kcg.edu

Exploratory Data Analysis and Visualization

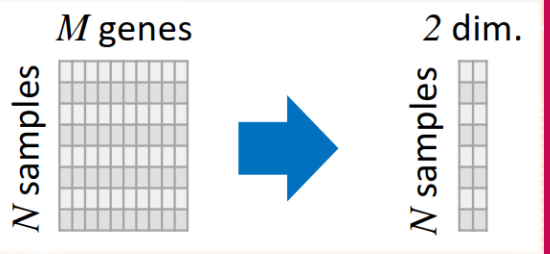
Data Visualization by t-SNE and UMAP

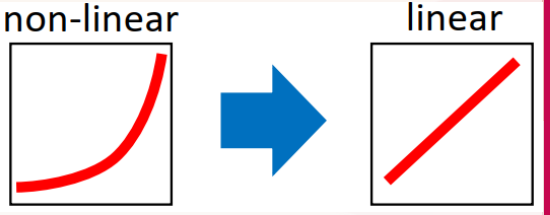
Instructor: Dr. Dinh Duy Tai
Email: t_dinh@kcg.ac.jp

Why dimensionality reduction?

kcg.edu

- Simplify complexity, so it becomes easier to work with.
  - Reduce the number of features
  - Transform non-linear relationships to linear
- Remove redundancies in the data
- Identify the most relevant information (find and filter noise)
- Reduce computational time
- Facilitate clustering, since some algorithms struggle with too many dimensions
- Data visualization





Some dimensionality reduction algorithms

kcg.edu

They can be divided into 3 major groups:

PCA	linear	Matrix Factorization		
ICA	linear	Matrix Factorization		
MDS	non-linear	Matrix Factorization		
Sparce NNMF	non-linear	Matrix Factorization	2010	https://pdfs.semanticscholar.org/664d/40258f12ad28ed0b7d4c272935ad72a150db.pdf
cPCA	non-linear	Matrix Factorization	2018	https://doi.org/10.1038/s41467-018-04608-8
ZIFA	non-linear	Matrix Factorization	2015	https://doi.org/10.1186/s13059-015-0805-z
ZINB-WaVE	non-linear	Matrix Factorization	2018	https://doi.org/10.1038/s41467-017-02554-5
Diffusion maps	non-linear	graph-based	2005	https://doi.org/10.1073/pnas.0500334102
Isomap	non-linear	graph-based	2000	10.1126/science.290.5500.2319
t-SNE	non-linear	graph-based	2008	https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf
- BH t-SNE	non-linear	graph-based	2014	https://lvdmaaten.github.io/publications/papers/JMLR_2014.pdf
- Flt-SNE	non-linear	graph-based	2017	arXiv:1712.09005
LargeVis	non-linear	graph-based	2018	arXiv:1602.00370
UMAP	non-linear	graph-based	2018	arXiv:1802.03426
PHATE	non-linear	graph-based	2017	https://www.biorxiv.org/content/biorxiv/early/2018/06/28/120378.full.pdf
scvis	non-linear	Autoencoder (MF)	2018	https://doi.org/10.1038/s41467-018-04368-5
VASC	non-linear	Autoencoder (MF)	2018	https://doi.org/10.1016/j.gpb.2018.08.003

