Dimension Reduction with UMAP

Maël Duverger
ENSAE Paris
mael.duverger@ensae.fr

Louis Geist
ENSAE Paris
louis.geist@ensae.fr

Abstract

A lot of datasets of interest are complex and high-dimensional. Nevertheless, we can often expect that most of the information can be captured with a geometric pattern of lower dimensions. Moreover, even if dimension reduction wastes an appreciable part of the information, it can be necessary from a computational point of view. Linear algorithms such as PCA allow to find a vector space of a certain lower dimension minimising the loss of information. However, some datasets have clearly a non-linear geometry which motivates to use more general geometrical objects. In this sense, Riemannian geometry and especially Riemannian manifolds are of particular interest. Riemannian geometry greatly extends the potential structures to explore while remaining compatible with numerical calculus. In this paper, we study an algorithm which exploits Riemannian manifolds for dimension reduction: UMAP (Uniform Manifold Approximation and Projection). We present in section 2 our understanding of UMAP mathematical foundations. In section 3, we analyse the empirical performance of UMAP on MNIST data. We have fully implemented the algorithm by ourselves, the code is available here.

1 Introduction

Dimension reduction is widely used in data science for mainly two purposes: visualisation and pre-processing. These two purposes are inherent to the analysis of large and diverse datasets. Visualisation gives an intelligible "picture" of large and complex datasets. Data pre-processing consists in transforming data to make it useable for a machine learning algorithm. For large datasets, an important step in data pre-processing is to reduce the dimension because some algorithms suffer from the curse of dimensionality: their performance (in terms of computing time or accuracy) decreases at an exponential rate of the dimension. Therefore, a lot of methods have been designed to operate this dimension reduction. An important historical method goes back to Karl Pearson in 1901 with the Principal Component Analysis (PCA) [6]. PCA finds the linear subspace minimising the loss of information. This linear method has inspired a lot of other methods (both linear and non-linear) such as the Kernel PCA [1] that performs PCA in a reproducing kernel Hilbert space thanks to the Kernel trick. More recently, new frameworks, non related to the linear framework, have led to the development of more general and adaptative methods. For instance, t-SNE algorithm is based on information theory and has been quite successful since its publication by Hinton and Roweis in 2002 [2]. Another class of dimension reduction algorithms are those based on manifolds. The common idea is to find a manifold on which data live nearby and then to perform the dimension reduction from this manifold. Among these algorithms, we will study in this paper UMAP, introduced in 2018 by McInnes et al. [4]. In addition to its empirical performances (that we will study in section 3), UMAP is for the most part mathematically founded which reduces the number of arbitrary choices (notably for hyperparameters). As far as we know, UMAP is one of the most theoretically founded algorithm among the manifold learning algorithms for dimension reduction, this is one of the reasons we decided to study it. Moreover, UMAP has great computational properties.

2 Mathematical ideas behind UMAP

In this first section, we will present the main mathematical concepts on which UMAP is based. The general idea of UMAP is to find a manifold in the ambient space such that data live nearby. Then, this manifold is matched with a manifold in a lower dimension space. The match is done in a way to preserve as much topological structure as possible.

Even if some mathematical concepts are quite abstract and look even more abstract given the final algorithm (based on graphs and nearest neighbours), we think that it is still worth presenting it. Indeed, it gives some insights on the final algorithm and it shows how some choices in the implementation are founded and justified. We don't have a complete understanding of certain algebraic or topological concepts, therefore we will only give the general ideas or intuitions on these aspects. We try as much as possible to make a clear link between the theoretical foundations and the final implementation.

