J. Appl. Prob. **57,** 458–476 (2020)
*doi:*10.1017/jpr.2020.21
© *Applied Probability Trust* 2020

RANDOMIZED NEAR-NEIGHBOR GRAPHS, GIANT COMPONENTS AND APPLICATIONS IN DATA SCIENCE

ARIEL JAFFE,[*]
YUVAL KLUGER,[**]
GEORGE C. LINDERMAN,[*]
GAL MISHNE,[***] AND
STEFAN STEINERBERGER,[****] *Yale University*

Abstract

If we pick n random points uniformly in $[0, 1]^d$ and connect each point to its $c_d \log n$ nearest neighbors, where $d \geq 2$ is the dimension and c_d is a constant depending on the dimension, then it is well known that the graph is connected with high probability. We prove that it suffices to connect every point to $c_{d,1} \log \log n$ points chosen randomly among its $c_{d,2} \log n$ nearest neighbors to ensure a giant component of size $n - o(n)$ with high probability. This construction yields a much sparser random graph with $\sim n \log \log n$ instead of $\sim n \log n$ edges that has comparable connectivity properties. This result has non-trivial implications for problems in data science where an affinity matrix is constructed: instead of connecting each point to its k nearest neighbors, one can often pick $k' \ll k$ random points out of the k nearest neighbors and only connect to those without sacrificing quality of results. This approach can simplify and accelerate computation; we illustrate this with experimental results in spectral clustering of large-scale datasets.

Keywords: k-nn graph; random graph; connectivity; sparsification

2010 Mathematics Subject Classification: Primary 60D05; 60K35
Secondary 05C40; 05C80

1. Introduction and main results

1.1. Introduction

The following problem is classical (we refer to the books of Penrose [26], Walters [39], and references therein). If n points are randomly chosen in $[0, 1]^2$ and we connect every point to its k nearest neighbors, what is the likelihood of obtaining a connected graph? It is not very difficult to see that $k \sim \log n$ is the right order of magnitude. Arguments for both directions are sketched in the first section of a paper by Balister, Bollobás, Sarkar, and Walters [2]. Establishing precise results is more challenging; the same paper shows that $k \leq 0.304 \log n$

Received 2 March 2018; revision received 16 December 2019.
* Postal address: Program in Applied Mathematics, Yale University, New Haven, CT 06511, USA.
Email address: a.jaffe@yale.edu; george.linderman@yale.edu
** Postal address: Department of Pathology and Applied Mathematics, Yale University, New Haven, CT 06511, USA.
Email address: yuval.kluger@yale.edu
*** Postal address: Halıcıoğlu Data Science Institute, UC San Diego, La Jolla, CA 92093, USA.
Email address: gmishne@ucsd.edu
**** Postal address: Department of Mathematics, Yale University, New Haven, CT 06511, USA.
Email address: stefan.steinerberger@yale.edu

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

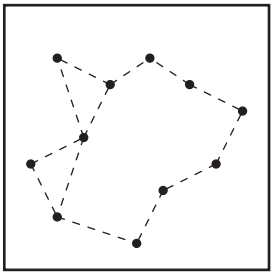


FIGURE 1: Random points: every point is connected to its 2 nearest neighbors.

leads to a disconnected graph and $k \geq 0.514 \log n$ leads to a connected graph with probabilities going to 1 as $n \to \infty$. See Figure 1. We refer to [3], [4], [10], [40], and [45] for other recent developments.

We contrast this problem with one that is encountered on a daily basis in applications.

> Suppose n points are randomly sampled from a set with some geometric structure (say, a submanifold in high dimensions). How should one create edges between these vertices to best reflect the underlying geometric structure?

This is an absolutely fundamental problem in data science. Information in data science is usually represented as points in high dimensions and for many applications one creates an *affinity matrix* that may be considered an estimate of how 'close' two elements in the data set are. Equivalently, this corresponds to building a weighted graph with data points as vertices. Taking the k nearest neighbors is a standard practice in the field (see e.g. [6], [12], and [32]) and will undoubtedly preserve locality. Points are only connected to nearby points and this gives rise to graphs that reflect the overall structure of the underlying geometry. The main point of our paper is that this approach, while correct at the local geometric perspective, is often not optimal for how it is used in applications. We first discuss the main results from a purely mathematical perspective and then explain what this implies for applications.

Notations. The notation $A \lesssim B$ denotes that $A \leq cB$ for a universal constant $c > 0$ that does not depend on A or B, and $A \lesssim_{c,d,\ldots} B$ to denote that this implicit constant may depend on c, d, ... The symbol $A \gtrsim B$ is used in the same sense; if both $A \lesssim B$ and $A \gtrsim B$, then we say that $A \sim B$ (with the same convention for dependences on other parameters).

1.2. k nearest neighbors

We now explore what happens if k is fixed and $n \to \infty$. More precisely, the question being treated in this section is as follows.

> Suppose n points are randomly sampled from a nice (compactly supported, absolutely continuous) probability distribution and every point is connected to its k nearest neighbors. What can be said about the number of connected components as $n \to \infty$? How do these connected components behave?

The results cited above already imply that the arising graph is disconnected with very high likelihood. We aim to answer these questions in more detail. By a standard reduction (a consequence of everything being local: see e.g. Beardwood, Halton and Hammersley [5]), it suffices to study the case of uniformly distributed points on the d-dimensional unit interval, i.e. $[0, 1]^d$, where $d \in \mathbb{N}$.

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

TABLE 1. Monte Carlo estimates for the value of $c_{d,k}$ (e.g. $k = 2$ nearest neighbors in $[0, 1]^2$ yield roughly $\sim 0.049n$ clusters). Larger values are difficult to obtain via sampling because $c_{d,k}$ decays very rapidly

$k \setminus d$	2	3	4
2	0.049	0.013	0.0061
3	0.0021	0.00032	0.000089
4	0.00011	0.0000089	0.0000014

The following theorem is a direct consequence of a theorem of Penrose and Yukich (Theorem 2.4 of [28]).

Theorem 1.1. (There are many connected components.) *Let $k \in \mathbb{N}$ be fixed and let X_n denote the number of connected components of a graph obtained from connecting n points sampled uniformly from $[0, 1]^d$ to their k nearest neighbors. There exists a constant $c_{d,k} > 0$ such that*

$$\lim_{n\to\infty} \frac{\mathbb{E}X_n}{n} = c_{d,k}.$$

In terms of number of clusters, this is the worst possible behavior: the number of clusters is comparable to the number of points. The reason why this problem is not an issue in applications is that the implicit constant $c_{d,k}$ decays quickly in both parameters (see Table 1).

We emphasize that this is a statement about the typical clusters and there are usually clusters that have very large diameter. This is also what turns k-nearest-neighbor graphs into a valuable tool in practice: one usually obtains a giant connected component. Results in this direction were established by Balister and Bollobás [1] and Teng and Yao [38]: more precisely, in two dimensions the 11-nearest-neighbor graph percolates (it is believed that 11 can be replaced by 3: see [1]). More precisely, our statements are very clearly located in the setting where k is fixed and $n \to \infty$.