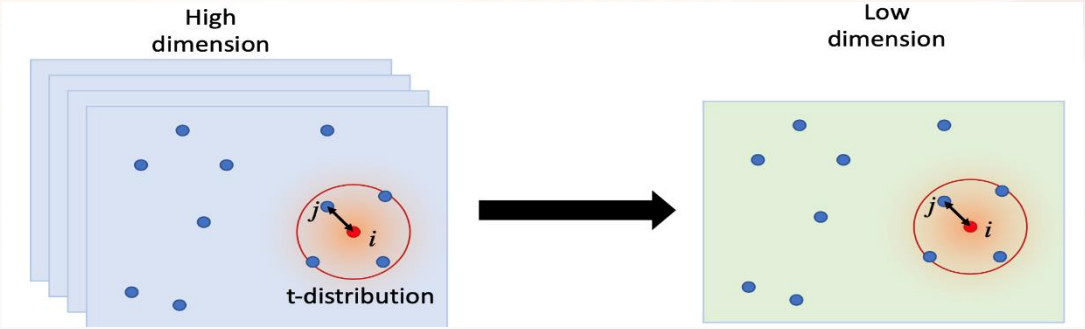
... and many more

Source: Paulo Czarnewski, ELIXIR-Sweden

General ideas

kcg.edu

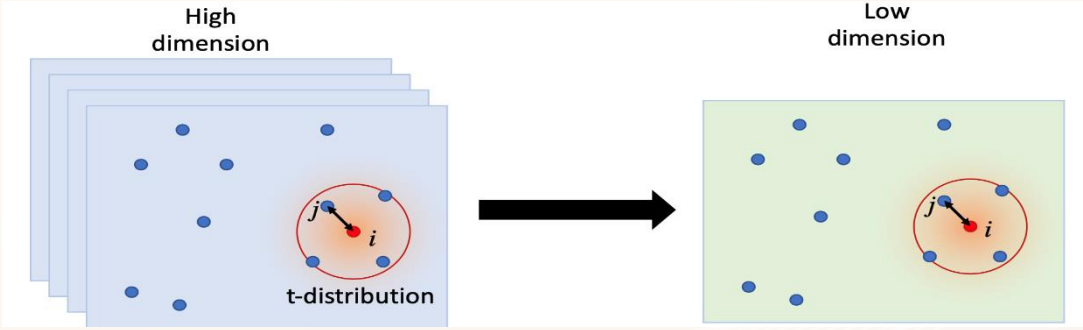
- Compute high dimensional probabilities p.
- Compute low dimensional probabilities q.
- Calculate the difference between the probabilities by a given cost function $C(p,q)$.
- Minimize the cost function.

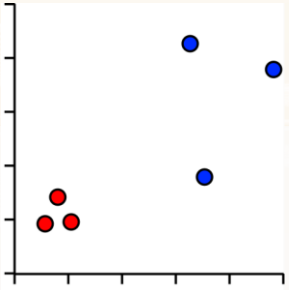


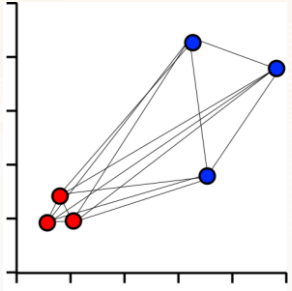
$p(j|i)$ $q(j|i)$

t-SNE: t-Stochastic Neighborhood Embedding

kcg.edu







	0	10	10	295	158	153
	9	0	1	217	227	213
	1	8	0	154	225	238
	205	189	260	0	23	45
	248	227	246	44	0	54
	233	176	184	41	36	0

Graph-based dimensionality reduction algorithms include the construction of a graph in high dimensions followed by an optimization step to find the most similar graph in lower dimensions.

t-SNE: t-Stochastic Neighborhood Embedding

kcg.edu

In what follows we are concerned with defining similarities between two objects i and j in the high dimensional input space X and low dimensional embedded space Y. These are normalized and symmetrized in various ways. In a typical implementation, these pair-wise quantities are stored and manipulated as (potentially sparse) matrices. Quantities with the subscript ij are symmetric, i.e. $v_{ij} = v_{ji}$. Extending the conditional probability notation used in t-SNE, $j \mid i$ indicates an asymmetric similarity, i.e. $v_{j|i} \neq v_{i|j}$.

t-SNE defines input probabilities in three stages. First, for each pair of points, i and j, in X, a pair-wise similarity, v_{ij}, is calculated, Gaussian with respect to the Euclidean distance between x_i and x_j:

$$v_{j|i} = \exp(-\|x_i - x_j\|_2^2 / 2\sigma_i^2) \tag{5}$$

where σ_i^2 is the variance of the Gaussian.

