Review over further developments of the PCA algorithm: classical and modern perspectives

Corina Dimitriu
Faculty of Computer Science
"Alexandru Ioan Cuza" University of Iași
Iași, Romania
corina.dimitriu.01@gmail.com

***Abstract*—This paper investigates the evolution in dimensionality reduction algorithms' family, from both theoretical and experimental perspective. Taken PCA's mathematical foundation as source of inspiration for performance and comparison, further proposed solutions for dimensionality reduction and data analysis, such as UMAP and t-SNE, have taken the competitive lead in fruitful advancements of the target research field. Even though these two algorithms have been designed for vizualization and dimensionality reduction purposes, they quickly diverge from PCA's interpretative support for feature selection and focus on clusterization and pattern recognition tasks, while still providing data compression facilities. Taking the comparison between t-SNE and UMAP aside, the former aims at higher fidelity, whereas the latter undergoes a trade-off in favor of scalability.**

***Index Terms*—PCA, UMAP, SNE, t-SNE, dimensionality reduction, vizualization, data analysis**

I. Introduction

On the premise of producing desirable results, complex machine learning and statistical tasks often require a thorough data processing step, which ultimately relies on performing a qualitative data analysis, both manually and automatically, depending on the use case. It is quite straightforward that data vizualization and dependencies removal, as well as broader automatization and reducing computational speed, all constitute a serious objective for many data science related optimization systems. As it is the case of dimensionality reduction — which is the main subject for optimization treated as purpose of the algorithms discussed in this paper — the data processing stage is filled with both limitations and benefits, as it should paradoxically and ideally preserve as much as possible of the original structure while removing a high number of variables.

For the rest of this paper, the PCA algorithm — having its very early origins before the Second World War — is assumed to be already known and only some key concepts defining it are highlighted [9]. The major drawback in the PCA algorithm is that we get the most of it in quite particular scenarios, under the heavy assumption that the resulted first two principal components should account for the vast majority of variance in the initial data. The two more modern approaches which followed PCA (and other techniques which filled the gap in between) were developed from the seed that a lower-dimensional space can be embedded within a higher-dimensional space. As a result, they seek for means — while reasoning in accordance with mathematical, topological and physical principles — to perform an efficient search towards an approximation of one, as optimum as possible, lower-dimensional space. The idea of translating a set of points from a high-dimensional space into a lower-dimensional modeled system that t-SNE and UMAP rely on, expressed for now as the core optimization from a high-dimensional clusterized input to a low-dimensional output preserving the initial clusterization of data, is depicted in Fig. 1.

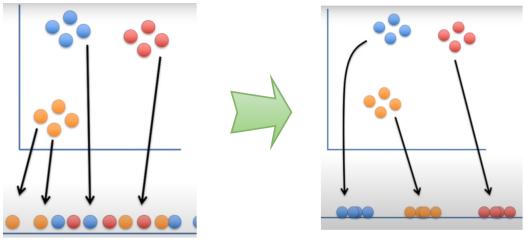


Fig. 1: General view over data translation from higher-dimensional to lower-dimensional space [5, 6].

Under the well justified assumption that the ideal modeling does not exist, each algorithm grows on the back of the others' drawbacks and missing support for certain practical applications; due to this premise, the paper includes a comparison section. Nevertheless, analyzed data always has its own spatial properties, particular distribution and ambiguous dependencies, which makes the current paper indirectly address three main topical issues:

- Why are there more algorithms out there addressing the same problem of dimensionality reduction?
- What are the strengths and drawbacks of each particular algorithm?
- Can we conclude which algorithm is the best, among the three discussed in the comparisons section?

The paper logic is structured as follows. Sections II, III and IV detail the three modern algorithms which followed the PCA algorithm: SNE, its improved and more widely used variant, t-SNE, and UMAP. Section V presents a suite of theoretical and experimental (pairwise and grouped) comparisons between the PCA algorithm, the t-SNE algorithm and the UMAP algorithm. Finally, Section VI presents the conclusions which derive from the full track being followed in the previous sections; further directions are also briefly highlighted.

II. SNE ALGORITHM

Stochastic Neighbor Embedding (SNE) was founded on the premise of inferring meaningful similarities between paired data objects, for the reason that these similarities can be further expressed in a probabilistic manner [1]. Rather than making use of collections of vectors depicting the whole high-dimensional search space, SNE relies on pairwise similarity metrics in order to summarize variance and decide over the new vizualization system. As it naturally follows, we can represent the similarity of one data object, let's call it x_j, to another data object, let it be called x_i, through a conditional probability $p_{j|i}$; semantically, this probability denotes the chances that x_j could be chosen as neighbor for x_i, all interpretation being settled under the assumption of a probability density function for the Gaussian distribution having x_i as the value of its mean and a reasonable variance σ_i. Obviously, for close data objects (datapoints) the value of the conditional probability is significantly higher, whereas for data objects far away from the x_i objective (with higher ϵ value if applying the reparameterization trick given the distribution) the probability is infinitesimally small [1]. The mathematical equation of the probability is visibly derived from the Gaussian distribution's equation:

$$p_{j|i} = \frac{\exp\left(\frac{-||x_i - x_j||^2}{2\cdot\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(\frac{-||x_i - x_k||^2}{2\cdot\sigma_i^2}\right)} \tag{1}$$

where $p_{i|i}$ is modeled as 0.

