Supplementary Material for Computationally Efficient Learning of Statistical Manifolds

Fan Cheng [*]
Monash University
and
Rob J Hyndman
Monash University
and
Anastasios Panagiotelis
University of Sydney

December 6, 2021

This is an online supplementary material for the paper titled Computationally Efficient Learning of Statistical Manifolds. This document presents a broad range of approximate nearest neighbor algorithms within manifold learning algorithms and evaluating their impact on embedding accuracy. Via a thorough empirical investigation based on the benchmark MNIST dataset, it is shown that approximate nearest neighbors lead to substantial improvements in computational time with little to no loss in the accuracy of the embedding produced by a manifold learning algorithm. This result is robust to the use of different manifold learning algorithms, to the use of different approximate nearest neighbor algorithms, and to the use of different measures of embedding accuracy.

1 MNIST dataset

The MNIST database [Modified National Institute of Standards and Technology database; LeCun, Cortes, and Burges (2010)] is a commonly used benchmark dataset for dimension

[*] `Fan.Cheng@monash.edu`

reduction techniques, consisting of 60,000 grayscale images of handwritten digits in a training set, and 10,000 grayscale images in a test set. It was constructed from a larger database called NIST. In the original NIST dataset, images in the training data were hand-written by American Census Bureau employees (Special Database 3, SD-3), while images in the test data were handwritten by high school students (Special Database 1, SD-1). LeCun, Cortes, and Burges (2010) normalized the size of the NIST images and centered them in a $28 \times 28 = 784$ pixel field. Then they mixed the samples from both SD-3 and SD-1 to form the MNIST dataset, where the training set contains 30,000 images from SD-3 and 30,000 images from SD-1, while the test set is composed of 5,000 images from SD-3 and 5,000 images from SD-1.

To examine manifold learning using ANN techniques, we use the MNIST test set of $N = 10,000$ observations, with a dimension equal to the number of pixels ($p = 784$). The embedding dimension for manifold learning is set as $d = 2$ and the number of nearest neighbors is set as $K = 20$. We apply different combinations of manifold learning algorithms and ANN methods to the data, recording the computational time and calculating the embedding quality measures discussed in Section 2. All experiments were run in parallel on a high-performance computing cluster with 2.70GHz Xeon-Gold-6150 CPUs.

2 Quality measures for manifold learning embedding

To examine the effect of using approximate nearest neighbor algorithms in manifold learning, measures of the quality of a low-dimensional representation must be defined. In the ANN benchmark tool built by Aumüller, Bernhardsson, and Faithfull (2020), recall rate is used to measure the accuracy of the approximate nearest neighbor searching methods. Recall rate is defined as the proportion of observations correctly classified as nearest neighbors compared to the true ones. However, even with a large proportion of true nearest neighbors, the local structure in the manifold learning embeddings might still be inaccurate. Therefore, the quality measures of the embeddings are essential.

There are a number of criteria that we consider, all of which measure the extent to which the nearest neighbor structure of the output points resembles the nearest neighbor structure of the input points. That is, all criteria measure the extent to which the topology of the manifold is preserved while the final criterion we consider (the Procrustes measure), additionally measures the extent to which the local geometry is preserved. For ease of comparison, we adjust all quality measures so that higher values of a measure indicate a higher quality embedding.

Recall that $U_K(i)$ was defined as the set of points j such that x_j is one of the K-nearest neighbors of x_i. We now also define $V_K(i)$ as the nearest neighborhood of observation i in the output space, i.e. a set of points j such that y_j is one of the K-nearest neighbors of y_i. We define the neighborhood ranking of x_j with respect to x_i as $\rho_{ij} = |\{\ell : \delta_{i\ell} < \delta_{ij}\}|$. For example, if x_j is the nearest neighbor of x_i, then $\rho_{ij} = 1$ since only $\delta_{ii} < \delta_{ij}$, where we assume without loss of generality that all input points are distinct. In case of tied distances, $\rho_{ij} = \big|\big\{\ell : \delta_{i\ell} < \delta_{ij} \text{ or } (\delta_{i\ell} = \delta_{ij} \text{ and } \ell < j)\big\}\big|$. The neighborhood rankings of output points, denoted r_{ij}, are similarly defined using d_{ij} in place of δ_{ij}. The value of K used to compute quality measures need not be the same as that used in the manifold learning algorithms, however, in all our simulations, we also set $K = 20$ for the purpose of computing quality measures.

