[Array 17 (2023) 100274](https://doi.org/10.1016/j.array.2022.100274)



Contents lists available at [ScienceDirect](https://www.elsevier.com/locate/array)

Array

journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array)

Random projection tree similarity metric for SpectralNet

Mashaan Alshammari [a](#_bookmark0),[∗](#_bookmark3), John Stavrakakis [b](#_bookmark1), Adel F. Ahmed [c](#_bookmark2), Masahiro Takatsuka [b](#_bookmark1)

a *Independent Researcher, Riyadh, Saudi Arabia*

b *School of Computer Science, The University of Sydney, NSW, Australia*

c *Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia*

A R T I C L E I N F O A B S T R A C T

*Keywords:*

*k*-nearest neighbor Random projection trees SpectralNet

Graph clustering Unsupervised learning

data. So far it was only used with *𝑘*-nn graphs, which are usually constructed using a distance metric (e.g., SpectralNet is a graph clustering method that uses neural network to find an embedding that separates the Euclidean distance). *𝑘*-nn graphs restrict the points to have a fixed number of neighbors regardless of the

local statistics around them. We proposed a new SpectralNet similarity metric based on random projection trees (rpTrees). Our experiments revealed that SpectralNet produces better clustering accuracy using rpTree

similarity metric compared to *𝑘*-nn graph with a distance metric. Also, we found out that rpTree parameters

direction. It is computationally efficient to keep the leaf size in order of log(*𝑛*), and project the points onto a do not affect the clustering accuracy. These parameters include the leaf size and the selection of projection

random direction instead of trying to find the direction with the maximum dispersion.

# Introduction

Graph clustering is one of the fundamental tasks in unsupervised learning. The flexibility of modeling any problem as a graph has made graph clustering very popular. Extracting clusters’ information from graph is computationally expensive, as it usually done via eigen decomposition in a method known as spectral clustering. A recently proposed method, named as SpectralNet [[1](#_bookmark21)], was able to detect clus- ters in a graph without passing through the expensive step of eigen decomposition.

SpectralNet starts by learning pairwise similarities between data points using Siamese nets [[2](#_bookmark22)]. The pairwise similarities are stored in

an affinity matrix *𝐴*, which is then passed through a deep network to

learn an embedding space. In that embedding space, pairs with large

can be clustered together by running *𝑘*-means in that embedding space. similarities fall in a close proximity to each other. Then, similar points

In order for SpectralNet to produce accurate results, it needs an affinity matrix with rich information about the clusters. Ideally, a pair of points in the same cluster should be connected with an edge carrying a large weight. If the pair belong to different clusters, they should be connected with an edge carrying a small weight, or no weight which is indicated by a zero entry in the affinity matrix.

SpectralNet uses Siamese nets to learn informative weights that ensure good clustering results. However, the Siamese nets need some information beforehand. They need some pairs to be labeled as negative and positive pairs. Negative label indicates a pair of points belonging

to different clusters, and a positive label indicates a pair of points in the same cluster. Obtaining negative and positive pairs can be done in a semi-supervised or unsupervised manner. The authors of SpectralNet have implemented it as a semi-supervised and an unsupervised method. Using the ground-truth labels to assign negative and positive labels, makes the SpectralNet semi-supervised. On the other hand, using a distance metric to label closer points as positive pairs and farther points as negative pairs, makes the SpectralNet unsupervised. In this study, we are only interested in an unsupervised SpectralNet.

and negative pairs. A common approach is to get the nearest *𝑘* neigh- Unsupervised SpectralNet uses a distance metric to assign positive

bors for each point and assign those neighbors as positive pairs. A random selection of farther points are labeled as negative pairs. But this approach restricts all points to have a fixed number of positive pairs, which is unsuitable if clusters have different densities. In this work, we proposed a similarity metric based on random projection trees (rpTrees) [[3](#_bookmark23),[4](#_bookmark24)]. An example of an rpTree is shown in [Fig.](#_bookmark4) [1](#_bookmark4). rpTrees do not restrict the number of positive pairs, as this depends on how many points in the leaf node.

The main contributions of this work can be summarized in the following points:

* Proposing a similarity metric for SpectralNet based on random projection trees (rpTrees) that does not restrict the number of positive pairs and produces better clustering accuracy.

∗ Corresponding author.

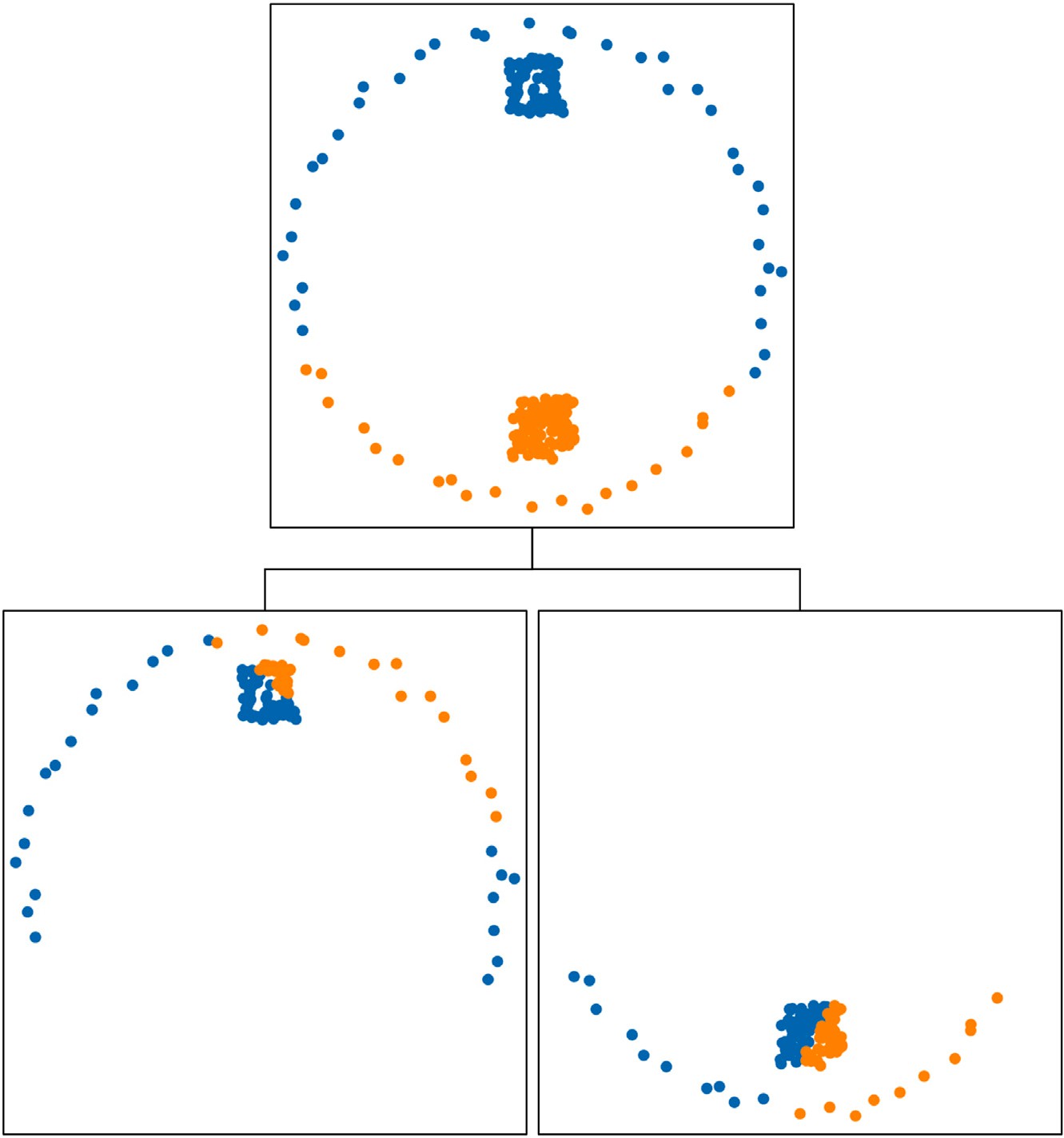
*E-mail addresses:* [mashaan.awad1930@alum.kfupm.edu.sa](mailto:mashaan.awad1930@alum.kfupm.edu.sa) (M. Alshammari), [john.stavrakakis@sydney.edu.au](mailto:john.stavrakakis@sydney.edu.au) (J. Stavrakakis), [adelahmed@kfupm.edu.sa](mailto:adelahmed@kfupm.edu.sa) (A.F. Ahmed), [masa.takatsuka@sydney.edu.au](mailto:masa.takatsuka@sydney.edu.au) (M. Takatsuka).

<https://doi.org/10.1016/j.array.2022.100274>

Received 2 November 2022; Received in revised form 17 December 2022; Accepted 17 December 2022

Available online 21 December 2022

2590-0056/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license ([http://creativecommons.org/licenses/by-](http://creativecommons.org/licenses/by-nc-nd/4.0/) [nc-nd/4.0/](http://creativecommons.org/licenses/by-nc-nd/4.0/)).



**/ig. 1.** An example of rpTree; points in blue are placed in the left branch and points in orange are placed in the right branch. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

* Investigating the influence of the leaf size parameter *𝑛*0 on the

clustering accuracy.

* Performing an in-depth analysis of the projection direction in rpTrees, and examine how it influences the clustering accuracy of SpectralNet.

# Related work

* 1. *Graph neural networks (GNNs)*

GNNs became researchers’ go-to option to perform graph repre- sentation learning. Due to its capability in fusing nodes’ attributes and graph structure, GNN has been widely used in many applications such as knowledge tracing [[5](#_bookmark25)] and sentiment analysis [[6](#_bookmark26)]. The most well-known form of GNN is graph convolutional network (GCN) [[7](#_bookmark27)].

et al. [[8](#_bookmark28)] have proposed to learn the adjacency matrix *𝐴* by running Researchers have been working on improving GCN. Franceschi GCN for multiple iterations and adjusting the graph edges in *𝐴* ac-

cordingly. Another problem with GCN is its vulnerability to adversarial attack. Yang et al. used GCN with domain adaptive learning [[9](#_bookmark29)]. Domain adaptive learning attempts to transfer the knowledge from a labeled source graph to unlabeled target graph. Unseen nodes from the target graph can later be used for node classification.

