MAXIMUM VARIANCE UNFOLDING ON DISJOINT MANIFOLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

An assumption underlying much of machine learning is that observed data are often sampled from a manifold of much lower dimension than the data space itself. While linear methods such as PCA can often be used to perform dimensionality reduction, they fail to capture nonlinear relationships in the data, which are often present in natural datasets. Maximum variance unfolding (MVU) is an established and well-studied neighborhood graph-based method for nonlinear dimensionality reduction with the unique property of retaining strong local isometry. However, its applicability to real-world data is limited due to its dependence on the connectivity of the underlying neighborhood graph: in natural datasets, data are often irregularly sampled, multimodal, or lie on disjoint manifolds, giving rise to clusters of points that are distant in the data space. In this work, we present a method that extends MVU to the common case where data lie on disjoint manifolds. By embedding neighborhood graph components in parallel and later building component connections, our method decreases both computation time and memory requirements, even with a single processor. Furthermore, we identify two practical failure modes of MVU, which our method inherently addresses, resulting in improved performance with respect to standard metrics that assess the extent to which the local structure of the data is preserved.

1 INTRODUCTION

Dimensionality reduction is a vast research field with the fundamental goal of transforming high-dimensional data into a lower-dimensional representation that captures its intrinsic structure. This is related to the manifold assumption, whereby we expect that data are sampled from distributions whose support lies on (or close to) a manifold embedded in the data space (Fefferman et al., 2016).

Despite being widely used as a preprocessing step in machine learning tasks, linear methods for dimensionality reduction, such as principal component analysis (PCA) (Pearson, 1901), are inadequate for capturing nonlinear relationships in the data. While PCA seeks to find a linear subspace that minimizes the reconstruction error of the data points, methods for nonlinear dimensionality reduction generalize this idea to smooth, nonlinear, and lower-dimensional geometries, i.e., manifolds. This is done by constructing a neighborhood graph where each data point is connected to its k-nearest neighbors. Each method then specifies a different problem resulting in embeddings with different properties, e.g., Isomap (Tenenbaum et al., 2000) maps data to a lower dimension while preserving geodesic distances along the manifold approximated by shortest paths along the neighborhood graph.

Due to their ability to recover nonlinear structure in data, and even in the age of deep learning techniques (see Figure 1), nonlinear dimensionality reduction methods are used across many domains and applications including the study of industrial (Wei et al., 2016), chemical (Boninsegna et al., 2015), and biological processes (Dsilva et al., 2018), brain imaging data (Tang et al., 2021), sentiment analysis (Kim & Lee, 2014), remote sensing (Song et al., 2024), facial recognition (Ge et al., 2024), and semi-supervised learning (Pitelis et al., 2014; Huang et al., 2019).

Maximum variance unfolding (MVU) (Weinberger & Saul, 2006a) is a well-studied method for nonlinear dimensionality reduction (Sun et al., 2006; Ghojogh et al., 2021; 2023), which attempts to pull data points apart, effectively "unfolding" them onto the embedding space. MVU has some interesting properties: unlike kernel-based methods such as kernel PCA (Schölkopf et al., 1998), it

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

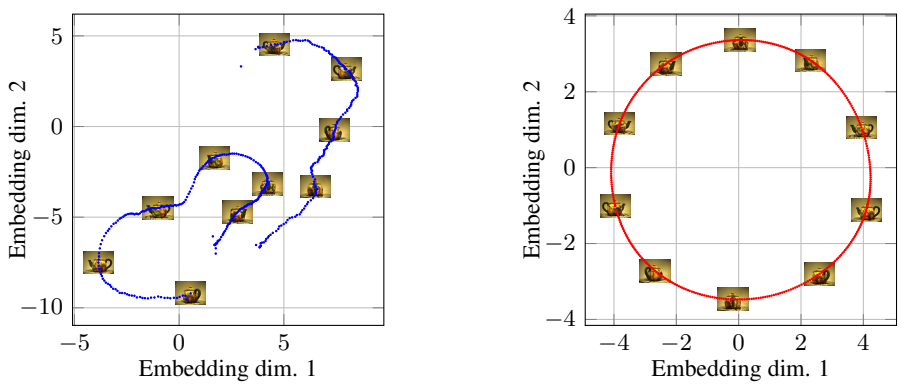


Figure 1: Embeddings of the teapots dataset Weinberger & Saul (2006a), which consists of 400 images of a rotating teapot. The embeddings on the left correspond to the latent representations of a convolutional autoencoder. The plot on the right shows embeddings discovered through maximum variance unfolding. The autoencoder is unable to adequately recover the intrinsic geometric structure of the rotation manifold.

does not require specifying a kernel, and can directly learn the intrinsic structure of the underlying manifold. In some applications, it may be difficult to find an appropriate kernel; e.g., Liu et al. (2014) show that MVU significantly outperforms kernel PCA in industrial process control. Furthermore, MVU is unique among its peers in that it provides local isometry guarantees, i.e., it enforces that distances between each point and its k-nearest neighbors should be equal in the data space and in the embedding space. In certain applications, this is a requirement, e.g., Simonetto et al. (2012) use MVU for sensor localization and robotic dispersion problems. Finally, unlike most other methods for nonlinear dimensionality reduction, MVU is resilient to the so-called "repeated eigendirection problem", whereby eigenvectors of embedding Jacobians are harmonics of previous ones (Dsilva et al., 2018; Meilă & Zhang, 2024). This is because of MVU's variance maximization objective, which works as a repulsion mechanism which makes it difficult for intrinsic dimensions to collapse.

MVU has been successfully used in many applications and types of data. Weinberger & Saul (2006b) use MVU to recover a $2D$ representation of $60,000$-dimensional text co-occurrence statistics and show that semantic relationships between words are preserved. Mahadevan et al. (2011) develop an extension to MVU which allows it to learn from bimodal data such as EEG-fMRI data and image-text pairs. Wang & Paynabar (2023) use MVU for regression in process optimization, and Song et al. (2007); Wei et al. (2016); Yang & Qi (2024) develop supervised variants of MVU.

However, MVU has two main drawbacks: most importantly, it cannot trivially be applied to data which form a disconnected neighborhood graph, i.e., when it has multiple disjoint components. Disconnected graphs arise naturally in the common case where data lie on multiple disjoint manifolds (e.g., multimodal data) or simply due to sampling irregularities. Secondly, MVU is computationally expensive, and applying it to datasets with thousands of samples may be prohibitive.

In this paper, we propose a simple solution to address both problems: we compute MVU embeddings for disjoint components in parallel, and later connect them in the embedding space. We call our method maximum variance unfolding on disjoint manifolds (MVU-DM). Our method inherently works when data form a disconnected neighborhood graph and, by decomposing the full MVU problem into separate MVU embeddings of each disjoint component, also greatly improves time and memory efficiency, even in the absence of multiple processors.

Finally, we identify two failure modes of MVU that occur in practice. By addressing both issues, we experimentally verify across several artificial and natural datasets that our method is able to not only improve computational performance but also embedding quality.[1]

[1]We provide all the software used to obtain the results presented in the paper in the supplementary material.

2 BACKGROUND

2.1 MAXIMUM VARIANCE UNFOLDING

Maximum variance unfolding (MVU) (Weinberger & Saul, 2006a) is a method for nonlinear dimensionality reduction which solves the problem of "unfolding" the data manifold by spreading points in the target space while maintaining local isometry. Given data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{x}_i \in \mathbb{R}^D$, such that $i \sim j$ indicates a k-nearest neighbor relation, we wish to find the Gramian of the embeddings $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ by solving:

$$\max_{\boldsymbol{K}} \quad \text{tr}(\boldsymbol{K}) \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{K}_{ii} - 2\boldsymbol{K}_{ij} + \boldsymbol{K}_{jj} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{2}$$

$$\boldsymbol{K} \succeq 0, \tag{3}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} \boldsymbol{K}_{ij} = 0. \tag{4}$$

This is a semidefinite programming problem (SDP). SDPs are solved using interior point methods, whose per-iteration computational complexity is $O(N^3 + (kN)^3)$: semidefinite cone operations scale with $O(N^3)$ and the number of constraints scales with $O(kN)$ (Borchers & Young, 2007), where k is a hyperparameter and denotes the k-nearest neighbors up to which isometry is to be retained. Furthermore, the memory requirement scales with $O(N^2)$ due to the Gramian. This cost is prohibitive for datasets with more than a few thousand samples, where convergence may take a long time or the data may not fit in memory at all, making MVU difficult to apply to real world data.

Finally, we can recover the actual embeddings $\boldsymbol{Y} \in \mathbb{R}^{N \times d}$, with $d << D$, by eigendecomposition of the kernel matrix and selecting the top d eigenvalues and corresponding eigenvectors: $\boldsymbol{Y} = \boldsymbol{Q}_d \boldsymbol{\Lambda}_d^{1/2}$.

2.2 RELATED WORK

Neighborhood graph-based methods for nonlinear dimensionality reduction fail when neighborhood graphs are not connected, i.e., when disconnected components arise from the k-nearest neighbor selection. In the case of MVU, if there are disjoint components, the problem becomes unbounded, as components could be taken arbitrarily far away from the origin, increasing variance to infinity.

A solution typically employed in other frameworks (Pedregosa et al., 2011) is to connect each component to all others. This fails in the case of MVU, since these constraints would make the geometry too rigid to unfold. Similarly, Fan et al. (2018) add edges between pairs of components until the rank of the set of edges equals the rank of the components themselves. This method both assumes that all components have the same intrinsic dimensionality and, similarly to the previous method, generates too many connections between components so that the intercomponent connections make it impossible for MVU to unfold the manifold.