Local Continuity Meta-Criterion (LCMC)

Chen and Buja (2009) proposed the local continuity criterion (LCMC) defined as

$$\text{LCMC}(K) = \frac{1}{NK} \sum_{i=1}^{N} \Big(\big|U_K(i) \bigcap V_K(i)\big| - \frac{K^2}{N-1} \Big). \tag{1}$$

The LCMC computes the average size of the overlap between the K-nearest neighborhood in the output space and K-nearest neighborhood in the input space. The value of LCMC(K), is bounded between zero and one with values closer to one indicating a larger overlap between nearest neighborhoods and therefore better quality embedding.

Trustworthiness & Continuity (T&C)

Venna and Kaski (2006) defined two quality measures, trustworthiness and continuity, that respectively distinguish two types of errors. For the first type of error, y_j is among the K-nearest neighbors of y_i (i.e. observations close in output space) but x_j is not among the

K-nearest neighbors of x_i (i.e. observations not close in the input space). Using all such points for each i, the trustworthiness of the embedding can be calculated as

$$M_T(K) = 1 - \frac{2}{G_K} \sum_{i=1}^{N} \sum_{\substack{j \in V_K(i) \\ j \notin U_K(i)}} (\rho_{ij} - K), \tag{2}$$

where the normalizing factor $G_K = \begin{cases} NK(2N - 3K - 1) & \text{if } K < N/2, \\ N(N-K)(N-K-1) & \text{if } K \geq N/2. \end{cases}$

This is bounded between zero and one, with values closer to one indicating a higher-quality representation. In contrast to LCMC, Trustworthiness uses information on the rankings of interpoint distances. In particular, points that are incorrectly included as nearest neighbors in the output space are penalized more when they are further away in the input space (ρ_{ij} is high).

For the second type of error, x_j is among the K nearest neighbors of x_i (i.e. observations close in input space) but y_j is not among the K-nearest neighbors of y_i (i.e. observations not close in the output space). Using these points, the continuity is defined as

$$M_C(K) = 1 - \frac{2}{G_K} \sum_{i=1}^{N} \sum_{\substack{j \in U_K(i) \\ j \notin V_K(i)}} (r_{ij} - K). \tag{3}$$

This is also normalized between zero and one and higher values indicate a higher-quality representation. Errors made for points further away in the output space (r_{ij}) are penalized to a greater extent.

Mean Relative Rank Errors (MRREs)

Lee and Verleysen (2008) developed two measures known as mean relative rank errors, based on similar principles as Trustworthiness and Continuity. These are defined as

$$\begin{aligned} W_n(K) &= \frac{1}{H_K} \sum_{i=1}^{N} \sum_{j \in U_K(i)} \frac{|\rho_{ij} - r_{ij}|}{\rho_{ij}}, \\ W_\nu(K) &= \frac{1}{H_k} \sum_{i=1}^{n} \sum_{j \in V_k(i)} \frac{|\rho_{ij} - r_{ij}|}{r_{ij}}, \end{aligned} \tag{4}$$

where H_K is the normalizing factor defined as $H_K = n \sum_{i=1}^{K} |N - 2i + 1|/i$.

The set of observations j that lie in the K-nearest neighborhood of i in both the input and output space do not affect the Trustworthiness and Continuity measures. In contrast, Mean Relative Rank Errors will penalize such points, particularly when the ranking of the distance between x_i and x_j differs substantially from the ranking of the distance between y_i and y_j. For ease of comparison to other quality measures, we will report $1 - W_n(K)$ and $1 - W_\nu(K)$ so that larger values indicate a better quality embedding.

Co-ranking Matrix ($Q_{NX}(K)$)

The co-ranking matrix criterion is defined as

$$Q_{NX}(K) = \frac{1}{KN} \sum_{k=1}^{K} \sum_{\ell=1}^{K} q_{k\ell}, \tag{5}$$

where

$$q_{k\ell} = |\{(i,j) : \rho_{ij} = k \text{ and } r_{ij} = \ell\}| \tag{6}$$

is the element in row k and column ℓ of the co-ranking matrix. For instance, the element in the first row and column of Q, q_{11}, denotes the number of pairs of observations for which the second observation is the nearest neighbor of the first observation in both the input and output space. The range of the criterion is $Q_{NX}(K) \in [0, 1]$, where 1 indicates a more accurate representation. It is also worth noting that the T&C, MRREs, and LCMC can be expressed in terms of the co-ranking matrix with details provided by Lee and Verleysen (2008).