* 1. *Graph clustering using deep networks*

GCN performs semi-supervised node classification. Due to limited availability of labeled data in some applications, researchers developed graph clustering using deep networks. Yang et al. developed a deep

(GCN) to encode the adjacency matrix *𝐴* and the feature matrix *𝑋*. They model for network clustering [[10](#_bookmark30)]. They used graph neural network also used multilayer perceptron (MLP) to encode the feature matrix *𝑋*.

The output is clustered using Gaussian mixture model (GMM), where GMM parameters are updated throughout training. A similar approach was used by Wang et al. [[11](#_bookmark31)], where they used autoencoders to learn latent representation. Then, they deploy the manifold learning tech- nique UMAP [[12](#_bookmark32)] to find a low dimensional space. The final clustering

assignments are given by *𝑘*-means. Affeldt et al. used autoencoders to

obtain *𝑚* representations of the input data [[13](#_bookmark33)]. The affinity matrices of these *𝑚* representations are merged into a single matrix. Then spectral

clustering was performed on the merged matrix. One drawback with this approach is that it still needs eigen decomposition to find the embedding space.

SpectralNet is another approach for graph clustering using deep

nets to construct the adjacency matrix *𝐴*, which is then passed through networks, which was proposed by Shaham et al. [[1](#_bookmark21)]. They used Siamese

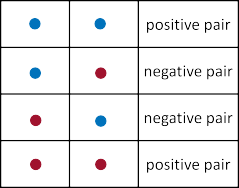
a deep network. Nodes in the embedding space can be clustered using

*𝑘*-means. An extension to SpectralNet was proposed by Huang et al.

[[14](#_bookmark34)], where multiple Siamese nets are trained on multiple views. Each

embedding spaces are fused in the final stage, and *𝑘*-means was run to view is passed into a neural network to find an embedding space. All

find the cluster labels. Another approach to employ deep learning for spectral clustering was introduced by Wada et al. [[15](#_bookmark35)]. Their method starts by identifying hub points, which serve as the core of clusters. These hub points are then passed to a deep network to obtain the cluster labels for the remaining points.



**/ig. 2.** An outline of the used algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

* 1. *Graph similarity metrics*

Every graph clustering method needs a metric to construct pairwise similarities. A shared neighbor similarity was introduced by Zhang et al. [[16](#_bookmark36)]. They applied their method to *attributed graphs*, a special type of graph where each node has feature attributes. They used shared neighbor similarities to highlight clusters’ discontinuity. The concept of shared neighbors could be traced back to Jarvis–Patrick algorithm [[17](#_bookmark37)]. It is important to mention the higher cost associated with shared neighbor similarity. Because all neighbors have to be matched, instead of computing one value such as the Euclidean distance.

Another way of constructing pairwise similarities was introduced

* 1. *SpectralNet*

The first step in SpectralNet is the Siamese network, which consists of two or more neural networks with the same structure and parame- ters. These networks has a single output unit that is connected to the output layers of the networks in the Siamese net. For simplicity let us

assume that the Siamese net consists of two neural networks *𝑁*1 and

*𝑁*2. Both networks received inputs *𝑥*1 and *𝑥*2 respectively, and produce two outputs *𝑧*1 and *𝑧*2. The output unit compared the two outputs using the Euclidean distance. The distance should be small if *𝑥*1 and *𝑥*2 are a

positive pair, and large if they are a negative pair. The Siamese net is trained to minimize contrastive loss, that is defined as:

by Wen et al. [[18](#_bookmark38)], where they utilized Locality Preserving Projection

{‖*𝑧*

– *𝑧* ‖2 *,* if (*𝑥 , 𝑥* ) is a positive pair

(LPP) and hypergraphs. First, all points are projected onto a space with

a heat kernel (Eq. ([1](#_bookmark6))). Second, a hypergraph Laplacian matrix *𝐿𝐻* is reduced dimensionality. The pairwise similarities are constructed using used to replace the regular graph Laplacian matrix *𝐿*. Hypergraphs

*𝐿*contrastive = ‖ 1 2‖ 1 2

*𝑚𝑎𝑥*(*𝑐* − ‖*𝑧*1 − *𝑧*2‖ *,* 0)*,* if (*𝑥*1*, 𝑥*2) is a negative pair*,*

(2)

would help to capture the higher relations between vertices. Two things needed to be considered when applying this method: (1) the

*𝜎* parameter in the heat kernel needs careful tuning [[19](#_bookmark39)], and (2)

the computational cost for hypergraph Laplacian matrix *𝐿𝐻* . Density

information were incorporated into pairwise similarity construction

by Kim et al. [[20](#_bookmark40)]. The method defines (locally dense points) that are separated from each other by (locally sparse points). This approach

where *𝑐* is a constant that is usually set to 1. Then the Euclidean

distance obtained via the Siamese net ‖*𝑧*1 − *𝑧*2‖ is used in the heat

construct the affinity matrix *𝐴*. kernel (see Eq. ([1](#_bookmark6))) to find the similarities between data points and

The SpectralNet uses a gradient step to optimize the loss function

*𝐿𝑆𝑝𝑒𝑐𝑡𝑟𝑎𝑙𝑁 𝑒𝑡*:

1 ∑*𝑚* ‖ ‖2

falls under the category of DBSCAN clustering [[21](#_bookmark41)]. These methods are iterative by nature and need a stopping criterion to be defined.

*𝐿𝑆𝑝𝑒𝑐𝑡𝑟𝑎𝑙𝑁 𝑒𝑡* = *𝑚*2

*𝑖,𝑗*=1

*𝑎𝑖,𝑗* ‖*𝑦𝑖* − *𝑦𝑗* ‖

(3)

*𝐴𝑖,𝑗* = *𝑒𝑥𝑝*

‖*𝑥𝑖* −*𝑥𝑗* ‖2

2*𝜎*2 (1)

2

where *𝑚* is the batch size; *𝑎* of size *𝑚* × *𝑚* is the affinity matrix of the sampled points; *𝑦𝑖* and *𝑦𝑗* are the expected labels of the samples *𝑥𝑖* and

*𝑥𝑗* . But the optimization of this functions is constrained, since the last

Considering the literature on graph representation learning, it is evident that SpectralNet [[1](#_bookmark21)]: (1) offers a cost-efficient method to per- form graph clustering using deep networks and (2) it does not require

labeled datasets. The problem is that it uses *𝑘*-nearest neighbor graph

with distance metric. This restricts points from pairing with more

neighbors if they are in a close proximity. A suitable alternative would be a similarity metric based on random projection trees [[3](#_bookmark23),[4](#_bookmark24)]. rpTrees similarity were already used in spectral clustering by [[22](#_bookmark42),[23](#_bookmark43)]. But they are yet to be extended to graph clustering using deep networks.

# SpectralNet and pairwise similarities

alongside the distance metric that was used for *𝑘*-nearest neighbor The proposed rpTree similarity metric was used in SpectralNet

graph. The SpectralNet algorithm consists of four steps: (1) identifying

negative pairs to construct the affinity matrix *𝐴*, (3) SpectralNet that positive and negative pairs, (2) running Siamese net using positive and

running *𝑘*-means in the embedding space. An illustration of these steps maps points onto an embedding space, and (4) clusters are detected by

is shown in [Fig.](#_bookmark5) [2](#_bookmark5). The next subsection explains the used neural net- works (Siamese and SpectralNet). The discussion of similarity metrics and their complexity is introduced in the following subsections.

layer is set to be a constraint layer that enforces orthogonalization. Therefore, SpectralNet has to alternate between orthogonalization and

gradient steps. Each of these steps uses a different random batch *𝑚*

from the original data *𝑋*. Once the SpectralNet is trained, all sam- ples *𝑥*1*, 𝑥*2*,* … *, 𝑥𝑛* are passed through network to get the predictions

*𝑦*1*, 𝑦*2*,* … *, 𝑦𝑛*. These predictions represent coordinates on the embedding

space, where *𝑘*-means operates and finds the clustering.

* 1. *Constructing pairwise similarities using 𝑘-nn*

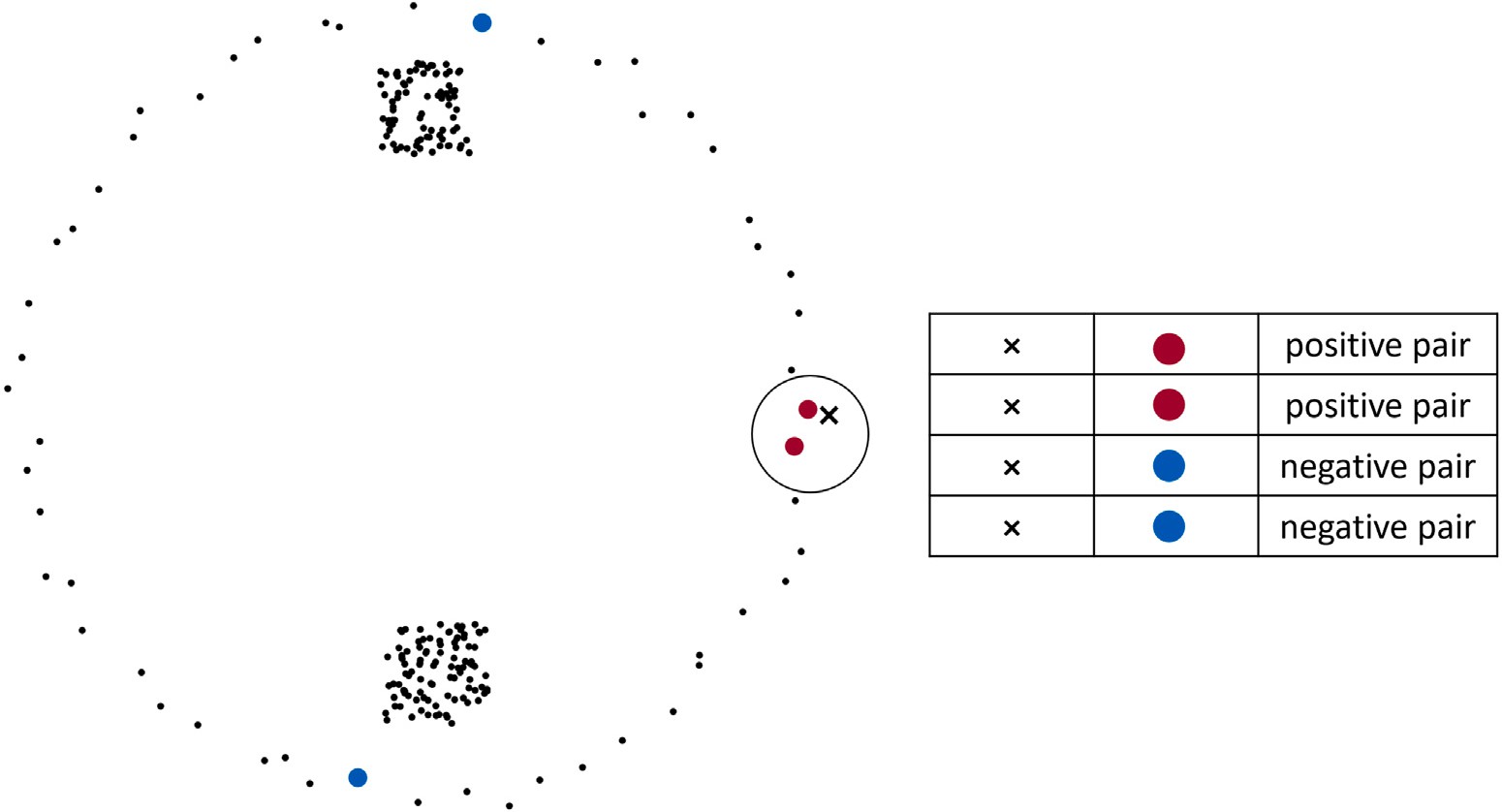
The original algorithm of SpectralNet [[1](#_bookmark21)] has used *𝑘*-nearest neigh- bor graph with distance metric to find positive and negative pairs. The

