Data Visualization with t-SNE algorithm

Dainese Nicola, Mancone Stefano, Vidaich Francesco

In this assignment the aim is to test the usage and efficacy of the t-stochastic neighbour embedding (t-SNE) algorithm for the purpose of data visualization.

I. VISUALIZATION TASK AND INFORMATION

Ideally, in a analysis project, one do not have to look at the data to formulate an hypothesis. It is however impossible to create a model without any insight about data. If previous knowledge is available, the safest way is to use it to create a model and then test it with some kind of metric. In absence of previous useful information, it is necessary to look at the data in some way as a starting point for further analysis. Here comes in visualization. The essence of the problem of visualizing a dataset with n features (where $n > 3$) is that we can't do it without loss of information. This is because the samples that we want to represent live in an n-dimensional space and we can dispose only of 2 or at most 3 dimensions to represent them. The case with $n \leq 3$ is easy because we can visualize it in our own space. With $n > 3$ we are forced to find a suitable combination of the initial coordinates (features) which reflects in some way the structure of our samples in feature space. Actually there is a case in which we can do it almost without loss of information, and this is when some of the features can only assume few values, like 0-1 in the simplest case. In fact we can encode this features in the color of the points, or using different symbols and sizes for the markers. However in this way we can account only for one or two extra dimensions and the price to pay is that the amount of information delivered in a single graph is overwhelming. Anyhow in this review we will focus only on the representation of samples with all the features that are continuous. The easiest approach to reduce the dimension of the samples minimizing the loss of information is Principal Component Analysis (PCA), where directions of most variance in the covariance matrix of the dataset are found. This can be used to find the most significant directions along which samples are distributed. Significance is understood in the sense that direction with small variance can be assumed as noise and therefore eliminated. The most limit aspect of PCA is that the transformation of the data is linear and thus it cannot be used with datasets with a complex structure like, for example, spirals.

II. T-STOCHASTIC NEIGHBOUR EMBEDDING

In this report we practice with a more sophisticated method for data visualization, that is called t-stochastic neighbour embedding (t-SNE).

The idea of stochastic neighbor embedding is to associate a probability distribution to the neighborhood of each data point in the feature space ($x \in \mathbb{R}^n$) using a gaussian likelihood

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

This implies that only points that are nearby x_i contribute to its distribution. To account this effect a symmetrized distribution $p_{ij} = (p_{i|j} + p_{j|i})/(2N)$ is used. This guarantees that $\sum_j p_{ij} > 1/(2N)$ for all data points making each data point significant contribute to the cost function to be defined below. t-SNE algorithm constructs a similar probability distribution q_{ij} in a low dimensional latent space ($y \in \mathbb{R}^{n'}$ where $n' < n$ is the dimension of the latent space)

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

Here q_{ij} is chosen to be a long tailed distribution. This preserves short distance information while strongly repelling two points that are far apart in the original space. The last step is to find the latent space coordinates y_i. In order to achieve this t-SNE algorithm minimizes the Kullback-Leiber (KL) divergence between q_{ij} and p_{ij}

$$C(Y) = D_{KL}(p||q) = \sum_{ij} p_{ij}\log\left(\frac{p_{ij}}{q_{ij}}\right)$$

via gradient descent. In order to acquire insight about how the t-SNE algorithm works we did some experiments with the *sklearn.manifold.TSNE* class from scikit learn library. Here the important hyperparameters that we can tune are:

- *n_components* defines the dimension of the latent space;
- *perplexity* sets the local entropy, which determines the σ_i in the gaussian likelihood;
- *early_exaggeration* controls how much distance in the embedded space will be between natural clusters in the original space;
- *learning_rate* fixes the learning rate in the gradient descent;
- *n_iter* is just the number of gradient updates;
- *n_iter_without_progress* and *min_grad_norm* are just early stopping parameters;
- *init* can be chosen between "random" and "PCA".

In order to experiment with this algorithm we were provided five different datasets of samples generated from five unknown functions. In every dataset there are five features and a color, which varies with continuity on the manifold from which the samples were generated. We start considering the first dataset, in order to understand how the different hyperparameters of the algorithm work. We have initially plotted the different pairwise scatter plots of the five features to get an idea about the structure of the dataset. Verified the non-trivial disposition of the samples, we applied the t-SNE transform for different choices of the hyperparameters. Since from the documentation seems that the perplexity is the main discriminator about how the algorithm behaves, the runs have been performed choosing the perplexity between $[2, 5, 50, 100]$ and changing the others hyperparameters one at the time.