1.3. Randomized near neighbors

Summarizing the previous sections, it is clear that if we are given n points uniformly distributed in $[0, 1]^d$, then the associated k-nearest-neighbor graph will have $\sim n$ connected components for k fixed (as $n \to \infty$) and will be connected with high likelihood as soon as $k \gtrsim \log n$. The main contribution of our paper is to show that there is a much sparser random construction that has better connectivity properties. This is of intrinsic interest but also has a number of remarkable applications in practice (and, indeed, was inspired by those).

Theorem 1.2. *There exist constants $c_{d,1}, c_{d,2} > 0$, depending only on the dimension, such that if we connect every one of n points, i.i.d. uniformly sampled from $[0, 1]^d$,*

$$\textit{to each of its } c_{d,1} \log n \textit{ nearest neighbors with likelihood } p = \frac{c_{d,2} \log\log n}{\log n},$$

then the arising graph has a connected component of size $n - o(n)$ with high probability.

(1) This allows us to create graphs on $\sim n \log\log n$ edges that have one large connected component of size proportional to $\sim n$ with high likelihood.

(2) While the difference between $\log n$ and $\log\log n$ may seem negligible for any practical applications, there is a sizeable improvement in the explicit constant that can have a big

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

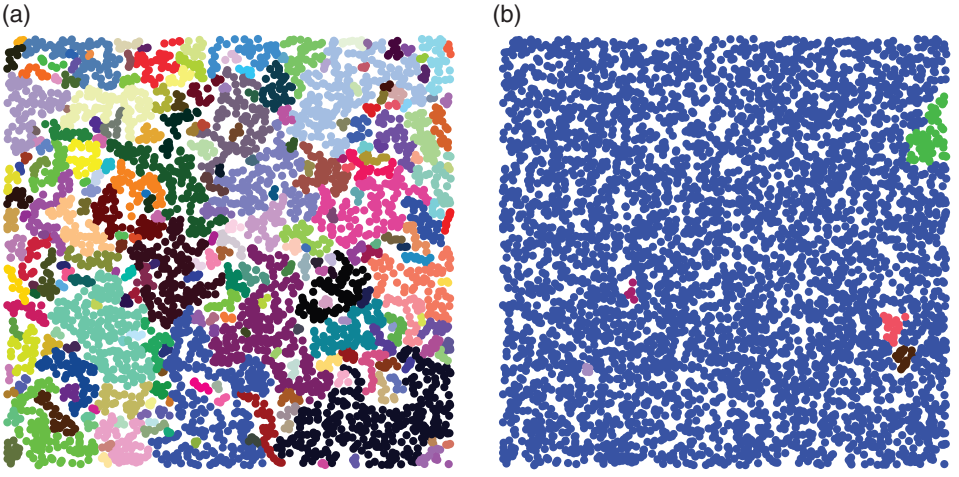


FIGURE 2: Theorems 1.1 and 1.2 illustrated: 5000 uniformly distributed random points are each connected to their two nearest neighbors (a) and two out of the four nearest neighbors, randomly selected (b). Connected components are distinguished by color; we observe a giant component in (b).

effect (we refer to Section 2.3 for numerical examples). This suggests that the constants $c_{d,1}$ and $c_{d,2}$ need not be very large: it could be of quite some interest to obtain both sharp estimates as well as effective results for n sufficiently small compared to d.

(3) In practical applications the constants scale favorably and the graphs are connected (in practice, even large n are too small for asymptotic effects).

See Figure 2 for an illustration of Theorems 1.1 and 1.2.

Related results. There are two classical ways of building a nearest-neighbor network [43]: (1) selecting the k nearest neighbors or (2) selecting all points that are at most a distance $r > 0$ away from a given point. Our Theorem 1.2 deals with (1); case (2) has been studied by Broutin, Devroye, Fraiman, and Lugosi [7]. We observe that these problems are structurally somewhat different but both are of obvious relevance in applications.

1.4. The big picture

We believe this result to have many applications in data science domains that involve the construction of a k-nearest-neighbor graph. For example, Maier, von Luxburg, and Hein [21, 22] address the question of how the choice of parameters for the k-nearest-neighbor graph influences the output of the spectral clustering algorithm. Naturally, in light of the results cited above, the parameter k must grow at least like $k \gtrsim \log n$ for such results to become applicable. Our approach allows for a much sparser random graph with comparable connectivity and several other useful properties.

However, we believe that there is also a much larger secondary effect that should be extremely interesting to study: suppose we are given a set of points $\{x_1, \ldots, x_n\} \subset [0, 1]^2$. If these points are well-separated then the k-nearest-neighbor graph is an accurate representation of the underlying structure; however, even very slight inhomogeneities in the data, some points landing in localized clusters, can have massive repercussions throughout the network (Figure 3). Even a slight degree of clustering will produce strongly localized clusters in the

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

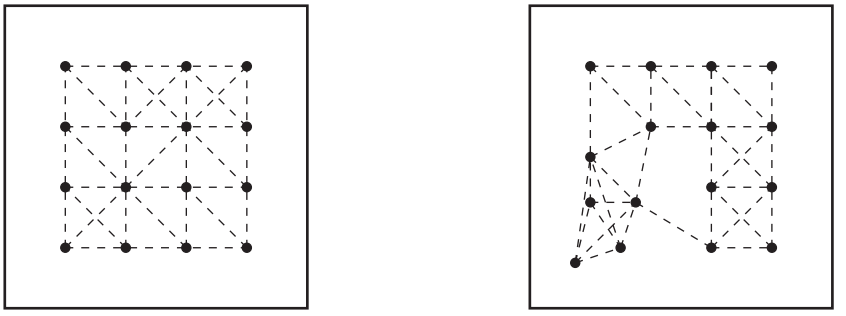


FIGURE 3: Structured points: every point is connected to its four nearest neighbors (some edges lie on top of each other); a slight perturbation of the points immediately creates a cluster in the 4-nearest-neighbor graph.

k-nearest-neighbor graph: the required degree of clustering is so slight that even randomly chosen points will be subjected to it (and this is one possible interpretation of Theorem 1.1).

Smoothing at logarithmic scales. It is easy to see that for points in $[0, 1]^d$ generated by a Poisson process with intensity n, local clustering is happening at spatial scale $\sim (c \log(n)/n)^{1/d}$. The number of points contained in a ball B with volume $|B| \sim c \log(n)/n$ is given by a Poisson distribution with parameter $\lambda \sim c \log n$, and satisfies

$$\mathbb{P}(B \text{ contains less than } \ell \text{ points}) \sim \frac{(c \log n)^{\ell}}{\ell!} \frac{1}{n^c} \lesssim \frac{1}{n^{c-\varepsilon}}.$$

This likelihood is actually quite small, which means that it is quite unlikely to find isolated clusters at that scale. In particular, an algorithm as proposed in Theorem 1.2 that picks random elements at that scale will then destroy the nonlinear concentration effect in the k-nearest-neighbor construction induced by local irregularities of uniform distribution. We believe this to be a general principle that should have many applications.

