THINKFUL

Dimensionality Reduction

Uniform Manifold Approximation and Projection (UMAP)

Course

Objective

Gain an understanding of the necessity and the usefulness of the UMAP dimensionality reduction method, how it relates to data science work, and how it relates to other non-linear dimensionality reduction methods.

Curriculum

Module:

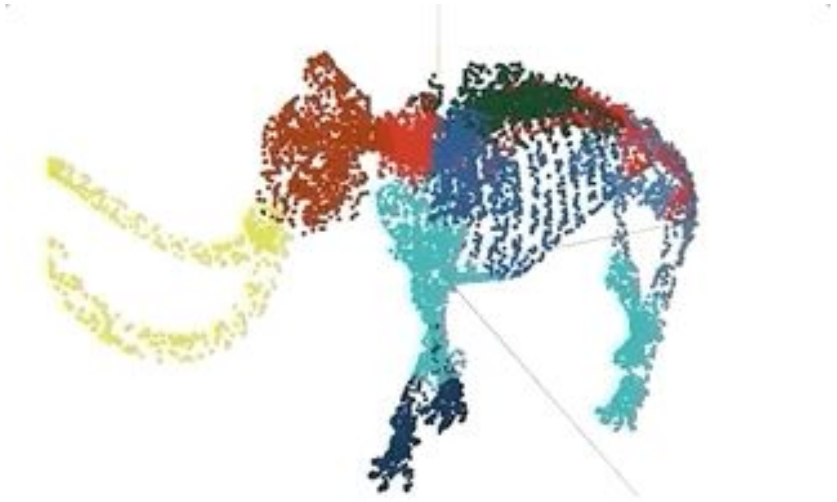
➔ Checkpoint 1 Title [linked]
➔ Checkpoint 2 Title [linked]

Agenda

- Warm Up
- Rationale of UMAP
- Theory behind UMAP
- Parameters of UMAP
- Applying UMAP
- Comparison with t-SNE and other DR methods
- Summary
- Assignment

Warm Up

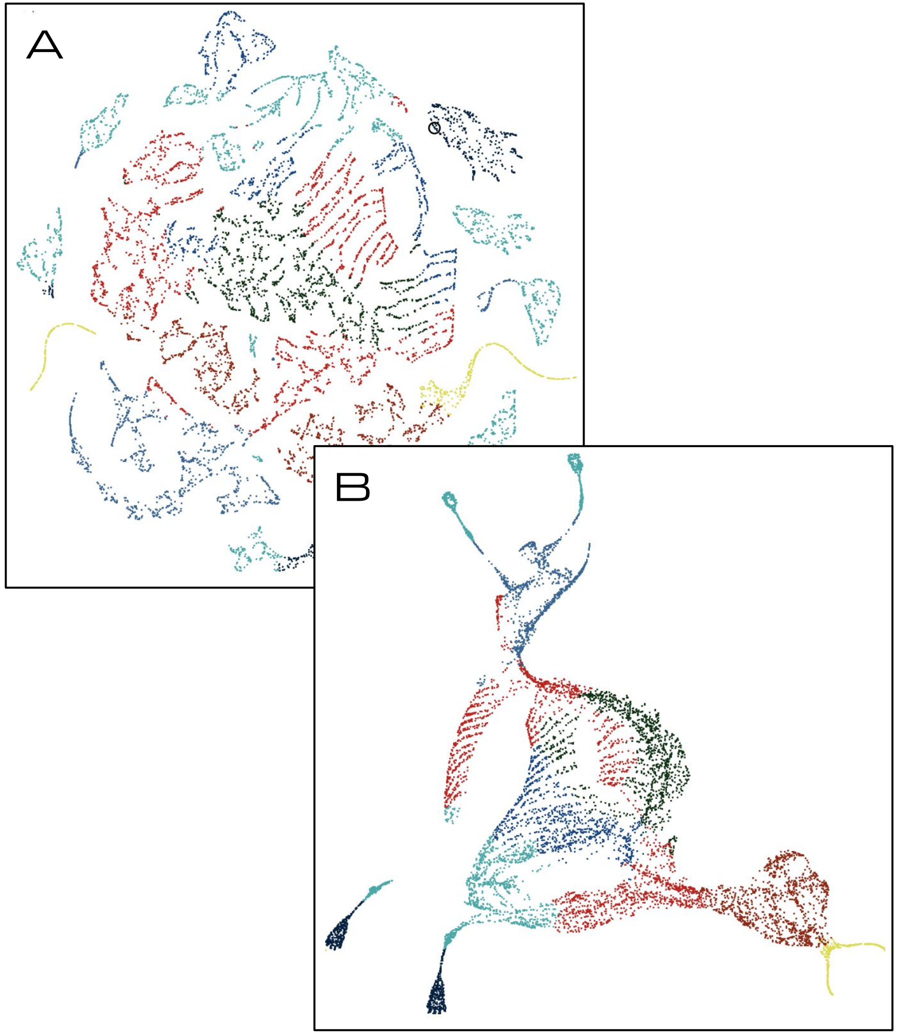
1. What does the perplexity parameter of t-SNE do? What trade-off is the parameter managing?