Second, the similarities are converted into N conditional probability distributions by normalization:

$$p_{j|i} = \frac{v_{j|i}}{\sum_{k \neq i} v_{k|i}} \tag{6}$$

σ_i is chosen by searching for a value such that the perplexity of the probability distribution $p_{\cdot|i}$ matches a user-specified value.

$$\text{Perplexity} = 2^{-\sum_j p_{j|i} \log_2 p_{j|i}}$$

t-SNE: t-Stochastic Neighborhood Embedding

kcg.edu

Third, these probability distributions are symmetrized and then further normalized over the entire matrix of values to give a joint probability distribution:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{7}$$

Similarities between pairs of points in the output space Y are defined using a Student t-distribution with one degree of freedom on the squared Euclidean distance:

$$w_{ij} = \left(1 + \|y_i - y_j\|_2^2\right)^{-1} \tag{8}$$

followed by the matrix-wise normalization, to form q_{ij}:

$$q_{ij} = \frac{w_{ij}}{\sum_{k \neq l} w_{kl}} \tag{9}$$

The t-SNE cost is the Kullback-Leibler divergence between the two probability distributions:

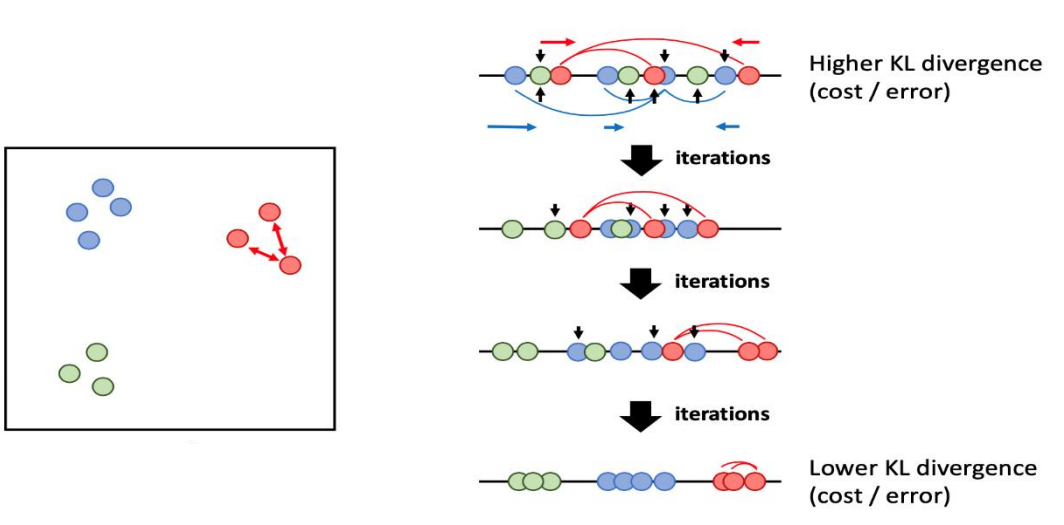
$$C_{t-SNE} = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{10}$$

this can be expanded into constant and non-constant contributions:

$$C_{t-SNE} = \sum_{i \neq j} p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \tag{11}$$

t-SNE: t-Stochastic Neighborhood Embedding

kcg.edu

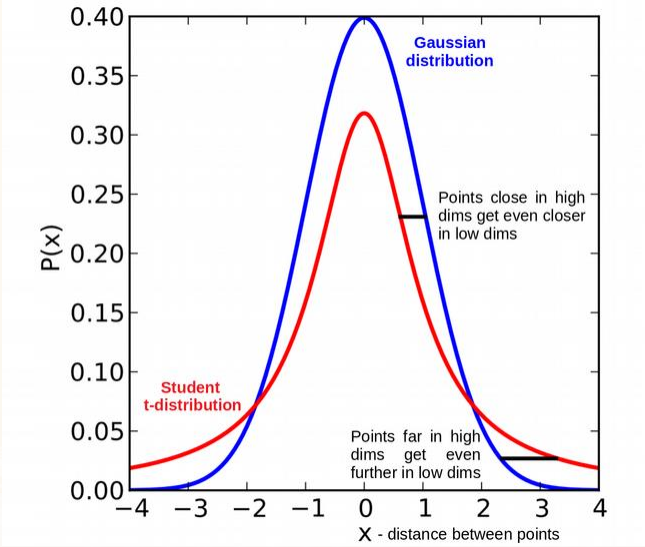


$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

- Kullback-Leibler (KL) divergence quantifies the difference between two probability distributions, P and Q. It measures how much one probability distribution differs from another.
- It quantifies the inefficiency of using Q to approximate P

t-SNE: t-Stochastic Neighborhood Embedding

kcg.edu



- The heavy tails of the student t-distribution are supposed to provide the global distance information as they push the points far apart in the high dimensions to be even further apart in the low dimensions.

