Morfismos, Vol. 27, No. 1, 2023, pp. 1–30

Dimension reduction algorithms and techniques

Tzolkin Garduño Alvarado[*], Feliú Sagols Troncoso[†], Gunnar Wolf[‡], Eddie Soulier[§], Francis Rousseaux[¶]

Abstract

The recent proliferation of multidimensional output models in AI has catalyzed advancements in dimension reduction research. While these models accommodate diverse application fields and dimensions, the sheer multiplicity doesn't always yield substantial insights. Dimension reduction emerges as a pivotal process for translating high-dimensional data into a more manageable lower-dimensional space. Achieving this requires preserving geometric and topological properties across both spaces. This article aims to explore fundamental dimension reduction techniques, essential for data science researchers venturing into this domain. Of particular focus is UMAP, currently recognized as the pinnacle of this field, offering an expansive research landscape for topologists, geometers, and mathematicians alike.

2020 Mathematics Subject Classification: 55N31, 62R40, 68T09.
Keywords and phrases: Dimension Reduction, PCA, ISOMAP, LLE, t-SNE, UMAP.

1 Introduction

In present day technology development the use of mathematical tools for resource optimization is crucial. The ability to implement an online

[*]Université de Technologie de Troyes, France tzolkin.garduno_alvarado@utt.fr, tzolkin.garduno@gmail.com

[†]Departamento de Matemáticas, CINVESTAV-I.P.N. México, CDMX fsagols@math.cinvestav.edu.mx

[‡]Instituto de Investigaciones Económicas UNAM / Facultad de Ingeniería UNAM gwolf@gwolf.org

[§]Université de Technologie de Troyes, France eddie.soulier@utt.fr

[¶]Université de Reims Champagne-Ardenne francis.rousseaux@univ-reims.fr

service and offer it to the target clientele in an agile and easy manner determines the success of the business. The recent Artificial Inteligence (AI) centered business surge has favored a wide variety of services across markets. Thus a thorough comprehension of the diverse models and methods used for service delivery using AI is greatly encouraged for the data scientists that choose and implement the corresponding models.

AI has helped in advancing analysis techniques for big sets of data. Specially in the field of Natural Language Processing (NLP), a great deal of effort has been made to find numerical representations of language. Generally speaking, the models used in this field transform the input data, in either text or voice, to a numerical output, in the form of a vector. The spaces where those vectors are found are called embedding spaces or embeddings for short.

The reader would wonder about the reason of using embedding spaces. The answer lies in the lack of understanding of what a numerical representation of natural language really means. Embeddings serve many purposes, they are used for input coding for later processing or to extract relative similarity among the elements of the input. They are usually, but not exclusively, high-dimensional spaces. For example, some early Large Language Models used for NLP had 300-dimension outputs [1]. More recent models use embedding space dimension as a hyperparameter [2], that is, it can be set *a priori* by the user. That means that the number of dimensions of the output is a parameter that the user is able to choose depending on their needs. This raises the question of how to choose the optimal number of dimensions for the output, which is an open problem to this day. One of the reasons for this is that there is no clear qualitative interpretation for the dimensions of embeddings. Thus, the research in the field has stuck to grid search, heuristics or optimization methods for determining the dimension that yields the best results according to a given metric. In consequence, large dimension spaces have been used in hopes of better capturing semantics. The latter poses some problems, given an apparent ever growing number of dimensions for embeddings. One of them is the processing power needed to analyze such kind of data, high dimension vector processing can be overwhelming for the end-user's computer. Additionally, increasing the number of dimensions does not necessarily improve metric performance. Moreover, mathematical and computer architecture considerations have to be made, for example:

1. **Curse of Dimensionality**: This term, introduced by Richard

Bellman, refers to the non-intuitive phenomena that arises when dealing with high-dimensional data. As the dimension of a dataset increases, the volume of the data space grows exponentially, meaning the data becomes sparse. This causes many statistical and machine learning techniques to become less effective, as it's harder to make meaningful inferences in such sparse spaces [3].

2. **Computational Cost**: Processing and analyzing high-dimensional data require a significant amount of computational resources. Algorithms become slower and require more memory. Reducing dimensionality can make data processing and analysis more computationally efficient.

3. **Avoiding Overfitting**: With a large number of features relative to observations, it's easy for a model to overfit[1] to the training data, capturing noise and losing generalization capability. By reducing dimensionality, we can eliminate redundant or non-informative features, allowing the model to focus on the most meaningful features and reducing the risk of overfitting.

4. **Interpretability**: In many contexts, working with a lower-dimensional dataset can make results more interpretable. If the reduced dimensions have clear meaning, results can more easily be subject to qualitative assessment.

5. **Visualization**: It's challenging to visualize data in more than three dimensions. By reducing a dataset's dimensionality to two or three components, graphical visualization can be used to explore and understand the data. Dimensionality reduction techniques like t-SNE or UMAP are especially popular for visualizing complex datasets in low-dimensional space while preserving local structures.

The previous list is not exhaustive, it only serves as a coarse list of the obstacles found when using high dimensional spaces. To overcome these problems, dimension reduction techniques have been developed using a vast array of mathematical tools in the areas of geometry, statistics,

[1] Overfitting refers to a model that approximates the training data *too well*, as the model works well with its training set, but fails to perform comparably when applied to new, unseen data. It occurs when a model is excessively complex, such as having too many parameters relative to the number of observations.

topology and optimization. These tools are used with the intention to preserve the properties of embeddings through a process of dimension contraction. In other words, the purpose of dimension reduction analysis is to preserve as much information as possible about a given set P found in a high dimension space while bringing it to a space with fewer dimensions. By information we mean a set of properties of interest that are intrinsic to P in terms of configuration and structure. Hence, topology plays a foundational part in dimension reduction techniques. For that, some topological constraints are considered:

1. **Preservation of Connectivity**: An ideal technique should preserve connectivity between points. If two points are connected in the original space, they should remain connected in the reduced space.

2. **Preservation of Local Proximity**: The local structure between points in the original space should be mirrored in the reduced-dimension space. That is, if two points are "close" in the high-dimensional space, they should remain "close" in the reduced space.

3. **Conservation of Cycles**: If the original dataset contains cycles or loops, a robust dimensionality reduction technique should attempt to preserve these cycles in the reduced space. Homology [4], which studies these cycles, is a powerful tool in topology, and techniques that preserve homology classes are especially valuable.