Warm Up

1. What does the perplexity parameter of t-SNE do? What trade-off is the parameter managing?

2. Both of these outputs are from applying t-SNE to the 3D plot on the previous slide. Do you think that output A or B used a higher value of perplexity? Why?



UP NEXT

Rationale of UMAP

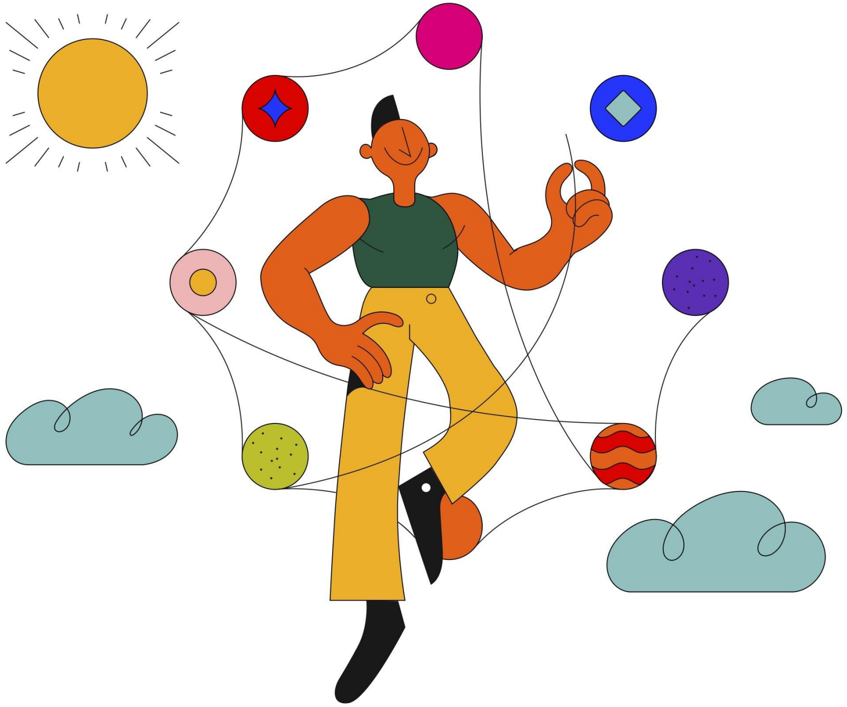
Why UMAP?

- Other methods might perform poorly on ‘real world’ data (LLE & Isomap), might be too slow (t-SNE), or might be unable to handle non-linear relationships (PCA).

- Uniform Manifold Approximation and Projection (UMAP) addresses the speed issues that come with t-SNE and seems to perform better at preserving global structure.