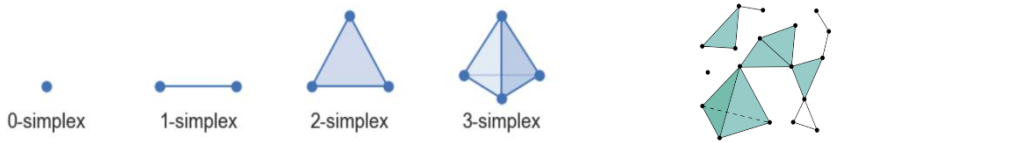
UMAP

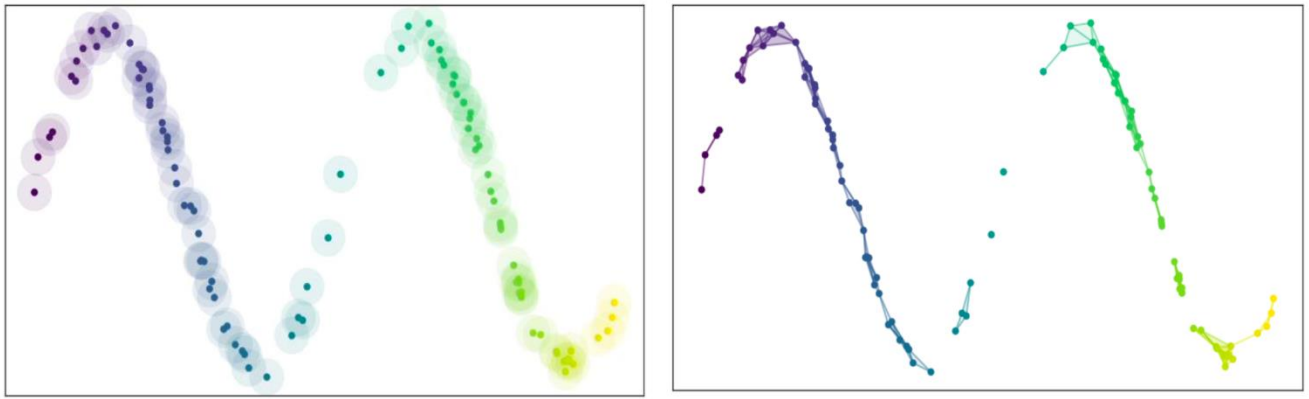
kcg.edu

It is based on **topological structures** in multidimensional space (simplices)

Points are connected with a line (edge) if the distance between them is below a threshold:

- Any distance metric can be used (euclidean)





This way, by constructing the simplicial complexes beforehand allows UMAP to calculate the relative point distances in the lower dimension, instead of randomly assigning as in tSNE.

UMAP

The first phase of UMAP can be thought of as the construction of a weighted k-neighbour graph. Let $X = \{x_1, \ldots, x_N\}$ be the input dataset, with a metric (or dissimilarity measure) $d : X \times X \to \mathbb{R}_{\geq 0}$. Given an input hyper-parameter k, for each x_i we compute the set $\{x_{i_1}, \ldots, x_{i_k}\}$ of the k nearest neighbors of x_i under the metric d. This computation can be performed via any nearest neighbour or approximate nearest neighbour search algorithm. For the purposes of our UMAP implemenation we prefer to use the nearest neighbor descent algorithm of [18].

For each x_i we will define ρ_i and σ_i. Let

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

and set σ_i to be the value such that

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

The selection of σ_i corresponds to (a smoothed) normalisation factor, defining the Riemannian metric local to the point x_i as described in Section 2.1.

We can now define a weighted directed graph $\bar{G} = (V, E, w)$. The vertices V of $\bar{G}$ are simply the set X. We can then form the set of directed edges $E = \{(x_i, x_{i_j}) \mid 1 \leq j \leq k, 1 \leq i \leq N\}$, and define the weight function w by setting

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

The low dimensional similarities are given by:

$$w_{ij} = \left(1 + a \left\|y_i - y_j\right\|_2^{2b}\right)^{-1} \tag{17}$$

where a and b are user-defined positive values. The procedure for finding them is given in Definition 11. Use of this procedure with the default values in the UMAP implementation results in $a \approx 1.929$ and $b \approx 0.7915$. Setting $a = 1$ and $b = 1$ results in the Student t-distribution used in t-SNE.