Procrustes measure ($R(X, Y)$)

As a final criterion for measuring the quality of a low-dimensional representation, we consider the Procrustes measure (Goldberg and Ritov 2009). The idea rests on the definition of a manifold as a being locally isomorphic to Euclidean space. More concretely, the Procrustes measure considers observations in the neighborhood of x_i, i.e observations $x_j : j \in U_K(i)$. The Procrustes rotation finds a $d \times p$ matrix A and a d-vector b that projects x_j onto a d-dimensional subspace. The aim is for the projected (and translated) points $Ax_j + b$ to be close to the corresponding output points y_j for $j \in U_K(i)$. This measure of closeness is the Procrustes statistic which for observation i, is defined as

$$G_i = \inf_{\{A,b:A'A=I\}} \sum_{j\in U_K(i)} \|x_j - Ay_j - b\|^2. \tag{7}$$

An overall measure of quality is found given by

$$G = \frac{1}{n}\sum_{i=1}^{N} \frac{G_i}{\sum_{j\in U_K(i)} \|x_j\|^2}, \tag{8}$$

where the denominator is a normalizing factor that ensures G will lie between zero and one. An advantage of this measure is that it will favor embeddings that preserve the geometric properties (e.g. distances, angles) of the manifold. Since larger values of G indicate a worse representation, to ease comparison with other measures we report $1 - G$.

3 Experimental results

Choosing different values of the tuning parameters for approximate nearest neighbor algorithms allows for a trade-off between computational time and accuracy. For k-d trees, we consider values of ε ranging from 0 to 5 in increments of 0.1. Note that setting $\varepsilon = 0$ allows us to compute exact nearest neighbors. For Annoy, we set values of *n_trees* ranging from 2 to 100 in increments of 2 and fix *search_k* as 500. For HNSW, we set *n_links* to range from 2 to 200 incremented by 2. For each set of parameter values, we record the computational time taken to find the nearest neighbor graph as well as the recall rate .

These results are summarized in Figure 1, where each point represents the recall rate and computational time for an ANN algorithm with a specific set of tuning parameters. A vertical line indicating the computation time for exact nearest neighbors is also added to the plot as a baseline for the ANN methods. A similar line is also added to subsequent figures to compare the accuracy and computation time. As expected, each ANN method shows a trade-off between computational time and accuracy with lower computational time associated with lower recall rates. If a curve lies entirely to the top and left of another curve, this suggests that one ANN method dominates the other with respect to the proportion of true nearest neighbors found. Figure 1 shows that the best performing ANN method is Annoy, followed by HNSW followed by k-d trees. This is in line with the finding of