UP NEXT

Theory behind UMAP

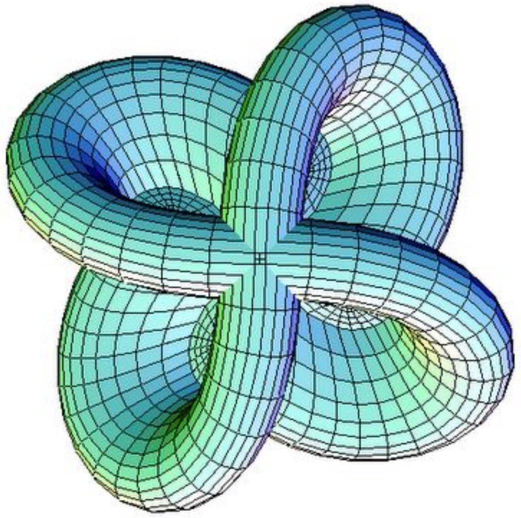


What and Why?

At a high level, the UMAP process can be thought of similarly to the t-SNE. That is, optimize a lower dimensional representation to be similar to the higher dimensional input data.

The big difference is that t-SNE is working with probability distributions, and UMAP is based in "topological" approach.

In taking this approach, UMAP processes faster than t-SNE and attempts to capture more global structure.

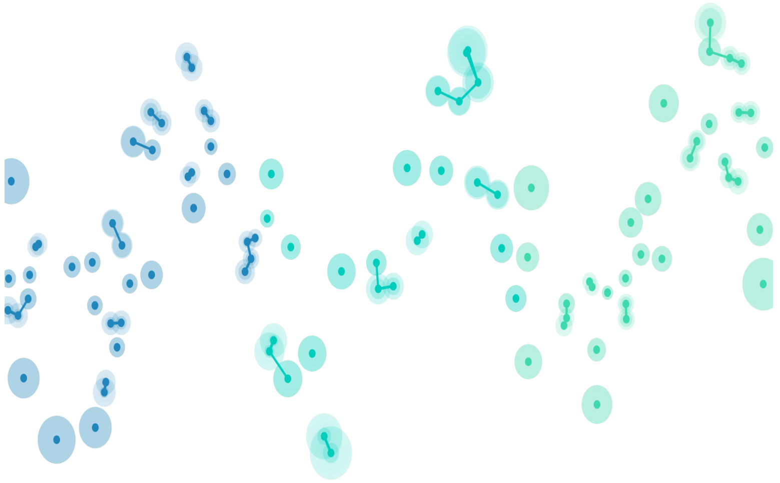


UMAP intuition

UMAP assesses the neighborhood of each datapoint by selecting a local radius for each data point, based on the point’s n nearest neighbors.

It then creates a fuzzy graph as it decreases the likelihood of connection among the data points, by increasing the aforementioned radius.

Finally, by setting the connectedness of each data point to its nearest neighbor as a requirement, it manages to preserve the local structure, all while ensuring that the global structure is preserved too to a large extent.



Source: UMAP applet on http://bit.ly/2U7KXXa

Underlying math framework

Just like t-SNE, UMAP makes use of graph theory for organizing the data into a lower dimensionality space

In particular, UMAP creates a high-dimensionality representation of the original dataset and then it configures a low-dimensionality representation that looks as similar to the original one as possible. The similarity has to do with the structural aspects of the dataset.

The math behind UMAP involves Riemannian geometry and Fuzzy Logic. It uses this math to configure a weighted graph, usually referred to as “fuzzy simplicial complex” that represents the original data.

QUESTION

Why is it important to preserve this global perspective / larger distances?