4. **Preservation of Geodesic Distance**: In many applications, we care not just about the Euclidean distance between points, but also the geodesic distance, which considers the "path" between points through the data structure. A good technique should try to preserve geodesic distances between points.

5. **Topological Stability**: A robust dimensionality reduction technique should be resilient to small perturbations in the data. Small perturbations in the original space shouldn't cause significant changes in the topological structure of the reduced space.

6. **Preservation of Density**: In some contexts, preserving the local density of points can be important. That is, dense regions in the original space should map to dense regions in the reduced space.

Certainly, it might be difficult (or even impossible) to preserve all of these properties simultaneously, especially when significantly reducing dimensions. The choice of technique often involves a balance between preserving different topological properties depending on the most relevant characteristics to preserve in a specific problem. Consequently, the field of application will determine the properties of interest.

1. **Image compression:** When storing or transmitting image data, the number of pixels represents a very high dimension. JPEG compression [5] works by doing dimension reduction, it transforms the image into a lower dimensional space using Discrete Cosine Transform (DCT) [6], retaining only the most important information. This reduces the storage space of images with minimal quality loss.

2. **Word embeddings:** Words are represented as high dimensional sparse vectors in a vocabulary. Word embeddings like word2vec [1, 7] reduce their dimension by mapping words into a much lower dimensional dense vector space, while preserving semantic relationships between words. This makes words easier to analyze.

3. **Autoencoders:** Autoencoders [8, 9] are neural networks which compress the input into a lower-dimensional code and then reconstruct the output from this code. The code is a reduced representation of the input, useful for dimensionality reduction, compression and denoising.

4. **Manifold learning:** Real-world high dimensional data often lies close to a lower dimensional manifold. Manifold learning algorithms try to learn the lower dimensional representation of the data while preserving its properties. This can lead to revealing hidden structures [10]. Figure 1 depicts an input and corresponding output example where a "rolled carpet"-like surface is represented in a lower dimensional space.

In summary, dimension reduction is useful to simplify high dimensional data into its most significative components while minimizing information loss. The subsequent uses of data in a reduced dimension space are regression, classification, noise removal and pattern recognition among others, [11]. All of which are families of methods that are key for AI service provision.

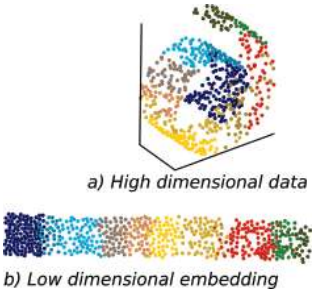


Figure 1: Manifold learning example.

This works layout is the following. In Section 2 we introduce Principal Component Analysis (PCA), which is the most frequently method used in data science. This method seeks to project data into a lower-dimensional subspace while preserving as much variance from the original data as possible and is useful when variables are highly correlated. Section 3 presents the ground theory used for the Isomap (Isometric Manifold Approximation) algorithm where the notion of intrinsic manifold approximation is developed along with the use of a loss function for point adjustment assessment first introduced. The loss function is first introduced in Section 3.1. Section 4 introduces the method known as Local Linear Embedding, which locally projects the high-dimensional space into the low-dimensional one, preserving the convex combinations of each point concerning its nearest neighbors. In Section 5, we introduce the Stochastic Neighbor Embedding method, which also carries out local re-embeddings preserving the probabilistic distributions of the nearest neighbors to each of the points in low dimensional space. In Section 6, we introduce the technique known as UMAP (Uniform Manifold Approximation and Projection), which has been developed in some ways from the techniques in the previous two sections but uses some topological concepts that had never been used before. Currently, this method is an active research area. Section 7 shows some examples of the results obtained from all the algorithms referred in previous sections with well known datasets. Finally, in Section 8, we present the conclusions of the work.

2 Principal Component Analysis

Principal Component Analysis PCA [12], is a classic dimensionality reduction method used for statistical analysis as well as machine learning [13]. This method transforms a set of observations from a given set of variables, that are not necessarily correlated, to a set of not linearly correlated variables called **principal components**. It is commonly used to visualize data in fewer dimensions, where each of the components, or axes of the reduced dimensionality coordinate system, is chosen in a decreasing order according to the variance found in the direction of the corresponding component. The first component can be interpreted as the direction in which the variance along the observations is the largest, that is, it helps explain the widest changes of the set. The second component is the direction for which variance is the second highest, and so on. However, it is important to remember that PCA is a linear technique and may not be suitable if the relationships in the data are nonlinear. Fig.2 shows a bidimensional data set with its two principal components.

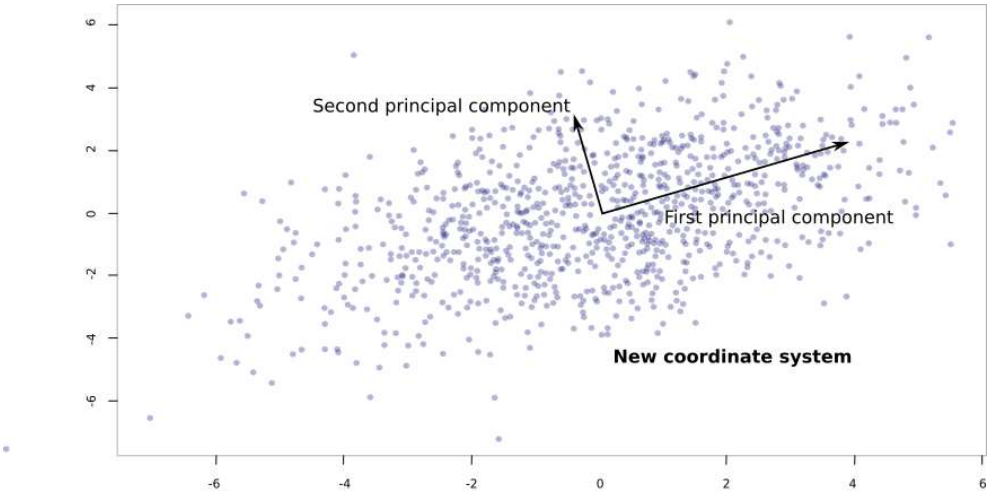


Figure 2: Principal components of a bi-dimensional set of observations.