An approach that does solve the disconnectivity problem for MVU is the use of minimum spanning trees (Dalmia & Sia, 2021) to discover connected neighborhood graphs. Another solution is the one employed in Van Der Maaten et al. (2009): iteratively connecting the largest component to the component that is closest to it until the graph is connected. While these methods allow us to perform MVU on datasets with disconnected neighborhood graphs, they do not address the computational limitations of MVU on large datasets.

Some extensions may be used to alleviate the computational burden of MVU. The Nyström approximation (Platt, 2005) is a low-rank matrix approximation used on kernel methods. It can be applied to MVU on disjoint components by embedding the largest component, and projecting the rest of the data points using a Gaussian kernel. However, by completely ignoring local structure, the Nyström approximation sacrifices the strong local isometry guarantees of MVU, achieving poor results on datasets with disconnected neighborhood graphs (Van Der Maaten et al., 2009). Landmark MVU (Weinberger et al., 2005) embeds a set of randomly sampled points, so-called "landmarks", and computes the embeddings for the remaining points as linear combinations of the landmarks. This

Algorithm 1 Maximum variance unfolding on disjoint manifolds

Input: $\boldsymbol{X} \in \mathbb{R}^{(N \times D)}, k \in \mathbb{N}$
Output: $\boldsymbol{Y} \in \mathbb{R}^{(N \times d)}$

1: $\mathcal{X}_1, \ldots, \mathcal{X}_C \leftarrow$ build_neighborhood_graph$(\boldsymbol{X}, k)$
2: $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_C \leftarrow \text{MVU}(\boldsymbol{X}, \mathcal{X}_p), \quad p = 1, \ldots, C$
3: $\mathcal{Z}_1, \ldots, \mathcal{Z}_C \leftarrow$ choose_representative_points$(\boldsymbol{Y}_p)$
4: $\mathcal{L} \leftarrow$ choose_intercomponent_connections$(\boldsymbol{X}, \mathcal{X}_p, \mathcal{Z}_p)$
5: $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_C \leftarrow$ connect_components$(\boldsymbol{Y}, \mathcal{Z}_p, \mathcal{L})$
6: $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_C \leftarrow$ translate_components$(\boldsymbol{Y}_p, \boldsymbol{Z}_p, \mathcal{Z}_p)$
7: **Return** $[\boldsymbol{Y}_1 \quad \cdots \quad \boldsymbol{Y}_C]$

method does not ignore local structure to such a drastic degree as the Nyström approximation and improves embedding computational performance, so we include it in our benchmarks.

Finally, while some very popular manifold learning methods such as t-SNE (Maaten & Hinton, 2008) and UMAP (Dalmia & Sia, 2021) are very successful at discovering embeddings that are useful for visualization, their performance in retaining local structure is weaker than that of methods such as MVU. We present results comparing our method to t-SNE and UMAP in the Appendix.

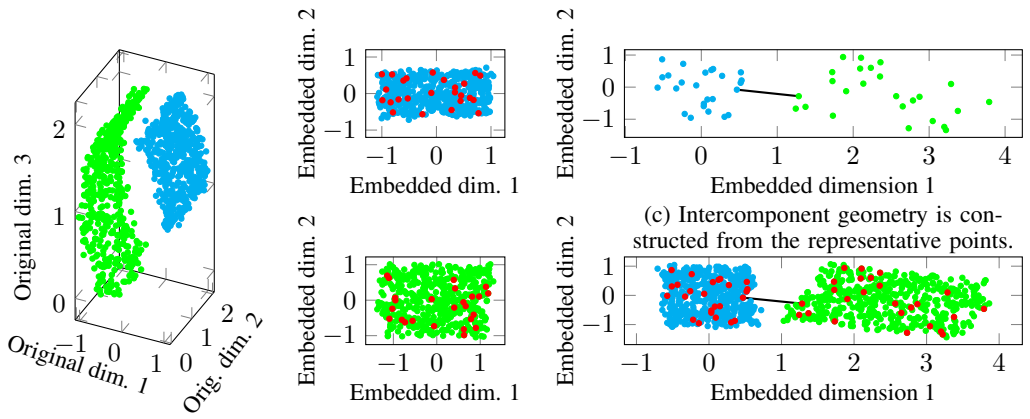
3 Maximum Variance Unfolding on Disjoint Manifolds

In this section, we describe our method in detail, which we summarize in pseudocode in Algorithm 1 and illustrate in Figure 2. The main idea is that, when data form a disconnected neighborhood graph, rather than trying to make it connected and applying MVU to the full Gramian at once, we separately embed each component and later reconnect the components in the embedding space using the intercomponent distances from the original data space.

Importantly, applying MVU to components in parallel allows the use of multiple processors to speed up embedding. Furthermore, if there are multiple components, the total memory occupied by the Gramians of the data decreases depending on the number of components and their sizes, e.g., 200 data points forming a connected graph would require a Gramian with $40,000$ entries, whereas if they were separated into four components with size 50 each, the Gramians would occupy a total of $10,000$ entries. Finally, another computational advantage of our method is that all Gramians need not be loaded into memory at the same time.

The steps of our algorithm, given data $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ and hyperparameter $k \in \mathbb{N}$, are as follows:

1. A neighborhood graph is built based on the k-nearest neighbors of each point. The components are then found by simply starting a breadth-first search on unvisited nodes until all nodes have been visited. The indices of the points that belong to each component are collected into sets $\mathcal{X}_1, \ldots, \mathcal{X}_C$, where C is the number of disjoint components. Note that:
   - The above definition implies that all components are pairwise disjoint and the union of all components covers the full dataset.
   - The number of components varies for the choice of k. The smaller the value of this hyperparameter, the larger the amount of components, and vice versa.
2. Maximum variance unfolding (MVU) is applied to each component (defined by a set of indices $\mathcal{X}_p$) separately, yielding embedded data $\boldsymbol{Y}_p \in \mathbb{R}^{|\mathcal{X}_p| \times d_p}$. Importantly, these computations can be done in parallel, since no intercomponent connections are considered at this time. We note that the dimensionality of the computed embeddings d_p depends on the intrinsic dimensionality of the corresponding manifold: like MVU, we retain the top eigenvalues of $\boldsymbol{K}$ which preserve some percentage of the variance in the original data. Furthermore, each embedded component $\boldsymbol{Y}_p$ is zero-centered, and not yet in its final position.
3. For each embedded component $\boldsymbol{Y}_p$, a set of indices of "representative points" $\mathcal{Z}_p$ is chosen uniformly at random. The goal is to choose a sample from each component that both adequately represents its large-scale geometry and that is small enough to keep the component connection step computationally cheap. In practice, we find that using 5% of the points in



(a) Original data in ambient space.

(b) Unfolded components; representative points in red.

(c) Intercomponent geometry is constructed from the representative points.

(d) Final embedded data.

Figure 2: Illustration of our method in a dataset with two components with intrinsic dimensionality of 2, embedded in a 3-dimensional ambient space (a). Each component is embedded separately, which can be done in parallel with multiple processors (b). The global structure, defined in terms of intercomponent connections, is constructed using a set of representative points from each component (c). Finally, we compute the translation of the remaining points to their final position relative to the set of representative points of their relative component (d).

each component gives adequate results in both local reconstruction and performance. In the Appendix, we show that the method is robust to different choices of representative points.

4. Connections are created between components until the neighborhood graph is connected. This is done in the same way as described in Section 2.1 for vanilla MVU: we find the largest component and the component closest to it, create a connection between the closest points between those two components, and treat them as a single component. This is done iteratively until the neighborhood graph is connected. Note that we only consider connections between points that are in the set of representative points of their respective component. This step is performed in the data space.
5. The embedded components are connected using the sets of representative points. First, to ensure that all components have the same dimensionality, we pad the embedded components with zeros. Then, for each pair of components for which there is a connection, we fix one component and compute a translation plus rotation of several random initializations of the moving component such that the exact intercomponent distance is preserved, while attempting to maximize variance. This step is explained in detail in Section 3.1.
6. The remaining points of each component $\boldsymbol{Y}_p$ are translated to their positions relative to the representative points of that component $\boldsymbol{Z}_p$. This is done by applying the rotation and translation matrices from the previous step to their respective components, leaving them in their final position.

3.1 Connecting the Embedded Components

After having embedded all components and selected their respective sets of representative points, we use the latter to recover distances between components in the original space. We do this with a simple algorithm where, for each component, we compute a rotation and translation such that intercomponent distance constraints are enforced. As noted before, all computations to connect the embedded components are performed on the set of representative points of each component for computational efficiency. Nevertheless, we refer to these sets as components for simplicity.