> When constructing a graph based on inhomogeneous data, it can be useful to connect each point to a random k-size subset of its K nearest neighbors, where $k \ll K$. Here, K should be chosen at such a scale that localized clustering effects disappear.

An important assumption here is that clusters at local scales are not intrinsic but induced by unavoidable sampling irregularities. We also emphasize that we believe the best way to implement this principle in practice and in applications to be very interesting and far from resolved.

We note that our theorem is closely related but does not exactly correspond to this setting. In our theorem, each node, *on average*, is connected to $k \sim \log\log(n)$ neighbors, whereas in this setting each node is connected to *exactly* k neighbors. In particular, there are no isolated nodes, and hence it likely has improved connectivity over the graph that we study in Theorem 1.2. We state the following open problem, which is related to Problem 38 of Stanisław Ulam in *The Scottish Book* [24].

> **Question.** If we are given n randomly chosen points in $[0, 1]^d$ and connect each vertex to exactly $c_1 \log\log n$ of its $c_2 \log n$ nearest neighbors, is the arising graph connected with high probability?

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

2. Applications

2.1. Implications for spectral clustering

The first step in spectral clustering of a set of points $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ into p clusters involves computation of a pairwise affinity kernel $w(x_i, x_j)$, resulting in an $n \times n$ matrix W. The kernel is commonly chosen to be a Gaussian with bandwidth σ:

$$w(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right).$$

W defines a graph with n nodes where the weight of the edge between x_i and x_j is $w(x_i, x_j)$. The normalized symmetric graph Laplacian [43] L_{sym} is defined as

$$L_{\text{sym}} = I - D^{-1/2} W D^{-1/2},$$

where D is a diagonal matrix with the sum of each row on the diagonal. The eigenvectors $\{v_1, \ldots, v_p\}$ corresponding to the p smallest eigenvalues of L_{sym} are then calculated and concatenated into the $n \times p$ matrix V. The rows of V are normalized to have unit norm and are then clustered into p clusters using the k-means algorithm. Crucially, the multiplicity of eigenvalue 0 of L_{sym} equals the number of connected components of the graph defined by L_{sym}, and the eigenvectors corresponding to eigenvalue 0 span a p-dimensional subspace that contains the vectors $\{D^{1/2}\mathbb{1}_j\}_{j=1}^{p}$, where $\mathbb{1}_j$ is piecewise constant on the jth connected component. In the case of p well-separated clusters, each connected component corresponds to a cluster, and the first p eigenvectors contain the information necessary to separate the clusters. In practice, eigenvalues corresponding to distinct clusters in the datasets will have eigenvalues *close to* zero, and even this happens only in the well-separated case.

The computational complexity and memory storage requirements of computing a full affinity matrix W scale quadratically with n, typically making computation intractable when n exceeds tens of thousands of points. Notably, as the distance between any two points x and y increases, $w(x, y)$ decays exponentially. Therefore W can be approximated by a sparse matrix W', where $W'_{ij} = w(x_i, x_j)$ if x_j is among x_i's k nearest neighbors or x_i is among x_j's nearest neighbors, and 0 otherwise.

Fast algorithms have been developed to find approximate nearest neighbors (e.g. [14]), allowing for efficient computation of W', and Lanczos methods can be used to efficiently compute its eigenvectors. When the number of neighbors, k, is chosen sufficiently large, W' is a sufficiently accurate approximation of W for most applications. However, when k cannot be chosen large enough (e.g. due to memory limitations when n is of the order of millions of points), the connectivity of the k-nearest-neighbor graph represented by W' can be highly sensitive to noise, leading W' to overfit the data and poorly model the overall structure. In the extreme case, W' can lead to a large number of disconnected components within each cluster, such that the smallest eigenvectors correspond to each of these disconnected components and not the true structure of the data.

On the other hand, choosing a random k-sized subset of K nearest neighbors, for $K > k$, results in a graph with the same number of edges which is much more likely to be connected within each cluster and hence allow for spectral clustering (Figure 4). This strategy is a more effective 'allocation' of the k edges, in the resource-limited setting. A naïve implementation is to calculate the K nearest neighbors of every point, and then select a random subset of k neighbors to connect to. This does not alleviate the nearest-neighbor search time, but has reduced memory requirements. In the following, we refer to this approach as the 'random near neighbors' or 'near' method.

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

Algorithm 1: Partition method for finding near neighbors

Input: Dataset $A = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, non-negative integers k, K with $k < K$
Output: Matrix M of size $n \times k$ where $M_{i1}, \ldots, M_{ik}$ are the indices of a random subset of $\sim K$ nearest neighbors of $x_i \in A$.

1 Let $\{B_j\}$ be a random partition of A into K/k subsets of size $m = kn/K$
2 For each $x_i \in A$, pick at random one of the subsets B_j and find the k nearest neighbors of x_i within the chosen subset.

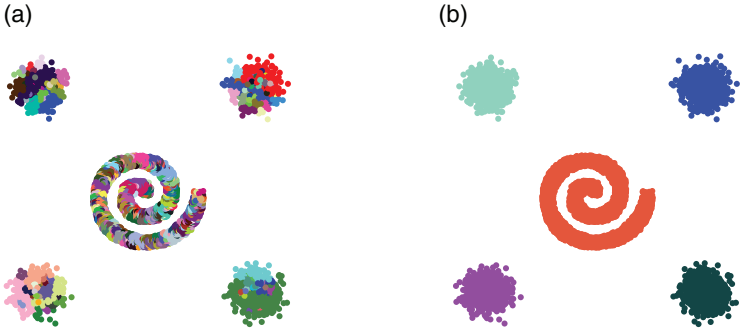


FIGURE 4: 16 000 points arranged in four clusters and a spiral. We compare connecting every point to its two nearest neighbors (a) and connecting every point to two randomly chosen out of its seven nearest neighbors (b). Connected components are colored: graph (a) has ~ 700 connected components; graph (b) consists of the actual five clusters.

2.2. Potential implementation

In order to apply this method to large datasets on resource-limited machines, an efficient algorithm for finding a k-sized subset for the K nearest neighbors of each point is needed. The naïve implementation of the near method is prohibitive for large datasets. Given a dataset so large that K nearest neighbors cannot be computed, how can we find k-sized random subsets of the K nearest neighbors for each point? Interestingly, this corresponds to an 'inaccurate' nearest-neighbor algorithm, in that the 'near' neighbors of each point are sought, not the 'nearest'. From this perspective, it seems an easier problem than that of finding the nearest neighbors. We suggest a simple and fast implementation which we have found to be empirically successful (Algorithm 1). Our approach is probabilistic: first we perform a random partition of the data into K/k subsets of equal size. For simplicity we assume that the number of data points n is a multiple of K and that K is a multiple of k. Then, for each point x_i we search for the k nearest neighbors among one of the preallocated subsets. Thus, on average k points from the K nearest neighbors in A will be among the subset samples. A k-nearest-neighbor search will therefore yield k out of the $\sim K$ nearest neighbors. We emphasize that the resulting graph is simply an approximation to the graphs described in Theorems 1.1 and 1.2, and hence these results do not necessarily apply directly.