To better understand the process PCA follows, lets suppose $P \subset \mathbb{R}^m$. To find a set $Q \subset \mathbb{R}^l$ representing P in a lower dimension, $l < m$, the PCA algorithm starts by calculating the covariance matrix of P and then calculating its eigenvectors and eigenvalues [12]. The eigenvectors define the directions of the resulting components and the eigenvalues define

their magnitude. Those magnitudes are used to sort the eigenvectors given eigenvalues represent the variance of P in the direction of their corresponding eigenvector.

Lets suppose that $P = (P_1, ..., P_m)$ is a multivariate random variable where $mean(P_k) = 0$ for all k. Then $C = Cov(P) = \frac{1}{n}\sum_{i=1}^{N} x_i x_i^T = \frac{1}{n}P^T P$ is the covariance matrix of P, where x_i are observations.

Lets suppose $w \in \mathbb{R}^m$ is a unit vector, then the variance of the observations in the direction of w, or the variance of the projections of the observations along the direction of w, is:

$$V_P(w) = \frac{1}{n}\sum_{i=1}^{n}(w^T x_i)^2 = w^T C w \tag{1}$$

The goal of PCA is to find the vector w_0 for which this variance is the highest. To achieve this, the Lagrangian multipliers method is applied to V along with the restriction of w being a unit vector.

$$\mathcal{L}(w, \lambda) = V_P(w) + \lambda\big((w^T \cdot w) - 1\big) = w^T C w + \lambda\big((w^T \cdot w) - 1\big) \tag{2}$$

where λ is the Lagrange multiplier. Then, calculating the derivative of $\mathcal{L}$ with respect to w and setting it to zero we get:

$$\frac{\partial \mathcal{L}}{\partial w} = 2Cw - 2\lambda w = 0 \implies Cw = \lambda w \tag{3}$$

which means that w is an eigenvector of S and λ its corresponding eigenvalue. Therefore, the variance of P in the direction of w is

$$V_P(w) = w^T C w = \lambda w^T w = \lambda, \tag{4}$$

so maximizing the variance is equivalent to maximizing the eigenvalue. Then, if we sort the eigenvalues in descending order, we will get a sequence of the corresponding eigenvectors sorted by variance amplitude in the direction they represent.

It remains to choose a number l between 1 and N to project our dataset with respect to the base obtained from $w_1, \ldots, w_l$. In this manner, the user can choose among the eigenvectors for the directions that are the most relevant to the problem in hand.

2.1 Observations

It is important to note that the ways in which the new components are chosen will depend upon the linearly independent eigenvectors obtained from $w_1, \ldots, w_l$, those correspond to the number of different eigenvalues. Therefore, if the number of observations n is less than m, then the rank r of $Cov(P)$ is such that $r < \min(n, m)$. Yielding r different eigenvectors. In those kind of cases, the range of l finds itself reduced.

Another issue has to do with the units of the original space axes. Say the units of measurement change, then the coordinates of P will change and thus the covariance matrix. This means that the eigenvectors obtained once the units are changed will not remain the same, [14].

This method has been subject to a process of evolution and adaptation to many applications,[14], it plays an important role in the most recent dimension reduction methods and is arguably one of the most sought after methods for dimension reduction.

3 Isometric Feature Mapping

The Isometric Feature Mapping (Isomap) algorithm was first introduced in Tenenbaum et al.'s seminal work [10]. It represents a method based on the concept of intrinsic manifold approximation. Its main contribution is the effort at maintaining the distances between points in a lower-dimensional space as closely aligned as possible with the geodesic distances within the intrinsic manifold itself. This innovative approach was initially proposed as a contemporary alternative to two prevalent methods at the time of its inception: PCA and Multidimensional Scaling (MDS) as discussed in Kruskal's influential paper [15]. These methods laid the foundational theoretical groundwork for the development of Isomap.

3.1 Multidimensional Scaling

Back in 1964, when the MDS paper was published [16], the words *loss function* were not commonly used. Regardless, the idea of setting a function as a measure of goodness of fit for point set approximation was well received. In this context, MDS introduced the definition of *stress* as a measurement of similarity between a given set of points P and a proposed set Q. *Stress* is defined as:

$$S(P,Q) = \sqrt{\frac{\sum_{i<j}(d_{ij} - d'_{ij})^2}{\sum_{i<j} d_{ij}^2}}$$

where d_{ij} is the distance between elements x_i and x_j of P and d'_{ij} is the distance between the corresponding points in Q. With stress as the loss function, MDS proposed a point reconfiguration of the points in Q using the method of steepest descent iteratively [16]. A key characteristic of the stress function is that it is independent of the dimension where $P \subset \mathbb{R}^m$ and $Q \subset \mathbb{R}^l$ are found. This raises the opportunity for point set adjustment in multidimensional space. The initial point configuration for Q is then chosen arbitrarily and then adjusted by optimizing stress.

Several observations can be made to the MDS algorithm. First, the algorithm setting does not rest on any assumption of the dimension of neither the space where P nor Q are found, that is, m and l can be equal or different. In fact, it is pointed out that this versatility can be used to search for the dimension in which the stress attains its minimum. Second, the initial configuration of points for Q does not approximate in any way the configuration of P, that is, it effectivity rests on the assumption that stress makes up for unfortunate initial configurations of Q. Third, the stress optimization is equivalent to a least-squares regression. The latter method will subsequently find itself at the core of several dimensionality reduction algorithms.

3.2 Isomap as joint MDS and PCA

It is clear that MDS already introduces an idea of topological approximation, though still diffuse given there is not an analysis *per se* of the preservation of the topological properties along the optimization process. Isomap takes MDS as a basis and adds three mathematical tools [10], which will be further explained below:

IM1 Point connectivity graph G.

IM2 Replace geodesic distances instead of absolute distances

IM3 Calculation of initial Q configuration with PCA of P.

For Point IM1, G is the neighborhood graph built using Floyd's algorithm [17], where for each $p \in P$ a set of neighbors is set using either ϵ-proximity or the k-nearest neighbors. For Point IM2, the Floyd's algorithm builds the distance matrix D of G iteratively, where D_{ij} is the

minimum path distance between vertices $q_i, q_j \in G$ corresponding to x_i and p_j in P. On Point IM3 it calculates an embedding $Q \subset \mathbb{R}^l$ of P, that is, the initial configuration of the points representing P in lower dimensional space. It does so by calculating a vector basis $\{y_i\}_1^l$ for $\mathbb{R}^l$ as $y_i = (\sqrt{\lambda}_1 v_{i1}, \sqrt{\lambda}_2 v_{i2}, ..., \sqrt{\lambda}_p v_{il})$, where $v_i = (v_{i1}, v_{i2}, ..., v_{il})$ and λ_i are the eigenvectors and eigenvalues of matrix

$$\tau(D) = \frac{-HCH}{2}$$

where $C_{ij} = D_{ij}^2$, $H_{ij} = \delta_{ij} - \frac{1}{N}$ with δ_{ij} as the Kronecker delta and N the cardinality of P. H is the *centering matrix* [10], a matrix with mean zero rows and columns.