Given the list $\mathcal{L}$ of intercomponent connections whose elements are 5-tuples (p, q, i, j, d), where $p, q \in \{1, \ldots, C\}$ are component indices, and i, j are the indices of the points in their respective components which must maintain distance d, we connect the embedded components as follows:

1. Set the largest component as fixed.
2. Select a moving component (i.e., not in its final position) $\boldsymbol{Z}_q \in \mathbb{R}^{|\mathcal{Z}_q| \times d}$ with an intercomponent connection to a fixed component $\boldsymbol{Z}_p \in \mathbb{R}^{|\mathcal{Z}_p| \times d}$. Let $\boldsymbol{z}_{p,i}, \boldsymbol{z}_{q,j} \in \mathbb{R}^{|\mathcal{Z}_p| \times d}$ be the i-th and j-th representative points of components p and q, respectively, which serve as "anchor points", i.e., for which there is an intercomponent constraint of distance $d \in \mathbb{R}$.
3. With $\boldsymbol{e}_i \in \mathbb{R}^d$ the canonical basis vectors, we consider the set of candidate directions $\mathcal{U} = \{\pm \boldsymbol{e}_i\}_{i=1}^d \subset \mathbb{R}^d$. These determine the set of different anchor placements to choose from. Note that $|\mathcal{U}| = 2d$.
4. For each $\boldsymbol{u} \in \mathcal{U}$ we compute a translation $t(\boldsymbol{u}) = \boldsymbol{z}_{p,i} + d\boldsymbol{u} - \boldsymbol{z}_{q,j}$ which enforces the intercomponent distance constraint, i.e., such that $\|\boldsymbol{z}_{p,i} - (\boldsymbol{z}_{q,j} + t(\boldsymbol{u}))\| = d$.
5. For each translated candidate component $\boldsymbol{Z}_q + t(\boldsymbol{u}) \in \mathbb{R}^{|\mathcal{Z}_q| \times d}$ we compute the optimal rotation $R(\boldsymbol{u}) \in SO(d)$ around $\boldsymbol{z}_{q,j}$ (i.e., the anchor of the candidate moving component is fixed) such that the distance between the centroids of both components, $\boldsymbol{\mu}_p, \boldsymbol{\mu}_q(\boldsymbol{u})$, is maximized.
6. The direction $\boldsymbol{u}^*$ maximizing $\|\boldsymbol{\mu}_p - R(\boldsymbol{u})\boldsymbol{\mu}_q(\boldsymbol{u})\|$ is selected, and the corresponding rotation and translation are applied to the moving component. The process is repeated until all the components are aligned.

The embedded component connection step of our method is computationally inexpensive, consisting only of closed-form computations. Because the iterative method of establishing intercomponent connections does not admit closed loops (instead, it forms a tree structure) and since there is at most a single connection between each pair of components, each component is aligned exactly once. With N the total number of representative points, C the number of components, and d the target embedding dimension, the cost of this step is $O(Nd^2 + Cd^3)$. Further details on the computational complexity of our method may be found in the Appendix.

We highlight that our method does not guarantee that the global variance of the embedded data is maximized: assuring that the translation plus rotations computed when aligning components maximizes global variance would require a nonconvex optimization over the rigid motions $SE(d)$. However, we retain the same local isometry guarantees as MVU, as all transformations computed by our method are translations and rotations, i.e., rigid motions. Furthermore, we find empirically that searching for anchor placements using the canonical basis vectors and their inverses, as previously described, yields good results.

4 Experimental Validation

In this section, we describe in detail the experimental evaluation performed to validate our method, maximum variance unfolding on disjoint manifolds (MVU-DM). We test our method against MVU Weinberger & Saul (2006a) and Landmark MVU (L-MVU) Weinberger et al. (2005). In both baselines, in the case of disconnected neighborhood graph components, we apply the same iterative method for creating a connected neighborhood graph as we do on MVU-DM. In L-MVU, we experimented with using 50% and 90% of the data as landmarks. However, using only 50% of the points as landmarks resulted in very bad performance across all of our benchmarks, so we do not report results for this setting.

We run each method with $k \in \{2, \ldots, 20\}$ and report the best-performing configuration for each dataset. For all methods, we select only the top d most informative dimensions of the embeddings, where d is the (possibly estimated) intrinsic dimensionality of each dataset. For the artificial datasets, $d = 2$; for the natural datasets, we use the likelihood-based intrinsic dimensionality estimator by Levina & Bickel (2004). We present studies on the effects of different k in the Appendix.

For the artificial datasets, we use a k-nearest neighbor neighborhood graph, since sampling is more regular. For the natural datasets, we use a mutual k-nearest neighbor relationship, which has been found to induce neighborhood graphs with desirable properties (Dalmia & Sia, 2021). In the latter, we also incorporate small components with up to 3 points into their closest component. Though results vary little between different initializations, we use 5 seeds for all configurations and report standard deviations in the Appendix.

Table 1: Trustworthiness results.

	Artificial Datasets			Natural Datasets				
Method	BS	SR	FM	Olivetti	CBCL	COIL-20	MPDD	AF
MVU	0.9746	0.9833	<u>0.9889</u>	<u>0.9208</u>	0.9607	<u>0.9786</u>	<u>0.9563</u>	<u>0.9247</u>
L-MVU	0.9522	0.9588	0.9706	0.7386	0.9111	0.9088	0.9200	0.8953
MVU-DM	<u>0.9995</u>	<u>0.9972</u>	0.9862	0.9192	<u>0.9737</u>	0.9567	0.9418	0.9192

4.1 Datasets

We included a variety of both artificial and natural datasets with varying global structures, scales, and intrinsic dimensionalities. We describe them here on a high level and refer to the Appendix for illustrations and further details.

4.1.1 Artificial Datasets

We considered all datasets from Fan et al. (2018). However, we exclude the Broken Swiss Roll dataset since it is largely redundant with the Broken S-curve dataset in terms of their properties. All artificial datasets consist of $2,000$ points, to which we added Gaussian noise with variance 0.05.

The selection of artificial datasets we use focuses on evaluating the way each method relates components that were found disconnected from the neighborhood graph:

- The Broken S-Curve (BSC) dataset consists of 4 sections of a bent $2D$ manifold forming a $3D$ 'S'-shaped structure.
- The Swiss Rolls (SR) dataset consists of two Swiss rolls at an arbitrary distance, and with different orientations.
- The Four Moons (FM) dataset consists of two pairs of $2D$ C-shaped manifolds embedded in $3D$ space. Each pair is composed of a smaller manifold nested within a larger one.

4.1.2 Natural Datasets

We test our method with a diverse set of natural image datasets varying in resolution, sampling density, and intrinsic dimensionality between modes of variability:

- The Olivetti faces dataset (Samaria & Harter, 1994) is a 400-photo dataset of the faces of 40 people taken from different angles at a 64×64 resolution.
- The MIT-CBCL dataset (Serre et al., 2007) is another face dataset with resolution of 64×64, but only from 10 subjects and a total of $2,059$ samples.
- The COIL-20 dataset (Nene et al., 1996) contains images of 20 different objects, where each object is placed on a turntable against a black background. Photos of each object were taken at intervals of 5 degrees of rotation, corresponding to 72 photos per object and a total of $1,440$ samples. The images have a 128×128 resolution.
- The Metal Parts Defect Detection (MPDD) dataset (Jezek et al., 2021) has $1,346$ images of normal and defective versions of 6 different industrial metal parts at a 256×256 resolution.
- The Animalfaces (AF) dataset (Si & Zhu, 2012) contains $2,409$ images, at 64×64 resolution, of the faces of 20 different species of animals. The intrinsic variability of its classes varies widely, with the 'Dog' class featuring species of very different colors, snouts, fur textures, etc., while images of the 'Panda' class have very little variability.

4.2 Metrics

We follow Van Der Maaten et al. (2009) and assess all methods according to the quality of their embeddings with respect to trustworthiness and continuity (Venna & Kaski, 2006) with $k = 12$. Trustworthiness and continuity assess how well neighborhoods around each point are preserved in the embedding space. For the i-th point, $\mathcal{U}_i^k$ is the set of its k-nearest neighbors in the embedding space. With j its $r(i,j)$-nearest neighbor in the input space, trustworthiness is:

Table 2: Continuity results.

	Artificial Datasets			Natural Datasets				
Method	BS	SR	FM	Olivetti	CBCL	COIL-20	MPDD	AF
MVU	0.9902	0.9932	0.9935	<u>0.8797</u>	0.9750	<u>0.9842</u>	0.9340	0.8548
L-MVU	0.9101	0.9093	0.9161	0.7312	0.8799	0.8672	0.8473	0.6858
MVU-DM	<u>0.9996</u>	<u>0.9989</u>	<u>0.9953</u>	0.8772	<u>0.9769</u>	0.9732	<u>0.9340</u>	<u>0.8827</u>

Table 3: Speedup of each method relative to MVU.

	Artificial Datasets			Natural Datasets				
Method	BS	SR	FM	Olivetti	CBCL	COIL-20	MPDD	AF
L-MVU	1.3×	1.4×	1.9×	1.4×	1.5×	1.8×	1.8×	1.5×
MVU-DM-1	63.6×	50.4×	10.2×	6.3×	10.3×	128.2×	6.4×	1.2×
MVU-DM-4	124.3×	126.6×	19.9×	7.4×	19.6×	137.2×	15.0×	2.4×

$$T(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in \mathcal{U}_i^k} \max(0, r(i,j) - k) \tag{5}$$

We can think of $r(i,j)$ as a ranking: a list, ordered by distance, of the neighbors of i in the original space. Then, since $\mathcal{U}_i^k$ is the set of neighbors of i in the embedding space, trustworthiness penalizes "intrusions" into the set of nearest neighbors after embedding.