The way the variance σ_i is selected has to do with the agglomerative properties illustrated by the dataset. In other words, a smaller value is suitable for denser subregions in space and a higher value is more appropriate for the sparse ones. This criterion makes even more sense if we take into account the logical relationship between variance and entropy: as the variance increases, the entropy increases as well. Consequently, in terms of algorithmic computation, the variance results from implementing a binary search algorithm over possible outcomes for a given perplexity. The perplexity measure has the following form and is fixed by the user:

$$Perp(P_i) = 2^{H(P_i)}, \tag{2}$$

where $H(P_i)$ corresponds to the Shannon entropy measured for the distribution probability induced by σ_i:

$$H(P_i) = -\sum_j p_{j|i}\log_2 p_{j|i}. \tag{3}$$

As the variance increases, the perplexity, viewed as a smooth version of the neighborhood cardinal, also monotonically increases, giving the user the commonly known range between 5 and 50 for perplexity choices.

Since we are interested in modeling a lower dimensional vizualization system, let's denote the components corresponding to x_i and x_j, in this newly proposed system, by y_i and y_j. Therefore, we compute another conditional probability funtion, $q_{j|i}$, with respect to a Gaussian distribution centered in y_i with $\frac{1}{\sqrt{2}}$ variance:

$$q_{j|i} = \frac{\exp\left(-||y_i - y_j||^2\right)}{\sum_{k\neq i}\exp\left(-||y_i - y_k||^2\right)}. \tag{4}$$

It is important to note that a different variance is employed for each Gaussian distribution (which corresponds to actually representing the neighborhood of each datapoint) in the original model; however, as far as the rescaled model is concerned, there is no conclusiveness regarding this property, due to the fact that a single distribution is being followed. Since each datapoint in the original model uses a different variance, embeddings of all datapoints from the rescaled model into the original one normally have to follow the same property; therefore, non-compliance with this rule implies that the data is not a perfect representation of itself anymore.

A correct and complete modeling system would result in equal conditional probabilities, $p_{j|i}$ and $q_{j|i}$. This already gives an insight into the objective/cost function the SNE algorithm proposes for the low-dimensional model, which aims at minimizing the discrepancy between $p_{j|i}$ and $q_{j|i}$, with the help of the Kullback-Leibler divergence. The cost function takes into account all neighborhoods, "surrounding" every data object, and targets each conditional probability, for all the other data objects except the one defining the neighborhood (by convention and for the sake of simplicity, we consider that $\log 0 = 0$):

$$C = \sum_i KL(P_i||Q_i) = \sum_i\sum_j p_{j|i}\log\frac{p_{j|i}}{q_{j|i}}. \tag{5}$$

The practical value of using this cost function given the algorithm's main target comes from the asymmetry of the Kullback-Leibler divergence. To be more specific, the cost involved by a rescaled mapping which widely separates data objects being originally close to each other, in terms of similarity, is large enough, whilst the cost involved by mapping widely separated data objects through nearby points is pretty small. This makes the algorithm concentrate more on properly defining the local structure of data rather than building a general overview of its dependencies.

The SNE algorithm executes stochastic gradient descent in order to perform minimization of the target function described above. Here is the formula for the cost's gradient wth respect to the y_i component:

$$\frac{\delta C}{\delta y_i} = 2\cdot\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j). \tag{6}$$

As far as the interpretation of the gradients is concerned, the power held by each attraction pool, depending on y_i and y_j respectively, is proportional to the distance between the two mappings of the data objects and to the discrepancy between the pairwise similarities they determine. The initialization stage consists in sampling random points from an isotropic Gaussian distribution centered around the (cartesian) origin with small variance. Additionally, the gradient descent formula

takes into account scaling according to previous gradients' optimization performance as well, by employing a momentum term at its end:

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \mu \cdot \frac{\delta C}{\delta \mathcal{Y}^{(t-1)}} + \alpha(t) \cdot \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right), \tag{7}$$

where $\mathcal{Y}^{(t)}$ stands for the solution corresponding to the tth iteration, μ stands for the learning rate and $\alpha(t)$ denotes the momentum ratio. Nonetheless, especially during the first iterations, Gaussian noise is sampled and added to the mappings, in order to escape local optima by creating a simulated annealing type of environment for the optimization process.