The latter aggregations of Isomap to MDS made it a widely used tool in the first two decades of this century, in which intrinsic manifold approximation gave rise to several nonlinear methods. An intrinsic manifold $\mathbb{M}$ is the actual manifold where P lies within $\mathbb{R}^m$, or roughly speaking, the manifold of which P is a sample. Point IM2 is where the hypothesis of an intrinsic manifold is found given the geodesic distances are calculated under the assumption that they are calculated within $\mathbb{M}$.

Finallly, for the optimization step, or the adjustment of Q, the cost function of Isomap is given by

$$E = \|\tau(D_P) - \tau(D_Q)\|$$

3.3 Observations

Isomap has several close denominations depending on the methods chosen for each of its steps. For example, if the method for the initial neighborhood graph is chosen to be ϵ-proximity, that is, all points of P within an ϵ neighborhood of each point $x \in P$, then Isomap is called $\epsilon-$Isomap. If it is chosen as the k-nearest neighbors of x, then it is called $k-$Isomap. Additionally, for the calculation of the shortest paths between the points of P, some other algorithms can be used, for example Dijskstra's algorithm [17].

The Isomap paper posits that the optimization process asymptotically approaches a global optimum for the error measure E as the number of observations P increases. This claim is substantiated under the premise that these observations, or samples, are obtained at a sufficiently high density. In the experimental section, we will explore the performance of Isomap on sets P characterized by 'well-behaved' distributions

as well as those that are not. Specifically, when P exhibits distributions that are densely populated in certain regions but sparse in others, the efficacy of Isomap may be surpassed by alternative algorithms.

4 Locally Linear Embedding

The Locally Linear Embedding (LLE) is a dimension reduction method primarily used for high-dimensional data analysis. LLE leverages the local geometric structure of the data to construct a lower-dimensional representation that preserves this structure. We follow the explanation provided by [18].

Let $P = \{x_i\}_1^N \subset \mathbb{R}^m$ and $Q = \{y_i\}_1^N \subset \mathbb{R}^l$, $l \leq m$, be the input and output sets of LLE, respectively, that is Q is an embedding of P. The LLE algorithm consists of three main steps, which are schematically illustrated in Figure 3:

LLE1 Neighbor localization

LLE2 Local point reconstruction

LLE3 Global embedding

In Step LLE1, the k nearest neighbors for each x_i in P are identified. Then in Step LLE2, a weight matrix $W = (w_{ij}) \in \mathbb{M}_{N\times N}$ is computed using the least squares method on the following cost function:

$$\varepsilon(W) = \sum_{i=1}^{N} \left\| x_i - \sum_{j=1}^{k} w_{ij} x_j \right\|^2 \tag{5}$$

Equation 5 is also constrained to two conditions. The first one is

$$\sum_{j=1}^{k} w_{ij} = 1 \tag{6}$$

this constraint is necessary to ensure the reconstruction is invariant to translation transformations. The second is that w_{ij} will be different to 0 only if p_j is neighbor of x_i. The aim is to find the weights that minimize the difference between each data point and the weighted sum of its neighbors. The result is a weight matrix $W = (w_{ij})$ where the local geometric structure of P is captured.

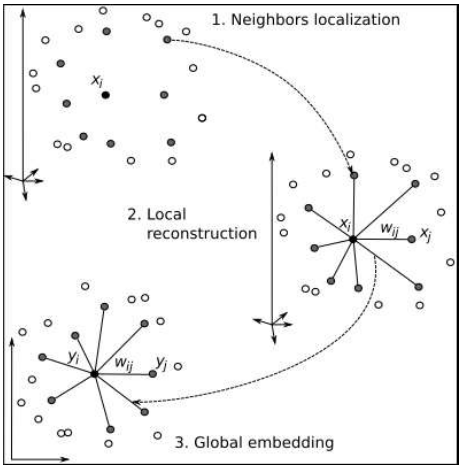


Figure 3: LLE schematic model reduction.

Once the weights have been calculated, a global embedding representation Q of P is computed in LLE3. The aim of the global embedding is to find a low-dimensional representation of P that minimizes the discrepancy between the high and low-dimensional representations measured with the following cost function:

$$\Phi(Q) = \sum_{i=1}^{N} \left\| y_i - \sum_{j=1}^{k} w_{ij} y_j \right\|^2 \tag{7}$$

where y_i is the low-dimensional vector representation of x_i. Also, y_j are the corresponding lower dimensional representations of the neighbors of x_i and w_{ij} their respective weights.

4.1 Observations

LLE has the advantage of not making linear assumptions about the data, given its approach considers the closest points to each point in the input set. That means that the global configuration of P is captured locally in Step LLE1 regardless of how "crooked" or twisted the intrinsic manifold of P is found, allowing LLE to capture non-linear structures.

It is also worth noting that the construction of Q is independent of rotation, translation and scaling of P given the relative conditions for building q from the neighbors of its corresponding p.

Another advantage of LLE is that it only has one hyperparameter. That is, it does not depend on conditions preset by the user other than the number of neighbors k used for building the weight matrix W

The solution to the global embedding optimization problem typically involves solving an eigenvalue problem involving W. More specifically, the data points in the low-dimensional representation Q are precisely the m lower eigenvectors of the quadratic form [18] induced by:

$$\Phi(Y) = \sum_{ij} M_{ij}(y_i \cdot y_j)$$

where $M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki}W_{kj}$. This means that the LLE final embedding is found by traditional linear algebra techniques.