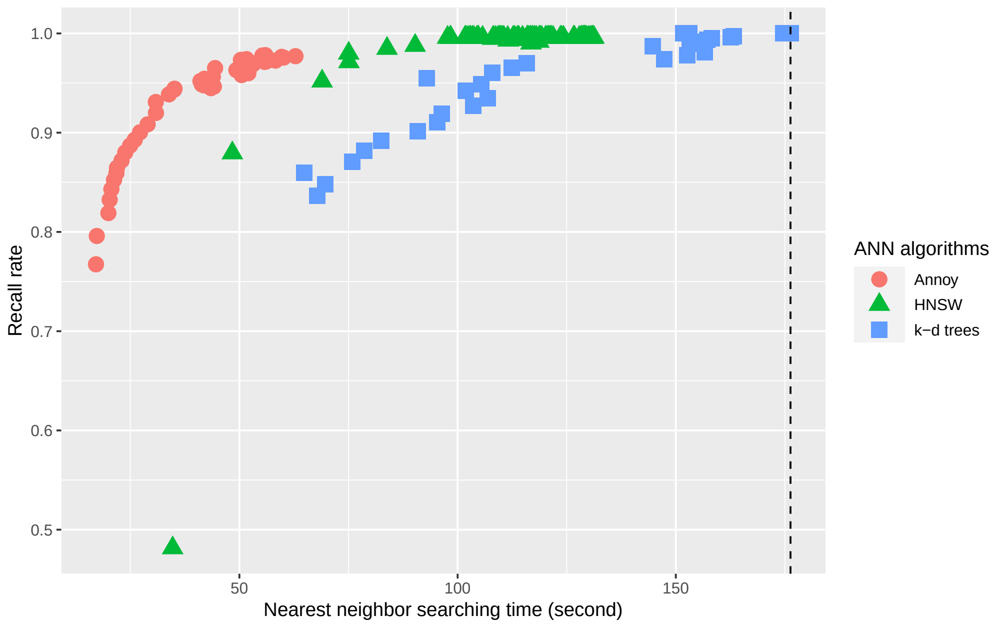

Figure 1: The comparison plot of recall rate for three ANN methods, k-d trees, Annoy, and HNSW. The points show the change of computation time (second) against the recall rate for different ANN parameter values. Points that are higher in recall rate and less in computation time (topleft) are relatively better. The black vertical line indicates the computation time for finding exact nearest neighbors.

Aumüller, Bernhardsson, and Faithfull (2020) who show that Annoy and HNSW both outperform k-d trees.

While the experiment gives a clear ranking of ANN methods according to recall rate, this ranking may not hold when the ANN methods are used as the first step of a manifold learning algorithm, and accuracy is defined according to quality measures of the resulting embedding. For the remainder of this section, we construct similar plots to Figure 1, but with different measures of the quality of a manifold learning embedding rather than recall rate.

Figure 2 shows the performance of different combinations of ANN and manifold learning algorithms according to the Trustworthiness measure. Once again ANN methods with a curve closer to the top left of the plot are to be preferred. Each panel refers to a different manifold learning algorithm method, namely ISOMAP, LLE, Laplacian Eigenmaps, Hessian LLE, t-SNE, and UMAP. Since the aim of the paper is not to find optimal manifold learning