of *𝑘*, the original method has *𝑘* set to be 2. The negative pairs were positive pairs are the nearest neighbors according to the selected value

selected randomly from the farther neighbors. An illustration of this process is shown in [Fig.](#_bookmark7) [3](#_bookmark7)

be a disadvantage of using *𝑘*-nn. That is a problem we are trying to Restricting the points to have a fixed number of positive pairs can

overcome by using rpTrees to construct positive and negative pairs. In rpTrees, there is no restriction on how many number of pairs for individual points. It depends on how many points ended up in the same leaf node.



**/ig. 3.** Constructing positive and negative pairs using *𝑘*-nn search; red points are the nearest neighbors when *𝑘* = 2; blue points are selected randomly. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

* 1. *Constructing pairwise similarities using rpTrees*

rpTrees start by choosing a random direction *⃗𝑟* from the unit sphere

*𝑆𝐷*−1, where *𝐷* is the number of dimensions. All points in the current node *𝑊* are projected onto *⃗𝑟*. On that reduced space R*𝐷*−1, the algo-

rithm picks a dimension uniformly at random and chooses the split

point *𝑐* randomly between [ 1 *,* 3 ]. The points less than the split point

rpTree and the number of points in each leaf node (*𝑛*0). The leaf size The proposed metric depends on the number of leaf nodes in the

*𝑛*0 is a parameter that can be tuned before running rpTree. It also

determines how many leaf nodes in the tree. Because the method will

stop partitioning when the number of points in a leaf node reaches a minimum limit. Then we have to pair all the points in the leaf node.

To visualize this effect, we have to fix all parameters and vary *𝑛*.

*𝑥 < 𝑐*

4 4

, and the points larger than the split

In [Fig.](#_bookmark10) [5](#_bookmark10), we set *𝑘* to be 2 and 10. The leaf size *𝑛*0 was set to 20 and

are placed in the left child *𝑊𝐿*

point *𝑥 > 𝑐* are placed in the right child *𝑊𝑅*. The algorithm continues

to partition the points recursively, and stops when the split produces a

100. The number of points *𝑛* was in the interval [100*,* 100000]. With

*𝑘*-nn graph we need *𝑛* × *𝑘* positive pairs, and in rpTree similarity we

node with points less than the leaf size parameter *𝑛*0.

need *𝑛* 2 × *𝑛*

*𝑛*0

0

= *𝑛* × *𝑛*0 positive pairs. So, both similarity metrics grow

To create positive pairs for the Simese net, we pair all points in one leaf node. So, points that fall onto the same leaf node are considered similar, and we mark them as positive pairs. For negative pairs, we

pick one leaf node *𝑊𝑥*, and from the remaining set of leaf nodes we

randomly pick *𝑊𝑦*. Then, we pair all points in *𝑊𝑥* with the points in

*𝑊𝑦*, and mark them as negative pairs (Eq. ([4](#_bookmark8))). An illustration of this

process is shown in [Fig.](#_bookmark9) [4](#_bookmark9).

(*𝑝, 𝑞*) ∈ *𝐸*(*𝑝𝑜𝑠𝑖𝑡𝑖𝑣𝑒*) ⇔ *𝑝* ∈ *𝑊𝑥 𝑎𝑛𝑑 𝑞* ∈ *𝑊𝑥*

linearly with *𝑛*. The main difference is how the points are partitioned.

*𝑘*-nn graph uses *𝑘𝑑*-tree which produces the same partition with each

points randomly, so the number of positive pairs will deviate from *𝑛*×*𝑛*0. run making the number of positive pairs fixed. But rpTrees partitions

# Experiments and discussions

In our experiments we compared the similarity metrics using *𝑘*- nearest neighbor and rpTree, in terms of: (1) clustering accuracy and

(*𝑝, 𝑞*) ∈ *𝐸*(*𝑛𝑒𝑔𝑎𝑡𝑖𝑣𝑒*) ⇔ *𝑝* ∈ *𝑊𝑥*

(4)

*𝑎𝑛𝑑 𝑞* ∈ *𝑊𝑦.*

Adjusted Rand Index (ARI) [[26](#_bookmark46)]. Given the true grouping *𝑇* and the (2) storage efficiency. The clustering accuracy was measured using predicted grouping *𝐿*, ARI is computed using pairwise comparisons.

* 1. *Complexity analysis for computing pairwise similarities*

We will use the number of positive pairs to analyze the complexity of the similarity metric used in the original SpectralNet method and the metric proposed in this paper. The original method uses the nearest

*𝑘* neighbors as positive pairs. This is obviously grows linearly with *𝑛*,

since we have to construct *𝑛* × *𝑘* pairs and pass them to the Siamese

net [[2](#_bookmark22)].

Before we analyze the proposed metric, we have to find how many points will fall into a leaf node of an rpTree. This question is usually

asked in proximity graphs [[24](#_bookmark44)]. If we place a squared tessellation *𝑇* on

top of *𝑛* data points (please refer to section 9.4.1 by Barthele√my [[25](#_bookmark45)] for more elaboration). *𝑇* has an area of *𝑛* and a side length of *𝑛*. Each small square *𝑠* in *𝑇* has an area of log(*𝑛*). The probability of having more than *𝑘* neighbors in *𝑠* is *𝑃* (*𝑙 > 𝑘*), where *𝑙* = *𝑘* + 1*,* … *, 𝑛*. The probability *𝑃* (*𝑙 > 𝑘*) follows the homogeneous Poisson process. This probability approximately equals 1 , which is very small, suggesting

*𝑛*

there is a significant chance of having at most log(*𝑛*) neighbors in a square *𝑠*. Since rpTrees follow the same approach of partitioning the

most log(*𝑛*) data points. search space, it is safe to assume that each leaf node would have at

*𝑛*11 if the pair belong to the same cluster in *𝑇* and *𝐿* groupings, and

*𝑛*00 if the pair in different clusters in *𝑇* and *𝐿* groupings. *𝑛*01 and *𝑛*10 if there is a mismatch between *𝑇* and *𝐿*. ARI is defined as:

*𝐴𝑅𝐼* (*𝑇 , 𝐿*) = 2(*𝑛*00*𝑛*11 − *𝑛*01*𝑛*10) *.* (5)

(*𝑛*00 + *𝑛*01)(*𝑛*01 + *𝑛*11) + (*𝑛*00 + *𝑛*10)(*𝑛*10 + *𝑛*11)

The storage efficiency was measured by the number of total pairs used. We avoid using machine dependent metrics like the running time. We also run additional experiments to investigate how the rpTrees parameters are affecting the similarity metric based on rpTree. The

first parameter was the leaf size parameter *𝑛*0, which determines the

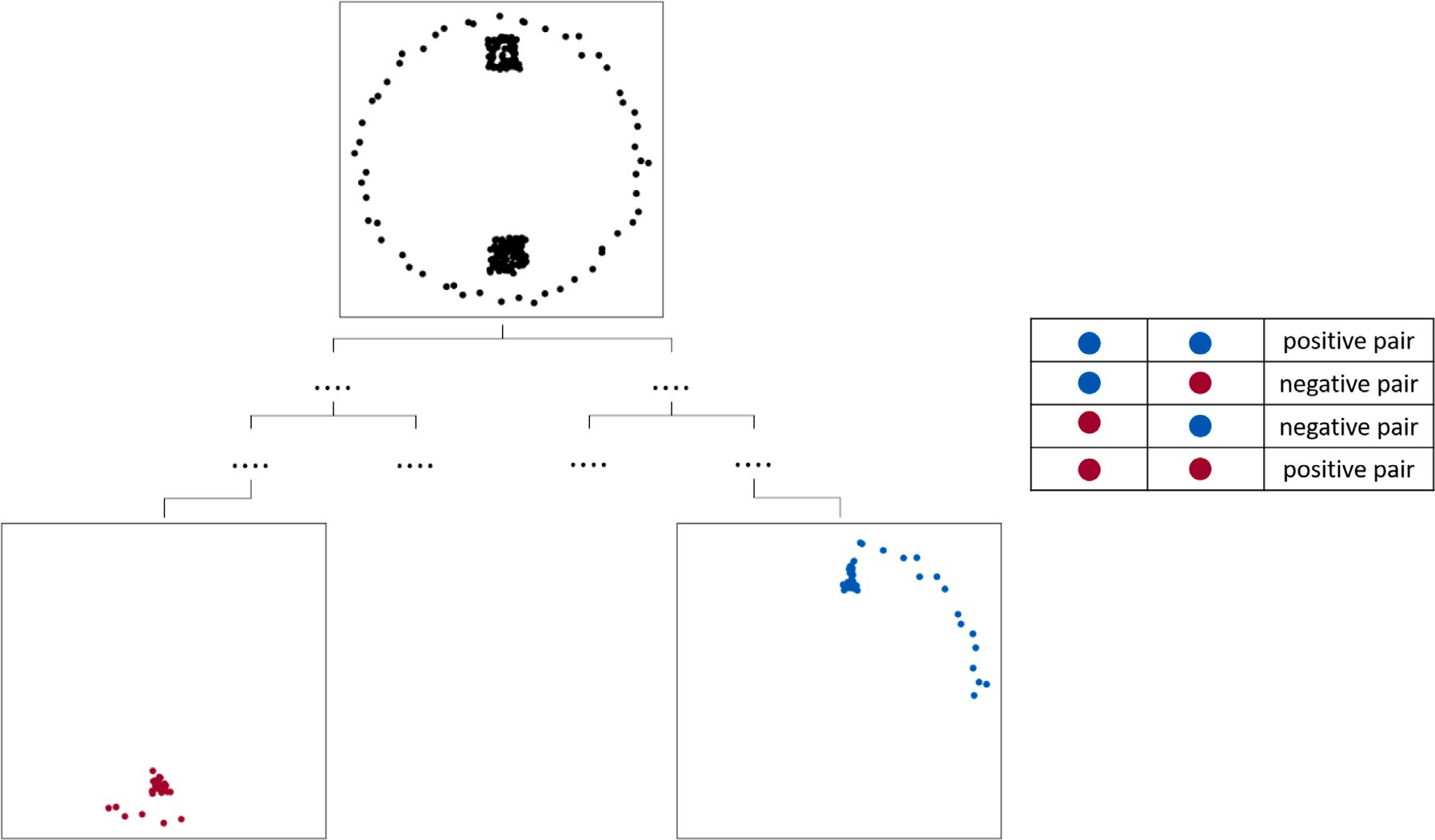
minimum number of points in a leaf node. The second parameter was

how to select the projections direction. There are a number of methods to choose the random direction. We tested these methods to see how they would affect the performance.