2.1 Riemannian Manifolds

We begin by informally introduce Riemannian manifolds. We define intuitively a Riemannian manifold as a set in $\mathbb{R}^n$ that can be locally approximated by a vector space. Thus, this set can be equipped with a family of metrics: for each point of the manifold a vector space (called tangent space) is associated and a metric is defined on this vector space. The metric of the manifold is the family of all these metrics. The way the coordinates change when moving from the metric of one point of the manifold to the metric of another point defines the smoothness of the manifold. A Riemannian manifold is smooth enough to be qualified of differentiable: when two points are close enough, their associated tangent spaces are not too different. This allows doing differential calculus (in some sense) on the manifold which is convenient for numerical applications. Riemannian manifold is a good compromise between generality (in a geometric sense) and numerical feasibility. Since UMAP uses exclusively Riemannian manifolds, in the following we will only write manifold for Riemannian manifold. Finally, we can define a notion of dimension for these manifolds, different from the dimension of the space where they live, say $\mathbb{R}^D$. A manifold is of dimension $p < D$ if a neighbourhood of each of its points can be *twisted* by a diffeomorphism into a linear subspace of dimension p. To illustrate, we show in figure 2.1 a torus in $\mathbb{R}^3$, a famous example of Riemannian manifold.

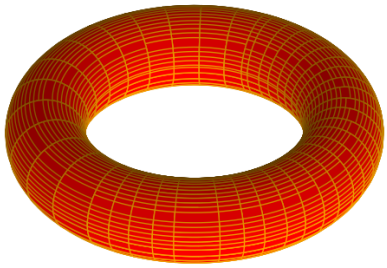

Figure 2.1: A torus in $\mathbb{R}^3$ defined by the equation $\left(R - \sqrt{x^2 + y^2}\right)^2 + z^2 \leq r^2$. Here, $R = 3$ and $r = 1$.

2.2 Manifold learning

UMAP starts by constructing a suitable topological representation of the dataset, based on manifolds. This first step doesn't reduce, at least from a numerical point of view, the dimension of the data, but it is crucial for the dimension reduction. Indeed suppose, that given a certain class of geometric objects in the ambient space, we are able to associate to each member of this class a geometrical object in a space of lower dimension. To preserve the topology of the data through this type of mechanism, we need to find in the ambiant space the right topological and geometrical representation of the data, this is the subject of this section. We will see in the following section the criteria to match topological representation of the dataset in a lower dimension space.

2.2.1 Discrete approximation of continuous topology in theory

As announced, the data, living in $\mathbb{R}^D$, are represented by a manifold of $\mathbb{R}^D$. The main assumption of UMAP is that data points are distributed uniformly on this manifold. This hypothesis is fairly strong. However, it does not appear overly restrictive, considering the variety of shapes within the set of all manifolds. To approximate this manifold (a continuous object), with a finite number of data points, UMAP uses simple combinatorial objects called simplices. Given $p+1$ points, a p-simplex is the convex hull of these points. Consequently, a 1-simplex is a segment and a 2-simplex is a triangle.

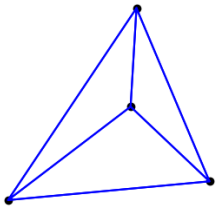

Figure 2.2: A 3-simplex in $\mathbb{R}^3$

We recall that each point of a smooth manifold has an associated tangent space. If two points are close enough, the system of coordinates on each tangent space is not too different thanks to smoothness. If we have for each point its tangent space, we can then find back the manifold. More precisely, given the map that makes the correspondence between coordinates of different tangent spaces, when the manifold is smooth, this map is continous. Thus we can represent this map by a class of open sets that recovers the manifold, leading to a topological space. It's precisely this topological space that UMAP tries to find back. A first approximation is to replace the open sets of the topological space by open balls of a certain radius. Then, Cech construction gives a way to approach this cover of balls. To be as clear as possible on this construction, we quote here the authors: *let each set in the cover be a 0-simplex; create a 1-simplex between two such sets if they have a non-empty intersection; create a 2-simplex between three such sets if the triple intersection of all three is non-empty; and so on*. Theorems state that this construction allows to capture most of the topology. We do not go deeper in this direction, the main idea is to *glue* these different simplices (*gluing* two simplices mean merging two of their faces) to get an approximation of the topology.