We use K/k preallocated subsets since this enables us to use special data structures to enhance the searching speed. For example, our MATLAB implementation constructs a Kd-tree on each partition and searches for all the neighbors of all points assigned to that partition simultaneously. As in [16], where a k-nearest-neighbor classifier is constructed using a subset

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

TABLE 2. Clustering quality metrics (NMI and ACC) for spectral clustering of four digits 0, 3, 6, 7 of InfiMNIST (∼3.27 million points) using a k-nearest-neighbor graph compared to using the near-neighbor and partition approaches. The subspace angle between leading eigenvectors of each construction and the leading eigenvectors of nearest-100 is also computed

	nearest-k				near-k/K	partition-k/K
k	10	20	50	100	2/100	2/100
subspace angle	1.564	1.501	1.571	0	0.050	0.063
NMI	0.386	0.823	0.964	0.971	0.967	0.962
ACC	0.462	0.744	0.991	0.994	0.993	0.992

of the points, we find that searching a much smaller number of points leads to a speed-up of several orders of magnitude (Tables 4 and 5).

2.3. Numerical results

As proof of concept, we applied spectral clustering to two large datasets and compared the quality of the embedding and computational times between the traditional nearest-neighbor approach (denoted nearest-k), random near neighbors (denoted near-k/K), and the partition method described in Algorithm 1 (denoted partition-k/K). For all three approaches the graph edges are weighted using the Gaussian kernel from above with an adaptive bandwidth σ_i equal to the point's squared distance to its kth neighbor. The first dataset is InfiMNIST [20], where we chose digits 0, 3, 6, 7, resulting in a dataset of $n \sim 3\,271\,995$ in a $d = 784$-dimensional space. The second dataset is Reuters [17], composed of 645 564 articles, across four root categories in a $d = 2000$-dimensional space. Note that while InfiMNIST is a balanced dataset, in Reuters the largest cluster is twice as large as two of the other clusters, and six times larger than the smallest cluster. The full details of the datasets and their preprocessing appear in Appendix A.

We used the Lanczos iterations as implemented in MATLAB's 'eigs' function to compute the first four eigenvectors of the Laplacians for InfiMNIST and first 20 for Reuters. For Reuters, we used more eigenvectors than the number of expected clusters, as some of the categories are composed of multiple smaller clusters, requiring more eigenvectors to reveal the underlying structure of the data.

For the InfiMNIST dataset, we plot the embedding of the datapoints computed with the first three eigenvectors of nearest-10, nearest-50 and nearest-100 in Figure 5(a–c) and near-2/100 and partition-2/100 in Figure 5(d, e). The first three eigenvectors of nearest-10 do not separate the digits, and for nearest-50 the embedding collapses to a curve with multiple outliers, providing a poor visualization of the data, that is, neither reveals the underlying manifold on which the digits lie. Increasing the number of nearest neighbors in nearest-100 provides a meaningful embedding. Remarkably, the same quality embedding can be obtained with near-2/100 and partition-2/100, despite it being a much sparser graph.

To quantify our results, we perform K-means on the eigenvectors and compare the resulting clusters to the true labels. To assess the quality of the clusters we use two different popular measures: normalized mutual information (NMI) and unsupervised clustering accuracy (ACC). The definition and details of these measures can be found in the Appendix A. Results are presented in Tables 2 and 3 (more detailed results across a wider range of k values for the near and partition approaches appear in Appendix A). We note that results for Reuters are comparable to those achieved by SpectralNet in [31], where a deep neural network is trained to map datapoints to the corresponding eigenspace of their associated graph Laplacian matrix.

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

TABLE 3. Quality metrics as in Table 2 for spectral clustering of Reuters ($\sim 650\,000$ points)

	nearest-k				near-k/K	partition-k/K
k	25	50	100	1000	5/1000	5/1000
subspace angle	1.57	1.57	1.57	0.00	0.20	1.48
NMI	0.048	0.129	0.158	0.422	0.418	0.407
ACC	0.420	0.371	0.417	0.637	0.638	0.671

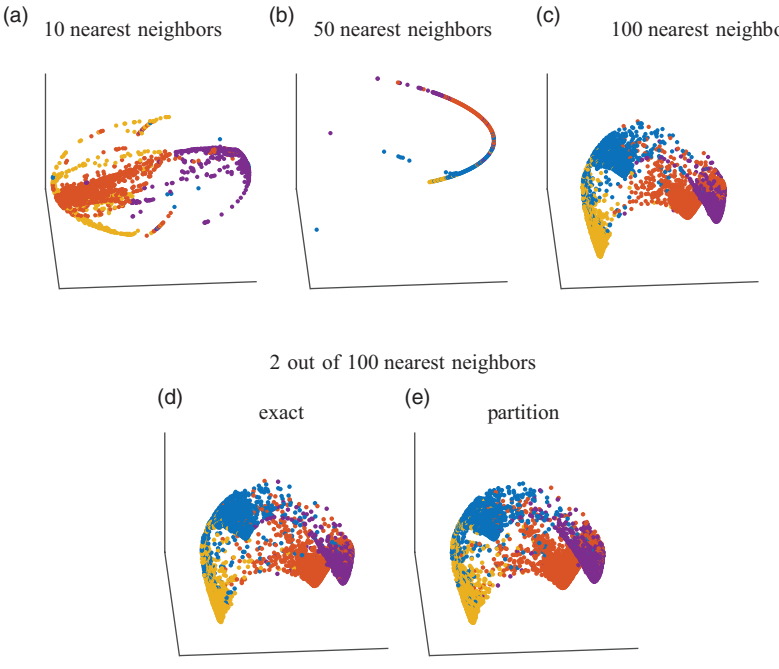


FIGURE 5: Datapoints embedding of four digits from the InfiMNIST data set computed by the sparse graph Laplacian leading eigenvectors. Connecting to k nearest neighbors when k is too small leads to catastrophic results (a, b), while a successful embedding is obtained for $k = 100$ (c). Connecting to two randomly chosen out of the 100 nearest neighbors gives a comparable embedding (d), while using far fewer edges. The partition method (Algorithm 1) also produces an embedding of equivalent quality (e).

We also calculate the subspace angle (principal angle) between the embedding for nearest-K versus near-k/K and partition-k/K embeddings for increasing k, where we consider the embedding obtained from nearest-K to be the 'ground-truth' embedding. For InfiMNIST, the angle between the 'ground-truth' embedding and the embeddings of both near-neighbor methods is very small, while for nearest-k the angle is practically maximal. For Reuters, the subspace angles near-k/K are small and decrease as k increases (see Appendix A). For partition-k/K, the angle is large but drops around $k = 50$ and decreases further as k increases.