Analogously, if we define $\mathcal{V}_i^k$ as the set of the k-nearest neighbors of i in the original space, and $\hat{r}(i,j)$ as the ranking of j in terms of nearest-neighbors of i in the embedding space, we get the metric of continuity, which measures how many neighbors of i in the data space are no longer its neighbors in the embedding space:

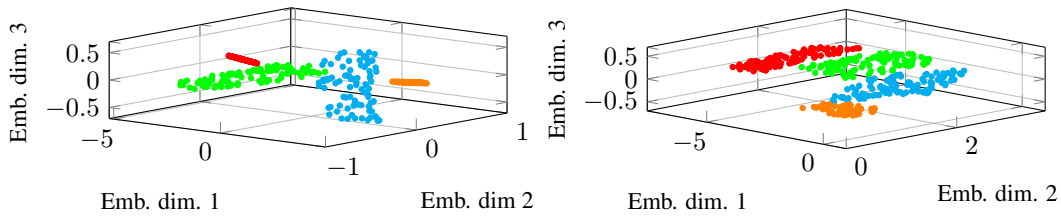
$$C(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in \mathcal{V}_i^k} \max(0, \hat{r}(i,j) - k) \tag{6}$$

4.3 Discussion

Tables 1 and 2 show the results for the trustworthiness and continuity metrics on the tested methods. These experiments show that not only did our method not sacrifice performance, it often resulted in higher quality embeddings as measured by the preservation of local structure. In fact, we identify two failure modes of MVU which may explain why it did not perform as well as our method:

1. The variance maximization objective of MVU may cause a component to collapse to a dimension lower than its intrinsic dimensionality. This is illustrated in Figure 3a, where the top and bottom components of a broken S curve are stretched along a line, collapsing an intrinsically $2D$ component into $1D$. This does not violate any of MVU's constraints because distances between neighbors can be maintained despite a shearing of the component.
2. While MVU maximizes variance in the embedding space, there is nothing guiding optimization towards concentrating information in the fewest dimensions possible. In Figure 3a, even though the two middle components have each been correctly linearized into a $2D$ representation, their embeddings are approximately orthogonal. If we eigendecomposed the kernel matrix found through MVU into a $2D$ subspace, we would lose much of the information of the cyan component. In other words, even if each component is correctly unfolded, the global arrangement of the embeddings may still result in information loss.

MVU-DM addresses both failure modes (Figure 3b). Since each component is embedded separately and zero-centered, unfolding in two orthogonal directions increases variance the most, so dimensional collapse is avoided. On the other hand, since components' Gram matrices are decomposed separately, the embeddings are placed in the d-dimensional target subspace such that their most informative dimensions are maintained.



(a) Pathological case of MVU's variance maximization objective causing disjoint components to collapse to a dimensionality lower than the intrinsic dimensionality of the data.

(b) MVU-DM unfolds both components into a flat $2D$ sheet and positioned such that a projection of all the embeddings onto a $2D$ subspace would result in virtually no loss of information.

Figure 3: Embeddings produced by MVU and MVU-DM, respectively, of the Broken S-Curve dataset. The solution was recovered from the $3D$ ambient space, rather than a $2D$ projection.

Furthermore, more detailed results provided in the Appendix also show that MVU-DM often achieves at least comparable performance to MVU for the vast majority of tested values of k, and that the results of MVU-DM are not sensitive to the choice of representative points or anchors of each component, as seen by very low standard deviation across several initializations.

Besides often improving embedding performance over MVU, we find that our method significantly speeds up execution. We present relative execution times of the three methods in Table 3 for the configurations that yielded the best results. For MVU-DM, we present results using one core (MVU-DM-1) and four cores (MVU-DM-4).

The speedups obtained with MVU-DM largely depend on the connectivity of the neighborhood graph: in the artificial datasets, whose components can easily be embedded in parallel and have equal sizes, speedups were massive; in the natural datasets, where components have varying sizes and intrinsic dimensionalities, there is greater variance in speedups. Nevertheless, MVU-DM consistently offered significant speedups on both artificial and natural datasets.

On average, the embedding performance of L-MVU fell short of that of vanilla MVU (Tables 1 and 2). This is expected, given its tradeoff of embedding only a subset of the data to achieve better computational performance. Indeed, Table 3 shows that L-MVU consistently reduced computation time relative to MVU. However, speedups were generally much lower than those of MVU-DM.

5 Conclusion

Neighborhood graphs are central to most methods for nonlinear dimensionality reduction, but may be naturally disconnected for a number of reasons, including the data lying on disjoint manifolds or due to sampling irregularities.

We introduce maximum variance unfolding on disjoint manifolds (MVU-DM), a simple but effective method that extends maximum variance unfolding (MVU) in the common setting of disconnected neighborhood graphs. We identify two failure modes of MVU and argue that our method effectively addresses both issues, which we find empirical support for in benchmarks across a varied set of artificial and natural datasets, where MVU-DM matches or surpasses MVU in embedding performance.

Furthermore, our method can be much more computationally efficient since it does not require loading the full Gramian of the data into memory and allows parallel embedding of components using multiple processing cores. However, we find that even with a single processing core, applying MVU separately to each component may already result in speedups of one or two orders of magnitude. Finally, our method does not introduce any additional hyperparameters to MVU and is not sensitive to different initializations nor choices of representative points.

An interesting direction for future work could be the development of methods for computing neighborhood graphs which not only take into account local properties (Mekuz & Tsotsos, 2006; Wei et al., 2008; Dalmia & Sia, 2021), but also how graph component partitioning might lend itself to decomposing the embedding problem such that manifold learning methods can be extended to even larger datasets.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs (after proofs were checked by humans), implement methods, create or modify scientific figures or images, creation of artifacts, and create or edit software code. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, formulate questions for surveys or interviews, suggest experimental parameters, draft parts of a research paper, transcribe recordings of research material, summarize or analyse existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature, format references, suggest a structure for a research paper, propose a title or keywords for a research paper. The authors verified any LLM-generated LaTeX and Python code, and LLMs were used to double-check mathematical proofs after those had been reviewed by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We submit, as supplementary material, the code required to run the experiments. It includes a README file with detailed instructions for running all methods and an explanation of all program parameters. The supplementary material also includes all the datasets used in the paper. In the Appendix, we describe how all three artificial datasets were generated. The natural datasets are cited in the main text. A proof of the computational complexity of the component connection step can also be found in the Appendix. Finally, we also disclose in the Appendix the hardware that was used as well as run time statistics for our experiments.

REFERENCES

MOSEK ApS. *The MOSEK Python Fusion API manual. Version 11.0.*, 2026. URL `https://docs.mosek.com/latest/pythonfusion/index.html`.

Lorenzo Boninsegna, Gianpaolo Gobbo, Frank Noé, and Cecilia Clementi. Investigating molecular kinetics by variationally optimized diffusion maps. *Journal of chemical theory and computation*, 11(12):5947–5960, 2015.

Brian Borchers and Joseph G Young. Implementation of a primal–dual method for sdp on a shared memory parallel architecture. *Computational Optimization and Applications*, 37:355–369, 2007.

Ayush Dalmia and Suzanna Sia. Clustering with UMAP: why and how connectivity matters. *CoRR*, abs/2108.05525, 2021. URL `https://arxiv.org/abs/2108.05525`.

Carmeline J Dsilva, Ronen Talmon, Ronald R Coifman, and Ioannis G Kevrekidis. Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study. *Applied and Computational Harmonic Analysis*, 44(3):759–773, 2018.

Jicong Fan, Tommy WS Chow, Mingbo Zhao, and John KL Ho. Nonlinear dimensionality reduction for data with disconnected neighborhood graph. *Neural Processing Letters*, 47(2):697–716, 2018.

Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

Huilin Ge, Zhiyu Zhu, Jiali Ouyang, Muhammad Awais Ashraf, Zhiwen Qiu, and Umar Muhammad Ibrahim. Integration of manifold learning and density estimation for fine-tuned face recognition. *Symmetry*, 16(6):765, 2024.

Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Unified framework for spectral dimensionality reduction, maximum variance unfolding, and kernel learning by semidefinite programming: Tutorial and survey. *arXiv preprint arXiv:2106.15379*, 2021.

Benyamin Ghojogh, Mark Crowley, Fakhri Karray, and Ali Ghodsi. Unified spectral framework and maximum variance unfolding. In *Elements of Dimensionality Reduction and Manifold Learning*, pp. 285–312. Springer, 2023.

Rui Huang, Guopeng Zhang, and Junli Chen. Semi-supervised discriminant isomap with application to visualization, image retrieval and classification. *International Journal of Machine Learning and Cybernetics*, 10(6):1269–1278, 2019.

Stepan Jezek, Martin Jonak, Radim Burget, Pavel Dvorak, and Milos Skotak. Deep learning-based defect detection of metal parts: evaluating current methods in complex conditions. In *2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pp. 66–71, 2021. doi: 10.1109/ICUMT54235.2021.9631567.

Kyoungok Kim and Jaewook Lee. Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction. *Pattern Recognition*, 47(2):758–768, 2014.

Elizaveta Levina and Peter Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in neural information processing systems*, 17, 2004.

Yuan-Jui Liu, Tao Chen, and Yuan Yao. Nonlinear process monitoring and fault isolation using extended maximum variance unfolding. *Journal of process control*, 24(6):880–891, 2014.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Vijay Mahadevan, Chi Wong, Jose Pereira, Tom Liu, Nuno Vasconcelos, and Lawrence Saul. Maximum covariance unfolding: Manifold learning for bimodal data. *Advances in Neural Information Processing Systems*, 24, 2011.

Marina Meilă and Hanyu Zhang. Manifold learning: What, how, and why. *Annual Review of Statistics and Its Application*, 11(1):393–417, 2024.

Nathan Mekuz and John K Tsotsos. Parameterless isomap with adaptive neighborhood selection. In *Joint Pattern Recognition Symposium*, pp. 364–373. Springer, 2006.

Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia object image library (coil-20). Technical report, Technical Report CUCS-005-96, Department of Computer Science, Columbia University, 1996.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Nikolaos Pitelis, Chris Russell, and Lourdes Agapito. Semi-supervised learning using an unsupervised atlas. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 565–580. Springer, 2014.

