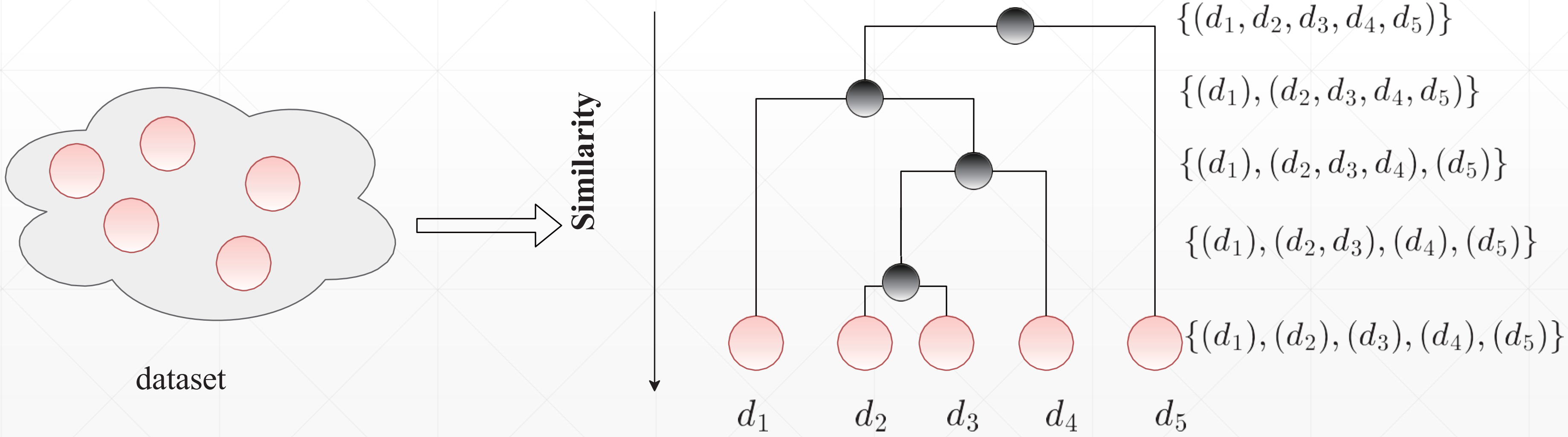
Streaming Hierarchical Clustering Based on Point-Set Kernel

Xin Han, Xi’an Shiyou University, China
Ye Zhu, Deakin University, Australia
Kai Ming Ting, **De-Chuan Zhan**, Nanjing University, China
Gang Li, Deakin University, Australia

Hierarchical Clustering

- Hierarchical clustering is one of the most popular clustering methods.
- It produces a cluster tree that each leaf contains a data point and each internal node represents a sub-cluster.



dataset
Similarity
$\{(d_1, d_2, d_3, d_4, d_5)\}$
$\{(d_1), (d_2, d_3, d_4, d_5)\}$
$\{(d_1), (d_2, d_3, d_4), (d_5)\}$
$\{(d_1), (d_2, d_3), (d_4), (d_5)\}$
$\{(d_1), (d_2), (d_3), (d_4), (d_5)\}$
d_1
d_2
d_3
d_4
d_5

Existing Methods

Classical AHC

- Single-linkage
- Average-linkage
- Complete-linkage

Disadvantages:

- They have at least computational complexity of $O(n^2)$.
- They cannot handle massive datasets and the cluster tree structure produced could not be changed easily.

Scalable AHC

- Sampling based.
- Mini-batches based.
- Approximations based.

Disadvantages:

- These approaches trade off the clustering quality for fast linkage/similarity calculations.

Contributions

- Proposing the first kernel-based incremental hierarchical clustering algorithm StreaKHC for clustering massive streaming data
- Developing an efficient tree updating strategy in real-time for StreaKHC. This updating strategy does not rely on any sampling, mini-batch or approximation method.
- Verifying the effectiveness and efficiency of StreaKHC on 17 synthetic and real-world datasets.