To compute these spectral embeddings there are two computationally intensive steps: near(est)-neighbor search and computation of the leading eigenvectors of the Laplacian (Tables 4 and 5). The partition method described in Algorithm 1 is 1–2 orders of magnitude

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

TABLE 4. Time taken by neighbor search and computation of leading eigenvectors for the InfiMNIST dataset (∼3.27 million points)

	nearest				near	partition
k	10	20	50	100	2/100	2/100
search time (sec)	2478	4645	7565	10903	–	338
eigs time (sec)	2472	5836	7572	3049	254	351

TABLE 5. Run-time as in Table 4 for the Reuters dataset (∼ 650 000 points)

	nearest				near	partition
k	25	50	100	1000	5/1000	5/1000
search time (sec)	835	942	1124	2084	–	15
eigs time (sec)	644	429	519	2137	42	38

faster than searching for the number of nearest neighbors necessary to obtain the above results. For the second step, the time taken to compute the eigenvectors is mainly determined by two factors: (i) the sparseness of the Laplacian matrix, which determines the complexity of a single iteration, and (ii) the spectrum of the Laplacian matrix, which determines the rate of convergence. The near-neighbor approach has an obvious advantage over the nearest-neighbor approach in terms of sparsity of the Laplacian. In terms of its spectral decay, however, the comparison depends on the values of k and K. If K is insufficient for a good embedding (i.e $K = 20$ for InfiMNIST) the convergence is extremely slow. On the other hand, if K is sufficient, then the convergence rate improves dramatically and thus might require fewer iterations than the k-near-neighbor approach. For InfiMNIST, the partition method took only 5 minutes for the near-neighbor search, and computing the first three eigenvectors took 6 minutes (in contrast to many hours for the nearest-neighbor approach). Similar speed-ups can be seen with the Reuters dataset, where both the near-neighbor search and eigenvector computation took less than a minute in total (in contrast to longer than an hour using the nearest-neighbor approach).

2.4. Further outlook

We demonstrate the application of our approach in the context of spectral clustering but this is only one example. There are a great many other methods of dimensionality reduction that start by constructing a graph that roughly approximates the data, for example t-distributed stochastic neighborhood embedding (t-SNE) [19, 42], diffusion maps [8], or Laplacian eigenmaps [6]. This refinement could be valuable for a very wide range of algorithms that construct graph approximations out of underlying point sets. Determining the precise conditions under which this method is effective for which algorithm will strongly depend on the context, but needless to say, we consider the experiments shown in this section to be extremely encouraging. We believe this paper suggests many possible directions for future research. Are there other natural randomized near-neighbor constructions? Other questions include the behavior

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

of the spectrum and the induced random walk. Here we would like to point out briefly that random graphs are natural expanders [15, 23, 29]. This should imply several additional degrees of stability that the standard k-nearest-neighbor construction does not have.

3. Proof of Theorem 1.2

3.1. The Erdős–Rényi lemma

Before embarking on the proof, we give a short description. The proof is not subtle and follows along completely classical lines but occurs in an unfamiliar regime: we are interested in ensuring that the likelihood of obtaining a disconnected graph is very small. The subsequent argument, which is not new but not easy to immediately spot in the literature, is included for the convenience of the reader (much stronger and more subtle results usually focus on the threshold $p = (1 \pm \varepsilon)(\log n)/n$).

Lemma 3.1. *Let G(n,p) be an Erdős–Rényi graph with $p > 10 \log n/n$. Then, for n sufficiently large,*

$$\mathbb{P}(G(n,p) \text{ is disconnected}) \lesssim \mathrm{e}^{-pn/3}.$$

Proof. It suffices to bound the likelihood of finding an isolated set of k vertices from above, where $1 \le k \le n/2$. For any fixed set of k vertices, the probability of being isolated is bounded from above by

$$\mathbb{P}(\text{fixed set of } k \text{ vertices being disconnected}) \le (1-p)^{k(n-k)}$$

and thus, using the union bound,

$$\mathbb{P}(G(n,p) \text{ is not connected}) \le \sum_{k=1}^{n/2} \binom{n}{k} (1-p)^{k(n-k)}.$$

We use

$$\binom{n}{k} \le \left(\frac{ne}{k}\right)^k$$

to rewrite the expression as

$$\begin{aligned}
\sum_{k=1}^{n/2} \binom{n}{k} (1-p)^{k(n-k)} &\le \sum_{k=1}^{n/2} \mathrm{e}^{k+k\log n + k\log k + [\log(1-p)]k(n-k)} \\
&\le \sum_{k=1}^{n/2} \mathrm{e}^{k(3\log n + [\log(1-p)](n-k))} \\
&\le \sum_{k=1}^{n/2} \mathrm{e}^{k(3\log n + [\log(1-p)](n/2))} \\
&\lesssim \mathrm{e}^{3\log n + [\log(1-p)]n/2},
\end{aligned}$$

where the last step is merely the summation of a geometric series and is valid as soon as

$$3\log n + [\log(1-p)]\frac{n}{2} < 0,$$

which is eventually true for n sufficiently large since $p > 10 \log n/n$. □

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

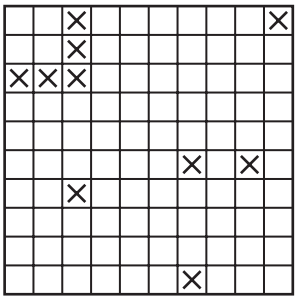


FIGURE 6: Removing tiny squares randomly: this random sample ends up removing a bit more from the giant component but is quite unlikely.

3.2. A mini-percolation lemma

The purpose of this section is to derive rough bounds for a percolation-type problem. These problems have been of great interest in mathematical physics and probability theory (see e.g. Penrose and Pisztora [26]). We provide a simple self-contained argument that, while not being sharp, results in a simple estimate sufficient for our purpose.

Lemma 3.2. *Suppose we are given a grid graph on* $\{1, 2, \ldots, n\}^d$ *for* $d \geq 2$ *and remove each of the* n^d *points with likelihood* $p = (\log n)^{-c}$ *for some* $c > 0$*. Then, for* n *sufficiently large, there is a giant component with expected size* $n^d - o(n^d)$*.*

Proof of Lemma 3.2. The proof is actually fairly lossy and proceeds by massive overcounting. The only way to remove mass from the giant block is to remove points in an organized manner: adjacent squares have to be removed in a way that encloses a number of squares that are not removed (see Figure 6).