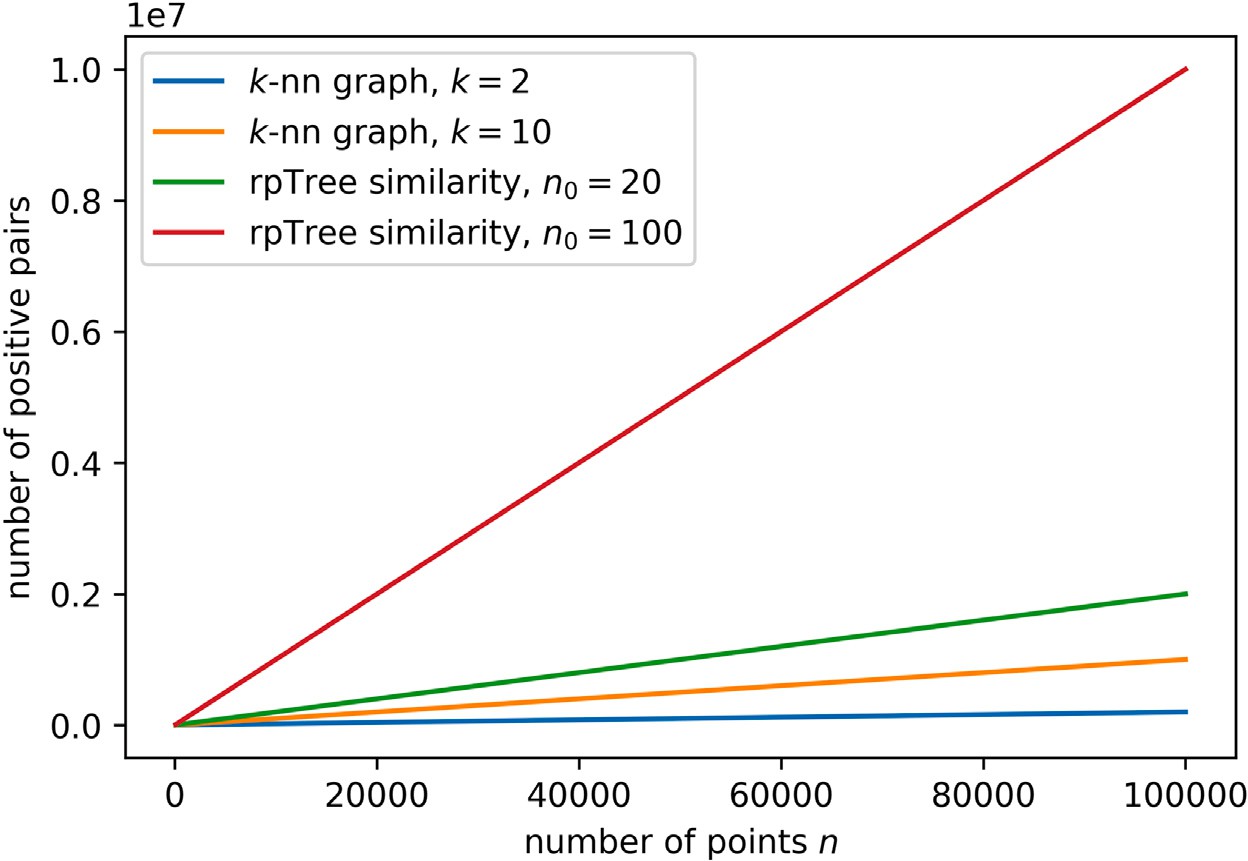
The two dimensional datasets used in our experiments are shown in [Fig.](#_bookmark11) [6](#_bookmark11). The remaining datasets were retrieved from scikit-learn library [[27](#_bookmark47),[28](#_bookmark48)], except for the mGamma dataset which was downloaded from UCI machine learning repository [[29](#_bookmark49)]. All experiments were coded in python 3 and run on a machine with 20 GB of memory and a

3.10 GHz Intel Core i5-10500 CPU. The code can be found on [https:](https://github.com/mashaan14/RPTree)

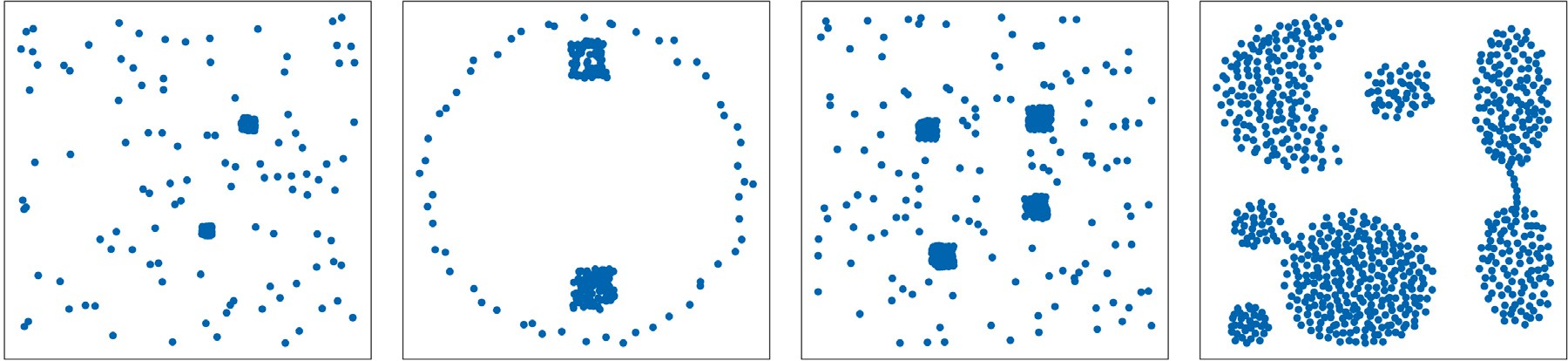
[//github.com/mashaan14/RPTree](https://github.com/mashaan14/RPTree).



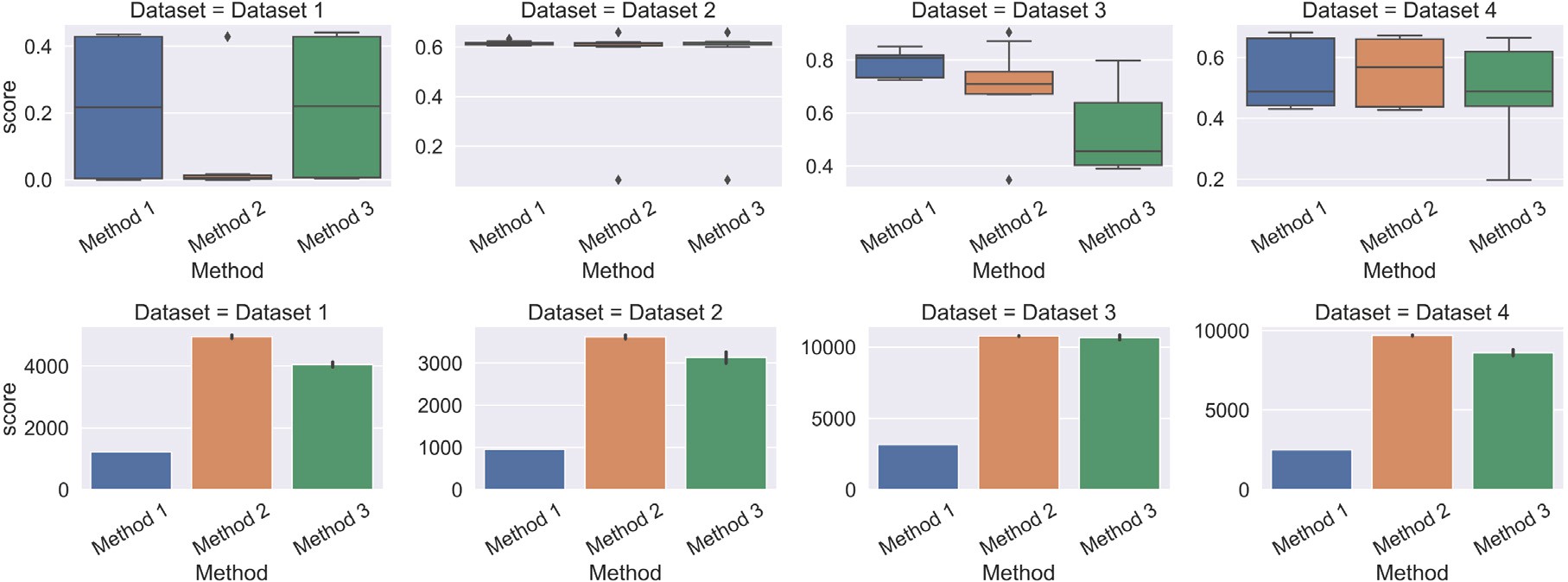
**/ig. 4.** Constructing positive and negative pairs using rpTree. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