John Platt. Fastmap, metricmap, and landmark mds are all nyström algorithms. In *International Workshop on Artificial Intelligence and Statistics*, pp. 261–268. PMLR, 2005.

Ferdinando S Samaria and Andy C Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision*, pp. 138–142. IEEE, 1994.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

Thomas Serre, Aude Oliva, and Tomaso Poggio. A feedforward architecture accounts for rapid categorization. *Proceedings of the national academy of sciences*, 104(15):6424–6429, 2007.

Zhangzhang Si and Song-Chun Zhu. Learning hybrid image templates (hit) by information projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1354–1367, 2012. doi: 10.1109/TPAMI.2011.227.

Andrea Simonetto, Tamás Keviczky, and Dimos V Dimarogonas. Distributed solution for a maximum variance unfolding problem with sensor and robotic network applications. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 63–70. IEEE, 2012.

Le Song, Arthur Gretton, Karsten Borgwardt, and Alex Smola. Colored maximum variance unfolding. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/55a7cf9c71f1c9c495413f934dd1a158-Paper.pdf.

Wenhui Song, Xin Zhang, Guozhu Yang, Yijin Chen, Lianchao Wang, and Hanghang Xu. A study on dimensionality reduction and parameters for hyperspectral imagery based on manifold learning. *Sensors*, 24(7):2089, 2024.

Jun Sun, Stephen Boyd, Lin Xiao, and Persi Diaconis. The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM review*, 48(4):681–699, 2006.

Yunbo Tang, Dan Chen, and Xiaoli Li. Dimensionality reduction methods for brain imaging data analysis. *ACM Computing Surveys (CSUR)*, 54(4):1–36, 2021.

Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Laurens Van Der Maaten, Eric O Postma, H Jaap Van Den Herik, et al. Dimensionality reduction: A comparative review. *Journal of machine learning research*, 10(66-71):13, 2009.

Jarkko Venna and Samuel Kaski. Visualizing gene interaction graphs with local multidimensional scaling. In *The European Symposium on Artificial Neural Networks*, 2006. URL https://api.semanticscholar.org/CorpusID:12239921.

Qian Wang and Kamran Paynabar. Maximum covariance unfolding regression: A novel covariate-based manifold learning approach for point cloud data. *arXiv preprint arXiv:2303.17852*, 2023.

Chihang Wei, Junghui Chen, and Zhihuan Song. Developments of two supervised maximum variance unfolding algorithms for process classification. *Chemometrics and Intelligent Laboratory Systems*, 159:31–44, 2016.

Jia Wei, Hong Peng, Yi-Shen Lin, Zhi-Mao Huang, and Jia-Bing Wang. Adaptive neighborhood selection for manifold learning. In *2008 international conference on machine learning and cybernetics*, volume 1, pp. 380–384. IEEE, 2008.

Kilian Weinberger, Benjamin Packer, and Lawrence Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In Robert G. Cowell and Zoubin Ghahramani (eds.), *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pp. 381–388. PMLR, 06–08 Jan 2005. URL https://proceedings.mlr.press/r5/weinberger05a.html. Reissued by PMLR on 30 March 2021.

Kilian Q Weinberger and Lawrence K Saul. Unsupervised learning of image manifolds by semidefinite programming. *International journal of computer vision*, 70:77–90, 2006a.

Kilian Q Weinberger and Lawrence K Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, volume 6, pp. 1683–1686, 2006b.

Deliang Yang and Hou-Duo Qi. Supervised maximum variance unfolding. *Machine Learning*, 113 (9):6197–6226, 2024.

A CONVEX MVU

In this Section we derive a convex version of MVU from the original formulation.

Given data $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^N, \boldsymbol{x}_i \in \mathbb{R}^D$, such that $i \sim j$ indicates a k-nearest neighbor relation, we wish to find embeddings $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^N, \boldsymbol{y}_i \in \mathbb{R}^d$, with $d \ll D$, by solving:

$$\max_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N \in \mathbb{R}^d} \quad \sum_{i=1}^N \|\boldsymbol{y}_i\|_2^2 \tag{7}$$

$$\text{s.t.} \quad \|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{8}$$

$$\sum_{i=1}^N \boldsymbol{y}_i = \boldsymbol{0} \tag{9}$$

The objective encodes our wish to spread the data as much as possible in the target space, i.e., to maximize the variance of the embeddings $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N \in \mathbb{R}^d$. With $i \sim j$ indicating a k-nearest neighborhood relationship between the i-th and j-th points, the constraint in equation 8 specifies that distances between neighbors in the target space should be equal to the original distances between those same points. Finally, a centering constraint is enforced in equation 9. This is necessary because if the data were not centered in the target space, the points could be taken indefinitely far away from the origin, maximizing variance but leading to an unbounded problem.

Note that the problem as stated is not convex: the objective consists of maximizing a quadratic function. Furthermore, the bilinear terms on the left hand side of constraint equation 8 define a quadratic equality on the decision variables, which does not, in general, define a convex set. Expanding the squared terms:

$$\max_{\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N} \quad \sum_{i=1}^N \boldsymbol{y}_i^\top \boldsymbol{y}_i \tag{10}$$

$$\text{s.t.} \quad \boldsymbol{y}_i^\top \boldsymbol{y}_i - 2\boldsymbol{y}_i^\top \boldsymbol{y}_j + \boldsymbol{y}_j^\top \boldsymbol{y}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{11}$$

$$\sum_{i,j=1}^n \boldsymbol{y}_i^\top \boldsymbol{y}_j = 0 \tag{12}$$

By collecting the embedded points into the columns of a matrix $\boldsymbol{Y} = [\boldsymbol{y}_1 \quad \cdots \quad \boldsymbol{y}_n] \in \mathbb{R}^{d \times N}$, we can rewrite the problem:

$$\max_{\boldsymbol{Y}} \quad \text{tr}(\boldsymbol{Y}^\top \boldsymbol{Y}) \tag{13}$$

$$\text{s.t.} \quad \boldsymbol{e}_i^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{14}$$

$$\boldsymbol{1}^\top \boldsymbol{Y}^\top \boldsymbol{Y} \boldsymbol{1} = 0, \tag{15}$$

where $\boldsymbol{e}_i$ is a "selection" or "one-hot" vector, i.e., its elements are all zero except for the element at the i-th index, which is 1. Note that, now, the whole problem depends on $\boldsymbol{Y}^\top \boldsymbol{Y}$, so we introduce a new variable $\boldsymbol{K} = \boldsymbol{Y}^\top \boldsymbol{Y} \in \mathbb{R}^{N \times N}$ to linearize the terms that depend on $\boldsymbol{Y}^\top \boldsymbol{Y}$:

$$\max_{\boldsymbol{K},\boldsymbol{Y}} \quad \text{tr}(\boldsymbol{K}) \tag{16}$$

$$\text{s.t.} \quad \boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{K} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j \tag{17}$$

$$\boldsymbol{1}^\top \boldsymbol{K} \boldsymbol{1} = 0 \tag{18}$$

$$\boldsymbol{K} = \boldsymbol{Y}^\top \boldsymbol{Y} \tag{19}$$

Because $\boldsymbol{K} = \boldsymbol{Y}^\top \boldsymbol{Y}$ defines an inner product matrix, we can replace that constraint if we make sure that $\boldsymbol{K}$ is both symmetric, positive semidefinite and that the rank of $\boldsymbol{K}$ is not greater than the dimension of the $\boldsymbol{y}_i$s:

$$\begin{aligned}
\max_{\boldsymbol{K}} \quad & \operatorname{tr}(\boldsymbol{K}) && (20)\\
\text{s.t.} \quad & \boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{K} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_l\|_2^2, \quad i \sim j && (21)\\
& \mathbf{1}^\top \boldsymbol{K} \mathbf{1} = 0 && (22)\\
& \boldsymbol{K} \succeq 0 && (23)\\
& \operatorname{rk}(\boldsymbol{K}) \leq n && (24)
\end{aligned}$$

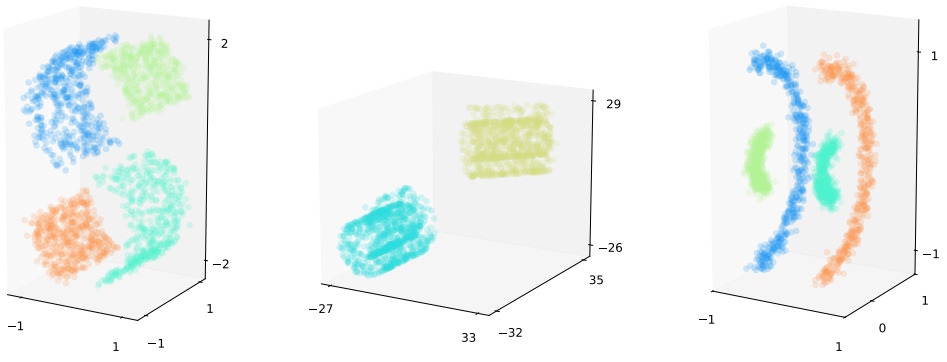
Now, the only nonconvexity in the problem is given by the rank constraint. If we remove it, we arrive at a convex relaxation of the original problem, which is the final formulation of MVU:

$$\begin{aligned}
\max_{\boldsymbol{K}} \quad & \operatorname{tr}(\boldsymbol{K}) && (25)\\
\text{s.t.} \quad & \boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_i - 2\boldsymbol{e}_i^\top \boldsymbol{K} \boldsymbol{e}_j + \boldsymbol{e}_j^\top \boldsymbol{K} \boldsymbol{e}_j = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2, \quad i \sim j && (26)\\
& \mathbf{1}^\top \boldsymbol{K} \mathbf{1} = 0 && (27)\\
& \boldsymbol{K} \succeq 0 && (28)
\end{aligned}$$

Here, $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is a Gramian (or inner product) matrix, so that maximizing its trace corresponds to maximizing the variance of the data in the target space.