The next question is how many other points can possibly be captured by a connected component on ℓ-nodes. The isoperimetric principle implies

$$\#\text{ blocks captured by } \ell \text{ nodes} \lesssim_d \ell^{d/(d-1)} \leq \ell^2.$$

Altogether, this implies that we expect to capture at most

$$\sum_{\ell=1}^{n^d} n^d (2^{3^d-1})^\ell (\log n)^{-c\ell} \ell^2 \leq n^d \sum_{\ell=1}^{\infty} \left(\frac{2^{3^d-1}}{(\log n)^c} \right)^\ell \ell^2 \lesssim \frac{n^d}{(\log n)^c},$$

where the last inequality holds as soon as $\log n^c \gg 2^{3^d-1}$ and follows from the derivative geometric series

$$\sum_{\ell=1}^{\infty} \ell^2 q^\ell = \frac{q(1+q)}{(1-q)^3} \quad \text{whenever } |q| < 1.$$

□

Remark. There are two spots where the argument is fairly lossy. First of all, every connected component on ℓ nodes is, generically, counted as ℓ connected components of length $\ell - 1$, as $\sim \ell^2$ connected components of size $\ell - 2$, and so on. The second part of the argument is the application of the isoperimetric inequality: a generic connected component on ℓ nodes will capture $\ll \ell^2$ other nodes. These problems seem incredibly close to existing research, and it

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

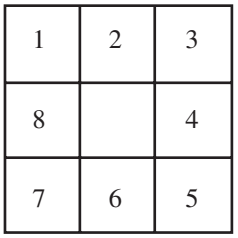


FIGURE 7: Fixing labels for the eight immediate neighbors.

seems likely that either they have been answered already or that techniques from percolation theory might provide rather immediate improvements.

We now show a separate result which we will use later. We call a subset $A \subset \{1, 2, \ldots, n\}^d$ *connected* if the resulting graph is connected: here, edges are given by connecting every node to all of its adjacent nodes that differ by at most one in each coordinate (that number is bounded from above by $3^d - 1$).

Lemma 3.3. *The number of connected components A in the grid graph over* $\{1, 2, \ldots, n\}^d$ *with cardinality* $|A| = \ell$ *is bounded from above:*

$$\#\ \textit{connected components of size}\ \ell \leq n^d \left(2^{3^d - 1}\right)^\ell.$$

Proof. The proof proceeds in a fairly standard way by constructing a combinatorial encoding. We show how this is done in two dimensions, giving an upper bound of $n^2 256^\ell$; the construction immediately transfers to higher dimensions in the obvious way. The encoding is given by a direct algorithm. See Figure 7.

(1) Pick an initial vertex $x_0 \in A$. Describe which of the eight adjacent squares are occupied by picking a subset of $\{1, 2, \ldots, 8\}$.

(2) Implement a depth-first search as follows: pick the smallest number in the set attached to x_0 and describe its neighbors, if any, that are distinct from previously selected nodes as a subset of $\{1, 2, \ldots, 8\}$.

(3) Repeat until all existing neighbors have been mapped out (the attached set is the empty set) and then go back and describe the next branch.

Just for clarification, we quickly show the algorithm in practice. Suppose we are given an initial point x_0 and the sequence of sets

$$\{4, 5\}, \{3, 4\}, \{\}, \{\}, \{5\}, \{4\}, \{\}.$$

Then this uniquely identifies the set shown in Figure 8.

Clearly, this description returns ℓ subsets of $\{1, \ldots, 8\}$, of which there are 256. Every element in A generates exactly one such subset and every connected component can thus be described by giving the n^d initial points and then a list of ℓ subsets of $\{1, \ldots, 8\}$. This implies the desired statement; we note that the actual number should be much smaller since this way of describing connected components has massive amounts of redundancy and overcounting. □

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

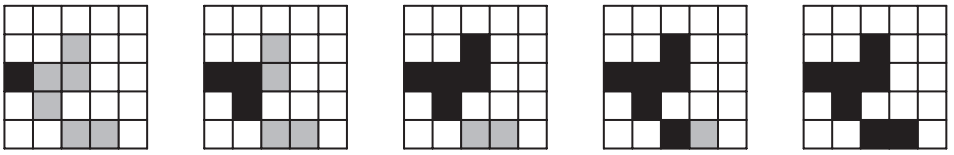


FIGURE 8: The starting point followed by generating the connected component described by the sequence of sets {4, 5}, {3, 4}, {}, {}, {5}, {4}, {}.

3.3. Outline of the proof

The proof proceeds in three steps.

(1) Partition the unit cube into smaller cubes such that each small cube has an expected number of $\sim \log n$ points (and thus the number of cubes is $\sim n/\log n$). Show that the likelihood of a single cube containing significantly more or significantly fewer points is small.

(2) Show that graphs within the cube are connected with high probability.

(3) Show that there are connections between the cubes that ensure connectivity.

3.4. Step 1

We start by partitioning $[0, 1]^d$ in the canonical manner into axis-parallel cubes having side-length $\sim (c \log n/n)^{1/d}$ for some constant c to be chosen later. There are roughly $\sim n/(c \log n)$ cubes and they have measure $\sim c \log (n)/n$. We start by bounding the likelihood of one such cube containing $\leq \log n/100$ points. Clearly, this likelihood can be written as a Bernoulli random variable:

$$\text{number of points in cube} = \mathcal{B}\left(n, \frac{c \log n}{n}\right).$$

The Chernoff–Hoeffding theorem [13] implies

$$\mathbb{P}\left(\mathcal{B}\left(n, \frac{c \log n}{n}\right) \leq \frac{\log n}{100}\right) \leq \exp\left(-nD\left(\frac{\log n}{100n} || \frac{c \log n}{n}\right)\right),$$

where D is the relative entropy

$$D(a||b) = a \log \frac{a}{b} + (1-a) \log \frac{1-a}{1-b}.$$

Here, we have, for n large,

$$D\left(\frac{\log n}{100n} || \frac{c \log n}{n}\right) \sim \frac{\log n}{n}\left(c - \frac{1}{100} + \frac{1}{100} \log \frac{1}{100c}\right).$$

This implies that, for c sufficiently large, we have

$$\mathbb{P}(\text{a fixed cube has fewer than } \log n/100 \text{ points}) \lesssim_{c,\varepsilon} \frac{1}{n^{c-\varepsilon}}$$

and the union bound implies

$$\mathbb{P}(\text{there exists a cube with fewer than } \log n/100 \text{ points}) \lesssim_{c,\varepsilon} \frac{1}{n^{c-1-\varepsilon}}.$$

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

The same argument also shows that

$$\mathbb{P}(\text{there exists a cube with more than } 10c \log n \text{ points}) \lesssim \frac{1}{n^c}.$$

This means we have established the existence of a constant c such that, with likelihood tending to 1 as $n \to \infty$ (at arbitrary inverse polynomial speed provided c is big enough), for all cubes Q,

$$\frac{\log n}{100} \leq \#\{\text{points in } Q\} \leq 10c \log n.$$

Henceforth we only deal with cases where these inequalities are satisfied for all cubes.

3.5. Step 2

We now study what happens within a fixed cube Q. The cube is surrounded by at most $3^d - 1$ other cubes, each of which contains at most $10c \log n$ points. This means that if, for any $x \in Q$, we compile a list of its $3^d 10c \log n$ nearest neighbors, we are guaranteed that every other element in Q is on that list. Let us suppose that the rule is that each point is connected to each of its $3^d 10c \log n$ nearest neighbors with likelihood

$$p = \frac{m}{3^d 10c \log n}.$$

Then Lemma 3.1 implies that, for $m \gtrsim 10 \log (3^d 10c \log (n)) \sim_{d,c} \log \log n$, the likelihood of obtaining a connected graph strictly within Q is at least $(\log n)^{-c}$. Lemma 3.2 then implies the result provided we can ensure that points in cubes connect to their neighboring cubes.