5 t-Stochastic Neighbor Embedding

The t-Distributed Stochastic Neighbor Embedding (t-SNE) technique is a prominent algorithm for visualizing high-dimensional data introduced by Maaten and Hinton in 2008 [19]. This method is grounded in stochastic dimensionality reduction techniques, such as Stochastic Neighbor Embedding (SNE), detailed in Hinton and Roweis [20]. The latter is a member of the non-linear algorithm family and serves as a precursor to t-SNE. As such, notable similarities are evident between these two algorithms, therefore we should begin with a quick exploration of SNE, before approaching t-SNE,

5.1 Stochastic Neighbor Embedding

Let $P = \{x_i\}_{i=1}^N$ be a set of points in m-dimensional space. SNE begins by calculating the probabilities of finding points in P as seen from a given x_i. SNE, [19] supposes that distribution of the points around x_i is Gaussian, and thus sets the following equation for the conditional probabilities for P from x_i:

$$p_{j|i} = \frac{\exp\left(\frac{-\|x_i - x_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\|x_i - x_k\|^2}{2\sigma_i^2}\right)}$$

Also, the value $p_{i|i} = 0$ is set. Suppose now that $Q = \{y_i\}_{i=1}^N$ is the set of points that correspond to those of P in lower dimensional space

such that x_i corresponds to y_i for every i. The distributions of the points of Q as seen from y_i should not be different from the corresponding $p_{j|i}$ calculated in higher dimensional space. In other words, the distributions of P and Q should be as close as possible. In order to redistribute the points for Q, the SNE algorithm sets the value of $q_{j|i}$ as a Gaussian distribution with variance $\sigma^2 = \frac{1}{\sqrt{2}}$:

$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}$$

To search for the σ_i terms for the initial distributions $p_{j|i}$ of each x_i, a binary search is executed using the perplexity as a hyperparameter. Perplexity is defined as:

$$Perp(P_i) = 2^{H(P_i)}$$

where P_i represents the conditional probability distribution over all other datapoints given data point x_i and $H(P_i)$ is the Shannon entropy:

$$H(P_i) = \sum_j p_{j|i} \log_2(p_{j|i})$$

The Shannon entropy is used to make a binary search for the values of σ_i. It is important to note that the values of σ_i do not change over the execution of SNE; they remain static throughout the run of the algorithm.

The initial configuration of Q from which the gradient descent starts is sampled from an isotropic Gaussian with small variance centered at the origin, yielding a set of N clumped points in lower dimensional space.

The next step in the t-SNE algorithm is to search for the optimal configuration of points in Q. The latter is achieved by minimizing the sum of the Kullbak-Leibler divergence [21], which is a measure of how one probability distribution differs from another. In this case, the Kullbak-Leibler divergence between points in P and Q is measured as:

$$C = \sum_i \sum_j p_{j|i} \log\left(\frac{p_{j|i}}{q_{j|i}}\right)$$

The search for the minimum is then made using the gradient descent method, which yields vectors as follows:

$$\frac{dC}{dp_i} = 2\sum_j (p_{j|i} + p_{i|j} - q_{j|i} - q_{i|j})(y_i - y_j)$$

Finally, a gradient update function is added to the optimization process by bringing some terms to the fold:

$$\gamma^{(t)} = \gamma^{(t-1)} + \eta\frac{dC}{d\gamma} + \alpha(t)\left(\gamma^{(t-1)} - \gamma^{(t-2)}\right)$$

These terms are the value of the gradient $\gamma_{(t)}$ at iteration t of the optimization, η the learning rate and $\alpha(t)$ the momentum at iteration t.

5.2 tSNE as readapted SNE

The initial SNE setting poses the problem of the difference between values $p_{j|i}$ and $q_{j|i}$, where the sense is clearly determined by vector $y_i - y_j$, but the direction and norm are determined by the forces exerted by both points with respect to the rest of the elements of P. To address that problem t-SNE proposes the use of joint probabilities for the calculation of the Kullback-Leibler entropy:

$$C = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

where p_{ij} and q_{ij} are calculated as follows:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}$$

where $p_{ii} = q_{ii} = 0$. The latter equations help to adjust each point's contribution to the loss function regardless of their relative position with respect to the rest of the points and also address the crowding problem [19, 22, 23]. In short, the crowding problem refers to the problem that arises when reducing the dimension for points that find themselves in a "crowded" area, that is, that find themselves in an area that is densely sampled. This is due to the exponential growth of volumes depending on the dimension where they are embedded, that growth means that the low dimensional space representation will need a greater surface for Q in the low dimensional space. For example, sets of points that are densely

clustered in two far apart regions will have their low dimension representation distorted for points that are close together. Thus modeling q_{ij} with a t-Student with one degree of freedom distribution will help preserve the relations between clusters by means of their distances and close points by means of their high relative probabilitie. It does so given t-Student distribution has "heavier tails" than the Gaussian, allowing t-SNE to be more effective in separating groups of observations.

Thus, the t-SNE provides a way to visualize high-dimensional data in a low-dimensional space, in such a way that the neighborhood structure of the high-dimensional data is preserved.

5.3 Observations

There are some important details of t-SNE that the developer has to take into consideration, some of them are:

- The complexity of t-SNE is quadratic on the number of points of P. This means that t-SNE does not scale well for large sets.
- The dimension reduction process is unclear for dimensions higher than 3. The efficiency of the reduction to more than 3 dimensions is measured indirectly with the assessment of the final outcome of the total process which uses the dimension reduction method as part of its steps.
- t-SNE does not cope well with the *curse of intrinsic dimensionality*, that stems from the assumption that the distances between points are linear. If P belongs to a too "twisted" manifold in high dimensionality space, the Euclidean distance used for joint probability calculation will not reflect the topological characteristics of P in lower dimensionality space.
- The cost function is not convex, which means that the optimization process can lead to local minima from which it won't be able to escape.

It should be noted that although t-SNE is widely used and can produce very intuitive visualizations, it also has some limitations. For example, non-convex optimization means that different results can be obtained each time the algorithm is run. Additionally, t-SNE does not correctly interpret distances in low-dimensional space , even though it

does a good job at preserving local relationships as it is designed to preserve neighborhood relationships, not absolute distances

6 Uniform Manifold Approximation and Projection

The Uniform Manifold Approximation and Projection (UMAP) technique is a novel method for dimensionality reduction and visualization of high-dimensional data [24]. It is particularly useful when dealing with data that has complex structures in high dimensions, as it can capture both the local and global structure of the data. UMAP has proven to be a powerful tool for data visualization and analysis. It remains an active research topic to fully understand its behavior and properties and how it can be best tailored for different data types and tasks. For this reason, it is advisable for specialists in topology and geometry to be involved in related tasks. Additionally, UMAP is remarkably efficient in computational terms. It can handle both large and high-dimensional datasets. Its Python implementation `umap-learn` is compatible with `scikit-learn`, facilitating its practical use in machine learning *pipelines* [25].

