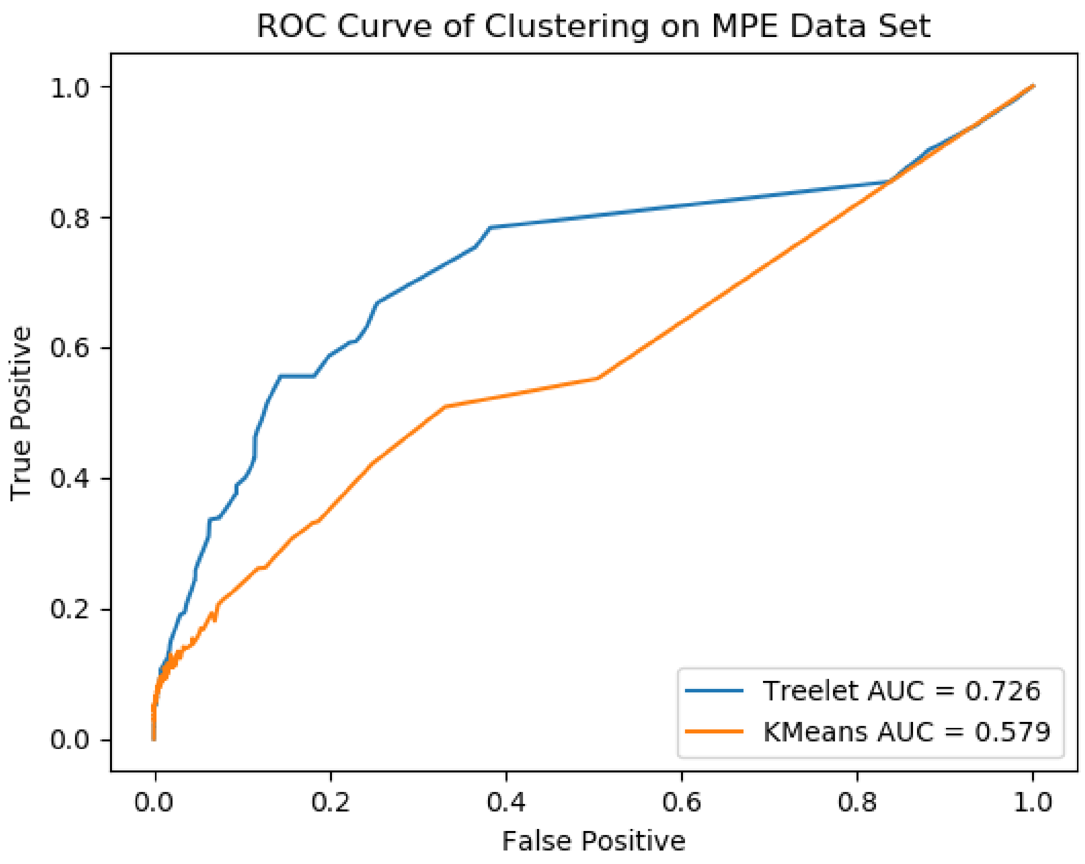

Figure 4: Comparison of KT and KMeans on MPE dataset

5 Conclusion and Future Plans

In the paper we described a variety of situations where KT can be applied as a hierarchical clustering methods. We provide examples to show that KT is as useful as other hierarchical clustering method in clustering numerical datasets, and is especially competitive for datasets with kernel but without numerical matrix representation since there does not exist a lot of alternatives. Our goals for the future are (1) device KT in clustering problems in a greater variety of fields with different kernels (2) find faster methods to approximate KT outcomes (3) extend KT to applications for semi-supervised tasks.

References

[1] 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.

[2] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

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

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

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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.

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

[15] 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.

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

[17] 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.

[18] 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.