B Artificial Datasets

A Gaussian perturbation $\epsilon \in \mathcal{N}(0, 0.05)$ was added to the generated data points of all artificial datasets. The three datasets are illustrated in Figure 4.



(a) Broken S-curve (b) Swiss Rolls (c) Four Moons

Figure 4: Visual representation of the synthetic datasets used.

Broken S-Curve - The dataset is defined by:

$$\boldsymbol{t}_0 \sim \mathcal{U}\left(-\frac{3\pi}{2}, -\frac{3\pi}{2} + 0.4\pi\right) \tag{29}$$

$$\boldsymbol{t}_1 \sim \mathcal{U}\left(-\frac{3\pi}{2} + 0.6\pi, -\frac{3\pi}{2} + 1.4\pi\right) \tag{30}$$

$$\boldsymbol{t}_2 \sim \mathcal{U}\left(-\frac{3\pi}{2} + 1.6\pi, -\frac{3\pi}{2} + 2.4\pi\right) \tag{31}$$

$$\boldsymbol{t}_3 \sim \mathcal{U}\left(-\frac{3\pi}{2} + 2.6\pi, -\frac{3\pi}{2} + 3.0\pi\right) \tag{32}$$

$$\boldsymbol{t} \leftarrow (\boldsymbol{t}_0, \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3) \tag{33}$$

$$\boldsymbol{h} \sim \mathcal{U}(0, 2) \tag{34}$$

$$\boldsymbol{X} = \begin{bmatrix} \sin(\boldsymbol{t}) \\ \boldsymbol{h} \\ \text{sign}(\boldsymbol{t}) \cdot (\cos(\boldsymbol{t}) - 1) \end{bmatrix} + \epsilon \tag{35}$$

Swiss Rolls - Defining a single Swiss Roll:

$$\boldsymbol{t} \sim \mathcal{U}\left(\frac{3\pi}{2}, 3\pi\right) \tag{36}$$

$$\boldsymbol{h} \sim \mathcal{U}(0, 30) \tag{37}$$

$$\boldsymbol{X}^{\text{single}} = \begin{bmatrix} \boldsymbol{t}\cos(\boldsymbol{t}) \\ \boldsymbol{h} \\ \boldsymbol{t}\sin(\boldsymbol{t}) \end{bmatrix} + \epsilon, \tag{38}$$

$$\tag{39}$$

$$\tag{40}$$

given transformations:

$$\boldsymbol{R} = \begin{bmatrix} \cos(-\pi/4) & -\sin(-\pi/4) & X \\ \sin(-\pi/4) & \cos(-\pi/4) & X \\ 0 & X & 1 \end{bmatrix} \tag{41}$$

$$\boldsymbol{t} = \begin{bmatrix} 20 \\ 20 \\ 30 \end{bmatrix}, \tag{42}$$

we define the dataset as:

$$\boldsymbol{X} = \boldsymbol{X}^{\text{single}} \cup \left(\boldsymbol{R}\boldsymbol{X}^{\text{single}} + \boldsymbol{t}\right) \tag{43}$$

Four Moons - Defining a single pair of moons as:

$$\boldsymbol{t} \sim \mathcal{U}(0, \pi) \tag{44}$$

$$\boldsymbol{X}_1 = \begin{bmatrix} 0 \\ \sin(\boldsymbol{t}) \\ \cos(\boldsymbol{t}) \end{bmatrix} + \epsilon \tag{45}$$

$$\boldsymbol{X}_2 = \begin{bmatrix} 0 \\ \frac{1-\sin(\boldsymbol{t})}{4} \\ \frac{-\cos(\boldsymbol{t})}{4} \end{bmatrix} + \epsilon \tag{46}$$

$$\boldsymbol{X}^{\text{single}} = \boldsymbol{X}_1 \cup \boldsymbol{X}_2, \tag{47}$$

the full dataset is defined:

$$\boldsymbol{X} = \boldsymbol{X}^{\text{single}} \cup \left(\boldsymbol{X}^{\text{single}} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) \tag{48}$$

Table 4: Trustworthiness results (artificial datasets).

Method	Broken S-Curve	Swiss Rolls	Four Moons
MVU	0.9746 ± 0.0000	0.9833 ± 0.0000	<u>0.9889</u> ± 0.0000
L-MVU	0.9540 ± 0.0027	0.9630 ± 0.0086	0.8947 ± 0.1500
MVU-DM	<u>0.9995</u> ± 0.0000	<u>0.9983</u> ± 0.0021	0.9736 ± 0.0154

Table 5: Trustworthiness results (natural datasets).

Method	Olivetti	CBCL	COIL-20	MPDD	Animalfaces
MVU	0.9138 ± 0.0000	0.9687 ± 0.0000	0.9636 ± 0.0000	<u>0.9480</u> ± 0.0000	<u>0.9272</u> ± 0.0000
L-MVU	0.6628 ± 0.0268	0.9281 ± 0.0111	0.8780 ± 0.0337	0.8865 ± 0.0076	0.7711 ± 0.0735
MVU-DM	<u>0.9221</u> ± 0.0033	<u>0.9958</u> ± 0.0000	<u>0.9943</u> ± 0.0015	0.9476 ± 0.0012	0.9118 ± 0.0005

C Computational Complexity of the Component Connection Step

With N the number of points, C the number of components, n_c the number of points in component c, and component dimensionality d, we study the computational complexity of the component connection step of our method.

The cost per constraint/component alignment is:

1. Translation enumeration. For each $\boldsymbol{u} \in \mathcal{U}$:
   - Computing the translation: $O(d)$;
   - Applying the translation: $O(n_c)$;
   - Computing the centroid of the moving component: $O(n_c d)$.
2. For each translation candidate, computing the rotation around $\boldsymbol{z}_{q,j}$ which maximizes distance between centroids:
   - Computing vectors $\boldsymbol{\mu}_p - \boldsymbol{z}_{p,i}$ and $\boldsymbol{\mu}_q - \boldsymbol{z}_{q,j}$: $O(d)$;
   - Closed-form rotation: $O(d^2)$.
3. To score each candidate translation plus rotation:
   - Applying the rotation to the centroid only: $O(d^2)$;
   - Computing the distance: $O(d)$.

Thus, per translation candidate, the cost is $O(n_c d + d^2)$. Considering canonical directions and their inverses, we multiply by $2d$, so that $O(n_c d^2 + d^3)$ is the cost to align a single component.

Since each component is aligned exactly once due to the tree structure induced by the iterative algorithm that joins components, the total cost of reconstructing the global geometry of the embedded data is:

$$\sum_{c=1}^{C} O(n_c d^2 + d^3) = O(N d^2 + C d^3). \tag{49}$$

D Full Results Tables

In tables 4 to 7 we reproduce the results presented in the main paper for the best configuration of each method, along with the corresponding standard deviation across 5 seeds.

E Ablations

E.1 Sensitivity of Embedding Quality to Choice of k

In figs. 5 to 12 we show the trustworthiness and continuity values for values of $k \in \{2, \ldots, 20\}$ for the three methods, in all datasets.

Table 6: Continuity results (artificial datasets).

Method	Broken S-Curve	Swiss Rolls	Four Moons
MVU	0.9902 ± 0.0000	0.9932 ± 0.0000	0.9935 ± 0.0000
L-MVU	0.9109 ± 0.0021	0.9093 ± 0.0021	0.8698 ± 0.0909
MVU-DM	<u>0.9996</u> ± 0.0000	<u>0.9989</u> ± 0.0001	<u>0.9952</u> ± 0.0012

Table 7: Continuity results (natural datasets).

Method	Olivetti	CBCL	COIL-20	MPDD	Animalfaces
MVU	<u>0.8639</u> ± 0.0000	0.9773 ± 0.0000	0.9781 ± 0.0000	0.9022 ± 0.0000	0.8441 ± 0.0000
L-MVU	0.7018 ± 0.0080	0.8697 ± 0.0165	0.8530 ± 0.0480	0.7979 ± 0.0132	0.6646 ± 0.0328
MVU-DM	0.8602 ± 0.0061	<u>0.9885</u> ± 0.0000	<u>0.9892</u> ± 0.0011	<u>0.9249</u> ± 0.0049	<u>0.8827</u> ± 0.0002

E.2 Sensitivity of Achieved Speedups to Choice of k

In Figure 13 we show the relative speedups of LMVU and MVU-DM (single core) relative to MVU.

F Computational Details

We implemented our experimental setup using Python3 and the MOSEK Fusion API (ApS, 2026). Our infrastructure consists of two AMD EPYC 9224 and 756GB of RAM.

G Comparison to t-SNE and UMAP

We present a comparison of MVU-DM against t-SNE Maaten & Hinton (2008) and UMAP Dalmia & Sia (2021) in terms of distance preservation, trustworthiness, and continuity in tables 8 to 10, respectively. Despite not enforcing isometry constraints, t-SNE and UMAP achieve impressive results on distance preservation in natural datasets. On the other hand, MVU-DM stands out in more regularly sampled manifolds.