UMAP is based on the idea of functorial optimization over fuzzy topological representations (seen as categories) of a high dimensional space set of points. This is, a series of optimization trials has to be performed over the space of reduced dimension representations of the input until the best representation is found. A full explanation of the concepts on which UMAP is built goes beyond the scope of this text, thus referral to the cited sources is advised.

The UMAP algorithm is composed by the three following steps [24], which are further explained subsequently:

1. Setting local fuzzy simplicial sets
2. Spectral embedding
3. Low dimensional optimization

Step 1 is pivotal in selecting an appropriate family of open covers that are homotopically equivalent. This requires ensuring that the chosen open covers are capable of morphing continuously among themselves. To facilitate this, Step 2 introduces a fuzzy approach. This method is particularly beneficial as it aids in effectively representing the

distribution of open sets within our chosen open cover. The most critical phase is Step 3, where the low-dimensional representation of the input set is determined. This is accomplished through the implementation of stochastic gradient descent, utilizing cross-entropy on the fuzzy sets as the loss function. The process is further refined by the application of a differentiable functor between open covers, enabling continuous variation. While some intricate details are omitted in this overview, it provides a comprehensive guideline for the enthusiastic reader.

6.1 Setting local fuzzy simplicial sets

The algorithm starts by identifying an effective method to build an open cover, which is instrumental in extracting topological information. Given that our primary focus is on point-based sets, it is essential to preserve neighbor relationships in high dimensional space as accurate as possible when transforming to lower-dimensional space. The aim is not merely to redistribute the points of P uniformly across their space, but rather to reinterpret the distances between these points in a way that effectively warps the space, creating an illusion of uniform distribution. The connectivity formed among the elements of this open cover will serve as the foundation for constructing simplicial sets [2].

An initial step involves defining the characteristics of the open cover. Key preliminary considerations include ensuring that the open cover is specifically constructed for the input set P. Also, that it accurately reflects the set's topological properties and is flexible enough to be transformed into similar open covers.

For addressing the first requirement, UMAP selects open neighborhoods N_i for each $p_i \in P$, with the radius determined as the smallest radius that encompasses the first k neighbors. It is crucial to note that k is a hyperparameter predetermined by the user. For every $p_i \in P$ and its first k-neighbors $\{q_{i,j}\}_{j=1}^{k}$, distances are calculated as follows:

$$d_{i,j} = \frac{\max(0, \text{dist}(p_i, q_{i,j}) - \nu_i)}{\sigma_i}$$

[2] A simplicial set X is a collection of sets $X_0, X_1, X_2, \ldots$ together with maps $\delta_i : X_n \to X_{n-1}$ and $s_i : X_n \to X_{n+1}$, $0 \leq i \leq n$, that fullfill the degeneracy and face operators conditions respectively. In other words, a simplicial set is a categorical model capturing those topological spaces that can be built up from simplices. Simplices can be viewed as generalizations of triangles for any dimension, [26].

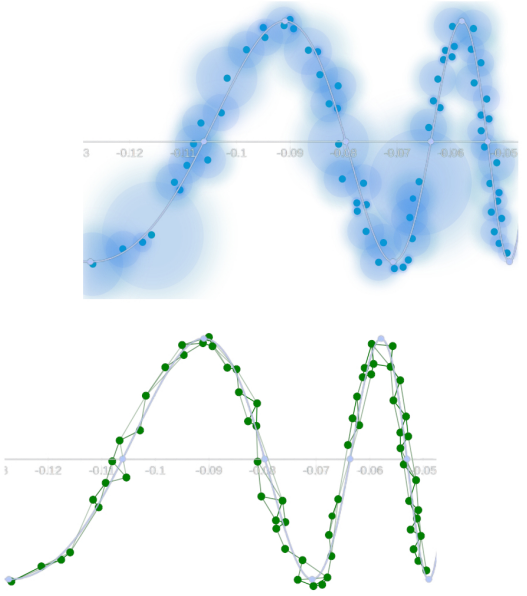


Figure 4: To the left, the open cover obtained using the fuzzy distance. To the right the simplicial complex of the 1-simplices from sets s_i.

where ν_i is the distance to the nearest neighbor of p_i, and σ_i is a smoothing term defined such that:

$$\sum_{j=1}^{k} \exp\left(\frac{d_{i,j}}{\sigma_i}\right) = \log_2(n)$$

These distances introduce a fuzzy local distance for each p_i and are used as weights for the simplicial set $S = \{s_i\}_{i=1}^{n}$, where every simplex $s_i = \{([p_i, q_{i,j}], \exp(-d_{i,j}))\}_{j=1}^{k}$ intersects at least another one. This property will be used to establish the initial embedding for P. A graphical representation of the fuzzy open sets and the corresponding simplicial set can be found in Figure 4.

6.2 Spectral embedding

Upon completing the step just described, S constitutes a $1-$skeleton, meaning the simplices s_i are confined to dimension of at most 1. The subsequent phase involves the construction of an initial embedding, employing spectral embedding techniques on S. This approach presup-

poses that S is embedded within a low-dimensional manifold in a higher-dimensional space, it aims at minimizing the expected squared distance between connected nodes. To achieve this, the weighted adjacency matrix A is calculated as

$$A = B + B^T - B \circ B^T$$

where B is the weighted adjacency matrix of the directed graph G obtained from S and $\circ$ is elementwise multiplication. In this sense, A becomes a symmetric matrix used to compute the sorted eigenvectors of

$$L = D^{\frac{1}{2}}(D - A)D^{\frac{1}{2}}$$

where D is the degree matrix and $D - A$ the Laplacian matrix. The sorted eigenvectors are then used for calculating the initial coordinates for each p_i.