III. t-SNE ALGORITHM

Although the base SNE algorithm demonstrates fairly good performance on experimental benchmarks, the cost function it optimizes, for the purpose of lower dimensional space conversion, determines several problems; apart from the significant workload, it deals with a phenomenon known as the "crowding problem". The succesor of SNE is called "t-Distributed Stochastic Neighbor Embedding" (*t*-SNE) and proposes two major improvements, consisting of optimizing a different cost function, with less heavy gradients, and replacing the use of a Gaussian distribution with the employment of the heavy tailed Student-t distribution.

A. Symmetric SNE

The improved cost function proposed by the t-SNE algorithm still relies on the background of the Kullback-Leibler divergence properties, but changes the target of individual minimization, which is supposed to take care of each and every probability distribution of neighborhood, finally gathered into a sum, to a target of joint minimization requiring joint probability distributions as arguments [1]:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{8}$$

This cost function is symmetric, as already implied by the title, which means that each probability, p_{ij}, is equal to its symmetrically denoted probability, p_{ji}, no matter the i and j values. Inspiration taken from Gaussian distribution is therefore reshaped to account for the joint probabilities p_{ij} and q_{ij}:

$$p_{ij} = \frac{\exp\left(\frac{-||x_i - x_j||^2}{2 \cdot \sigma_i^2}\right)}{\sum_{k \neq l} \exp\left(\frac{-||x_k - x_l||^2}{2 \cdot \sigma_i^2}\right)} \tag{9}$$

$$q_{ij} = \frac{\exp\left(-||y_i - y_j||^2\right)}{\sum_{k \neq l} \exp\left(-||y_k - y_l||^2\right)}. \tag{10}$$

However, this cost function is problematic in terms of outliers' influence, since for such outlier datapoints the joint probability in both models is too small to have a significant impact on the cost function, even though these outliers often constitute a topical issue, subject to detection in the data analysis process; consequently, there will be (distant) points which would almost not matter at all when establishing the position of one specific point. The solution to the presented problem comes in the shape of a slightly modified joint probability density function for the original datapoints:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2 \cdot n}, \tag{11}$$

which assigns a minimum degree of importance to every point in the dataset by ensuring that $\sum_j p_{ij} > \frac{1}{2 \cdot n}, \forall i$. The equation for computing the similarities in the low-dimensional space is preserved as it is depicted by (10).

One reason for employing this whole different similarity computation procedure refers to the gradients' simpler form, which makes the overall algorithm benefit from better time performance [1]:

$$\frac{\delta C}{\delta y_i} = 4 \cdot \sum_j (p_{ij} - q_{ij})(y_i - y_j). \tag{12}$$

B. The "crowding problem"

According to basic and intuitive mathematical knowledge, it is completely feasible to represent 10 mutually equidistant datapoints in a space consisting of 11 dimensions, but what if the low-dimensional model results in having only 2 or 3 dimensions [1]? The new low-dimensional representation would definitely not be trustworthy anymore. Related to this issue, the entropy among the pairwise similarity distributions would be very high, meaning too much diversity in the mapping. Intuitively, the ratio between the attraction pool encompassed by two moderately distant points and the one covered by two really close points would be too small as well, which determines a conglomerate of "crowded" and inaccurate subregions to be represented in the final mapping. Therefore, too many moderately distant points — which are also likely to be the most frequent ones since they commonly tend to produce similarities close to the average distance — will attract a lot of mappings in their gravity center (or, the center of the final map) and, unfortunately, diminish the visibility of the natural clusters' configuration.

C. "Mismatched Tails can Compensate for Mismatched Dimensionalities"

As the main improvement of the base SNE algorithm, the introduction of the Student t-distribution in the computing similarity scores stage, naturally resolves the main problem the base algoritm used to have, specifically the tendency of the rescaled model to distort the original clusterization of data, which was briefly described in the previous section under the name of "crowding problem" [1]. Since the symmetric cost function we introduced earlier pairs joint probabilities coming from the original and the rescaled model, respectively, instead of actual distances between data objects, the high dimensional space might keep the Gaussian distribution to compute probabilities, but the low dimensional space needs to increase the projection of the original distance on the rescaled model between moderately distant points, which genuinely translates as the need for a more heavily tailed distribution. The Student t-distribution assigns larger values to originally

average like similarities and transforms the rescaled joint probabilities formula as follows:

$$q_{ij} = \frac{\left(1 + ||y_i - y_j||^2\right)^{-1}}{\sum_{k \neq l} \left(1 + ||y_k - y_l||^2\right)^{-1}}. \quad (13)$$

Particularly, the above formula applies to all current t-SNE experimental setups, as it uses the Cauchy distribution which is equivalent to the Student t-distribution with one degree of freedom and has the very useful property that far away, big clusters interact with each other in the same way the individual points composing them would do. This way, the optimization process keeps a regular pace and operate systematically for both local and global optimization as the algorithm advances [Algorithm 1].

Algorithm 1: Pseudocode of the t-SNE algorithm [1]. Regarding notations used, T stands for a previously decided maximum number of iterations.