Isolation Kernel

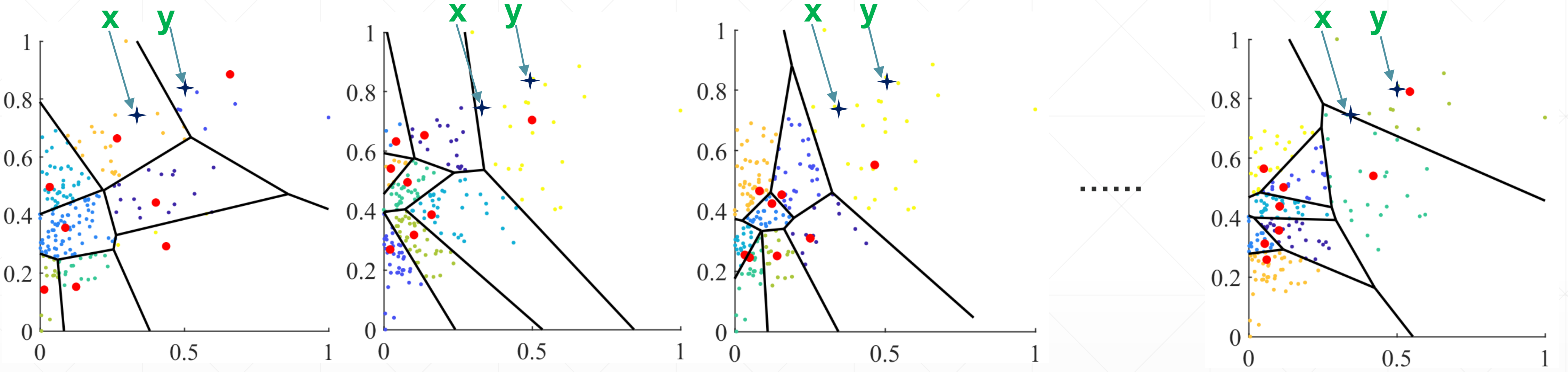
The key idea of the Isolation kernel [1] is using a space partitioning strategy to split the whole data space into ψ non-overlapping partitions based on a random sample of ψ points from a given dataset. The similarity between any two points is the expectation that these two points are found in the same partition.

$$K_{\psi}(\mathbf{x},\mathbf{y}|D) = \mathbb{E}_{\mathcal{H}_{\psi}(D)}[\mathbb{I}(\mathbf{x},\mathbf{y} \in \theta|\theta \in H)] \cong \frac{1}{t}\sum_{i=1}^{t} \mathbb{I}(\mathbf{x},\mathbf{y} \in \theta|\theta \in H_i\,),$$

where $H \in \mathcal{H}_{\psi}(D)$ is one partitioning based on a subsample with size ψ, *and* $\mathbb{I}$ is an indicator function.

[1] Qin, X., Ting, K. M., Zhu, Y., & Lee, V. C. (2019, July). Nearest-neighbour-induced isolation similarity and its impact on density-based clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 4755-4762).