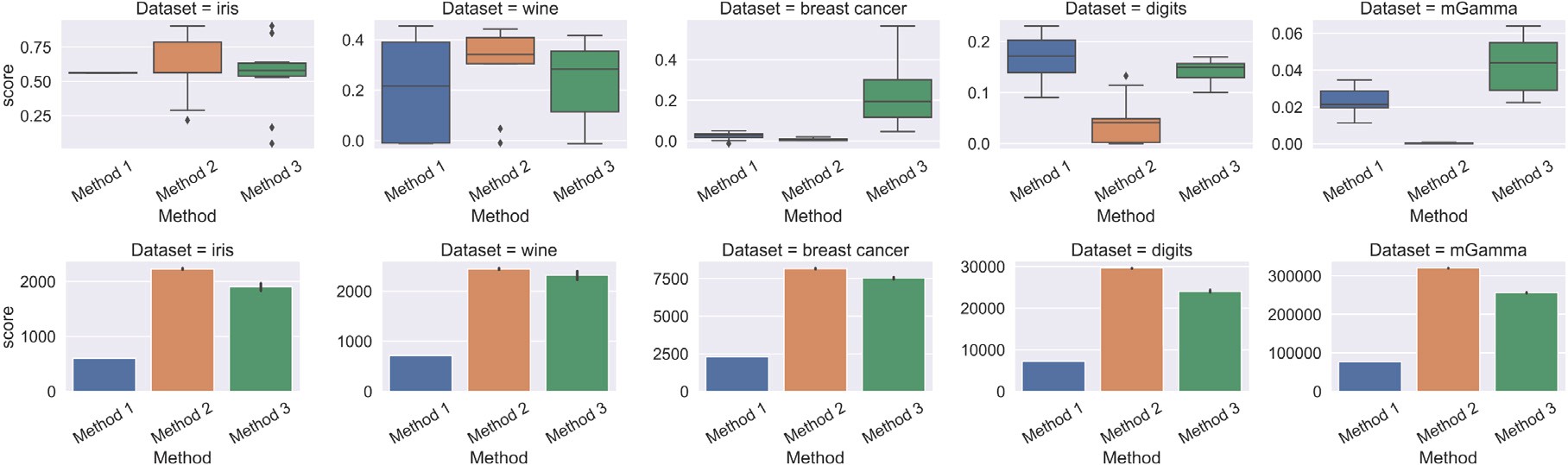
**/ig. 5.** The expected number of positive pairs using *𝑘*-nn and rpTree similarities. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 6.** Synthetic datasets used in the experiments; from left to right Dataset 1 to Dataset 4.



**/ig. 7.** Experiments with synthetic datasets; Method 1 is *𝑘*-nn graph with *𝑘* = 2, Method 2 is *𝑘*-nn graph with varying *𝑘*, and Method 3 is rpTree similarity with *𝑛*0 = 20; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 8.** Experiments with real datasets; Method 1 is *𝑘*-nn graph with *𝑘* = 2, Method 2 is *𝑘*-nn graph with varying *𝑘*, and Method 3 is rpTree similarity with *𝑛*0 = 20; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

* 1. *Experiments using 𝑘-nn and rpTree similarity metrics*

Three methods were used in this experiment. Method 1 is the

developed by Alshammari et al. [[30](#_bookmark50)], it sets *𝑘* dynamically based on the original SpectralNet method by Shaham et al. [[1](#_bookmark21)]. Method 2 was

uses an rpTree similarity instead of *𝑘*-nn graph. statistics around the points. Method 3 is the proposed method which

With the four synthetic datasets, all three methods delivered sim- ilar performances shown in [Fig.](#_bookmark12) [7](#_bookmark12). Apart from Dataset 3, where rpTree similarity performed lower than other methods. This could be attributed to how the clusters are distributed in this dataset. The rpTree splits separated points from the same cluster, which lowers the ARI. Method 2 has the maximum number of pairs over all three methods. The number of pairs in Method 2 and Method 3 deviated slightly from the mean, unlike Method 1 which has the same number of pairs

with each run because *𝑘* was fixed (*𝑘* = 2).

rpTrees similarity outperformed other methods in three out of the

in [Fig.](#_bookmark13) [8](#_bookmark13). *𝑘*-nn with Euclidean distance performed poorly in breast five real datasets iris, breast cancer, and mGamma as shown

cancer, which suggests that connecting to two neighbors was not enough to accurately detect the clusters. Yan et al. reported a similar finding where clustering using rpTree similarity was better than cluster- ing using Gaussian kernel with Euclidean distance [[23](#_bookmark43)]. They showed the heatmap of the similarity matrix generated by the Gaussian kernel and by rpTree.

As for the number of pairs, the proposed similarity metric was the second lowest method that used total pairs across all five datasets. Because of the randomness involved in rpTree splits, the proposed similarity metric has a higher standard deviation for the number of total pairs.

* 1. *Investigating the influence of the leaf size parameter 𝑛*0

One of the important parameters in rpTrees is the leaf size *𝑛*0. It determines when the rpTree stops growing. If the number of points in

a leaf node is less than the leaf size *𝑛*0, that leaf node would not be split

further.

By looking at the clustering performance in synthetic datasets shown in [Fig.](#_bookmark14) [9](#_bookmark14) (top), we can see that we are not gaining much by

increasing the leaf size *𝑛*0. In fact, increasing the leaf size *𝑛*0 might

number of pairs is also related with the leaf size *𝑛*0, as it grows with affect the clustering accuracy like what happened in Dataset 3. The

*𝑛*0. This is shown in [Fig.](#_bookmark14) [9](#_bookmark14) (bottom).

Increasing the leaf size *𝑛*0 helped us to get higher ARI with breast

cancer and mGamma as shown in [Fig.](#_bookmark15) [10](#_bookmark15). With other real datasets it

observed that the number of pairs increases as we increase *𝑛*0. was not improving the clustering accuracy measured by ARI. We also

parameter *𝑛*0. They stated that *𝑛*0 controls the balance between global Ram and Sinha [[31](#_bookmark51)] provided a discussion on how to set the search and local search. Overall, they stated that *𝑛*0 effect on search

accuracy is ‘‘quite benign’’.

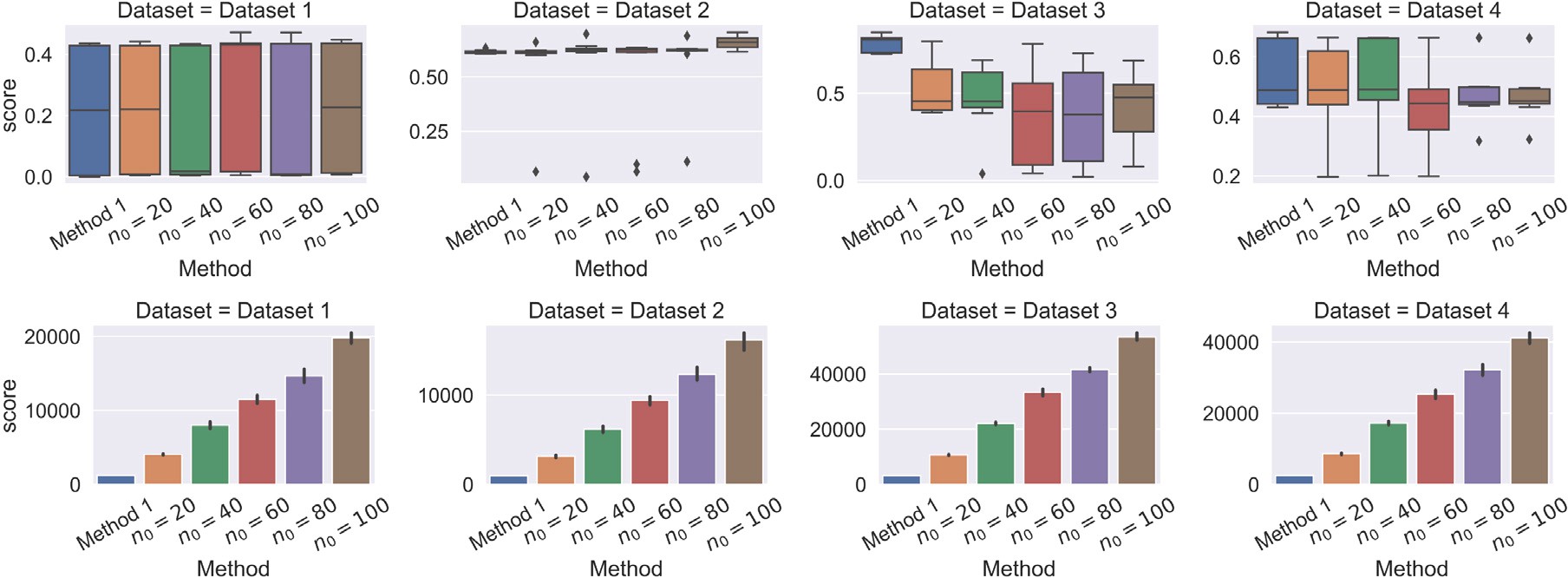
* 1. *Investigating the influence of the dispersion of points along the projec- tion direction*

The original algorithm of rpTrees [[3](#_bookmark23)] suggests using a random direction selected at random. But a recent application of rpTree by Yan

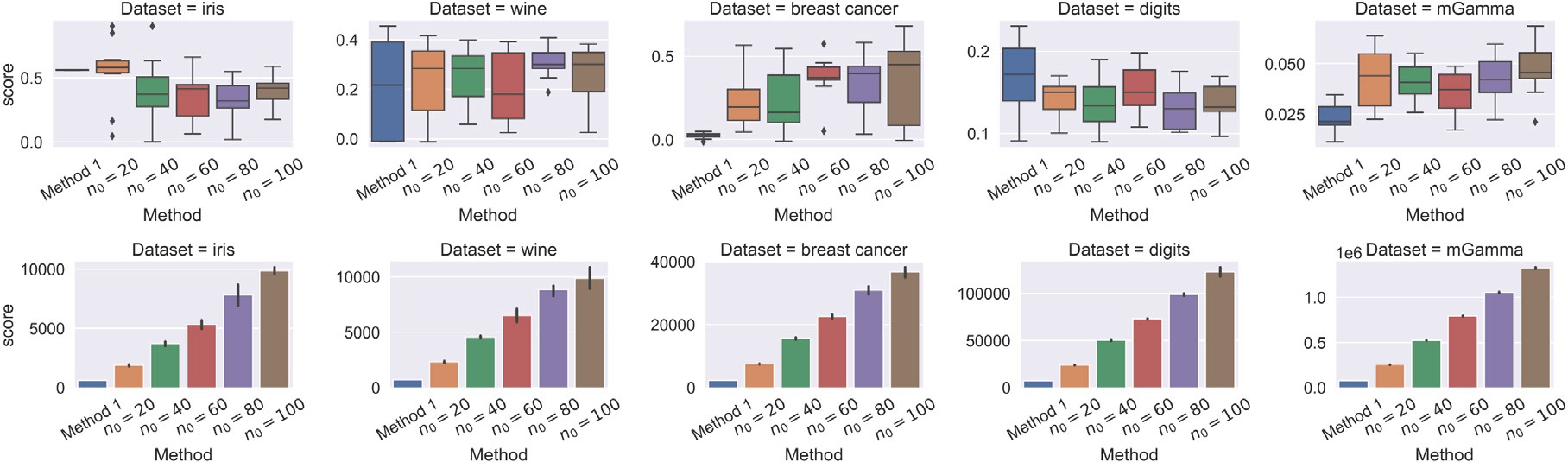
et al. [[32](#_bookmark52)] recommended picking three random directions (*𝑛𝑇 𝑟𝑦* = 3)

and use the one that provides the maximum spread of data points.