Register $X = \{x_1, x_2, x_3, ..., x_n\}$ dataset;
Choose $Perp$ perplexity metric;
compute pairwise affinities $p_{j|i}$ using $Perp$ and (1);
$p_{i|j} \leftarrow \frac{p_{j|i} + p_{i|j}}{2 \cdot n}$;
$Y^{(0)} = \{y_1, y_2, ..., y_n\} \leftarrow \mathcal{N}(0, 10^{-4} I)$ initial solution;
for *t = 1 to T* **do**
    compute low-dimensional affinities $q_{i|j}$ using (10);
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ using (12);
    update $Y^{(t)}$ using (7);
end
return $Y^{(T)}$

Ultimately, this section provides a suite of images depicting a very basic example of translating a set of 2-dimensional points into a set of 1-dimensional points preserving the original clusterization as faithfully as possible [Fig. 2]. This example was retrieved — as an excerpt from a video — from a presentation on the topic made at university.

IV. UMAP ALGORITHM

UMAP (Uniform Manifold Approximation and Projection) finds its strength in a complex mathematical foundation, involving Riemannian geometry and manifold theory, as well as topological algebraic analysis [2, Fig. 3]. Firstly, the data is assumed to lie on a (n-dimensional, where n is the number of features) manifold which is to be approximated by the algorithm. The manifold as geometric structure — we can think of it as resembling the Earth — was chosen due to its property of embedding lower-dimensional spaces. To simplify things, we only add to the previously made assumption that the manifold has a Riemannian metric on it and that the manifold is locally connected[1]. Informally, that relates to the use of fuzzy simplicial sets during the course of the algorithm. The algorithm, seen as a whole, optimizes the data layout

[1]A locally connected manifold allows the slicing of data into local neighborhoods.

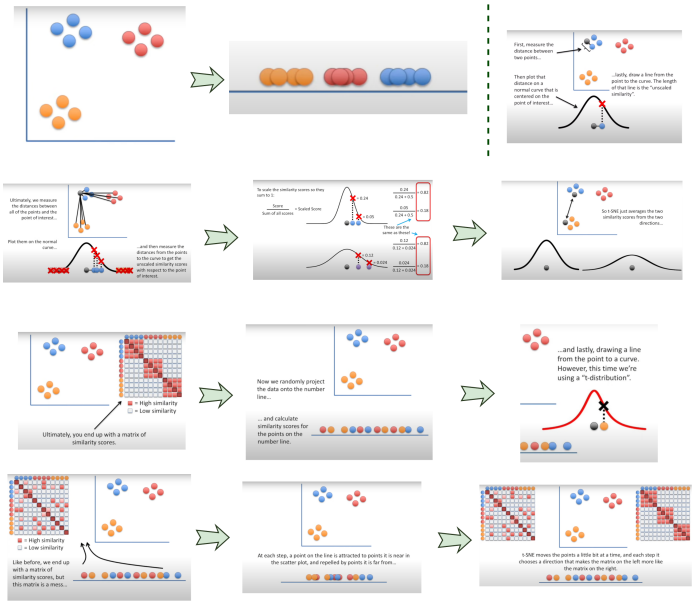


Fig. 2: Toy example for understanding the t-SNE algorithm [5].

in the low-dimensional space by minimizing the difference between two topological representations slicing the data in neighborhood regions, which should already lead thoughts to a strong association with graph theory [2].

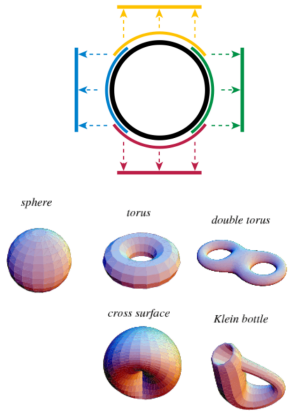


Fig. 3: The way manifolds function [8].

For the purpose of this paper, we keep the deeper rest of the mathematical background aside and further focus on the computational track of the algorithm. Given the approximation of the manifold on which the data is uniformly distributed, it is one of the main premises in the target vizualitaion field that this manifold's topological structure has to be preserved as much as possible by the new representation model the algorithm constructs. In the long run, a weighted k-neighbor graph is built upon the manifold's topological representation and a suite of processing steps is applied to it; afterwards, the new representation is forced to preserve the characteristics

of this graph, which ultimately translates into minimizing an objective function embedding them. These two steps are commonly known in dedicated literature as graph construction and graph layout.