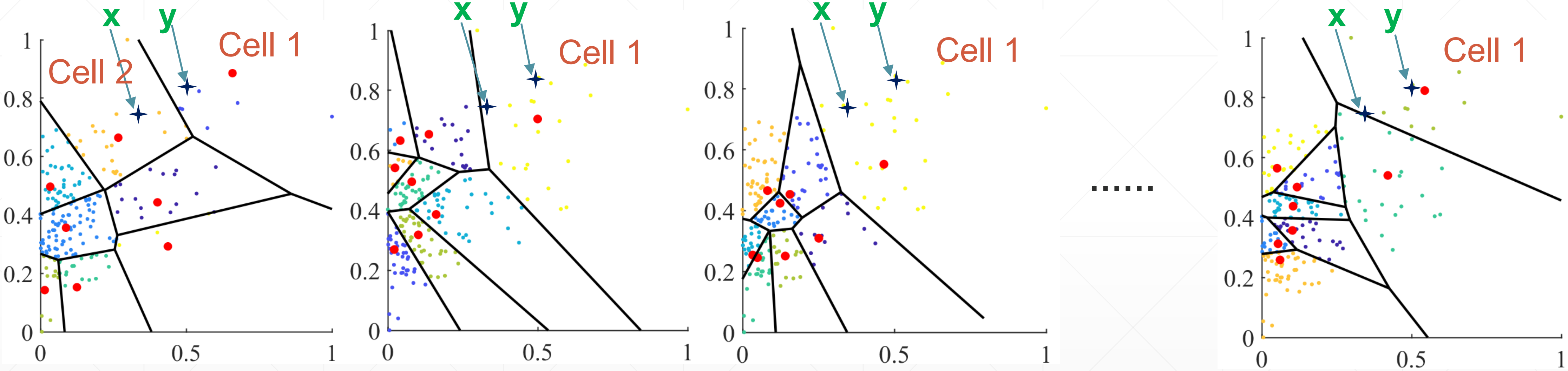
Isolation Kernel Calculation



We can use a nearest neighbour method to split a data space into 8 non-overlapping partitions, and independently conduct this partitioning strategy for t=100 trials. If two points $\mathbf{x}$ and $\mathbf{y}$ are located in the same partition (sharing the same nearest subsample point) in 25 out of 100 trials, then the similarity between $\mathbf{x}$ and $\mathbf{y}$ is estimated as 0.25, i.e., $K_8\ (\mathbf{x}, \mathbf{y}|D) = 0.25$.

Isolation Kernel Feature Map

$\Phi(\mathbf{x})$ is a binary vector that represents the partitions in all the partitionings, where **x** falls in to only one of ψ cells in each partitioning.



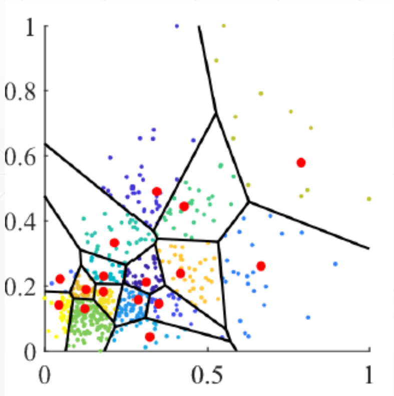
$\Phi(\mathbf{x})$ - > [0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ...... 1 0 0 0 0 0 0 0]
$\Phi(\boldsymbol{y})$ - > [1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 ...... 1 0 0 0 0 0 0 0]

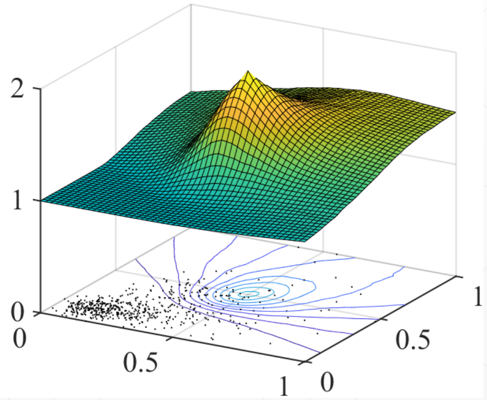
$$K_{\psi}(\mathbf{x},\boldsymbol{y}|D) = \frac{1}{t} < \Phi(\mathbf{x}), \Phi(\boldsymbol{y})) >$$

Isolation Kernel Properties

- Isolation kernel adapts to local density distribution. The isolation mechanism of IK produces large partitions in sparse regions and small partitions in dense regions, based on the random subsamples.

- The probability of two points from the dense cluster falling into the same isolating partition is lower than two points of equal inter-point distance from the sparse cluster, i.e., two points in a sparse region are more similar than two points of equal inter-point distance in a dense region.





Contours with reference to point (0.5, 0.5).