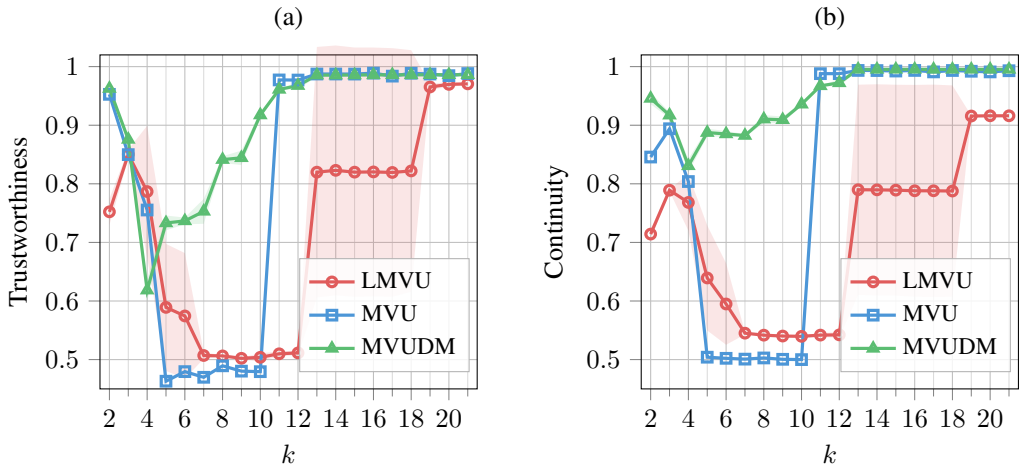


Figure 5: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the Four Moons artificial dataset. Shaded regions show standard deviation across 5 seeds.

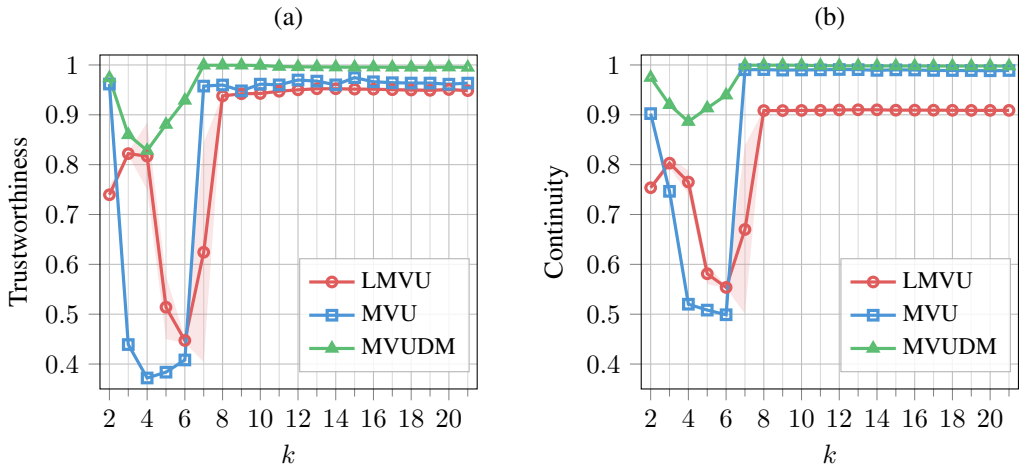


Figure 6: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the Broken S-Curve artificial dataset. Shaded regions show standard deviation across 5 seeds.

Table 8: Distance Preservation Error results.

	Artificial Datasets			Natural Datasets				
Method	BS	SR	FM	Olivetti	CBCL	COIL-20	MPDD	AF
t-SNE	0.2800	0.3360	0.3984	0.4582	0.4027	0.4235	0.5250	0.5515
UMAP	0.5113	0.4944	0.6110	0.6713	0.8344	0.6987	0.7585	0.5417
MVU-DM	0.1323	0.0256	0.1771	0.4587	0.4899	0.5351	0.5553	0.5186

Table 9: Trustworthiness results.

	Artificial Datasets			Natural Datasets				
Method	BS	SR	FM	Olivetti	CBCL	COIL-20	MPDD	AF
t-SNE	0.9992	0.9995	0.9977	0.9495	0.9939	0.9891	0.9770	0.9693
UMAP	0.9986	0.9988	0.9954	0.9566	0.9903	0.9883	0.9764	0.9700
MVU-DM	0.9995	0.9983	0.9736	0.9221	0.9958	0.9943	0.9476	0.9118

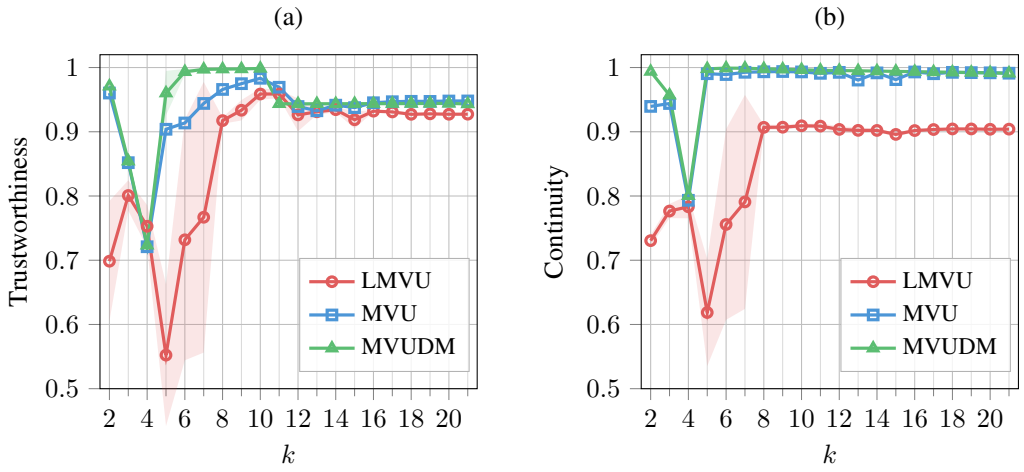


Figure 7: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the Swiss Rolls artificial dataset. Shaded regions show standard deviation across 5 seeds.

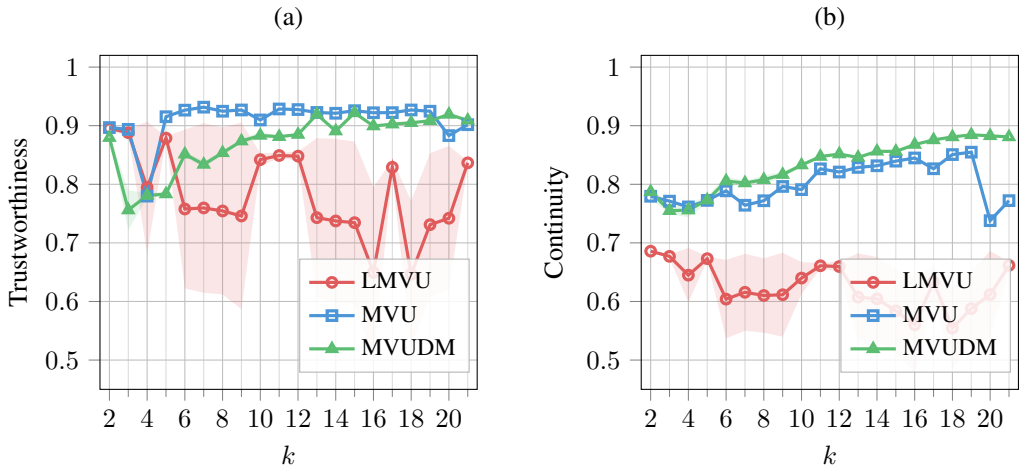


Figure 8: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the Animal Faces dataset. Shaded regions show standard deviation across 5 seeds.

Table 10: Continuity results.

	Artificial Datasets			Natural Datasets				
Method	BS	SR	FM	Olivetti	CBCL	COIL-20	MPDD	AF
t-SNE	0.9994	0.9956	<u>0.9977</u>	0.9248	<u>0.9940</u>	0.9879	<u>0.9784</u>	0.9482
UMAP	0.9983	0.9941	0.9962	<u>0.9289</u>	0.9893	0.9808	0.9780	<u>0.9708</u>
MVU-DM	<u>0.9996</u>	<u>0.9989</u>	0.9952	0.8602	0.9885	<u>0.9892</u>	0.9249	0.8827

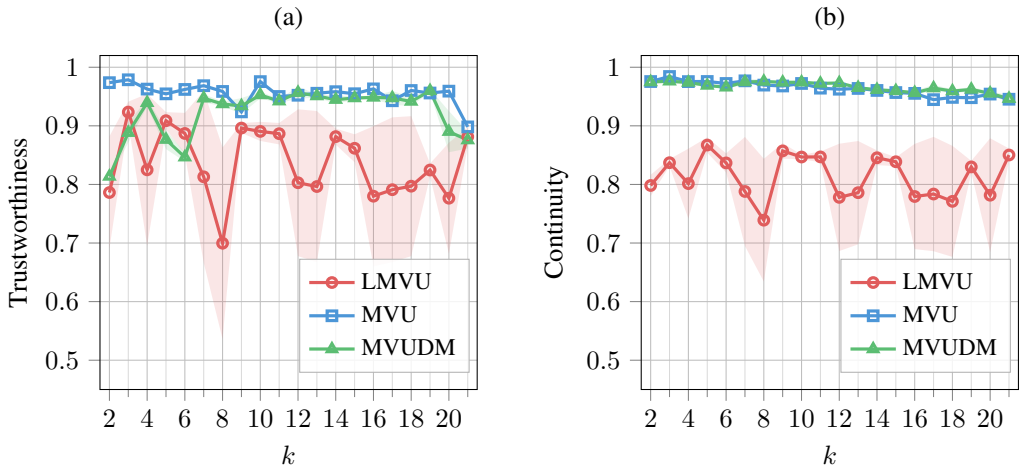


Figure 9: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the COIL20 dataset. Shaded regions show standard deviation across 5 seeds.