Specific metrics used in UMAP

To measure how similar the two representations are, the cross-entropy is used:

$$C((A,\mu),(A,\nu)) \triangleq \sum_{a\in A}\left(\mu(a)\log\left(\frac{\mu(a)}{\nu(a)}\right) + (1-\mu(a))\log\left(\frac{1-\mu(a)}{1-\nu(a)}\right)\right)$$

Where (A, μ) & (A, ν) are two fuzzy representations, α is a point in set A, and μ & ν the membership functions for each one of the fuzzy sets

In practice, we aim to minimize the following functions which is tied to the previous metric:

$$-\sum_{a\in A}(\mu(a)\log(\nu(a)) + (1-\mu(a))\log(1-\nu(a)))$$

Other requirements of UMAP

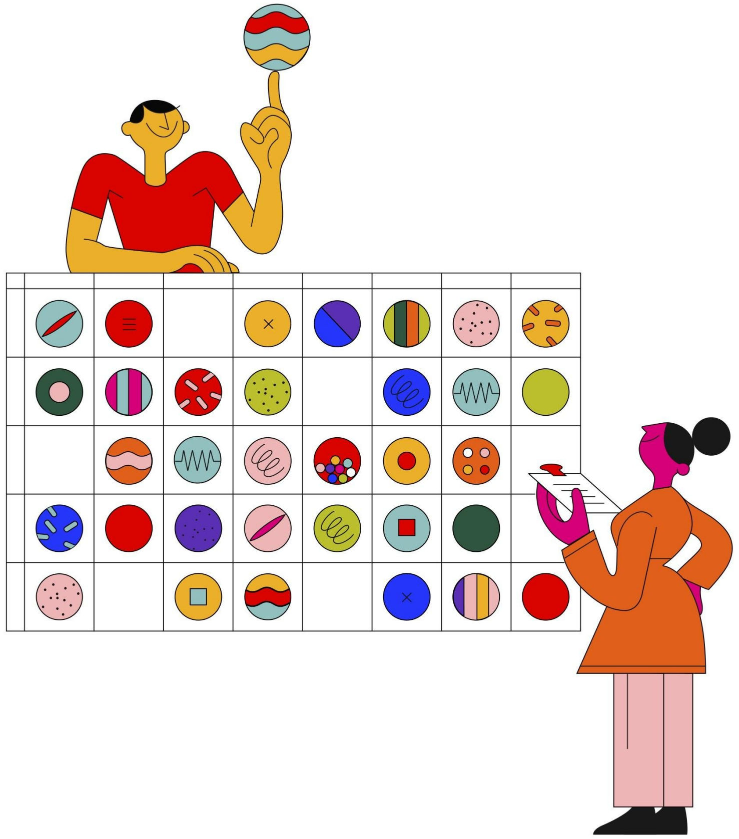
Just like other manifold based methods, UMAP needs to get a sense of the neighborhood of each data point to understand what the local structure of the dataset is like.

It does this using an approximate k-nearest neighbor calculation (Nearest Neighbor Descent algorithm).

In order to optimize the whole process, a couple of sampling methods are used, namely *probabilistic edge sampling* and *negative sampling*. This makes the process more efficient and removes the need of normalization beforehand.

UP NEXT

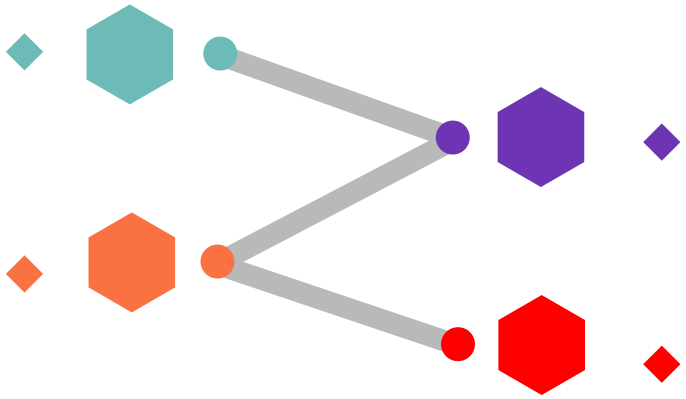
Parameters of UMAP



UMAP hyper-parameters

Number of neighbors to consider (n_neighbors)

Desired separation between close points in the embedding Space (min_dist)