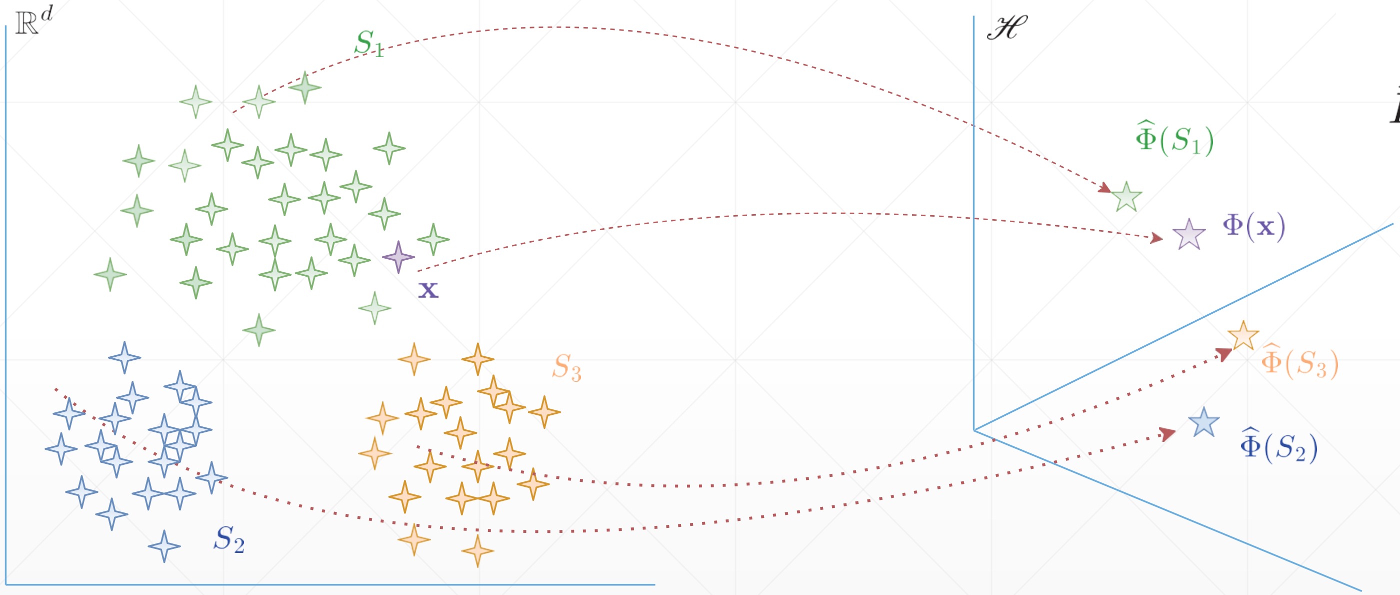
Point-Set Kernel

Given a point $\mathbf{x}$ and a set $A = \{\mathbf{y}_i\}_{i=1}^{p}$, and $\mathbf{x}, \mathbf{y_i} \in R^d$, the point-set similarity between $\mathbf{x}$ and A is the average pairwise similarity between $\mathbf{x}$ and every point in A, defined as follows:

$$\widehat{K}_\psi(\mathbf{x}, \mathrm{A}|D) = \frac{1}{|\mathrm{A}|}\sum_{\mathbf{y}\in A} K_\psi(\mathbf{x}, \boldsymbol{y}|D) = \frac{1}{t} < \Phi(\mathbf{x}), \widehat{\Phi}(A) >$$

Where $\widehat{\Phi}(A) = \frac{1}{|A|}\sum_{\mathbf{y}} \Phi(\mathbf{y})$ is the kernel mean map of K_ψ.

Point-Set Kernel (cont.)



we normalise it to $[0, 1]$ as

$$\widehat{K}_{\psi}(\mathbf{x}, A|D) = \frac{\langle \Phi(\mathbf{x}), \widehat{\Phi}(A) \rangle}{\sqrt{\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle} \sqrt{\langle \widehat{\Phi}(A), \widehat{\Phi}(A) \rangle}}$$

$$\widehat{\Phi}(A) = \frac{1}{|A|} \sum_{\mathbf{y} \in A} \Phi(\mathbf{y})$$

Because $\widehat{\Phi}(A)$ can be pre-calculated, estimating the similarity between a point and a set points costs constant time $O(1)$.