The choice of A is due to the differences on the weights between $s_i = ([p_i, q_{i,j}], \exp(-d_{i,j}))$ and $s_l = ([p_l, q_{l,m}], \exp(-d_{l,m}))$ where $p_i = q_{l,m}$ and $p_l = q_{i,j}$, yet their distances or weights differ, i.e., $\exp(-d_{i,j}) \neq \exp(-d_{l,m})$. In this scenario, UMAP replaces both s_i and s_j with a single simplex $s = ([p_i, p_l], \exp(-d_{i,j}) + \exp(-d_{i,j}) - \exp(-d_{i,j} - d_{i,j}))$ on matrix A. This modification in distance measurement is the spectral representation of the points in P. The resulting set of simplices is designed to preserve the local topological properties of its low-dimensional representation. The ensuing step involves adjusting the low-dimensional representation T of S, derived from the spectral embedding, in accordance with a specific loss function.

6.3 Low dimensional optimization

The next step on the algorithm consists of varying the distances or weights of the elements of T in order to find the best choice of distances in lower dimensional space within certain constraints. These constraints come in the form of hyperparameters for the general optimization process. The first step is to determine the function to be optimized, that is, the loss function. In the case of UMAP, *cross entropy* has been used given its resemblance to force-directed layout algorithms. The cross entropy function in question is the following:

$$\sum_{s \in S'} \left(w_h(s) \log\left(\frac{w_h(s)}{w_l(s)}\right) - \Big(1 - w_h(s)\Big) \log\left(\frac{1 - w_h(s)}{1 - w_l(s)}\right) \right)$$

Where w_h and w_l are the weights in higher and lower dimensionality spaces respectively. The starting point of the optimization is the spectral embedding T made in the previous section. Then stochastic gradient descent is used in n epochs.

In the preceding formula, the first term is expected to reflect an attractive force. That is, whenever w_l is bigger than w_h, the whole term reduces the entropy. An analogous analysis could be made to the second term, which provides a repulsive force to the equation. In consequence, the optimization process will search for an equilibrium between these two forces.

6.4 Some observations on UMAP

It is important to consider that there are several free parameters that will certainly make a difference on the results yielded by UMAP. Those are:

1. k: The number of neighbors for calculating S.

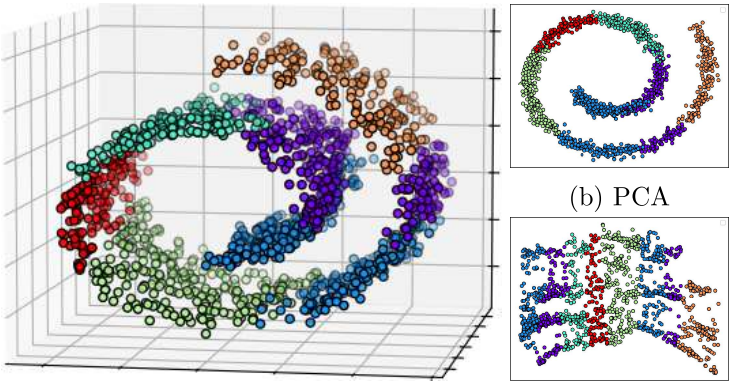
2. d: The target embedding dimension.

3. $min - dist$: The desired separation between close points of P, the original coordinates.

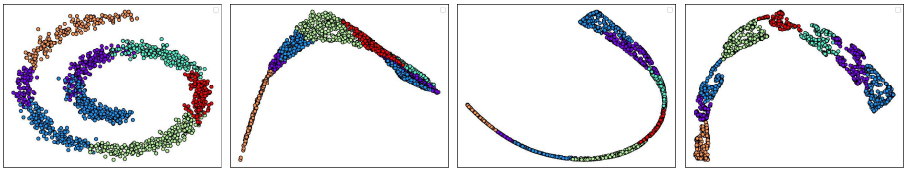
4. n: The number of epochs used on the optimization.

The foundational theory that justifies the validity of the process lies within category theory. In brief, the optimization process is searching the best representation for S' within a space of categories representing the Cech complex of S', which is also the skeleton and the 1-simplex structureThat makes all the elements of the category homotopically equivalent, according to the Nerve theorem [27]. In other words, the optimization process operates as a functor joining categories in its domain and codomain which correspond to the original embedding space, or high dimensionality space, and the lower dimensionality space.

No mention has been made when talking about the sampling used on the optimization process. Also, the final point distribution on the display followed by UMAP is the product of attraction and repulsion forces in the cross entropy function.



(a) Original Swiss Roll set in $\mathbb{R}^3$ (b) PCA (c) Isomap



(d) MDS (e) LLE (f) *t*-SNE (g) UMAP

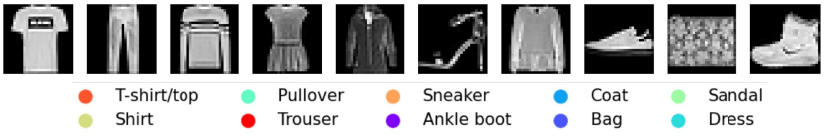
Figure 5: Result of processing the Swiss Roll set with each of the methods described in this article

7 Use case examples

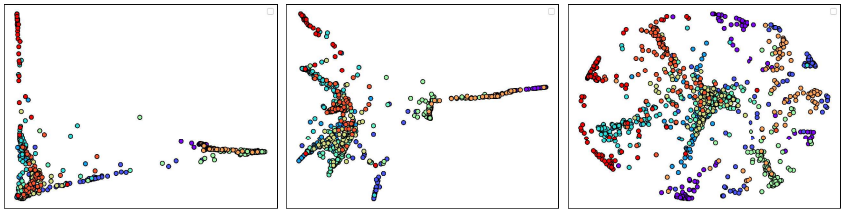
In this section we show some examples of applying the methods described in previous sections. To better illustrate them, two different sets of points were chosen from two well known sources [28, 29]

The first one is known as the Swiss Roll set, depicted in Figure 5a [28]. It is a set that is bounded in $\mathbb{R}^3$ and poses a non-linear problem for dimension reduction. This set has had gaussian noise with a 0.5 standard deviation added to it. The reason for this is to better depict the different outputs for a 2-dimensional space on each of the methods, the result of each of which is presented in Figure 5. The differences are apparent on each of the cases and they depict a rather intuitive approach of what each of the methods do to a 3-dimensional set layout that is easy to grasp.