To investigate the effect of this parameter, we used four methods for picking a projection direction: (1) picking one random direction, (2)



**/ig. 9.** Experiments with synthetic datasets; Method 1 is *𝑘*-nn graph with *𝑘* = 2, other methods use rpTree similarity with varying *𝑛*0; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 10.** Experiments with real datasets; Method 1 is *𝑘*-nn graph with *𝑘* = 2, other methods use rpTree similarity with varying *𝑛*0; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

picking three random direction (*𝑛𝑇 𝑟𝑦* = 3) and use the one with maximum spread, (3) picking nine random directions (*𝑛𝑇 𝑟𝑦* = 9), and

(4) using principal component analysis (PCA) to find the direction with the maximum spread.

By looking at ARI numbers for synthetic datasets ([Fig.](#_bookmark16) [11](#_bookmark16)) and real datasets ([Fig.](#_bookmark17) [12](#_bookmark17)), we observed that we are not gaining much by trying to maximize the spread of projected points. This parameter has very little effect. Also, all methods with different strategies to pick the projection direction have used the same number of pairs.

In a final experiment, we measured the accuracy differences be- tween a choosing random projection direction against an ideal projec- tion direction. As there are infinite number of projection directions, we instead sample up to 1000 different directions uniformly in the unit sphere, and then pick the best performing among those (see [Fig.](#_bookmark18) [13](#_bookmark18)). For the tested datasets, we compared the best performing direction against the random direction. We found no significant difference among mean of those 100 or 1000 samples with the random vector as shown in [Figs.](#_bookmark19) [14](#_bookmark19) and [15](#_bookmark20). Our finding is supported by the recent efforts in the literature [[33](#_bookmark53)] to limit the number of random directions to make rpTrees more storage efficient.

# Conclusion

The conventional way for graph clustering involves the expensive step of eigen decomposition. SpectralNet presents a suitable alterna- tive that does not use eigen decomposition. Instead the embedding is achieved by neural networks.

metric for *𝑘*-nearest neighbor graph. This approach restricts points from The similarity metric that was used in SpectralNet was a distance being paired with further neighbors because *𝑘* is fixed. A similarity

metric based on random projection trees (rpTrees) eases this restriction and allows points to pair with all points falling in the same leaf node. The proposed similarity metric improved the clustering performance on the tested datasets.

There are number of parameters associated with rpTree similarity metric. Parameters like the minimum number of points in a leaf node

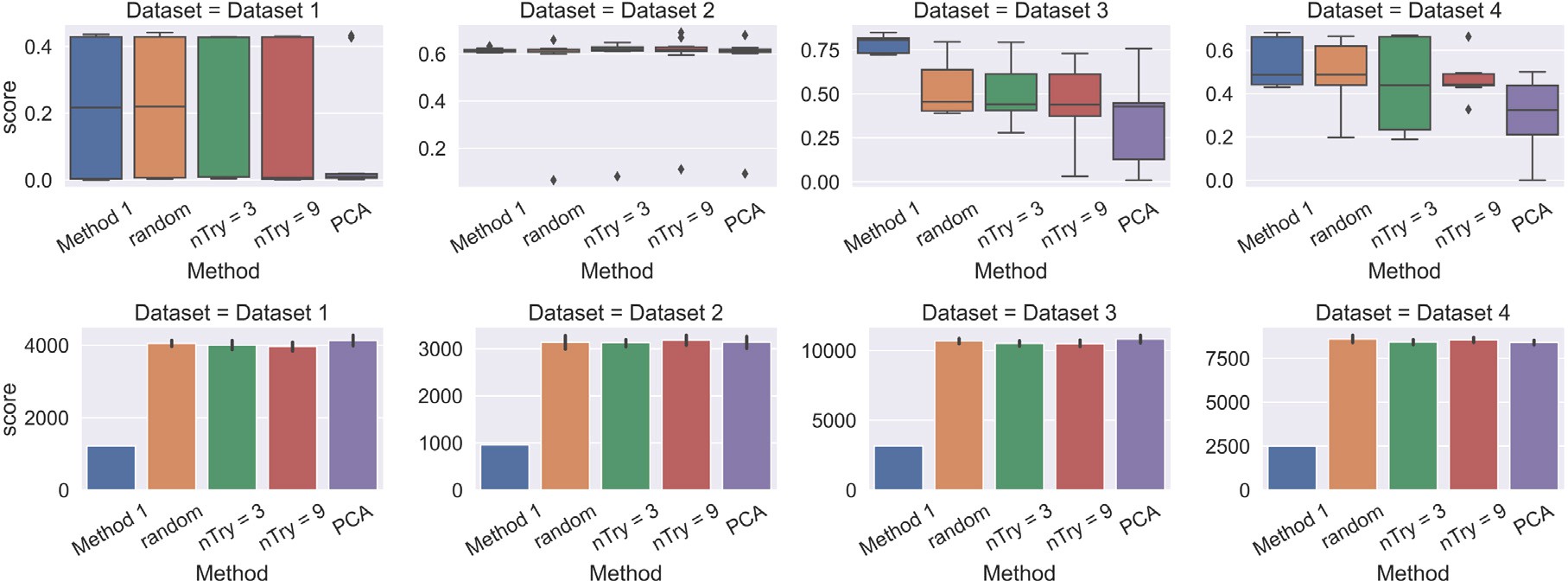
*𝑛*0, and how to select the projection direction to split the points. After

running experiments while varying these parameters, we found that

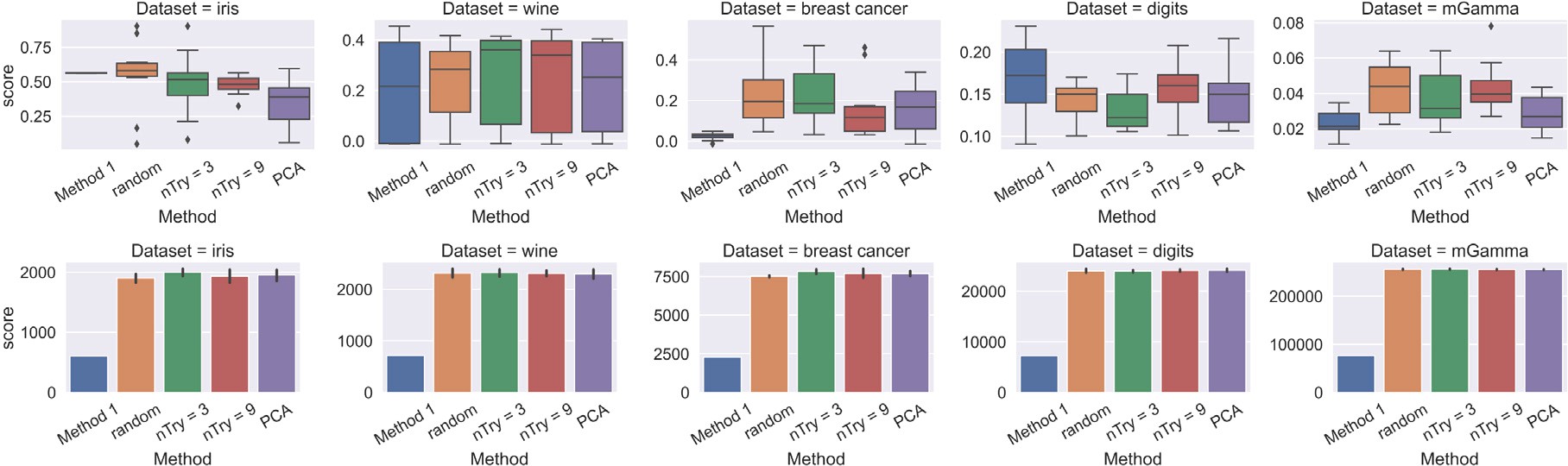
So we recommend keeping the number of points in a leaf node *𝑛*0 in rpTrees parameters have a limited effect on the clustering performance. order of *𝑙𝑜𝑔*(*𝑛*). Also, it is more efficient to project the points onto a

random direction, instead of trying to find the direction with the max- imum dispersion. We conclude that random projection trees (rpTrees) can be used as a similarity metric, where they are applied efficiently as described in this paper.