3.6. Step 3

We now establish that the likelihood of a cube Q having, for every adjacent cube R, a point that connects to a point in R is large. The adjacent cube has $\sim \log n$ points. The likelihood of a fixed point in Q not connecting to any point in R is

$$\leq \left(1 - \frac{\log n/100}{3^d 10c \log n}\right)^{c \log \log n} = \left(1 - \frac{1}{3^d 1000c}\right)^{c \log \log n} \lesssim (\log n)^{-\varepsilon_{c,d}}.$$

The likelihood that this is indeed true for every point is then bounded from above by

$$(\log n)^{-\varepsilon_{c,d} \log n/100} \lesssim n^{-1},$$

which means, appealing again to the union bound, that this event occurs with a likelihood going to 0 as $n \to \infty$. □

Connectedness. It is not difficult to see that this graph is unlikely to be connected. For a fixed vertex v, there are $\sim c \log n$ possible other vertices it could connect to and $\sim c \log n$ other vertices might possibly connect to v. Thus

$$\mathbb{P}(v \text{ is isolated}) \lesssim \left(1 - \frac{c_2 \log \log n}{\log n}\right)^{c_3 \log n} \leq \mathrm{e}^{-c_2 c_3 \log \log n} = \frac{1}{(\log n)^{c_2 c_3}}.$$

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

This shows that we can expect at least $n(\log n)^{-c_2c_3}$ isolated vertices. This also shows that the main obstruction to connectedness is the non-trivial likelihood of vertices not forming edges to other vertices.

Appendix A. Experimental results

A.1. Data and preprocessing

The first dataset is the MNIST8M dataset generated by InfiMNIST [20], which provides an unlimited supply of handwritten digits derived from MNIST using random translations and permutations. For simplicity of visualization, we chose digits 0, 3, 6, 7, resulting in a dataset of $n \sim 3\,271\,995$ in a $d = 784$-dimensional space. The second dataset is Reuters [17], composed of 645 564 articles, across four root categories: corporate/industrial, government/social, markets, and economics (documents with multiple root categories were removed). Each article is represented using tf-idf features on the 2000 most frequently occurring word stems, as in [44]. In addition, we removed all points that were exact duplicates. For both datasets, we reduced the dimensionality of the samples via the randomized PCA algorithm [30] implemented in [18]. All neighbor searches were done in the reduced dimension.

A.2. Clustering measures

Normalized mutual information (NMI) is defined as

$$\mathrm{NMI}(l, c) = \frac{I(l; c)}{\max\{H(l), H(c)\}},$$

where $H(c), H(l)$ denote the entropy of the clustering result c and the true labels l, respectively, and $I(l; c)$ denote the mutual information between them. Unsupervised clustering accuracy (ACC), is defined as

$$\mathrm{ACC}(l, c) = \frac{1}{n} \max_{\pi \in \Pi} \sum_{i=1}^{n} \mathbb{1}\{l_i = \pi(c_i)\},$$

where Π is the collection of all permutations of $\{1, \ldots, k\}$. The optimal permutation π can be computed using the Kuhn–Munkres algorithm [25].

A.3. Extended simulations

TABLE 6. Measures for InfiMNIST dataset

	near			partition			nearest			
k/K	2 /100	5 /100	10 /100	2 /100	5 /100	10 /100	10	20	50	100
subspace	0.050	0.025	0.016	0.063	0.031	0.020	1.564	1.501	1.571	0
NMI	0.967	0.970	0.971	0.962	0.968	0.970	0.386	0.823	0.964	0.971
ACC	0.993	0.993	0.993	0.992	0.993	0.993	0.462	0.744	0.991	0.994
eigs time	254.0	340.4	530.6	350.6	400.4	645.2	2471.7	5836.1	7572.4	3048.6
search time	–	–	–	337.6	913.6	1627.2	2478.3	4644.7	7565.5	10903.2

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

TABLE 7. Measures for Reuters dataset

	k	5	10	25	50	100	1000
subspace angle	nearest			1.57	1.57	1.57	0.00
	near	0.20	0.14	0.09	0.06	0.04	
	partition	1.48	1.47	1.57	0.53	0.23	
NMI	nearest			0.048	0.129	0.158	0.422
	near	0.418	0.420	0.421	0.421	0.422	
	partition	0.407	0.411	0.425	0.424	0.423	
ACC	nearest			0.420	0.371	0.417	0.637
	near	0.638	0.638	0.637	0.637	0.637	
	partition	0.671	0.673	0.642	0.641	0.639	
eigs time	nearest			643.9	428.9	518.8	2137.0
	near	41.8	58.4	96.2	162.3	268.8	
	partition	38.2	60.1	110.6	181.6	315.6	
search time	nearest			1524.5	1394.5	1400.2	1315.0
	partition	14.6	18.6	31.9	56.7	126.4	

Acknowledgements

This work was supported by NIH grants F30HG010102 (to GCL), R01HG008383-01A1 (to GCL and YK), R01GM131642 (to YK), R01GM135928 (to YK), P50CA121974 (to YK), R01NS100049 (to GM), NIH MSTP Training Grant T32GM007205 (to GCL).

References

[1] BALISTER, P. AND BOLLOBÁS, B. (2013). Percolation in the *k*-nearest neighbor graph. In *Recent Results in Designs and Graphs: A Tribute to Lucia Gionfriddo* (Quaderni di Matematica **28**), eds M. Buratti *et al.*, pp. 83–100. Aracne.

[2] BALISTER, P., BOLLOBÁS, B., SARKAR, A. AND WALTERS, M. (2005). Connectivity of random *k*-nearest-neighbour graphs. *Adv. Appl. Prob.* **37** (1), 1–24.

[3] BALISTER, P., BOLLOBÁS, B., SARKAR, A. AND WALTERS, M. (2009). A critical constant for the *k*-nearest-neighbour model. *Adv. Appl. Prob.* **41** (1), 1–12.

[4] BALISTER, P., BOLLOBÁS, B., SARKAR, A. AND WALTERS, M. (2009). Highly connected random geometric graphs. *Discrete Appl. Math.* **157** (2), 309–320.

[5] BEARDWOOD, J., HALTON, J. H. AND HAMMERSLEY, J. M. (1959). The shortest path through many points. *Proc. Cambridge Philos. Soc*. **55**, 299–327.

[6] BELKIN, M. AND NIYOGI, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **15**, 1373–1396.

[7] BROUTIN, N., DEVROYE, L., FRAIMAN, N. AND LUGOSI, G. (2014). Connectivity threshold of Bluetooth graphs. *Random Struct. Algorithms* **44** (1), 45–66.

[8] COIFMAN, R. AND LAFON, S. (2006). Diffusion maps. *Appl. Comput. Harmon. Anal.* **21** (1), 5–30.