Let's denote the dataset as $X = \{x_1, x_2, ..., x_N\}$, having a dissimilarity measure $d : X \times X \to \mathbb{R}_{\geq 0}$.[2]. Considering the definition of d and a hyperparameter k, a k-neighborhood $x_{i_1}, ..., x_{i_k}$ is set to be revolving around each x_i. The k-neighborhood mapping operation might be accomplished by a k-nearest neighbors (i.e. k-NN) algorithm, which benefits from a practically unexisting training step. Each data object x_i associates itself with two metrics, ρ_i and σ_i which satisfy:

$$\rho_i = \min\{d(x_i, x_{i_j}) | 1 \leq j \leq k, d(x_i, x_{i_j}) > 0\}, \tag{14}$$

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2 k. \tag{15}$$

Intuitively, σ_i can be looked at as a normalisation factor and the origins of its formula lie behind Riemannian geometry interpretations. Naturally following, a weighted directed graph will be represented as:

$$\bar{G} = (X, E, \omega), \tag{16}$$

with the edges set naturally defined as:

$$E = \{(x_i, x_{i_j}) | 1 \leq j \leq k, 1 \leq i \leq N\} \tag{17}$$

and the following σ-related weight function:

$$\omega((x_i, x_j)) = \exp\left(\frac{-\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right). \tag{18}$$

The term "directed" used for describing the graph already suggests that the above mentioned dissimilarity measure induces asymmetric neighborhood relations between data objects, which raises a couple of topology similarity issues requiring to be solved. To be specific, similar to the motivation of employing the symmetric SNE within the t-SNE algorithm — in place of the cost function used within base SNE algorithm — there are several irregularities deriving from assymetry in making analogies to clusterization.

Diving deeper, obtaining the weighted graph turns on the creation of the fuzzy simplicial set, dedicated to the local structure centered around x_i. The need for such a fuzzy set, composed of 1-simplex shaped connections, is justified by the universality of its properties, applying to local neighborhoods with distinct metrics and thus allowing their blending. The metric of the simplicial fuzzy set is contained within a combination deriving from (14) and (15).

In an attempt to transform G into an undirected graph in order to solve the above mentioned topology related problems, denoting the weighted adjacency matrix of the initial graph as A allows writing the following definition of a new symmetric adjaceny matrix [2]:

$$B = A + A^T - A \circ A^T \tag{19}$$

[2]The notion of similarity measure was defined and exemplified in the SNE algorithm section. The notion of dissimilarity measure immediately follows.

where $\circ$ stands for the Hadamard (i.e. pointwise) product. Taking into account that A_{ij} is associated with the existence of an edge pointing from x_i to x_j, B_{ij} would denote nothing else than the probability that at least one of the initially assymetric connections — pointing from x_i to x_j and from x_j to x_i — exists. This is the moment when the graph construction stage, which takes place in the high-dimensional space, might be considered complete. As far as the use of the manifold is concerned, any ball having x_i as its center and containing exactly k neighbors of x_i has to measure an approximately equal volume, no matter the choice of i, making the manifold the actual reason why simplicial sets had to be employed for blending different measures of distances, characteristic to each x_i aside [2].

Going further, the graph layout phase implies borrowing the action of two opposite forces from essential physics principles: attraction and repulsion, causing the effective clusterization of points [5, 6]. The attraction is assigned to the edges and the repulsion belongs to the vertices. Consequently, the algorithm falls into the category of solving non-convex optimization problems, with the following formula for defining the attractive force:

$$\frac{-2 \cdot a \cdot b \cdot ||y_i - y_j||_2^{2 \cdot (b-1)}}{1 + ||y_i - y_j||_2^2} \cdot \omega((x_i, x_j)) \cdot (y_i - y_j) \tag{20}$$

having a, b as hyperparameters.

Due to limited computational resources, each repulsive force can be quickly sampled if having all attractive forces already decided, but the landmark repulsive forces formula can be written as:

$$\frac{2 \cdot b}{(\epsilon + ||y_i - y_j||_2^2) \cdot (1 + a \cdot ||y_i - y_j||_2^{2 \cdot b})} \cdot (1 - \omega((x_i, x_j))) \cdot (y_i - y_j) \tag{21}$$

where ϵ helps avoiding errors caused by any attempt to divide by zero.

However, a smooth approximation of the attraction force — also called the neighbor cost — between two points is used and the repulsive force — also called the non-neighbor cost — is therefore considered to be the complementary with respect to 1.0:

$$\phi(x, y) = \left(1 + a \cdot (||x - y||_2^2)^b\right)^{-1}, \tag{22}$$

$$C = \psi(\log(\phi) + \log(1 - \phi)). \tag{23}$$

where ψ marks an additional processing to the cost function in order for it to fit the minimization purpose.

Like (t-)SNE, for decreasing the attraction and increasing the repulsion until obtaining a viable low-dimensional representation, the UMAP algorithm applies stochastic gradient descent. The notable differences arise from the gradients' formulas, which correspond to the forces' formulas previously described, as the objective function is chosen to resemble the cross-entropy between the topological representation depicted by the graph G and a low-dimensional topological representation, associated to a new weighted graph — let's denote it

by H — made out of complementary datapoints denoted as $y_1, ..., y_n$ in the formulas [Algorithm 2].