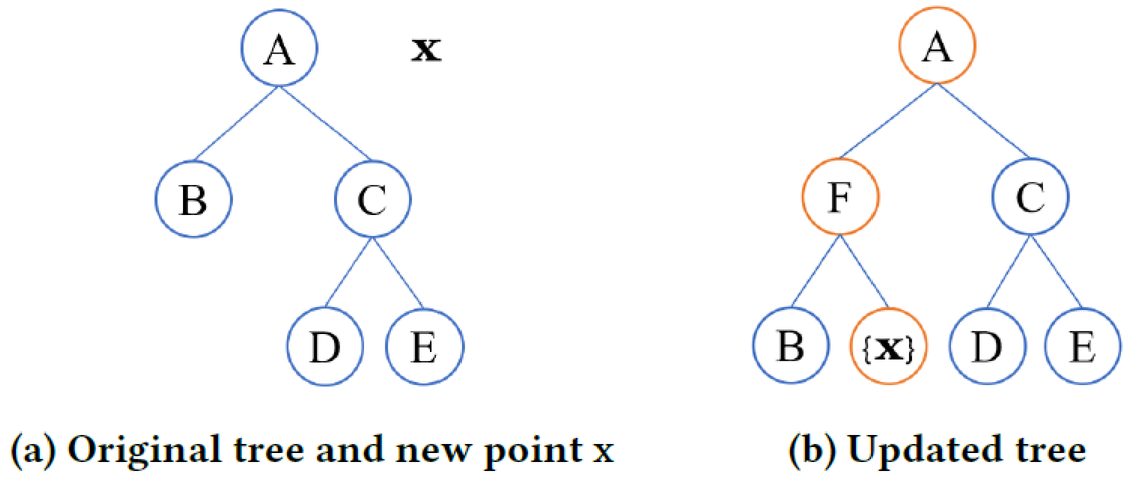
StreaKHC

StreaKHC is a kernel-based incremental hierarchical clustering algorithm.

- Building an Isolation kernel and mapping points to its feature space.
- Employing a top-down search strategy to grow a tree with a new point, i.e., it recursively add the new point $\mathbf{x}$ with $\mathbf{x}$'s most similar node at each level long the path from the root to a leaf node η.
- Pruning the existing leaf to control the size of the hierarchical cluster tree, i.e., it stores the most recent l points only to reduce the over-size hierarchical tree to a pre-set size limit.

StreaKHC - Example of Adding a Point

1. Add $\mathbf{x}$ into the root
2. Find $\widehat{K}_\psi(\mathbf{x}, S_B) \geq \widehat{K}_\psi(\mathbf{x}, S_C)$
3. Replace B with a subtree having a new parent node containing $F = B \cup \{\mathbf{x}\}$



(a) Original tree and new point x

(b) Updated tree

The time and space complexity of StreaKHC are $O(n)$ and $O(l\psi t)$, respectively.

The distance version StreaKHC-D uses the average-linkage function for point-set dissimilarity calculation as $D(\mathbf{x}, S) = \sum_{\mathbf{y} \in \mathbf{S}} ||\mathbf{x} - \mathbf{y}||_2 / |S|$.

Empirical Evaluation - Baselines

1. **Batch Hierarchical Clustering:**

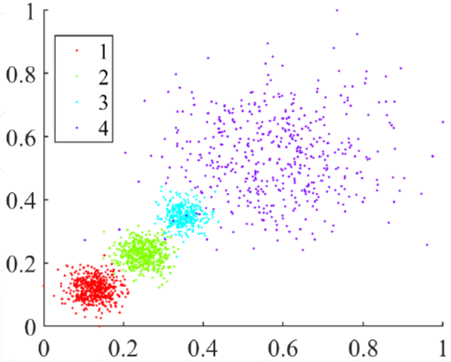
- AHC with average-linkage (Avg.).
- AHC with single-linkage (Sing.).
- AHC with complete-linkage (Comp.)
- PHA. It uses a potential-based linkage function.

2. **Online Hierarchical Clustering:**

- PERCH. It adds a new point to the nearest leaf node of an existing cluster tree and then re-arrange/rotate the tree once detecting a masking situation.
- GRINCH. It is similar to PERCH with an additional grafting subroutine. Two versions of GRINCH are used: approximate average-linkage (GRINCH-A) and cosine similarity linkage (GRINCH-C).