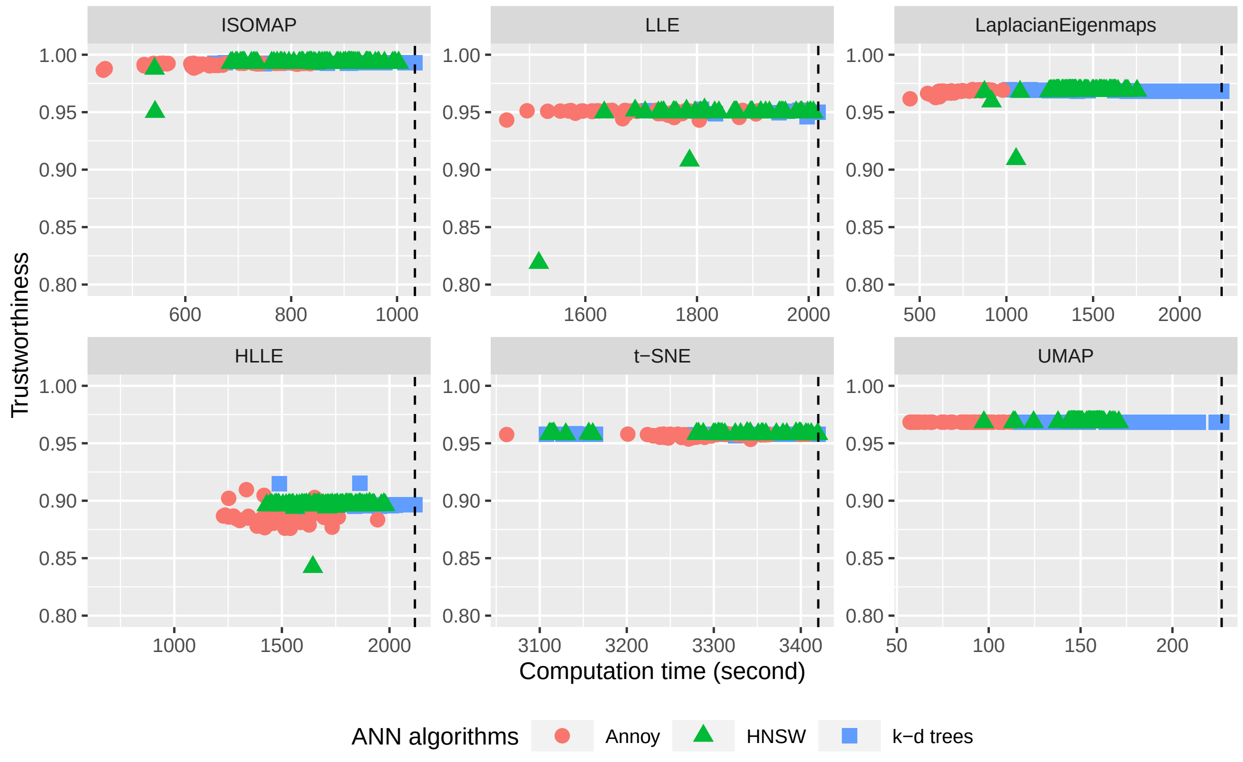

Figure 2: Trustworthiness against computation time for Annoy, HNSW and k-d trees in six manifold learning methods: ISOMAP, LLE, Laplacian Eigenmaps, Hessian LLE, t-SNE, and UMAP, with each point representing a different parameter value in ANN algorithms. Points that are higher in Trustworthiness and less in computation time are relatively better.

methods but to introduce ANN in them, we should focus on comparing across different ANN methods in subsequent figures, i.e. comparing points in different shapes and colors.

Figure 2 provides a number of insights. First, the use of approximate nearest neighbors can reduce the computational time taken to carry out manifold learning. For example, using k-d trees with ISOMAP allows computational time to be reduced from around 1,030 seconds to 650 seconds. This reduction in computational time is greatest when Annoy is used, a result consistent with Figure 1. The speed-up is also particularly noticeable for Laplacian Eigenmaps where Annoy can achieve a roughly four-fold improvement in computational time compared to exact nearest neighbors. This result can be explained by the fact that finding the nearest neighbors graph represents more of a computational bottleneck for Laplacian Eigenmaps compared to other algorithms. Second, in a result that stands in contrast to Figure 1, the improvement in computational time does not come at the cost of substantially lower Trustworthiness. In fact, the effect of using a different manifold learning algorithm seems to have a much greater impact on Trustworthiness than the use of an approximate nearest neighbor algorithm. For example, the Trustworthiness is almost always between 0.98 and 1 when ISOMAP is used (the best performing algorithm for this particular dataset), while it is almost always between 0.87 to 0.92 for Hessian LLE (the worst-performing algorithm for this particular dataset). As a minor caveat to the conclusion that ANN has a negligible impact on embedding accuracy, we note that HNSW has a small number of very inaccurate embeddings that lead to a substantially lower Trustworthiness. This may be explained by the propensity of the greedy search upon which HNSW is built to return nearest neighbors that are quite far from the true nearest neighbor.

The results from Figure 2 are, for the most part, robust to the use of metrics used to evaluate embedding quality. For instance, Figure 3 shows results of percentage change of all quality measures discussed in Section 2 for ISOMAP. For all measures, the computational time of the algorithm can be cut in half with an associated reduction in the accuracy of less than 10% (with a few exceptions due to the instability of HNSW). Similar conclusions can be drawn for LLE, Laplacian Eigenmaps, t-SNE, and UMAP so these figures[1] are omitted

[1]available at `https://github.com/ffancheng/paper-mlann/tree/public/paper/figures/public`.