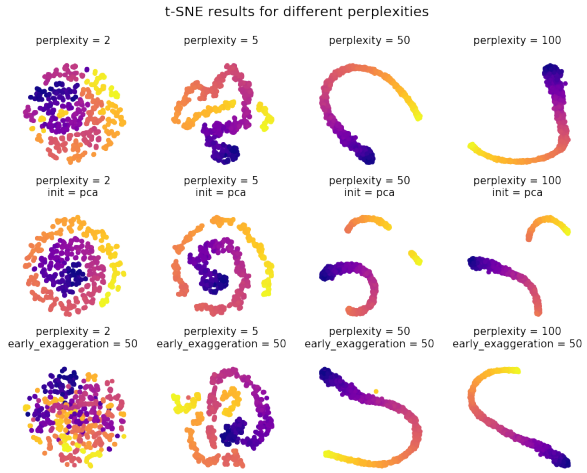

FIG. 1. t-SNE algorithm with perplexity in [2,5,50,100], init as PCA and early exaggeration 36

In Figure 1 we have reported the runs with only changes in perplexity, with *init* chosen as PCA and with the early exaggeration parameter set to 36. It is clear that a different choice of perplexity parameter lead to a qualitative different result. The strategy that we find most efficient is to use all the different results together for infer some properties of the data. For example in this case, from the the runs with all default values and perplexity $[5, 50, 100]$ seems clear that the structure of the data develops on a single dimensional manifold. Looking instead at the runs with initialization with PCA, we can see that it appears a structure similar to a spiral. Taking together these two piece of information we already have a pretty good idea about the structure of the first dataset.

From other experiments, whose graphs we do not report, we observe that, unlike what happens with other algorithms, in this case most of the hyperparameters choices lead only to slightly different results. For example the *learning_rate* best choice is the default one and the behaviour does not change, unless it is varied of order of magnitude, in which case one will get very poor results.

III. T-SNE AND UNSUPERVISED COLOR-LABELLING

A spontaneous objection to this strategy is that we used the colors we were given to understand the structure of the data, but they are a labeling feature we do not have in real world visualization tasks. It is then interesting to find a strategy for coloring the samples, which otherwise would look colorless and more difficult to interpret. Since the embedding can be performed to an arbitrary latent space, one can choose a one dimensional latent space, map the output coordinate in the interval [0,1] and use it as color labels also for 2D and 3D target spaces; we will refer to this method as forced projection. This seems more promising that simply using the last coordinate from an n'-dimensional space as color, because if $n' = 1$ the algorithm is forced to infer a meaningful structure for the color. This however has a really limited utility; in fact it loses effectiveness as soon as the dataset topology leaves the case of a simple non-closed curve. Another strategy is to perform a k-means clustering choosing a priori or iteratively the number of interesting components and then using the mapped colors to look at the data in different realizations of the t-SNE algorithms.

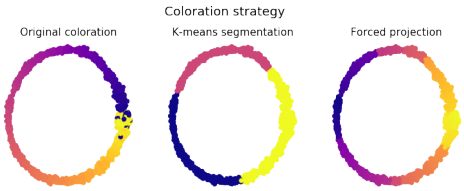

FIG. 2. Different coloration strategies for unsupervised learning

In Figure 2 we had taken the second dataset we had available to test the efficacy of the two methods. As it can be seen, while the k-means method is quite coarse, the forced projection method is not much reliable. Therefore, even if the k-means method is not perfect, can be effective to gain insight in those cases where a previously given coloration was not provided.

The last dataset on which we did some tests was the fifth. In Figure 3 we show the results of executing the algorithm with some hyperparameters. The data structure however remained not easly understandable. After some other attemps we decided to set the latent space dimension to 3. The obtained result are shown in Figure 4. Is then immediately clear what the structure of the fifth dataset is. Also the 2D plots in Figure 3 are now more interpretable.