2.2.2 Practical approximation

A question arises from the previous section: how to choose the radius r of the balls cover? On one hand, a very small radius would lead to a lot of 1-simplices which is not satisfying. On the other hand, a large radius would give a too rough approximation of the topology. To circumvent this tough choice, UMAP relies on Riemannian geometry once again. The idea is that thanks to the hypothesis of uniform distribution, each unit ball, according to the manifold metric this time, around a point should contain, in expectation, the same number of points. This leads to choose for each point of the dataset the ball containing its k-nearest neighbours. The simplex associated to a point is the k-simplex formed by the point and its k-nearest neighbours. We end up with an open covering (by balls of different radius according to the ambiant metric) of the manifold. The parameter k describes how locally we want to estimate the manifold. A small k would give a precise approximation of the manifold but with a large variance if the amount of data is not very important. As often, we find back the so-called biais-variance trade-off. As we have balls of different radius whereas the initial approximation was with a constant radius, we have to normalise the distances. For each point, the distances with its k nearest neighbours are divided by the distance with its k-th nearest neighbour. This leads in defining a metric for each point of the dataset.

To sum up, for each point, we define a weighted graph: the point itself and its neighbours are the vertices and each neighbour has an edge to the initial point, weighted by the normalised distance. The crucial point is that all these graphs estimate the topology of the manifold. We denote by $X_1^N = \{X_1, ..., X_N\}$ the N points of our dataset (these are elements of $\mathbb{R}^D$). As each point has its own metric, the distance between X_1 and X_2 can differ depending on the metric we choose: the one of X_1 gives $d_{1,2}$ and the one of X_2 gives $d_{2,1}$. $d_{1,2}$ and $d_{2,1}$ can be different because they can have been normalised by different constants. This situation doesn't look satisfying and rather problematic for numerical implementation. Mathematical arguments, that we do not present here, justify that the

right operation to merge these distances is $d_1 + d_2 - d_1 \times d_2$. We can give a probabilistic interpretation of this formula. If we consider d_1 as the probability that X_1 is connected to X_2 (denoted $X_1 \to X_2$) and conversely d_2 is the probability that X_2 is connected to X_1 ($X_1 \leftarrow X_2$) . Supposing that these two random variables are independent, the probability of an unoriented connection is:

$$\begin{aligned}\mathbb{P}(X_1 \leftrightarrow X_2) &= \mathbb{P}(X_1 \to X_2 \cup X_1 \leftarrow X_2) \\ &= d_1 + d_2 - d_1 \times d_2 \quad \textit{by independence}\end{aligned}$$

This motivates the formula since we do not detail the mathematical reasoning behind.

The result of all these steps is a weighted graph connecting the point of the datasets and that approximates the manifold on which data is supposed to live. This result looks quite simple compared to the mathematics behind. But these mathematics justify entirely the use of this graph for manifold learning.

2.3 Dimension reduction

With this topological representation of the dataset, let's see how the dimension reduction is performed: we are looking for N points $Y_1^N = \{Y_1, ..., Y_N\}$ in $\mathbb{R}^d$ (with $d << D$) minimising the *"topological difference"* with $\{X_1, ..., X_N\} \in \mathbb{R}^D$. We have represented the topology of $\{X_1, ..., X_N\}$ by a weighted graph and we given an interpretation of the weights as probabilities. Hence, UMAP looks for $\{Y_1, ..., Y_N\}$ such that their induced graph, as defined in section 2.2.2, is as close as possible to the one of $\{X_1, ..., X_N\}$. Writing E the set of all possible edges and let $e \in E$, $w_X(e)$ $\big($respectively $w_Y(e)\big)$ is the weight on e in the graph induced by X_1^N (respectively Y_1^N). The criterion to determine if two graphs are similar is the cross entropy denoted CV, which has to be minimised:

$$CV(X_1^n, Y_1^n) = \sum_{e \in E} w_X(e) \log\left(\frac{w_X(e)}{w_Y(e)}\right) + \big(1 - w_X(e)\big) \log\left(\frac{1 - w_X(e)}{1 - w_Y(e)}\right)$$

When $w_X(e)$ is close to 1, the term $w_X(e) \log\big(w_X(e)/w_Y(e)\big)$ is predominant compared to $\big(1 - w_X(e)\big) \log\big((1 - w_X(e))/(1 - w_Y(e))\big)$. Hence, the sum of these two terms is minimised by taking a high value for $w_Y(e)$, so by making close the two points in Y_1^N forming this edge e. In the same way, when $w_X(e)$ is close to 0, the minimization is obtained by distancing the points. The authors compare these two observations to attractive and repulsive forces. We notice that the notion of cross entropy comes from Information Theory which is at the basis of some other dimension reduction algorithms such as t-SNE.