Clustering Results in Dendrogram Purity

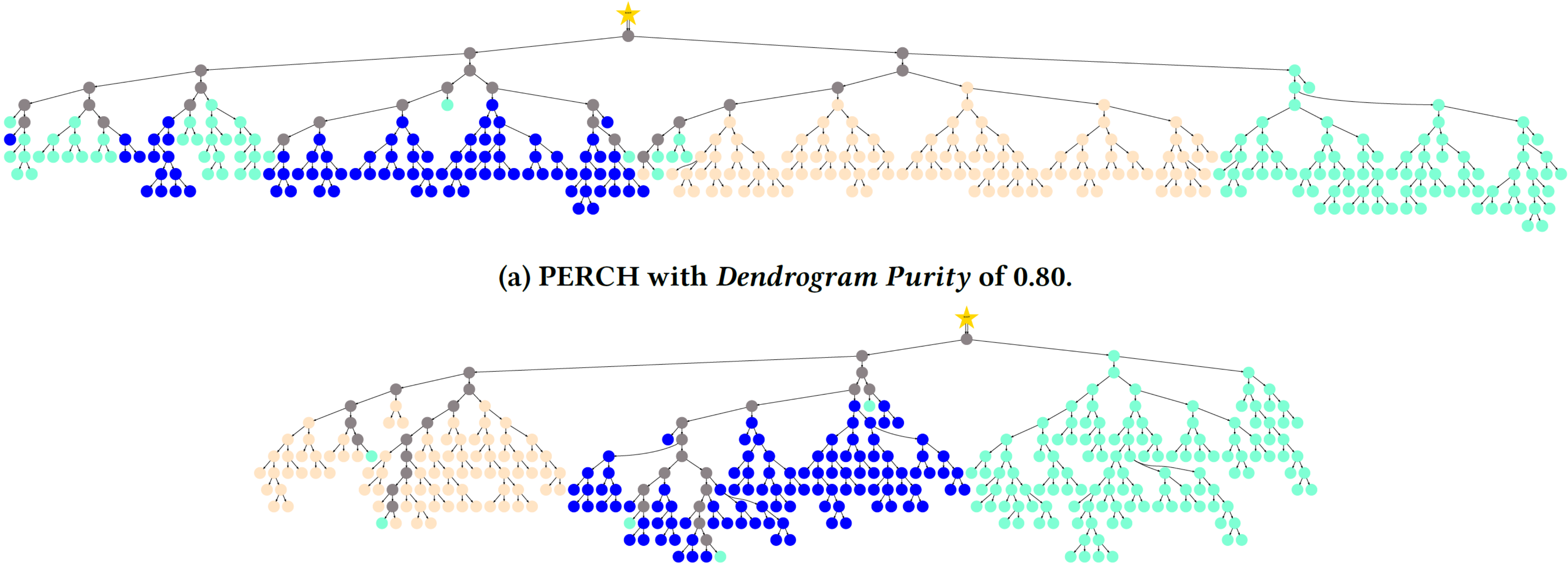
Dataset	#Points	#Dim.	#Clus.	Avg.	Sing.	Comp.	PHA	PERCH	GRINCH-C	GRINCH-A	StreaKHC	StreaKHC-D	PERCH-IK
ALLAML	72	7,129	2	.60	.68	.67	.68	.61 ± .04	.67 ± .03	.67 ± .01	**.69 ± .03**	.68 ± .05	**.72 ± .04**
LSVT	126	310	2	.63	.58	.62	.59	**.65 ± .03**	.61 ± .01	.62 ± .01	**.65 ± .01**	.61 ± .01	**.67 ± .02**
Wine	178	13	3	.89	.68	**.92**	.73	.72 ± .09	.80 ± .06	.88 ± .04	**.91 ± .03**	.85 ± .02	.87 ± .03
Seeds	210	7	3	**.85**	.69	.75	**.84**	.69 ± .08	.68 ± .04	.79 ± .04	.83 ± .01	.81 ± .01	.79 ± .04
Musk	476	166	2	.55	.54	**.56**	.54	.55 ± .01	.55 ± .01	.55 ± .00	.55 ± .01	.54 ± .01	**.57 ± .01**
WDBC	569	30	2	**.86**	.71	.79	.73	.71 ± .05	.64 ± .02	.83 ± .05	**.89 ± .01**	.83 ± .02	.82 ± .05
LandCover	675	147	9	**.56**	.30	.52	.44	.42 ± .03	.43 ± .04	.50 ± .03	**.55 ± .03**	.45 ± .02	.48 ± .02
Hill	1,212	100	2	.50	.50	.50	.50	.50 ± .00	**.57 ± .01**	.50 ± .00	**.51 ± .00**	.50 ± .00	**.51 ± .00**
Banknote	1,372	4	2	.68	**.92**	.63	.62	.66 ± .04	.63 ± .03	.71 ± .05	**.80 ± .05**	.63 ± .01	.77 ± .07
Synthetic-1	1,800	2	4	**.95**	.77	.88	.86	.78 ± .07	.28 ± .00	.87 ± .05	**.95 ± .00**	.89 ± .03	.94 ± .02
Spam	4,601	57	2	.58	.59	.57	.55	.57 ± .01	.56 ± .02	.58 ± .01	**.68 ± .01**	.62 ± .02	**.63 ± .02**
ImageNet-10	13,000	128	10	**.87**	.77	.27	.69	.67 ± .06	.70 ± .04	.71 ± .02	**.86 ± .01**	.76 ± .03	.70 ± .01
STL-10	13,000	128	10	**.62**	.50	.22	.53	.41 ± .04	.41 ± .02	.42 ± .01	**.61 ± .01**	.54 ± .02	.42 ± .02
CIFAR-10	60,000	128	10	-	-	-	-	.42 ± .02	.43 ± .03	.45 ± .03	**.68 ± .01**	**.61 ± .03**	.45 ± .02
Mnist	70,000	128	10	-	-	-	-	.20 ± .00	.23 ± .02	.24 ± .02	**.41 ± .01**	**.36 ± .01**	.32 ± .02
CoverType	581,012	54	7	-	-	-	-	**.45 ± .00**	.43 ± .00	.43 ± .00	.43 ± .01	.41 ± .01	**.46 ± .03**
Synthetic-2	1,800,000	2	4	-	-	-	-	**.73 ± .02**	-	-	**.81 ± .03**	-	-
Average of the first 13 datasets				.70	.63	.61	.64	.61	.58	.66	.73	.67	.68