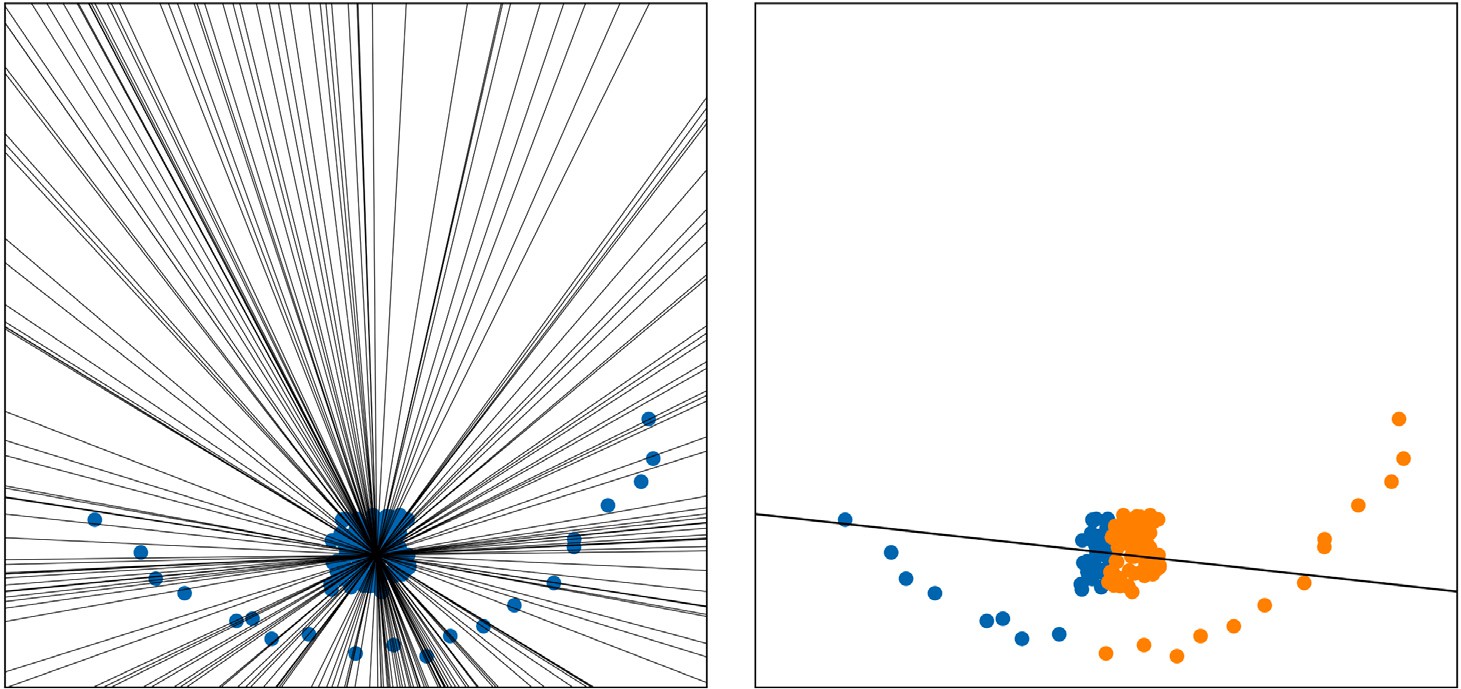
This work can be extended by changing how the pairwise similarity is computed inside the Siamese net. Currently it is done via a heat kernel. Also, one could use other random projection methods such as random projection forests (rpForest) or rpTrees with reduced space complexity. It would be beneficial for the field to see how these space-partitioning trees perform with clustering in deep networks.



**/ig. 11.** Experiments with synthetic datasets; Method 1 is *𝑘*-nn graph with *𝑘* = 2, other methods use different strategies to pick the projection direction; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 12.** Experiments with real datasets; Method 1 is *𝑘*-nn graph with *𝑘* = 2, other methods use different strategies to pick the projection direction; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 13.** (left) Sampling 100 projection directions with maximum orthogonality between them; (right) splitting points along the direction with maximum dispersion. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

# CRediT authorship contribution statement

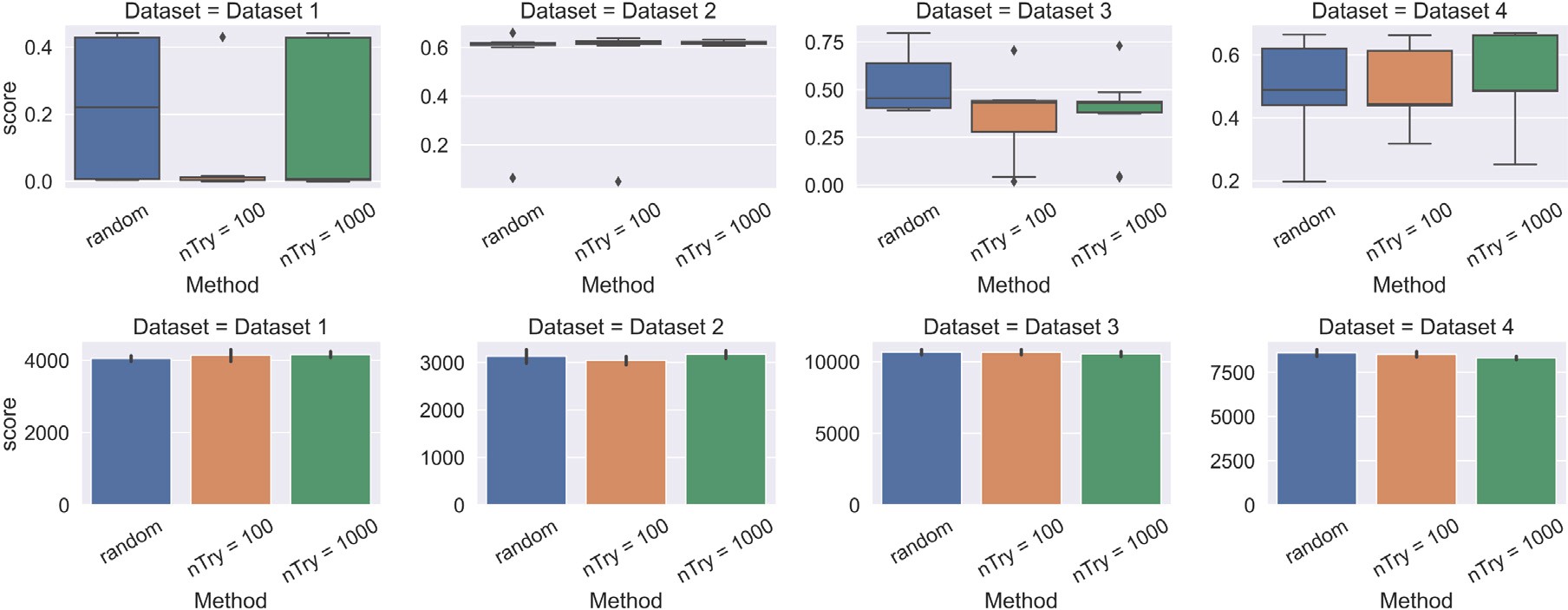
**Mashaan Alshammari:** Conceptualization, Formal analysis, Inves- tigation, Methodology, Project administration, Software, Visualization, Writing – original draft. **John Stavrakakis:** Conceptualization, Formal analysis, Investigation, Methodology, Validation, Writing – review & editing. **Adel /. Ahmed:** Conceptualization, Formal analysis, Super- vision, Validation, Writing – review & editing, Funding acquisition. **Masahiro Takatsuka:** Conceptualization, Formal analysis, Supervision, Validation, Writing – review & editing.

# Declaration of competing interest

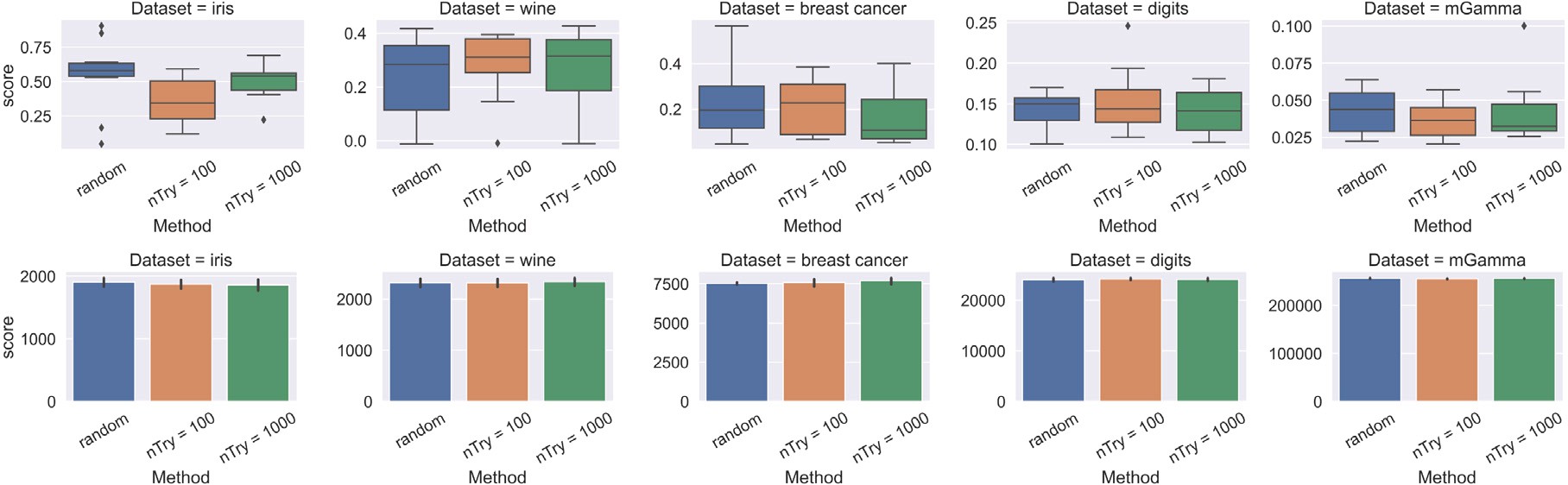
No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work.

# Data availability

I have shared the link to my data/code.