3 Empirical analysis

In this section, we analyse the performance of UMAP on synthetic data and on the MNIST database [3]. We begin by specifying some choices we made through the implementation and which are potentially different from the authors' choices. We recall that our code is available here. We implemented UMAP algorithm from scratch with the PyTorch framework. We have faced some issues, as exploding gradients. Clipping gradients to a certain low range (for instance $[-4, 4]$ in our case) has fixed that issue and did not seem to affect the qualitative results compared to the original implementation. Furthermore, as the number of negative sampling is not defined in the original paper of McInnes et al., we have arbitrarly fixed it to 200.

3.1 Synthetic data

In the first place, we have checked that our algorithm has a good behaviour on synthetic data. To do so, we have uniformly sampled on a torus in $\mathbb{R}^3$ defined by the following equation:

$$\left(3 - \sqrt{x^2 + y^2}\right)^2 + z^2 \leq 1$$

This sampling has been done with an acceptance-rejection algorithm. We have added some Gaussian noise with standard deviation 0.5, which gives the samples displayed in figure 3.2. Figure 3.3 shows the result when we reduce with UMAP the noisy torus of $\mathbb{R}^3$ to $\mathbb{R}^2$.

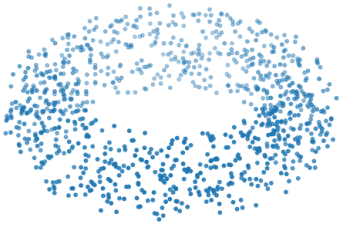

Figure 3.1: 1 000 uniform samples on a torus

Figure 3.2: 1 000 noisy uniform samples on a torus

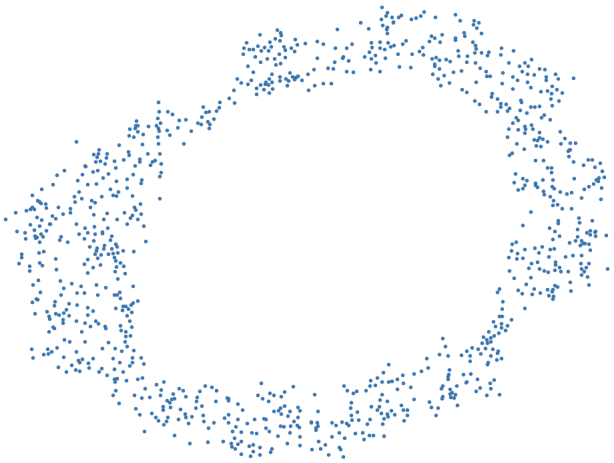

Figure 3.3: Noisy torus of figure 3.2 projected by UMAP on $\mathbb{R}^2$ (400 epochs).

Despite added noise, UMAP seems able to find back the geometry of the torus in $\mathbb{R}^3$ by getting a kind of thick ring in $\mathbb{R}^2$. The outlines of this ring look even cleaner than those of the noisy torus. This result is acceptable, especially as the number of points is not very large.

To analyse more quantitatively these results on synthetic data, we could have used some known scores. We preferred to focus on real data analysis (which is done in a quantitative way in the next section). We still would like to mention an interesting quantitative method of assessment. The idea would be to apply an inverse transform to the embeddings in order to check that we recover something close to the initial data. To be more precise, it would consist in embedding the dataset in a low dimensional space and sample (with also interpolation) in these embeddings. Then, the inverse transform find the data in the high dimensional space that would have generated a sample as close as possible to this one. The final step is to compare this new high-dimensional dataset with the initial one. This could be done for instance with a divergence.

3.2 Real data: MNIST

3.2.1 Embeddings

We now turn to the study of UMAP results for the real-world dataset, MNIST. We present in figure 3.4 the embeddings obtained with UMAP for 10 000 MNIST samples (the size of the full dataset is 60 000).