Synthetic 1 and 2

- StreaKHC-D is StreaKHC using Euclidean distance only.
- PERCH-IK: The same PERCH algorithm running on Isolation kernel space.

Visualisation on Wine Dataset



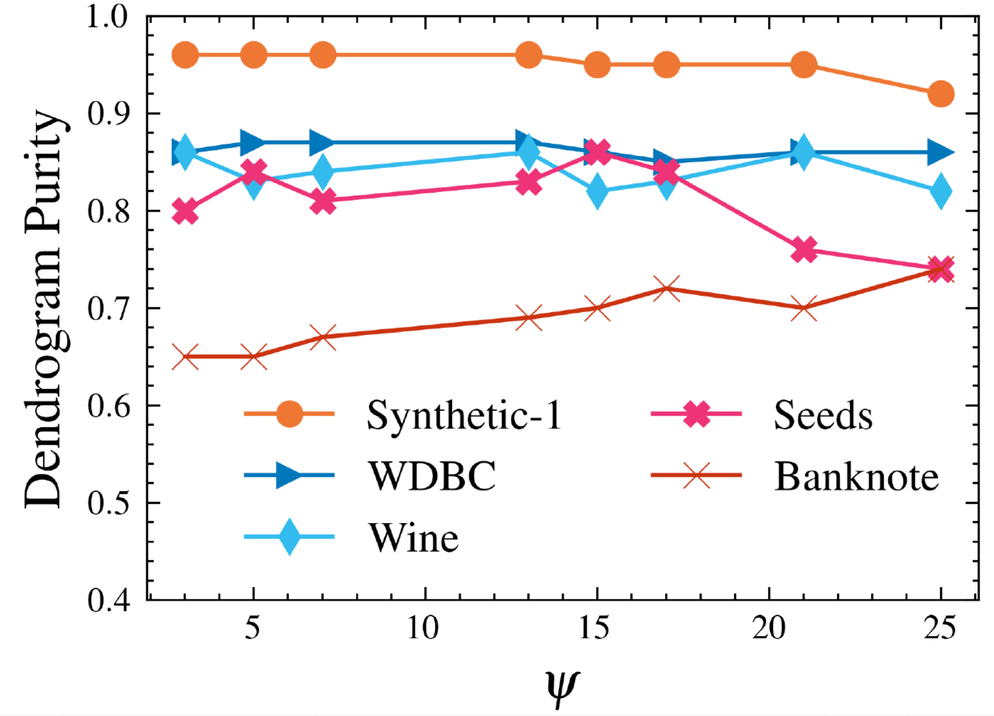
(a) PERCH with *Dendrogram Purity* of 0.80.