**/ig. 14.** Experiments with synthetic datasets; random represents picking one random projection direction, *𝑛𝑇 𝑟𝑦* = 100 and *𝑛𝑇 𝑟𝑦* = 1000 is the number of sampled directions; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



**/ig. 15.** Experiments with real datasets; random represents picking one random projection direction, *𝑛𝑇 𝑟𝑦* = 100 and *𝑛𝑇 𝑟𝑦* = 1000 is the number of sampled directions; (top) ARI scores for 10 runs, (bottom) number of total pairs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

# References

1. Shaham U, Stanton K, Li H, Nadler B, Basri R, Kluger Y. Spectral- Net: Spectral clustering using deep neural networks. In: 6th international conference on learning representations, ICLR 2018 - conference track pro- ceedings. 2018, URL [https://www.scopus.com/inward/record.uri?eid=2-s2.0-](https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083950872&partnerID=40md5%3Da1d859bb0bc2080faa2ee428a30201b1) [85083950872&partnerID=40md5=a1d859bb0bc2080faa2ee428a30201b1](https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083950872&partnerID=40md5%3Da1d859bb0bc2080faa2ee428a30201b1).
2. [Bromley J, Guyon I, LeCun Y, Säckinger E, Shah R. Signature verification using](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb2) [a ‘‘Siamese’’ time delay neural network. In: Proceedings of the 6th international](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb2) [conference on neural information processing systems. San Francisco, CA, USA:](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb2) [Morgan Kaufmann Publishers Inc.; 1993, p. 737–44.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb2)
3. Dasgupta S, Freund Y. Random projection trees and low dimensional manifolds. In: Proceedings of the fortieth annual ACM symposium on theory of computing. New York, NY, USA: Association for Computing Machinery; 2008, p. 537–46. <http://dx.doi.org/10.1145/1374376.1374452>.
4. Freund Y, Dasgupta S, Kabra M, Verma N. Learning the structure of man- ifolds using random projections. In: Platt J, Koller D, Singer Y, Roweis S, editors. Advances in neural information processing systems, Vol. 20. Cur- ran Associates, Inc.; 2008, URL [https://proceedings.neurips.cc/paper/2007/file/](https://proceedings.neurips.cc/paper/2007/file/9fc3d7152ba9336a670e36d0ed79bc43-Paper.pdf) [9fc3d7152ba9336a670e36d0ed79bc43-Paper.pdf](https://proceedings.neurips.cc/paper/2007/file/9fc3d7152ba9336a670e36d0ed79bc43-Paper.pdf).
5. Song X, Li J, Cai T, Yang S, Yang T, Liu C. A survey on deep learn- ing based knowledge tracing. Knowl-Based Syst 2022;258:110036. [http://](http://dx.doi.org/10.1016/j.knosys.2022.110036) [dx.doi.org/10.1016/j.knosys.2022.110036](http://dx.doi.org/10.1016/j.knosys.2022.110036), URL [https://www.sciencedirect.com/](https://www.sciencedirect.com/science/article/pii/S0950705122011297) [science/article/pii/S0950705122011297](https://www.sciencedirect.com/science/article/pii/S0950705122011297).
6. Zhou J, Huang JX, Hu QV, He L. SK-GCN: Modeling syntax and knowl- edge via graph convolutional network for aspect-level sentiment classification. Knowl-Based Syst 2020;205:106292. [http://dx.doi.org/10.1016/j.knosys.2020.](http://dx.doi.org/10.1016/j.knosys.2020.106292) [106292](http://dx.doi.org/10.1016/j.knosys.2020.106292).
7. [Kipf TN, Welling M. Semi-supervised classification with graph convolutional](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb7) [networks. In: International conference on learning representations. 2017.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb7)
8. [Franceschi L, Niepert M, Pontil M, He X. Learning discrete structures for graph](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb8) [neural networks. In: Proceedings of the 36th international conference on machine](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb8) [learning. 2019.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb8)
9. Yang S, Cai B, Cai T, Song X, Jiang J, Li B, et al. Robust cross-network node classification via constrained graph mutual information. Knowl-Based Syst 2022;257:109852. <http://dx.doi.org/10.1016/j.knosys.2022.109852>, URL [https:](https://www.sciencedirect.com/science/article/pii/S0950705122009455)

[//www.sciencedirect.com/science/article/pii/S0950705122009455](https://www.sciencedirect.com/science/article/pii/S0950705122009455).

1. Yang S, Verma S, Cai B, Jiang J, Yu K, Chen F, et al. Variational co-embedding learning for attributed network clustering. 2021, [http://dx.doi.org/10.48550/](http://dx.doi.org/10.48550/ARXIV.2104.07295) [ARXIV.2104.07295](http://dx.doi.org/10.48550/ARXIV.2104.07295), arXiv.
2. Wang Y, Chang D, Fu Z, Zhao Y. Learning a Bi-directional discriminative representation for deep clustering. Pattern Recognit 2022;109237. [http://dx.doi.](http://dx.doi.org/10.1016/j.patcog.2022.109237) [org/10.1016/j.patcog.2022.109237](http://dx.doi.org/10.1016/j.patcog.2022.109237).
3. McInnes L, Healy J, Melville J. UMAP: Uniform manifold approximation and projection for dimension reduction. 2018, [http://dx.doi.org/10.48550/ARXIV.](http://dx.doi.org/10.48550/ARXIV.1802.03426) [1802.03426](http://dx.doi.org/10.48550/ARXIV.1802.03426), arXiv.
4. Affeldt S, Labiod L, Nadif M. Spectral clustering via ensemble deep au- toencoder learning (SC-EDAE). Pattern Recognit 2020;108:107522. [http://](http://dx.doi.org/10.1016/j.patcog.2020.107522) [dx.doi.org/10.1016/j.patcog.2020.107522](http://dx.doi.org/10.1016/j.patcog.2020.107522), URL [https://www.sciencedirect.com/](https://www.sciencedirect.com/science/article/pii/S0031320320303253) [science/article/pii/S0031320320303253](https://www.sciencedirect.com/science/article/pii/S0031320320303253).
5. Huang S, Ota K, Dong M, Li F. MultiSpectralNet: Spectral clustering us- ing deep neural network for multi-view data. IEEE Trans Comput Soc Syst 2019;6(4):749–60. <http://dx.doi.org/10.1109/TCSS.2019.2926450>.
6. Wada Y, Miyamoto S, Nakagama T, Andéol L, Kumagai W, Kanamori T. Spectral embedded deep clustering. Entropy 2019;21(8). [http://dx.doi.org/10.](http://dx.doi.org/10.3390/e21080795) [3390/e21080795](http://dx.doi.org/10.3390/e21080795), URL <https://www.mdpi.com/1099-4300/21/8/795>.
7. Zhang X, Liu H, Wu X-M, Zhang X, Liu X. Spectral embedding net- work for attributed graph clustering. Neural Netw 2021;142:388–96. [http://](http://dx.doi.org/10.1016/j.neunet.2021.05.026) [dx.doi.org/10.1016/j.neunet.2021.05.026](http://dx.doi.org/10.1016/j.neunet.2021.05.026), URL [https://www.sciencedirect.com/](https://www.sciencedirect.com/science/article/pii/S0893608021002227) [science/article/pii/S0893608021002227](https://www.sciencedirect.com/science/article/pii/S0893608021002227).
8. Jarvis R, Patrick E. Clustering using a similarity measure based on shared near neighbors. IEEE Trans Comput 1973;C-22(11):1025–34. [http://dx.doi.org/](http://dx.doi.org/10.1109/T-C.1973.223640) [10.1109/T-C.1973.223640](http://dx.doi.org/10.1109/T-C.1973.223640).
9. Wen G, Zhu Y, Zheng W. Spectral representation learning for one-step spectral ro- tation clustering. Neurocomputing 2020;406:361–70. [http://dx.doi.org/10.1016/](http://dx.doi.org/10.1016/j.neucom.2019.09.108) [j.neucom.2019.09.108](http://dx.doi.org/10.1016/j.neucom.2019.09.108), URL [https://www.sciencedirect.com/science/article/pii/](https://www.sciencedirect.com/science/article/pii/S0925231220303477) [S0925231220303477](https://www.sciencedirect.com/science/article/pii/S0925231220303477).
10. [Zelnik-Manor L, Perona P. Self-tuning spectral clustering. Adv Neural Inf Process](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb19) [Syst 2005;1601–8.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb19)
11. Kim J-H, Choi J-H, Park Y-H, Leung CK-S, Nasridinov A. KNN-SC: Novel spectral clustering algorithm using k-nearest neighbors. IEEE Access 2021;9:152616–27. <http://dx.doi.org/10.1109/ACCESS.2021.3126854>.
12. [Ester M, Kriegel H-P, Sander J, Xu X. A density-based algorithm for discovering](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb21) [clusters in large spatial databases with noise. In: Proceedings of the second](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb21) [international conference on knowledge discovery and data mining. AAAI Press;](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb21) [1996, p. 226–31.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb21)
13. Yan D, Huang L, Jordan MI. Fast approximate spectral clustering. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. New York, NY, USA: Association for Computing Machinery; 2009,

p. 907–16. <http://dx.doi.org/10.1145/1557019.1557118>.

1. Yan D, Gu S, Xu Y, Qin Z. Similarity kernel and clustering via random projection forests. 2019, CoRR abs/1908.10506, [arXiv:1908.10506](http://arxiv.org/abs/1908.10506).
2. Gilbert EN. Random plane networks. J Soc Ind Appl Math 1961;9(4):533–43. <http://dx.doi.org/10.1137/0109045>.
3. [Barthelemy M. Morphogenesis of spatial networks. Lecture notes in morphogen-](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb25) [esis, Springer International Publishing; 2017.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb25)
4. Hubert L, Arabie P. Comparing partitions. J Classification 1985;2(1):193–218. <http://dx.doi.org/10.1007/BF01908075>.
5. [Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb27) [Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb27)
6. [Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, et al. API](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb28) [design for machine learning software: Experiences from the scikit-learn project.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb28) [In: ECML PKDD workshop: languages for data mining and machine learning.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb28) [2013, p. 108–22.](http://refhub.elsevier.com/S2590-0056(22)00107-2/sb28)
7. Dua D, Graff C. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences; 2017, URL [http://archive.ics.uci.](http://archive.ics.uci.edu/ml) [edu/ml](http://archive.ics.uci.edu/ml).
8. Alshammari M, Stavrakakis J, Takatsuka M. Refining a k-nearest neighbor graph for a computationally efficient spectral clustering. Pattern Recognit 2021;114:107869. <http://dx.doi.org/10.1016/j.patcog.2021.107869>, URL [https:](https://www.sciencedirect.com/science/article/pii/S003132032100056X)

[//www.sciencedirect.com/science/article/pii/S003132032100056X](https://www.sciencedirect.com/science/article/pii/S003132032100056X).

1. Ram P, Sinha K. Revisiting kd-tree for nearest neighbor search. New York, NY, USA: Association for Computing Machinery; 2019, p. 1378–88. [http://dx.doi.](http://dx.doi.org/10.1145/3292500.3330875) [org/10.1145/3292500.3330875](http://dx.doi.org/10.1145/3292500.3330875).
2. Yan D, Wang Y, Wang J, Wang H, Li Z. K-nearest neighbor search by random projection forests. IEEE Trans Big Data 2021;7(1):147–57. [http://dx.doi.org/10.](http://dx.doi.org/10.1109/TBDATA.2019.2908178) [1109/TBDATA.2019.2908178](http://dx.doi.org/10.1109/TBDATA.2019.2908178).
3. Keivani O, Sinha K. Random projection-based auxiliary information can improve tree-based nearest neighbor search. Inform Sci 2021;546:526–42. [http://dx.doi.](http://dx.doi.org/10.1016/j.ins.2020.08.054) [org/10.1016/j.ins.2020.08.054](http://dx.doi.org/10.1016/j.ins.2020.08.054).