Algorithm 2: Pseudocode focusing on the UMAP algorithm's embedding optimization step [2]. Regarding notations used, T stands for a previously decided maximum number of iterations and $neg - samples$ is the number of updates to execute with respect to the points rejected by y_a.

Register $X = \{x_1, x_2, x_3, ..., x_n\}$ dataset;
Choose σ according to (15);
compute simplicial fuzzy sets $fs - set(x_i), \forall i = \overline{1, n}$;
$Y^{(0)} \leftarrow spectral\_embedding(k)$;
$\alpha \leftarrow 1.0$ learning rate;
for t = *1 to* T **do**
  for $[a, b, p] \in E(H)$ **do**
    $rand \leftarrow Random(0, 1)$;
    if $rand \leq p$ **then**
      $y_a \leftarrow y_a + \alpha \cdot \nabla(\log(\phi))(y_a, y_b)$;
      for $i \leftarrow 1, ..., neg - samples$ **do**
        $y_a \leftarrow y_a + \alpha \cdot \nabla(\log(1 - \phi))(y_a, y_c)$;
      end
    end
  end
  $\alpha \leftarrow 1.0 - \frac{t}{T}$;
end
return $Y^{(T)}$

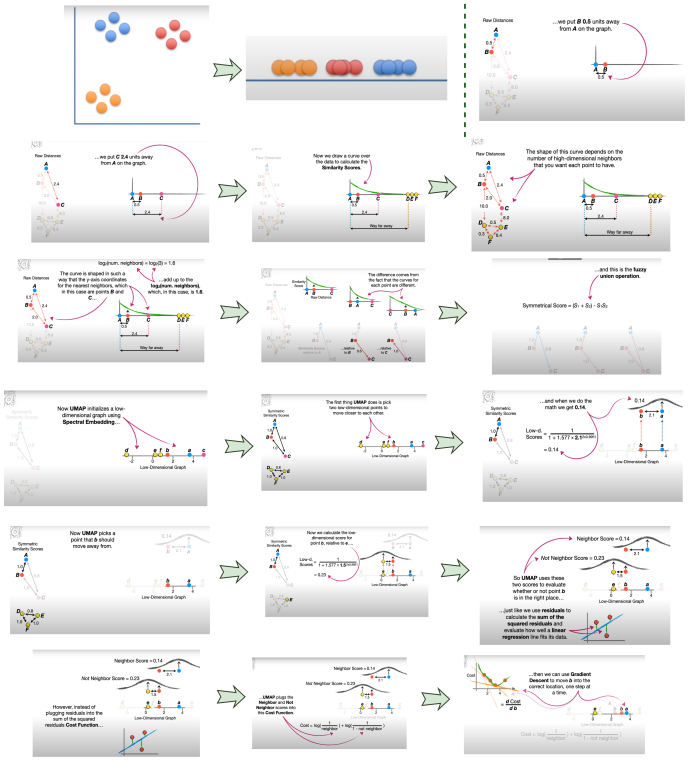


Fig. 4: Toy example for understanding the UMAP algorithm [6, 7].

Ultimately, this section provides a suite of images depicting a very basic example of translating a set of 2-dimensional points into a set of 1-dimensional points preserving the original clusterization as faithfully as possible [Fig. 2]. This example was retrieved — as an excerpt from a video — from a presentation on the topic made at university.

V. Similarities, differences and applications

A. Theoretical comparison

1) Is the Principal Component Analysis trully the principal inspiration component for UMAP and t-SNE?: Placing the comparison of the two newer approaches with PCA on the cutting board first, PCA strives for a balance between feature selection and noise reduction, whilst UMAP and t-SNE both fit better within pattern recognition and clusterization perspectives, with the additional mention that UMAP allows for higher scalability [3]. According to [2], PCA is the only technique focusing on preserving global strcuture among the three discussed in the current section. Moreover, PCA uses linear conventions while the other two solutions tackle non-linear optimization. Needless to say, t-SNE and UMAP lack PCA's feasible and more natural interpretability, by linearly connecting variance and importance, as well as its explanatory properties as a manual data analysis tool when it comes to visualizing the statistical summary (mean, variance etc.) of the original and the rescaled model side by side. Under these considerations, is PCA still considered a source of inspiration for the more modern approaches?

Yes and no. PCA has definitely set the tone for dimensionality reduction using mathematical interpretation and proved to be efficient in low dimensions, which is why it remains a milestone in its field. However, as the main common target of the three approaches is clearly the dimensionality reduction while still preserving individual and collective data tendencies, t-SNE and UMAP outperform PCA due to their power of capturing non-linear dependencies, but, since PCA is less computationally expensive, it is still frequently applied before the other two algorithms — especially before t-SNE if more than 2 or 3 dimensions are required in the low-dimensional model — for the purpose of resources optimization.