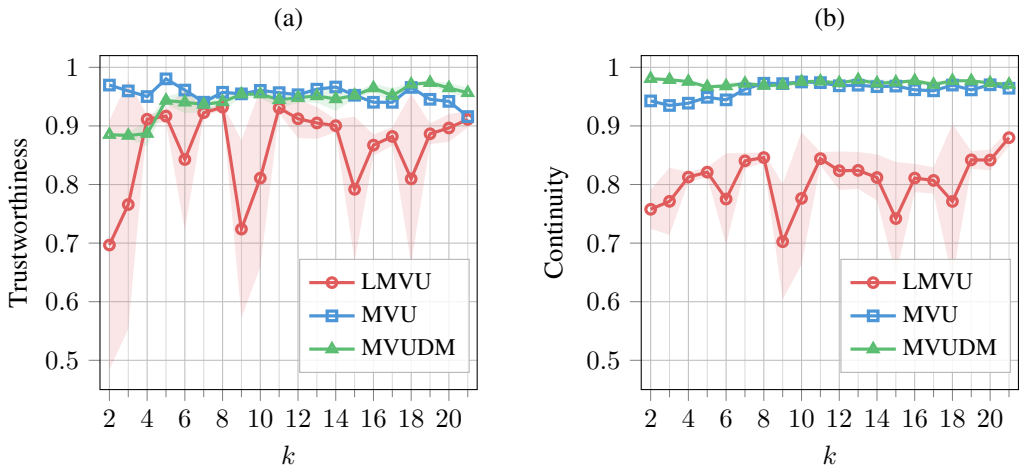


Figure 10: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the MIT-CBCL dataset. Shaded regions show standard deviation across 5 seeds.

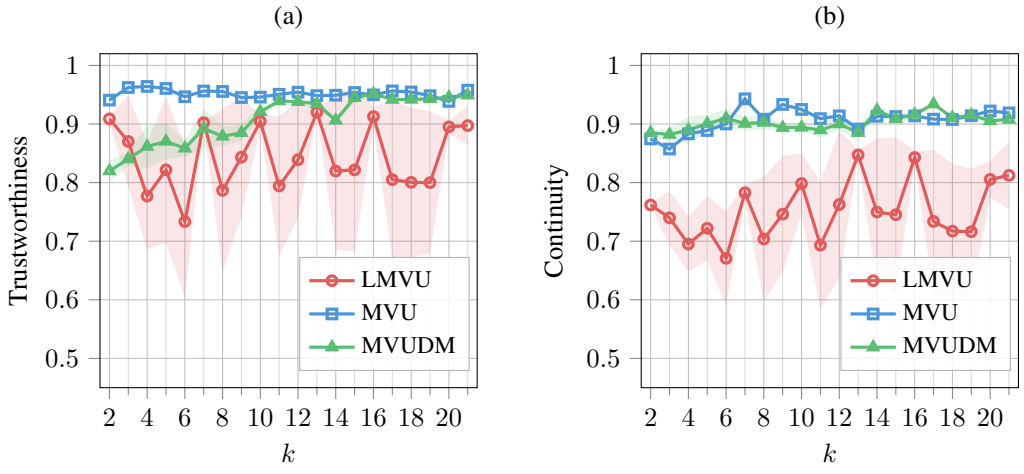


Figure 11: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the MPDD dataset. Shaded regions show standard deviation across 5 seeds.

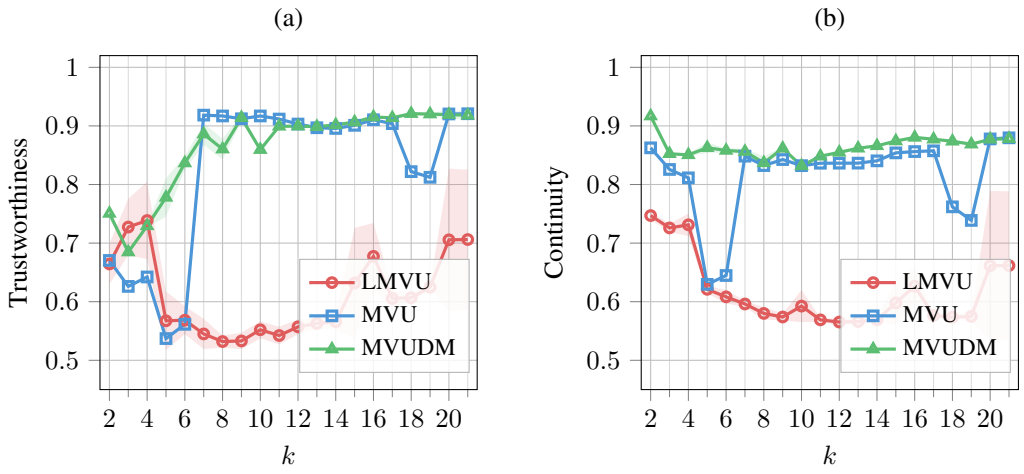


Figure 12: Sensitivity of trustworthiness (a) and continuity (b) to the neighbourhood parameter k for each method on the Olivetti dataset. Shaded regions show standard deviation across 5 seeds.

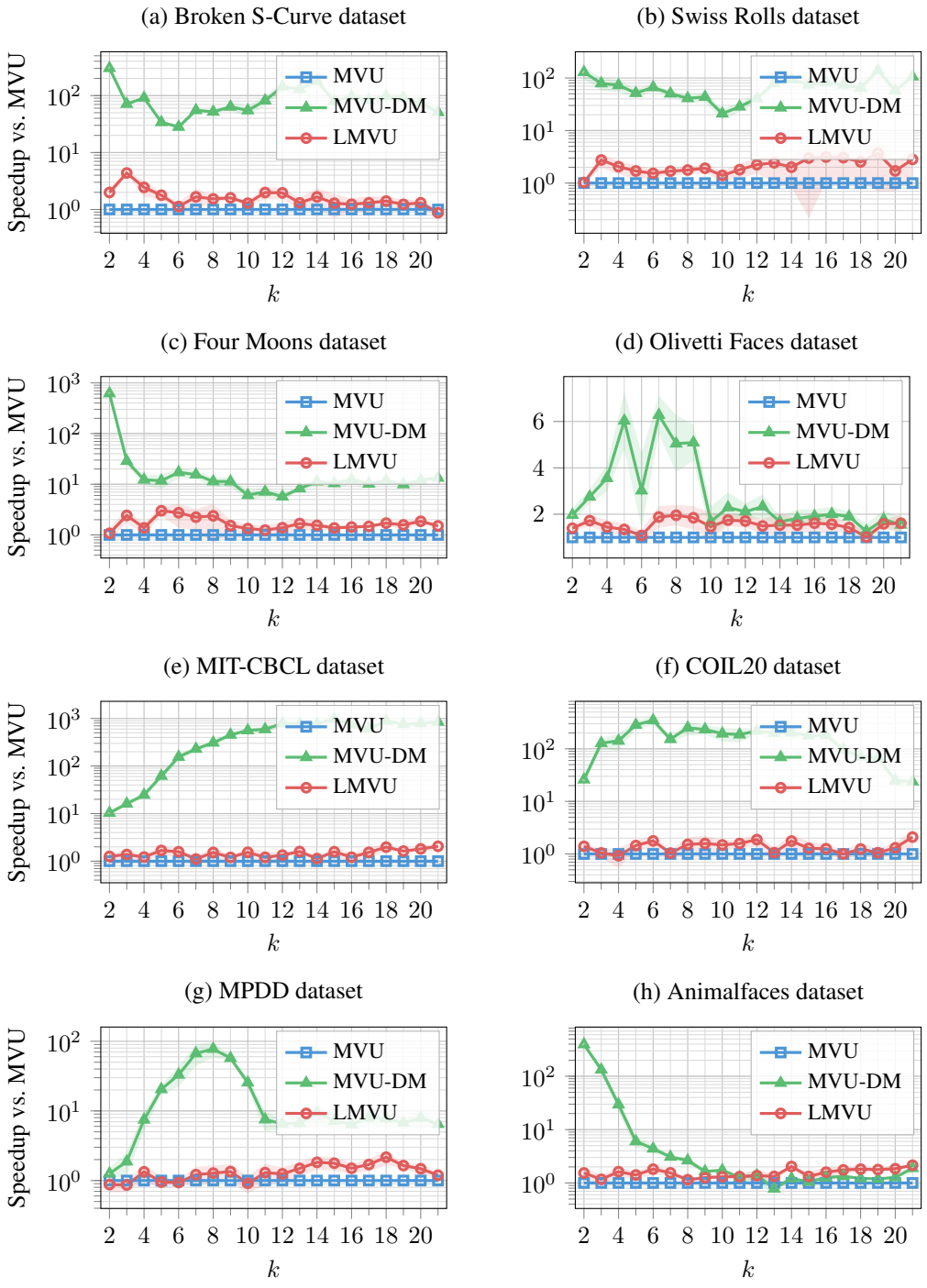


Figure 13: Runtime speedup of LMVU and MVU-DM (single CPU) relative to MVU (speedup $= t_{\text{MVU}}/t_{\text{method}}$, so values > 1 indicate faster than MVU) as a function of the number of neighbors k, across all eight datasets. MVU is fixed at 1 by definition. Shaded regions show the standard deviation across 5 seeds. Note the log scale in most plots.