Ignoring specific definitions of v_{ij} and w_{ij}, the UMAP cost function, the cross entropy, is:

$$C_{UMAP} = \sum_{i \neq j} v_{ij} \log \left(\frac{v_{ij}}{w_{ij}} \right) + (1 - v_{ij}) \log \left(\frac{1 - v_{ij}}{1 - w_{ij}} \right) \tag{13}$$

Like the Kullback-Leibler divergence, this can be arranged into two constant contributions (those containing v_{ij} only) and two optimizable contributions (containing w_{ij}):

$$\begin{aligned} C_{UMAP} = \sum_{i \neq j} & v_{ij} \log v_{ij} + (1 - v_{ij}) \log (1 - v_{ij}) \\ & - v_{ij} \log w_{ij} - (1 - v_{ij}) \log (1 - w_{ij}) \end{aligned} \tag{14}$$

Hyperparameters of UMAP

kcg.edu

- Min-dist:
minimal distance between
points in low-dimensional space

- N_neighbours:
k in the kNN graph

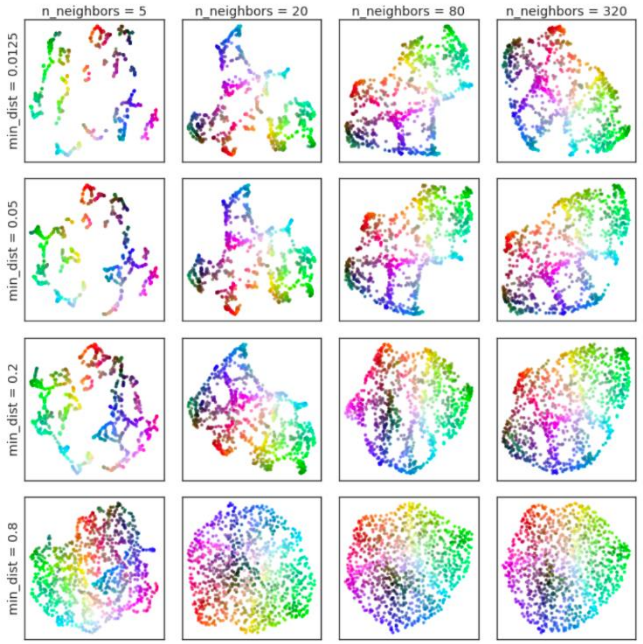
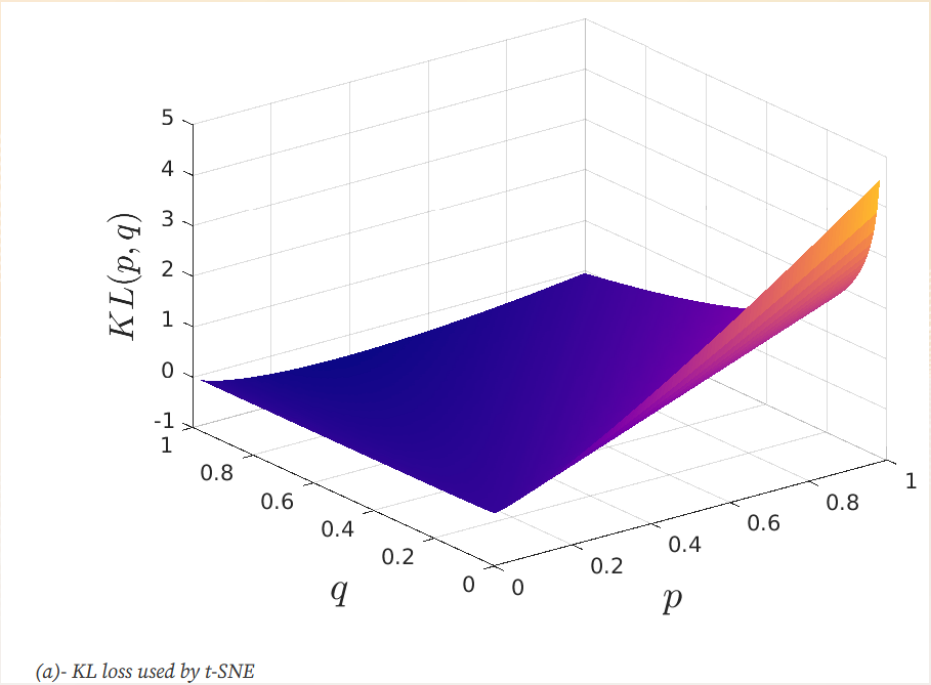
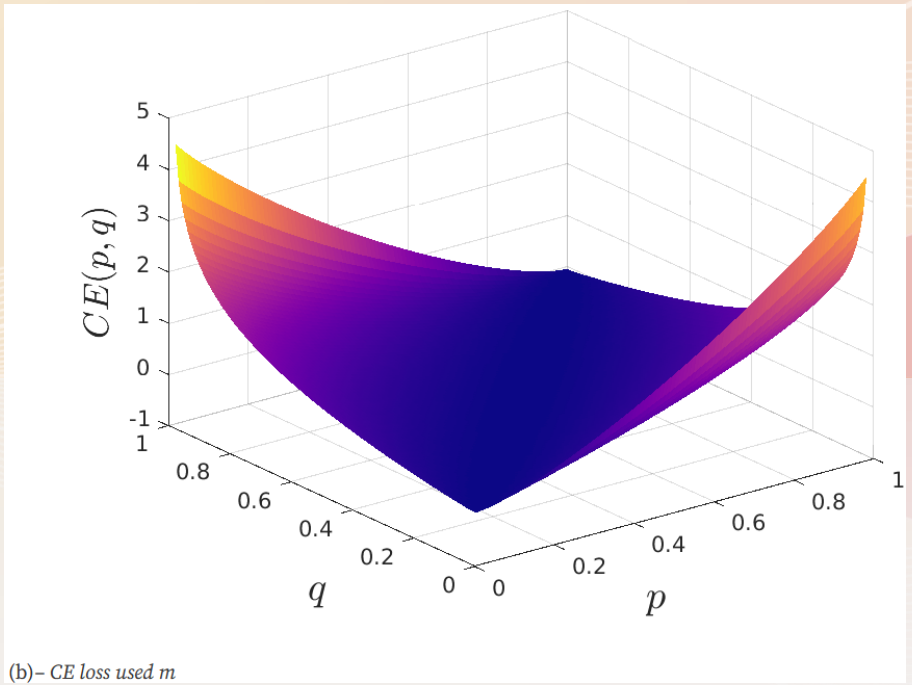