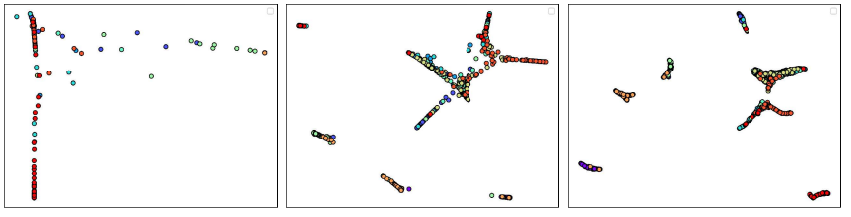
The second set, called Fashion-MNIST [29], is a set composed of 28×28 gray scale labeled images of $70,000$ fashion products from 10 categories, with $7,000$ images per category. Figure 6a shows some examples



(a) Class examples from Fashion MNIST database along with color code. Colors chosen are for depiction purposes only.



(b) PCA (c) Isomap (d) MDS



(e) LLE (f) t-SNE (g) UMAP

Figure 6: Some Fashion-MNIST image examples for each class of clothing, and the result of processing the dataset with the methods described in this article.

of the images in Fashion-MNIST, downloaded from [30]. For depiction ease, 1000 images were uniformly sampled and used to train a neural network, refer to Appendix A for a further description of the neural network architecture. The product of the neural network is a 10-dimensional embedding space representation for each of the images. The data points were then submitted to all of the methods mentioned above, Figure 6. Though it is not possible to look at the data in high-dimensional space, the product of the dimension reduction methods helps at visualizing how the data can be represented in 2 dimensions.

8 Conclusions

Throughout this work, we presented several important works on dimension reduction that strongly contribute to recent technological developments. The algorithms described in this compendium are historically significant in that they present the general theory and mixed techniques that are the state of the art. The choice of the technique will always depend on the characteristics of each problem, available resources and the business model. Therefore, there is not a best choice or one-solution-fits-all technique, this will be determined by the use case in specific fields of application.

The field of dimension reduction research is closely related to AI, hence its recent surge. Nevertheless, dimension reduction is not the only contribution of these algorithms, one could argue that manifold approximation methods developed for embedding the input point set play a fundamental role in these algorithms. The quality of the initial manifold approximation is what yields a good final interpretation of the problem at hand.

There is also an elephant in the room that we have not mentioned and that is the choice of hyper-parameters for each of the previously presented algorithms. The value of which plays a determinant role on the results obtained. Hyperparameters are mainly set by an optimization process chosen from a wide variety of methods available in the literature. This process composes itself with the analogous step of the embedding model. That is, the input data has to be modeled in at least two steps: 1) AI model hyperparameter optimization, 2) dimension reduction hyperparameter optimization. Each of these steps go through an optimization process that makes some field specialists weary of the trustworthiness of the results because of the apparent difference when changing hyperparameter values.

Finally, we found that PCA has proven to be an almost unavoidable method for feature selection. The theoretical justifications of PCA make it a building block for other methods. In particular, the combination of PCA and spectral analysis extends the reach of manifold approximation methods. This is due the ease of representation of the vicinities of the points of the input set. We think that the dimension reduction algorithms developed in the near future will use spectral analysis along with PCA as the basis for manifold approximation methods, given the granularity it allows to achieve.

Acronyms used

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[2] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022.

[3] Richard Bellman. *Adaptive control processes: a guided tour*. Princeton University Press, 1961.

[4] Allen Hatcher. *Algebraic topology*. Cambridge University Press, Cambridge, UK, 2002.

[5] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.

[6] Nasir Ahmed, T Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3111–3119, 2013.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[10] Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, December 2000.

[11] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[12] Markus Ringnér. What is principal component analysis? *Nature biotechnology*, 26(3):303–304, 2008.

[13] Ian Jolliffe and Jorge Cadima. *Principal Component Analysis*. Springer, 2016.

[14] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, April 2016.

[15] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, March 1964.

[16] J. B. Kruskal. Nonmetric multidimensional scaling: A numerical method. *Psychometrika*, 29(2):115–129, June 1964.

[17] Thomas Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, Cambridge (Mass.), 2nd. ed edition, 2001.

[18] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[20] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. *Advances in Neural Information Processing Systems*, 15:857–864, 2002.

[21] Solomon Kullback. *Information Theory and Statistics*. Dover Publications, 1959.

[22] James Cook, Ilya Sutskever, Andriy Mnih, and Geoffrey Hinton. Visualizing similarity data with a mixture of maps. In *Artificial intelligence and statistics*, pages 67–74. Proceedings of Machine Learning Research, 2007.

[23] Kilian Q. Weinberger and Lawrence K. Saul. Unsupervised Learning of Image Manifolds by Semidefinite Programming. *International Journal of Computer Vision*, 70(1):77–90, October 2006.

[24] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[26] W.G. Dwyer and J. Spalinski. Chapter 2 - homotopy theories and model categories. In I.M. JAMES, editor, *Handbook of Algebraic Topology*, pages 73–126. North-Holland, Amsterdam, 1995.

[27] Jon Peter May. *Simplicial objects in algebraic topology*. Chicago lectures in mathematics. Univ. of Chicago Pr, Chicago, 1992.

[28] Multiple contributors. Sklearn Datasets Swiss Roll.

[29] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. 2017. Publisher: arXiv Version Number: 2.

[30] Keras.

[31] Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. Fundamental Concepts of Convolutional Neural Network. In Valentina E. Balas, Raghvendra Kumar, and Rajshree Srivastava, editors, *Recent Trends and Advances in Artificial Intelligence and Internet of Things*, volume 172, pages 519–567. Springer International Publishing, Cham, 2020. Series Title: Intelligent Systems Reference Library.

Appendices

A Neural network architecture

This is a description of the architecture used to generate the embedding for the Fashion MNIST [29] labeled dataset described in section 7. The code was written using Python programming language along with the Keras library [30]. The architecture chosen for this particular example is a sequential convolutional neural network [31] with the following features:

- Input convolutional layer with 32 filters size 3×3 and *relu* activation.
- Convolutional layer with 64 filters of size 3×3 and *relu* activation.
- Max pooling layer with 3×3 window size.
- Convolutional layer with 32 filters of size 3×3 and *relu* activation.
- Max pooling layer with 3×3 window size.
- Flatten layer.
- Dense layer with 4096 units and *relu* activation.

- Dropout layer of 0.5.
- Dense layer with 10 units and $softmax$ activation.

The loss function chosen for the training was categorical cross-entropy using the Adadelta optimizer [30] and accuracy was chosen for the final assessment. Additionally, 50 epochs were performed with batches of size 32.

It is important to point out that the neural network architecture chosen is a typical convolutional neural network.