IV. T-SNE SCOPE AND LIMITATIONS

The last example highlights one t-SNE algorithm limits. If the samples have an intricate structure, for example presents some knots, the embedded space dimen-

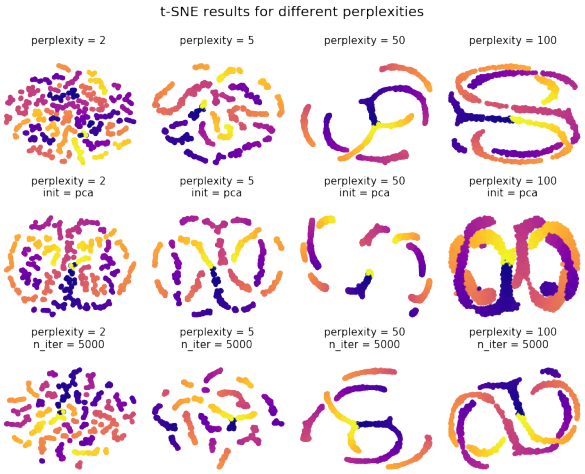

FIG. 3. Fifth dataset rappresentation in different 2D latent spaces

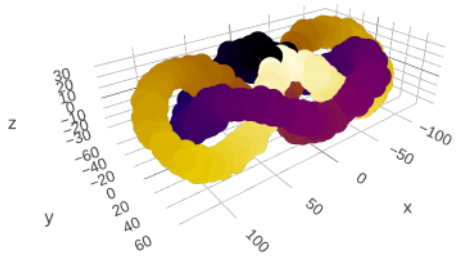

FIG. 4. Fifth dataset rappresentation in a 3D latent space

sion must be large enough to represent these complexities topologically. But, although this is actually a limitation, we are already far ahead of other unsupervised algorithms such as k-means clustering in terms of applicability. If however one wants to find an algorithm that is not affected by this limit, one should take a look to the k-Nearest-Neighbors or hierarchical clustering which could be used together with t-SNE.

The last tests we performed were with the Modified National Institute of Standards and Technology (MNIST) dataset of handwritten digits. This is a standard classification dataset and thus it is used as a benchmark for classification problems. Our goal is to understand if with these unsupervised techniques we can recognize a structure that can then be learned in principle with supervised learning.

Every sample of this dataset is a $28 \times 28 = 784$ pixels image where every pixel is a number between 0 and 255 which correspond to the pixel gray scale. We used the t-SNE algorithm on 6000 samples over 10000 iterations with a perplexity of 30 and a learning rate of 200. Then we used the so obtained coordinates to train a k-means algorithm; finally we labelled each cluster with the corresponding digit, to check if we got to catch the structure of the dataset.

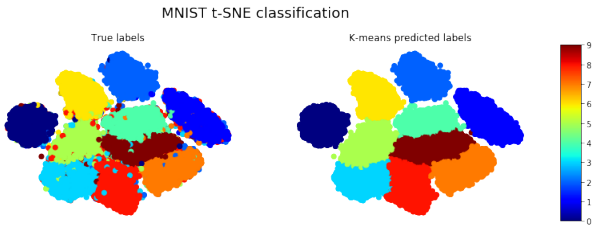

FIG. 5. MNIST dataset classification with t-SNE e k-means algorithms

With this very naive method we got an accuracy of 95.3%. Which mean that the actual structure has been obtained quite accurately. One thing to note here is that, because the digit clusters are not very separated perform the k-means algorithm with exactly 10 centroids lead to poor results because the algorithm collocate them in unfavorable positions. As it is shown in Figure 6 a solution can be initialize the algorithm with a doubled number of centroids and then reduce the effective number of clusters using a democratical approach to assign to each cluster the corresponding label.

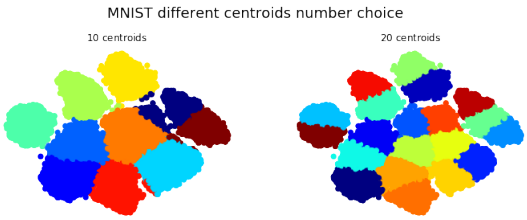

FIG. 6. Different cluster representation conditioned by the centroids number choice

V. CONCLUSIONS

We had therefore sperimented with the usage of the t-SNE algorithm. We had seen that despite having some limitation (no free lunch theorem) it is very useful to gain insight about the structure and the disposition of the data. Its usage is particularly recommended when data belong to a low dimensional manifold without complex topological behaviour, which is usually the case. At the end of the day its scope is limited only to visualization tasks, that are the ones for which it is designed. Instead t-SNE can not be used in learning tasks because it does not create a map between the feature space and the embedded space as, for example, the PCA. It only act directly on the sample points so there is not a obvious way to use it for dimensionality reduction. Beyond then to its use for data visualization it has limited use also due to his heavy computational cost.