[9] ERDŐS, P. AND RÉNYI, A. (1959). On random graphs I. *Publ. Math. Debrecen* **6**, 290–297.

[10] FALGAS-RAVRY, V. AND WALTERS, M. (2012). Sharpness in the *k*-nearest-neighbours random geometric graph model. *Adv. Appl. Prob.* **44** (3), 617–634.

[11] FEW, L. (1955). The shortest path and the shortest road through *n* points. *Mathematika* **2**, 141–144.

[12] HEIN, M., AUDIBERT, J.-Y. AND VON LUXBURG, U. (2005). From graphs to manifolds: weak and strong pointwise consistency of graph Laplacians. In *Learning Theory* (Lecture Notes Comput. Sci. **3559**), pp. 470–485. Springer, Berlin.

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

[13] HOEFFDING, W. (1963). Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* **58** (301), 13–30.
[14] JONES, P. W., OSIPOV, A. AND ROKHLIN, V. (2011). Randomized approximate nearest neighbors algorithm. *Proc. Nat. Acad. Sci*. **108** (38), 15679–15686.
[15] KOLMOGOROV, A. N. AND BARZDIN, Y. (1993). On the realization of networks in three-dimensional space. In *Selected Works of A. N. Kolmogorov* (Mathematics and Its Applications (Soviet Series) **27**), ed. A. N. Shiryayev, pp. 194–202. Springer, Dordrecht.
[16] KUSNER, M. J., TYREE, S., WEINBERGER, K. AND AGRAWAL, K. (2014). Stochastic neighbor compression. In *31st International Conference on Machine Learning* (Proceedings of Machine Learning Research **32**), pp. 622–630. PMLR.
[17] LEWIS, D., YANG, Y., ROSEN, T. AND LI, F. (2004). RCV1: a new benchmark collection for text categorization research. *J. Machine Learning Res.* **5**, 361–397.
[18] LI, H., LINDERMAN, G. C., SZLAM, A., STANTON, K. P., KLUGER, Y. AND TYGERT, M. (2017). Algorithm 971: an implementation of a randomized algorithm for principal component analysis. *ACM Trans. Math. Software* **43** (3), 28.
[19] LINDERMAN, G. C. AND STEINERBERGER, S. (2019). Clustering with t-SNE, provably. *SIAM J. Math Data Science* **1**, 313–332.
[20] LOOSLI, G., CANU, S. AND BOTTOU, L. (2007). Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*, eds L. Bottou *et al.*, pp. 301–320. MIT Press.
[21] MAIER, M., VON LUXBURG, U. AND HEIN, M. (2009). Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems 21 (2008)*, pp. 1025–1032.
[22] MAIER, M., VON LUXBURG, U. AND HEIN, M. (2013). How the result of graph clustering methods depends on the construction of the graph. *ESAIM Prob. Statist.* **17**, 370–418.
[23] MARGULIS, G. (1973). Explicit constructions of concentrators. *Problemy Peredachi Informatsii* **9** (4), 71–80. English version: *Problems Inform. Transmission* **10**, 325–332 (1975).
[24] MAULDIN, R. D. (ed.) (2015). *The Scottish Book: Mathematics from The Scottish Café, with Selected Problems from The New Scottish Book*, 2nd edn. Birkhäuser/Springer, Cham.
[25] MUNKRES, J. (1957). Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math.* **5** (1), 32–38.
[26] PENROSE, M. (2003). *Random Geometric Graphs* (Oxford Studies in Probability **5**). Oxford University Press, Oxford.
[27] PENROSE, M. AND PISZTORA, A. (1996). Large deviations for discrete and continuous percolation. *Adv. Appl. Prob.* **28** (1), 29–52.
[28] PENROSE, M. AND YUKICH, J. (2003). Weak laws of large numbers in geometric probability. *Ann. Appl. Prob.* **13** (1), 277–303.
[29] PINSKER, M. S. (1973). On the complexity of a concentrator. In *Seventh International Teletraffic Congress (Stockholm, 1973)*, paper # 318.
[30] ROKHLIN, V., SZLAM, A. AND TYGERT, M. (2009). A randomized algorithm for principal component analysis. *SIAM J. Matrix Anal. Appl.* **31** (3), 1100–1124.
[31] SHAHAM, U., STANTON, K., LI, H., BASRI, R., NADLER, B. AND KLUGER, Y. (2018). SpectralNet: spectral clustering using deep neural networks. In *International Conference on Learning Representations, 2018*.
[32] SINGER, A. (2006). From graph to manifold Laplacian: the convergence rate. *Appl. Comput. Harmon. Anal.* **21** (1), 128–134.
[33] STEELE, J. M. (1980). Shortest paths through pseudorandom points in the d-cube. *Proc. Amer. Math. Soc*. **80** (1), 130–134.
[34] STEELE, J. M. (1981). Subadditive Euclidean functionals and nonlinear growth in geometric probability. *Ann. Prob.* **9** (3), 365–376.
[35] STEELE, J. M. (1997). *Probability Theory and Combinatorial Optimization* (CBMS-NSF Regional Conference Series in Applied Mathematics **69**). SIAM, Philadelphia.
[36] STEINERBERGER, S. (2010). A new lower bound for the geometric traveling salesman problem in terms of discrepancy. *Oper. Res. Lett.* **38** (4), 318–319.
[37] STEINERBERGER, S. (2015). New bounds for the traveling salesman constant. *Adv. Appl. Prob.* **47**, 27–36
[38] TENG, S.-H. AND YAO, F. (2007). k-nearest-neighbor clustering and percolation theory. *Algorithmica* **49** (3), 192–211.
[39] WALTERS, M. (2011). Random geometric graphs. *Surveys Combin.* **392**, 365–402.
[40] WALTERS, M. (2012). Small components in k-nearest neighbour graphs. *Discrete Appl. Math.* **160** (13–14), 2037–2047.
[41] VAN DER MAATEN, L. (2014). Accelerating t-SNE using tree-based algorithms. *J. Machine Learning Res.* **15** (1), 3221–3245.

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press

[42] VAN DER MAATEN, L. AND HINTON, G. (2008). Visualizing data using t-SNE. *J. Machine Learning Res.* **9**, 2579–2605.
[43] VON LUXBURG, U. (2007). A tutorial on spectral clustering. *Statist. Comput.* **17** (4), 395–416.
[44] XIE, J., GIRSHICK, R. AND FARHADI, A. (2016). Unsupervised deep embedding for clustering analysis. In *33rd International Conference on Machine Learning* (Proceedings of Machine Learning Research **48**), pp. 478–487. JMLR.
[45] XUE, F. AND KUMAR, P. R. (2004). The number of neighbors needed for connectivity of wireless networks. *Wireless Networks* **10**, 169–181.
[46] YUKICH, J. (1998). *Probability Theory of Classical Euclidean Optimization Problems* (Lecture Notes Math. **1675**). Springer, Berlin.

https://doi.org/10.1017/jpr.2020.21 Published online by Cambridge University Press