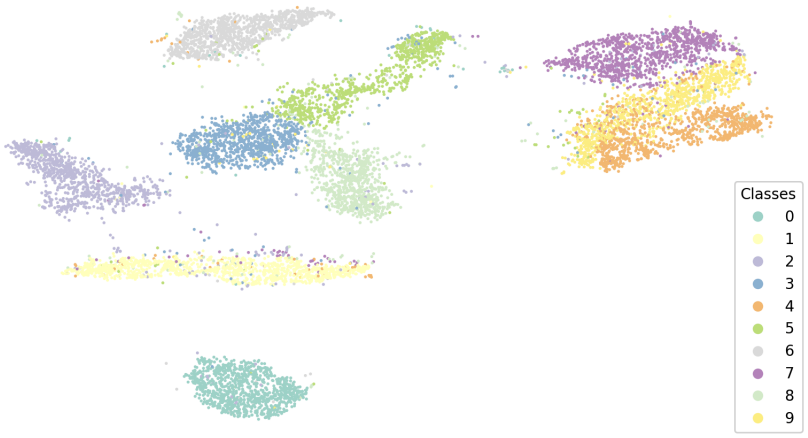

Figure 3.4: 10 000 samples of MNIST projected by UMAP in $\mathbb{R}^2$
(with 400 epochs). The colours indicate the digit class associated to a point.

Visually, this first result is satisfying. Indeed, the embeddings are globally grouped into clusters of the same label. Something not satisfying is the proximity of the embeddings for the digits 4,7 and 9 on the right of the picture. We can think that a clustering algorithm would struggle to separate them. Yet, these groups look almost linearly separable and for this reason we are not so worried about that. At this stage, we have repeated this embedding with different numbers of points and we present the resulting computation times.

Number of points	1 000	5 000	10 000
Fuzzy sets	0.29	1.48	3.45
Spectral embedding	0.32	46.1	358
Embedding optimisation	83.8	784.5	2366

Table 1: Computation times (in seconds) of the 3 main parts of UMAP algorithm.

Unsurprisingly, the longest operation is the embedding one. The computation time of this task looks to be more than linear compared to the number of points.

As for synthetic data, we could have used tools like the inverse transform to give a quantitative evaluation of these results. Instead, we have chosen another approach to evaluate it. This approach is based on clustering and is also quantitative. We are interested by UMAP applications for ML tasks, so we have chosen one possible application that is clustering.

3.2.2 Motivations for clustering on MNIST Data

To assess UMAP performance on the real-world dataset MNIST, we have chosen to evaluate the benefits of UMAP for a clustering task with the K-means algorithm. Indeed, as we said in the introduction, dimension reduction is a popular tool for pre-processing, in particular for clustering pre-processing. This is based on the idea that dimension reduction can help to remove the proportion of noise in the dataset. They are still some concerns about this method because UMAP do not preserve necessarly some properties of the initial distribution, notably related to the density. This approach is more based on intuition and heuristics than mathematically founded. Though, we can still assess with a quantitive score the results obtained by doing clustering with and without UMAP.

3.2.3 Adjusted Mutual Information score

We will use K-means algorithm with $K = 10$ (the number of labels) to cluster raw MNIST data and MNIST data transformed by UMAP. To see if UMAP improves the quality of the clustering, we will use the AMI score, standing for Adjusted Mutual Information. We explain briefly how this

information criterion is obtained and we give some properties. This is further discussed (among with other criteria) in [5]. So, let's denote $\mathcal{U}$ and $\mathcal{T}$ two partitions of a set in K subsets. For our purpose, $\mathcal{T}$ will be the partition by the true labels; $\mathcal{U}$ will be the partition obtained with K-means (on UMAP data or raw data). The Mutual Information (MI) allows comparing two partitions, it is defined by:

$$MI(\mathcal{U}, \mathcal{T}) := \sum_{1 \leq i,j \leq K} p_{\mathcal{U},\mathcal{T}}(i,j) \log \left(\frac{p_{\mathcal{U},\mathcal{T}}(i,j)}{p_{\mathcal{U}}(i) p_{\mathcal{T}}(j)} \right)$$

with:

- $p_{\mathcal{U}}(i)$ the number of points in the cluster $\mathcal{U}_i$ divided by the total number of points,
- $p_{\mathcal{T}}(j)$ the number of points in the cluster $\mathcal{T}_j$ divided by the total number of points,
- $p_{\mathcal{U},\mathcal{T}}(i,j)$ the number of points that are both in clusters $\mathcal{U}_i$ and $\mathcal{T}_j$ divided by the total number of points.