Figure 1: Variation of UMAP hyperparameters n and min-dist result in different embeddings. The data is uniform random samples from a 3-dimensional color-cube, allowing for easy visualization of the original 3-dimensional coordinates in the embedding space by using the corresponding RGB colour. Low values of n spuriously interpret structure from the random sampling noise – see Section 6 for further discussion of this phenomena.



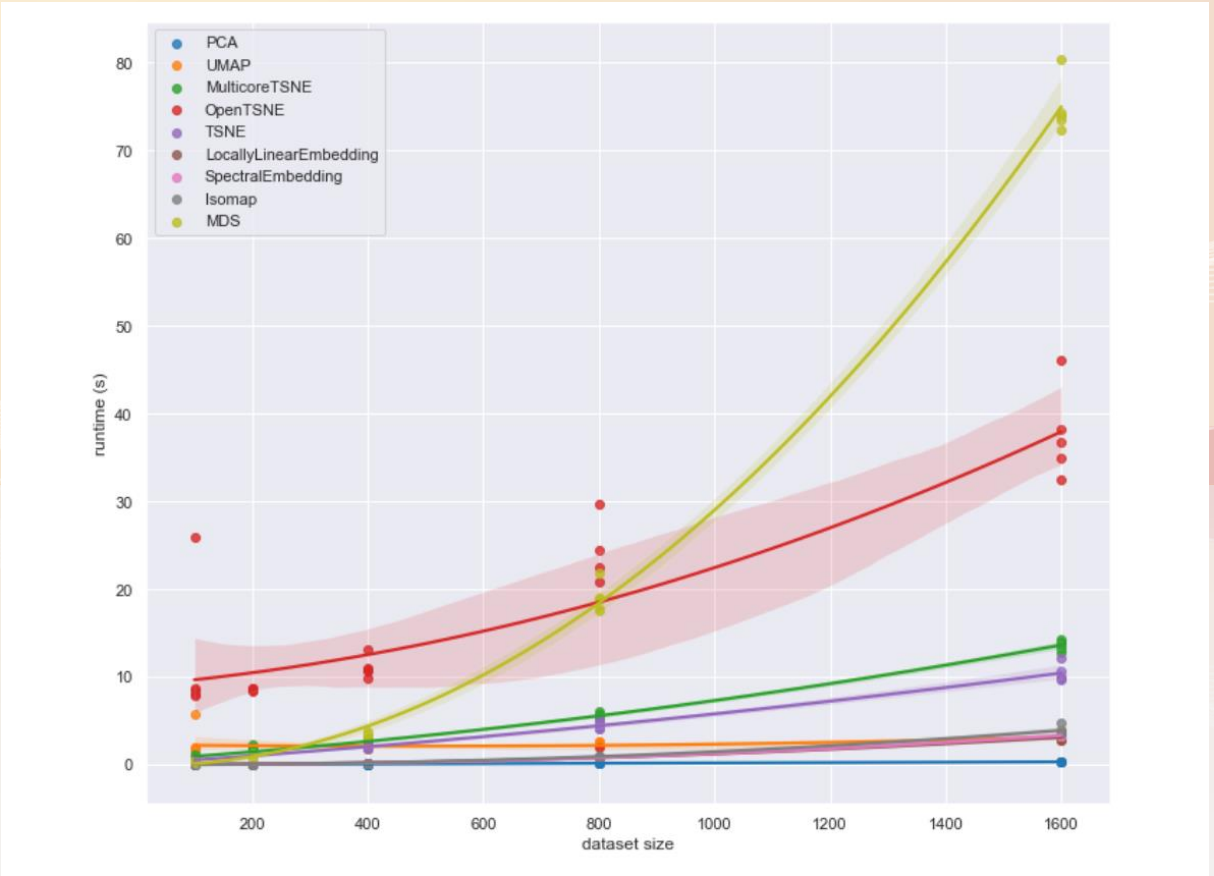
(a)- KL loss used by t-SNE



(b)– *CE loss used m*

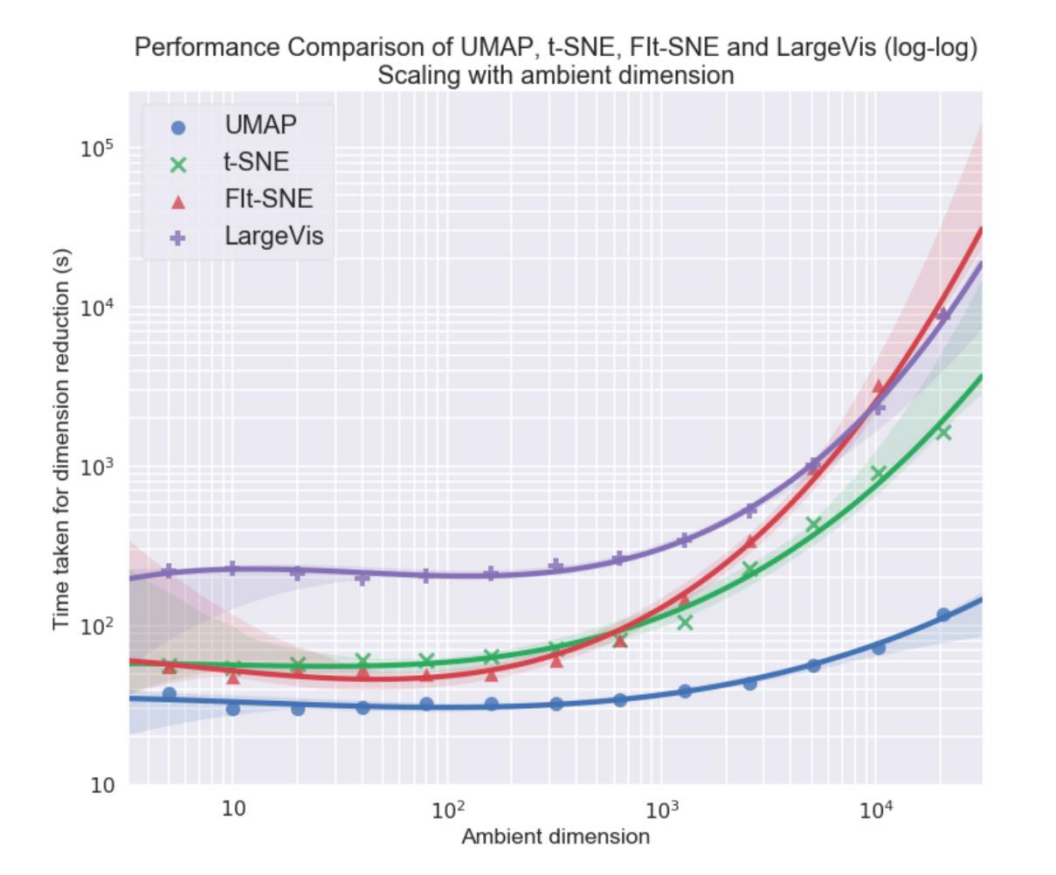
Speed of t-SNE and UMAP

kcg.edu



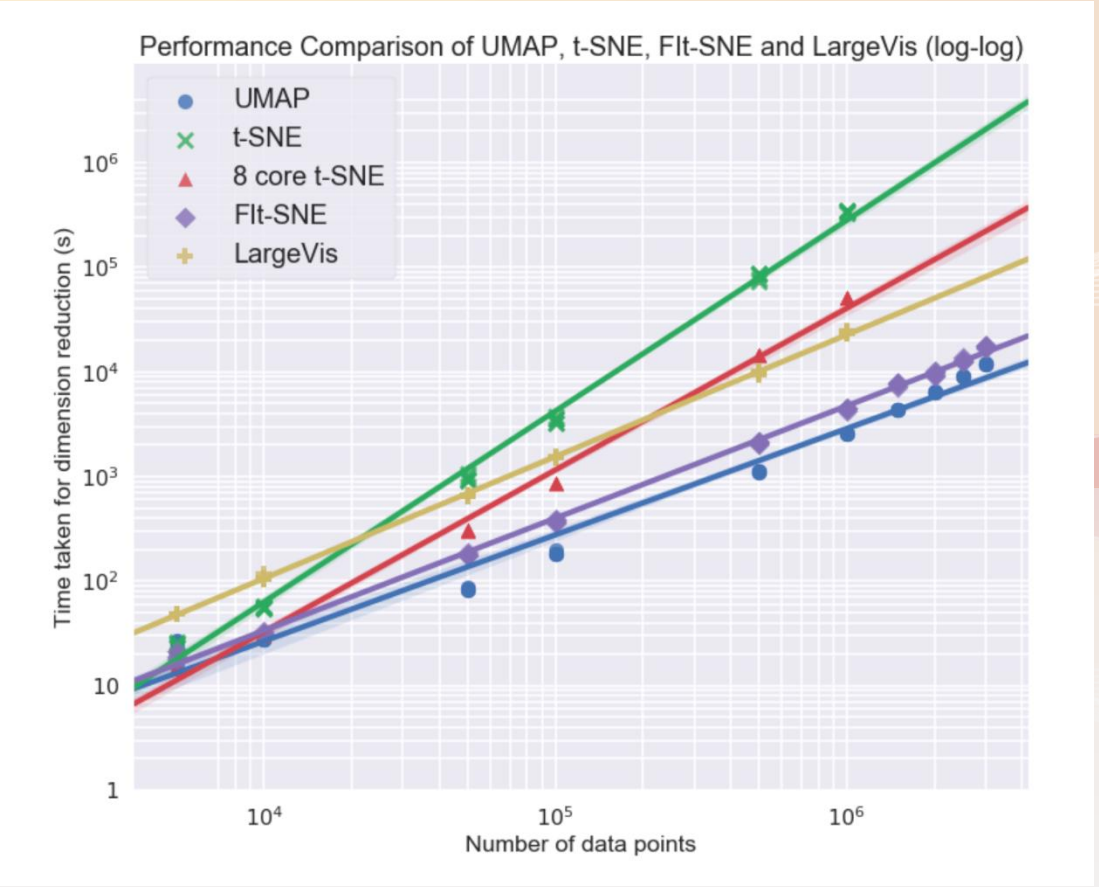
Speed of t-SNE and UMAP

kcg.edu



Speed of t-SNE and UMAP

kcg.edu



https://umap-learn.readthedocs.io/en/latest/
https://pair-code.github.io/understanding-umap/