2) From t-SNE to UMAP: As far as dimensionality reduction is concerned, the UMAP algorithm is a competitive alternative to the t-SNE algorithm's visualization quality, with better time performance and more emphasis on preserving the global structure of the original system. However, given the computational use of weighted graphs — which suits for neighborhood exploration — and the classification of dimensionality reduction methods proposed in [2], UMAP tends to fall into the category of local structure preserving algorithms, like t-SNE, which also favors the transfer of local distances over the maintenance of global layout.

Not surprisingly at all, although the graph structure is never mentioned alongside the t-SNE algorithm, the edges of the UMAP graph can be thought of as probabilities for

the respective connections to trully exist. As both algorithms proceed similarly in approaching non-convex optimization, only two notable differences stand out: firstly, an initialization procedure with random behaviour is adopted by t-SNE, while UMAP might enjoy slightly more predictible convergence by initializing the weighted graph with its symmetric Laplacian; secondly, UMAP moves only a small batch of points (randomly chosen, even up to only one point) at any step during stochastic gradient descent, while t-SNE thoroughly rearranges all points at each iteration. This last difference situates t-SNE on top when it comes to precision, but simultaneously nominates UMAP as a better alternative for large datasets.

In terms of time performance, UMAP runs undoubtedly faster, and does not impose additional optimization through data structures, such as t-SNE, which relies on space trees and, therefore, is biased through dimensions that can be exponentially represented [4]. However, this difference in performance is not only due to the stochastic gradient procedure, but also due to the global normalisation operation, which UMAP performs implicitly with (15) while t-SNE demands for additional processing to form probabilities [7]. However, t-SNE's model's quality on outputing more than 2 or 3 dimensions is usually superior. This is partially the fault of the critical assumptions the UMAP algorithm relies on, as it confidently proceeds on the uniform distribution of data on a manifold, which is definitely debatable for small datasets.

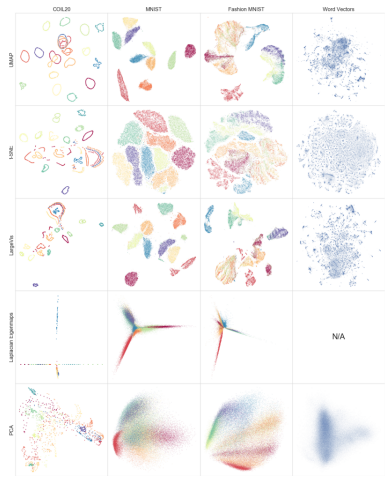
Setting the values $a = 1$ and $b = 1$ for the UMAP's hyperparameters involved in the similarities computations specific to the low-dimensional representation reuslts in the t-SNE's similarity computation equation for the rescaled model [2].

B. Experimental comparison

As far as experiments are concerned, Fig. 5 shows comparative results of the three algorithms and eventually another well performing solutions in terms of quality, time performance and stability. Covering multiple metrics, the experiments prove that a winner algorithm does not exist on its own, but it is always paired with the corresponding data source and the analyzed criteria.

VI. Conclusions

Traversing sections one after another, hopefully this paper has achieved its purpose in terms of arguing that the selection of an algorithm depends on the higher task's specifics. All in all, there is no right or wrong in terms of preferring one algorithm over another, as the conclusions to be drawn from vizualized data are subject to variation and particular requirements and all three of them scale upon strong mathematical foundations. Further directions might revolve around designing and analyzing hybrid approaches consisting of full combinations of the three algorithms, in order to cover their individual, diverse targets all at once. As the progress in science favors bold ideas first and then the overanalysis of them, there is room for experiencing with methodical parameter variations and ingenious pipelines, so as to found even more effective



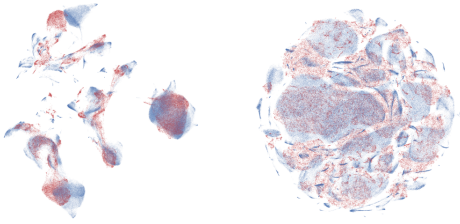
(a) Comparison with respect to results' quality [2].

	UMAP	FIt-SNE	t-SNE	LargeVis	Eigenmaps	Isomap
Pen Digits (1797x64)	9s	48s	17s	20s	**2s**	**2s**
COIL20 (1440x16384)	**12s**	75s	22s	82s	47s	58s
COIL100 (7200x49152)	**85s**	2681s	810s	3197s	3268s	3210s
scRNA (21086x1000)	**28s**	131s	258s	377s	470s	923s
Shuttle (58000x9)	**94s**	108s	714s	615s	133s	–
MNIST (70000x784)	**87s**	292s	1450s	1298s	40709s	–
F-MNIST (70000x784)	**65s**	278s	934s	1173s	6356s	–
Flow (100000x17)	**102s**	164s	1135s	1127s	30654s	–
Google News (200000x300)	**361s**	652s	16906s	5392s	–	–

(b) Comparison with respect to runtime performance [2].