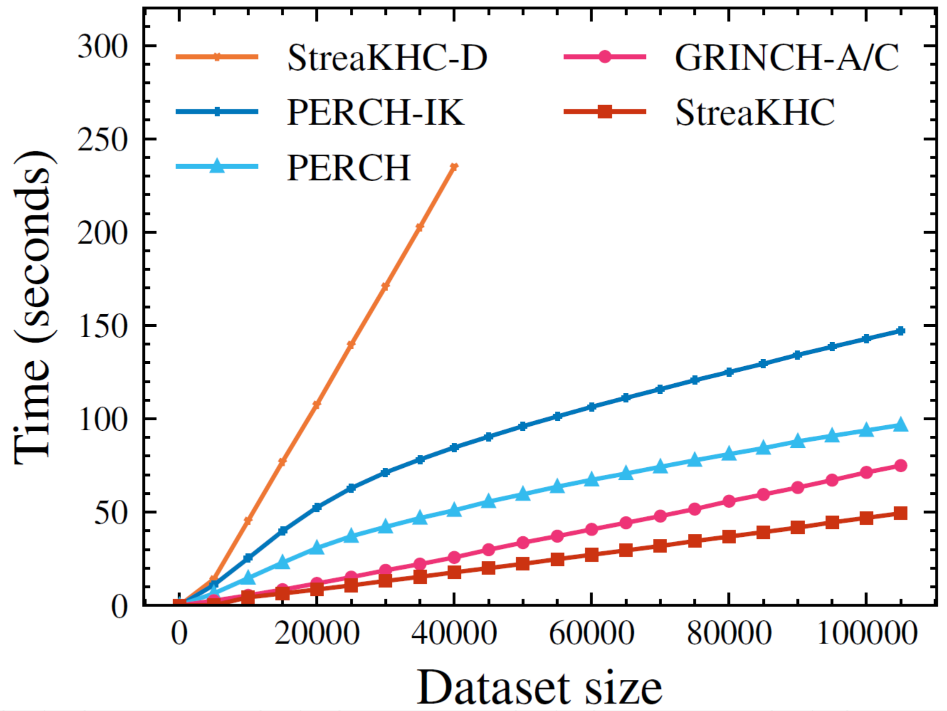
(b) `StreaKHC` with *Dendrogram Purity* of 0.95.

Figure 4: Clustering dendrograms (cluster trees) on the Wine dataset. Each leaf node contains a data point, and each colour represents a ground-truth label of a node if all the points in it belong to the same label, otherwise, the node is coloured grey.

Sensitivity and Scalability Test

- Maintains a stable clustering quality within a certain parameter range.
- The fastest among existing online hierarchical clustering algorithms.





Conclusion

- StreaKHC conducts its search in a top-down manner, avoiding searching all nodes in the cluster tree as required by existing methods.

- It is designed based on the point-set kernel that has constant time for each similarity computation, it updates the cluster tree very efficiently with each emerging new data point, and maintains a high-quality cluster tree in real-time.

- It utilises the data-dependent property of Isolation kernel to effectively detect clusters of varied densities in which most existing algorithms have difficulty separating.

StreaKHC can be obtained from:
https://github.com/tulip-lab/open-code/tree/master/StreaKHC