It can be interpreted as the amount of information obtained on one clustering by observing the other one. It quantifies how much these two partitions are linked. In addition, MI is upper bounded by the entropy of each partition, denoted $H(\mathcal{U})$ and $H(\mathcal{T})$. So, to get a quantity upper bounded by 1, the MI can be normalised with these entropies. Moreover, a lot of clustering algorithms are random (K-means included because of its initialization) and thus the MI evaluated on the resulting partitions is also random. [5] justify to consequently center the MI (in fact it's a little more subtle because the expectation is also substracted to the denominator, see the formula below). Therefore, the AMI has been introduced in the following way:

$$AMI(\mathcal{U}, \mathcal{T}) := \frac{MI(\mathcal{U}, \mathcal{T}) - \mathbb{E}\big[MI(\mathcal{U}, \mathcal{T})\big]}{\max\{H(\mathcal{U}), H(\mathcal{T})\} - \mathbb{E}\big[MI(\mathcal{U}, \mathcal{T})\big]}.$$

The interpretation is the same than for MI; we notice that in the extreme case where the partitions are equal, the AMI is equal to 1.

3.2.4 Results and explanation

To see the benefit of UMAP, we compute two K-means, one on raw data and another one on UMAP data. For each K-means, we compute the AMI score with respect to the true label clusters. Table 2 presents the results on MNIST samples of different size.

Number of samples	500	1000	5 000	10 000
MNIST UMAP	37.0	44.6	58.2	67.2
Raw MNIST	36.1	35.0	36.8	32.6

Table 2: AMI score (in percentage) (with UMAP parameters : 100 nearest neighbours, 0.05 minimal distance and 400 epochs.

We see that as the number of points increases, the AMI for K-means on raw MNIST stays constant (around 35%) while the AMI for "UMAP MNIST" increases gradually from 37% to 67.2%. Whereas UMAP brings no improvement with 500 samples (but doesn't reduce the AMI neither), the improvement is great for 10 000 samples with a dobled AMI. Since the size of MNIST dataset is 60 000, we can expect even more striking results with the full dataset. Unfortunately, because of long computation time (already nearly one hour for 10 000 samples, see table 1), we cannot present the results for the full dataset. Figure 3.5 gives a graphical representation of these results.

As UMAP has improved the clustering, the initial idea is strengthened: we think that UMAP has reduced the proportion of uninformative data leading to a better clustering. This is specific to the dataset (MNIST), to the clustering algorithm (K-means) and to the score (AMI) but we hope that it's to some extent generalisable. Finally, we propose an explanation for the increase of the AMI with the number of MNIST UMAP samples. We think that this increase is entirely due to a better manifold learning. Indeed, increasing the number of samples allows approximating more locally the manifold and therefore to gain in accuracy. We don't see clear reasons justifying that the other parts of the process are affected by the number of points. This is why we think it's all about manifold learning.