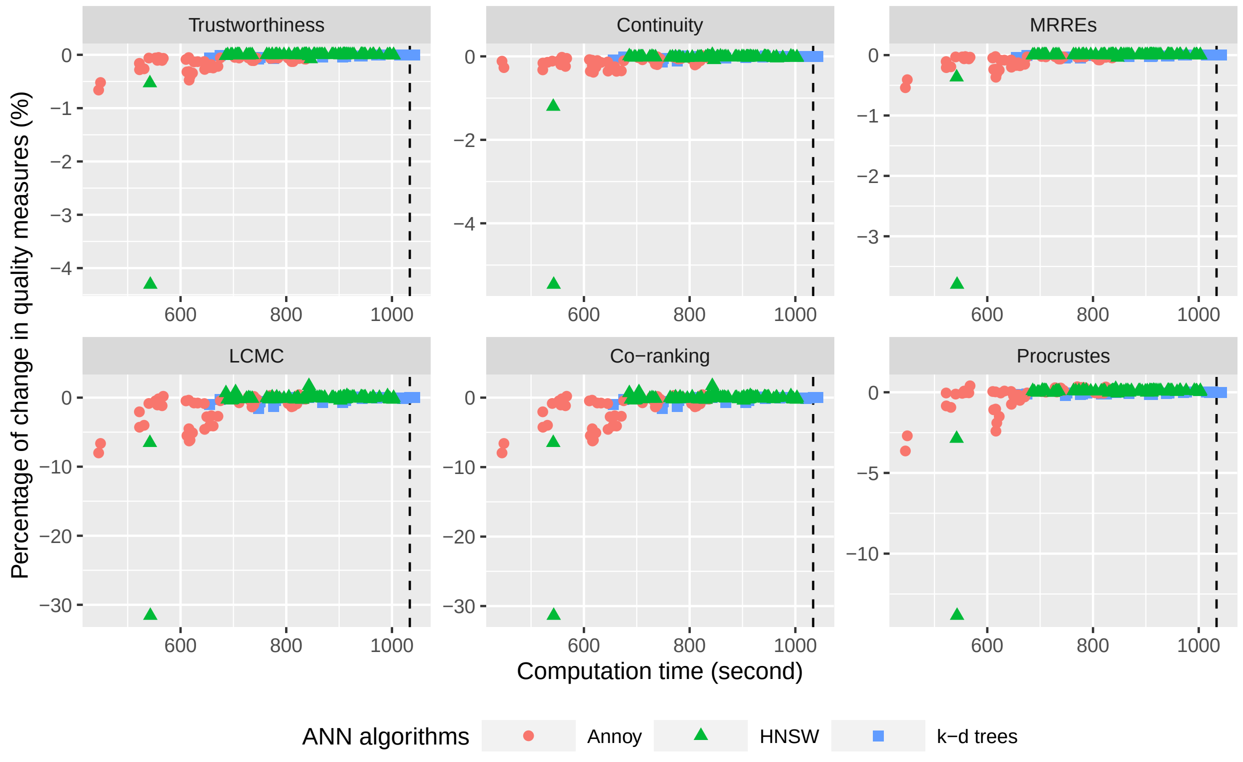

Figure 3: Comparison of ISOMAP embedding quality measures against computation time for three ANN methods, Annoy, HNSW, and k-d trees. The percentage change compared to true nearest neighbors of six quality measures are shown in the order of Trustworthiness, Continuity, MRREs, LCMC, Co-ranking matrix, and Procrustes measure.