	k	t-SNE	UMAP	LargeVis	Eigenmaps	PCA
Shuttle	100	**0.994** (± 0.002)	0.993 (± 0.002)	0.992 (± 0.003)	0.962 (± 0.004)	0.833 (± 0.013)
	200	**0.992** (± 0.002)	0.990 (± 0.002)	0.987 (± 0.003)	0.957 (± 0.006)	0.821 (± 0.007)
	400	**0.990** (± 0.002)	0.988 (± 0.002)	0.976 (± 0.003)	0.949 (± 0.006)	0.815 (± 0.007)
	800	0.969 (± 0.005)	**0.988** (± 0.002)	0.957 (± 0.004)	0.942 (± 0.006)	0.804 (± 0.003)
	1600	0.927 (± 0.005)	**0.981** (± 0.002)	0.904 (± 0.007)	0.918 (± 0.006)	0.792 (± 0.003)
	3200	0.828 (± 0.004)	**0.957** (± 0.005)	0.850 (± 0.008)	0.895 (± 0.006)	0.786 (± 0.001)
MNIST	100	**0.967** (± 0.015)	**0.967** (± 0.014)	0.962 (± 0.015)	0.668 (± 0.016)	0.462 (± 0.023)
	200	0.966 (± 0.015)	**0.967** (± 0.014)	0.962 (± 0.015)	0.667 (± 0.016)	0.467 (± 0.023)
	400	0.964 (± 0.015)	**0.967** (± 0.014)	0.961 (± 0.015)	0.664 (± 0.016)	0.468 (± 0.024)
	800	0.963 (± 0.016)	**0.967** (± 0.014)	0.961 (± 0.015)	0.660 (± 0.017)	0.468 (± 0.023)
	1600	0.959 (± 0.016)	**0.966** (± 0.014)	0.947 (± 0.015)	0.651 (± 0.014)	0.467 (± 0.0233)
	3200	0.946 (± 0.017)	**0.964** (± 0.014)	0.920 (± 0.017)	0.639 (± 0.017)	0.459 (± 0.022)
Fashion-MNIST	100	**0.818** (± 0.012)	0.790 (± 0.013)	0.808 (± 0.014)	0.631 (± 0.010)	0.564 (± 0.018)
	200	**0.810** (± 0.013)	0.785 (± 0.014)	0.805 (± 0.013)	0.624 (± 0.013)	0.565 (± 0.016)
	400	**0.801** (± 0.013)	0.780 (± 0.013)	0.796 (± 0.013)	0.612 (± 0.011)	0.564 (± 0.017)
	800	**0.784** (± 0.011)	0.767 (± 0.014)	0.771 (± 0.014)	0.600 (± 0.012)	0.560 (± 0.017)
	1600	**0.754** (± 0.011)	0.747 (± 0.013)	0.742 (± 0.013)	0.580 (± 0.014)	0.550 (± 0.017)
	3200	0.727 (± 0.011)	**0.730** (± 0.011)	0.726 (± 0.012)	0.542 (± 0.014)	0.533 (± 0.017)

(c) Comparison with respect to stability if varying k value [2].



(a) UMAP (b) t-SNE

(d) Comparison with respect to stability using Procrustes distance on Flow Cytometry dataset [2].

Fig. 5: Comparative experimental results with regard to different metrics

systems suiting the properties and behavior of the objective datasets throughout the analyzed space.

REFERENCES

[1] L. McInnes, J. Healy and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," ArXiv e-prints 1802.03426, September, 2020.

[2] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE, ", Journal of Machine Learning Research, vol. 9, August 2008.

[3] S. M. Park, "Easy explanation of the dimension reduction (PCA, t-SNE, and UMAP)," LinkedIn, Available: https://www.linkedin.com/pulse/easy-explanation-dimension-reduction-pca-t-sne-umap-sung-mo-park/ [Accessed: Jan. 9, 2024].

[4] L. van der Maaten, "Accelerating t-SNE using Tree-Based Algorithms," Journal of Machine Learning Research, vol. 15, October 2014.

[5] Josh Starmer, "StatQuest: t-SNE, Clearly Explained," YouTube, Available: https://www.youtube.com/watch?v=NEaUSP4YerM [Accessed: Jan. 7, 2024].

[6] Josh Starmer, "UMAP Dimension Reduction, Main Ideas," YouTube, Available: https://www.youtube.com/watch?v=eN0wFzBA4Sc [Accessed: Jan. 8, 2024].

[7] Josh Starmer, "UMAP: Mathematical Details," YouTube, Available: https://www.youtube.com/watch?v=jth4kEvJ3P8 [Accessed: Jan. 9, 2024].

[8] B. Keng, "Manifolds: A Gentle Introduction," Bounded Rationality, Available: https://bjlkeng.io/posts/manifolds/ [Accessed: Jan. 9, 2024].

[9] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (PCA)," Computers & Geosciences, vol. 19, March 1993.