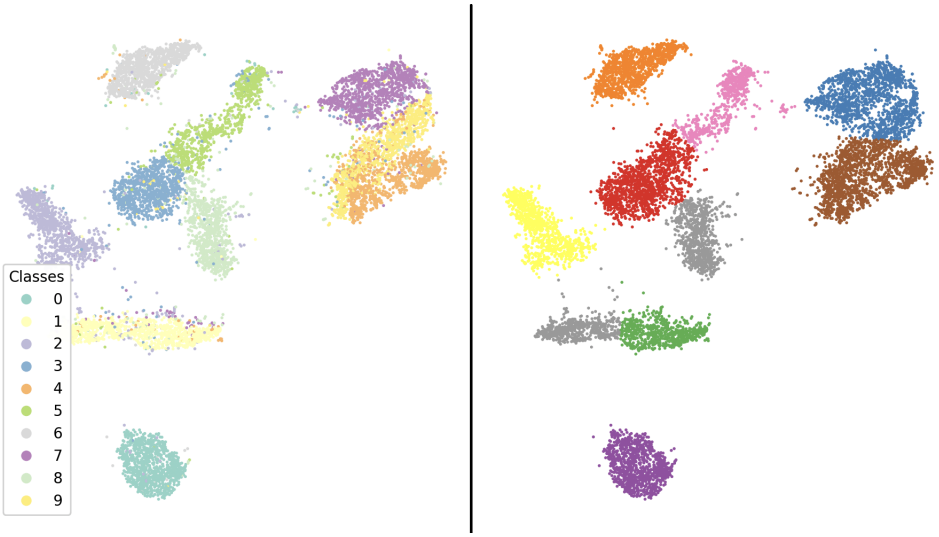

Figure 3.5: Both sides: 10 000 samples of MNIST reduced by UMAP to a 2 dimensional space (with 400 epochs, 100 nearest neighbours, 0.05 of minimal distance). On the left the colours correspond to the true digits clusters, on the right the colours correspond to the K-means clusters.

References

[1] Alexander Smola Bernhard Schölkopf and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". *Neural Computation* (1998).

[2] Geoffrey Hinton and Sam Roweis. "Stochastic neighbor embedding". *Neural Information Processing Systems* (2002).

[3] Y. LeCun. "The mnist database of handwritten digits". *http://yann. lecun. com/exdb/mnist/* (1998).

[4] John Healy Leland McInnes and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". *arXiv:1802.03426* (2018).

[5] Julien Epps Nguyen Xuan Vinh and James Bailey. "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance". *Journal of Machine Learning Research* (2010).

[6] Karl Pearson. "On lines and planes of closest fit to systems ofpoints in space". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* (1901).

4 Appendix

We precise in this appendix two points of our implementation.

First, for reasons of simplicity, we wanted to fix the value of the minimum distance for the rest of our study. We can see from Table 3 that in terms of AMI, compared with the results on 1000 samples of MNIST raw with a score of 0.350, this parameter has little impact). So, we chose min_dist = 0.05.

min_dist	score
0.01	41
0.05	45
0.1	43

Table 3: Minimal distance parameter and corresponding Adjusted Mutual Information (AMI) score (in percentage), for 1000 samples of MNIST with 400 epochs and 100 nearest neighbours in UMAP.

Secondly, the figure 4.1 shows that our approximation of Ψ for calculating the forces of repulsion and attraction seems acceptable.

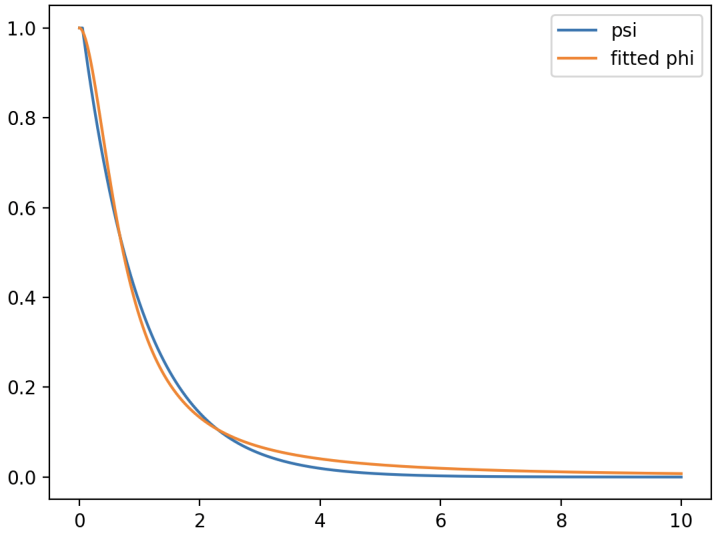

Figure 4.1: Smooth approximation Φ of the function Ψ, with parameter $min\_dist = 0.05$.