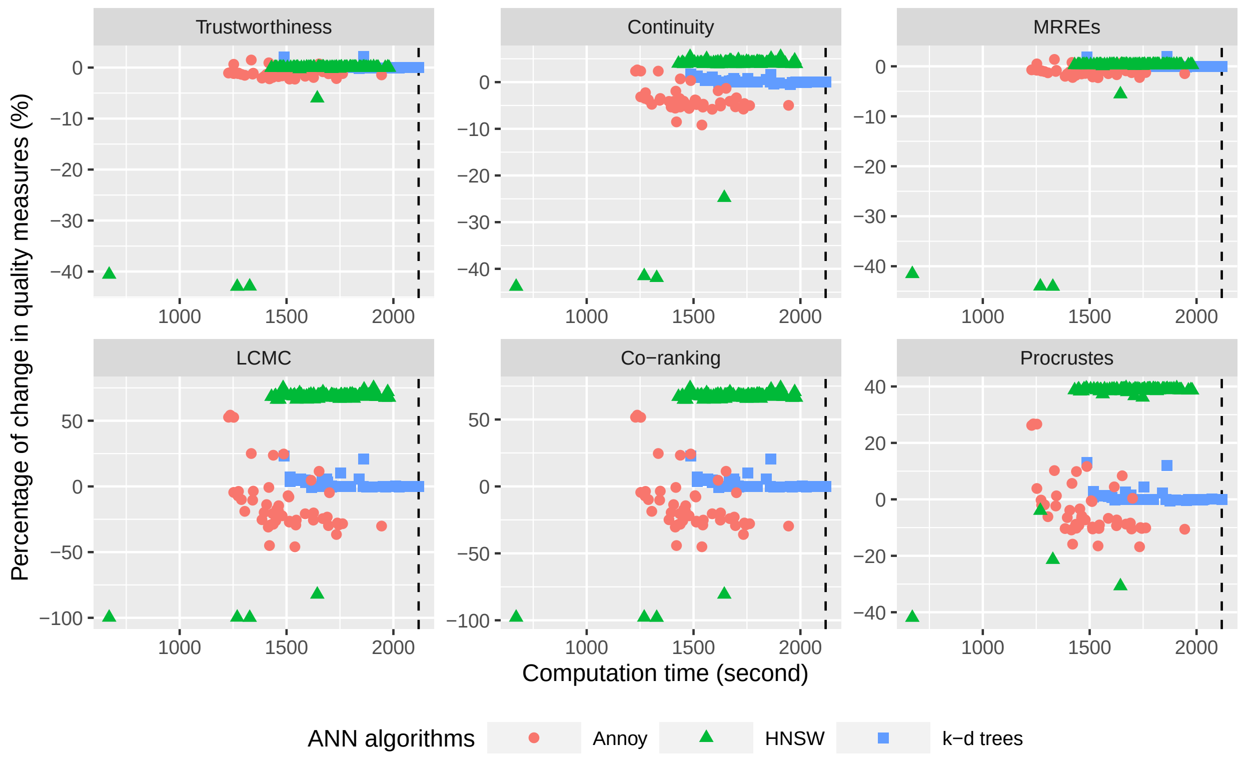

Figure 4: Comparison of percentage change in six Hessian LLE embedding quality measures against computation time for three ANN methods, Annoy, HNSW and k-d trees.

for brevity. On the other hand, Figure 4 does show that for Hessian LLE, HNSW can achieve a higher level of accuracy for some measures (Continuity, LCMC, Co-ranking, and Procrustes), even compared to exact nearest neighbors. While this result is quite counter-intuitive, it should be noted that Hessian LLE performs considerably worse than ISOMAP (in general ISOMAP achieves an LCMC of above 0.3 for ISOMAP, the corresponding figure never exceeds 0.2 for Hessian LLE). In general, with a wide range of different ANN parameter values, the combination of ANN methods with all six manifold learning methods has shown an obvious reduction in computation time with an accuracy loss of less than 10%.

Overall, the results from this benchmark study do seem to suggest that approximate nearest neighbors are suitable for use in manifold learning algorithms. In particular, we would recommend Annoy for the greatest computational speed up and warn that care should be taken if using HNSW due to the instability of this algorithm in a small number of cases. When using Annoy, the improvement in computational time comes at the cost of at most a small reduction in the accuracy. This result is robust to the use of different manifold learning algorithms, different measures of embedding accuracy, and different choices of tuning parameters for approximate nearest neighbor algorithms.

References

Aumüller, Martin, Erik Bernhardsson, and Alexander Faithfull. 2020. “ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms.” *Information Systems* 87 (January): 101374.

Chen, Lisha, and Andreas Buja. 2009. “Local Multidimensional Scaling for Nonlinear Dimension Reduction, Graph Drawing, and Proximity Analysis.” *J. Am. Stat. Assoc.* 104 (485): 209–19.

Goldberg, Yair, and Ya’acov Ritov. 2009. “Local Procrustes for Manifold Embedding: A Measure of Embedding Quality and Embedding Algorithms.” *Mach. Learn.* 77 (1): 1–25.

LeCun, Yann, Corinna Cortes, and CJ Burges. 2010. “MNIST Handwritten Digit Database.” *ATT Labs [Online]. Available: Http://Yann.lecun.com/Exdb/Mnist.*

Lee, John, and Michel Verleysen. 2008. “Quality Assessment of Nonlinear Dimensionality Reduction Based on k-Ary Neighborhoods.” In *Proceedings of the Workshop on New Challenges for Feature Selection in Data Mining and Knowledge Discovery at ECML/PKDD 2008*, edited by Yvan Saeys, Huan Liu, Iñaki Inza, Louis Wehenkel, and Yves Van de Pee, 4:21–35. Proceedings of Machine Learning Research. Antwerp, Belgium: PMLR.

Venna, Jarkko, and Samuel Kaski. 2006. “Local Multidimensional Scaling.” *Neural Netw.* 19 (6-7): 889–99.