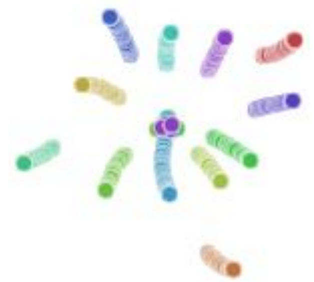
Number of embeddings (n_components)

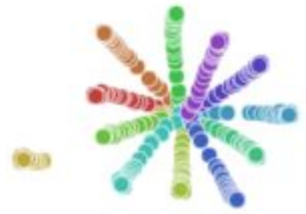
Number of iterations to use in opt. process (n_epochs)

n_neighbors parameter

- This parameter, along with `min_dist`, manages the local vs. global trade-off
  - Lower values => local structure emphasized (loss of global perspective)
  - Higher values => global structure more important (loss of detail)

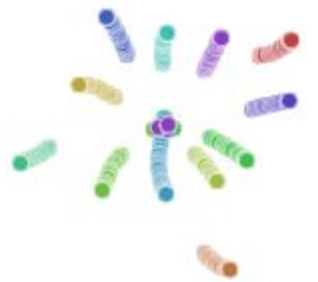
- It's best to play around with various values and visualize the results to find the best output for the task at hand.

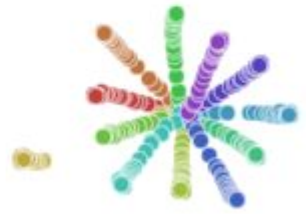

min_dist 0, n_neighbors 5


min_dist 0.1, n_neighbors 50

min_dist parameter

- This parameter has to do with the minimum distance between the points in the low-dimensional space
  - Low values => more tightly packed embeddings
  - High values => data points are packed more loosely (focus is on the preservation of the broad topological structure)


min_dist 0, n_neighbors 5


min_dist 0.1, n_neighbors 50

UP NEXT

Applying UMAP

UMAP applications

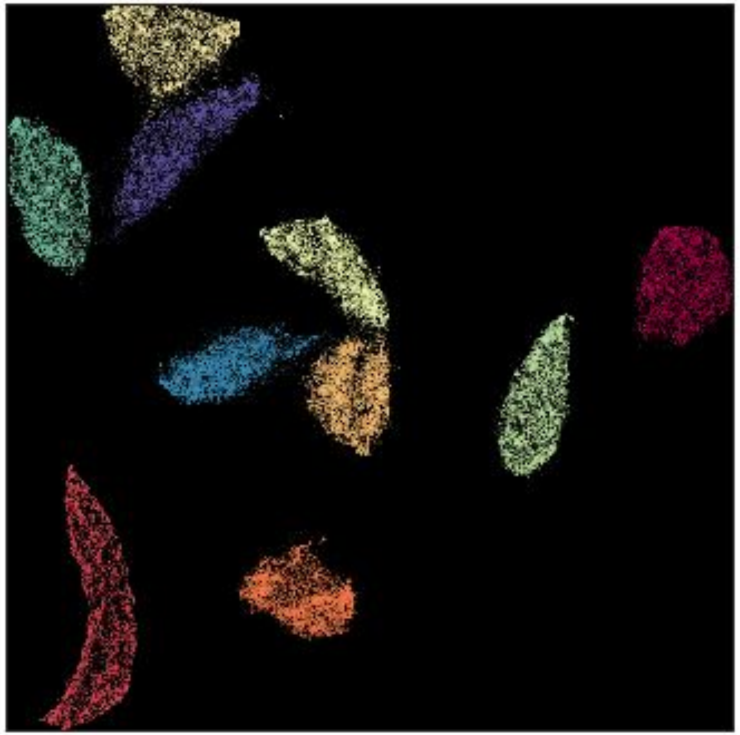
- All use cases of t-SNE and other manifold-based methods (in addition to larger datasets where t-SNE might fail)
- DR so as to obtain meta-features to use in a data model
  - Not just for visualization purposes; however, this is still the main use case.
  - For choosing the number of features you might optimize in a pipeline.
- Cases where interpretability is not a requirement (in cases like that PCA would be preferable)

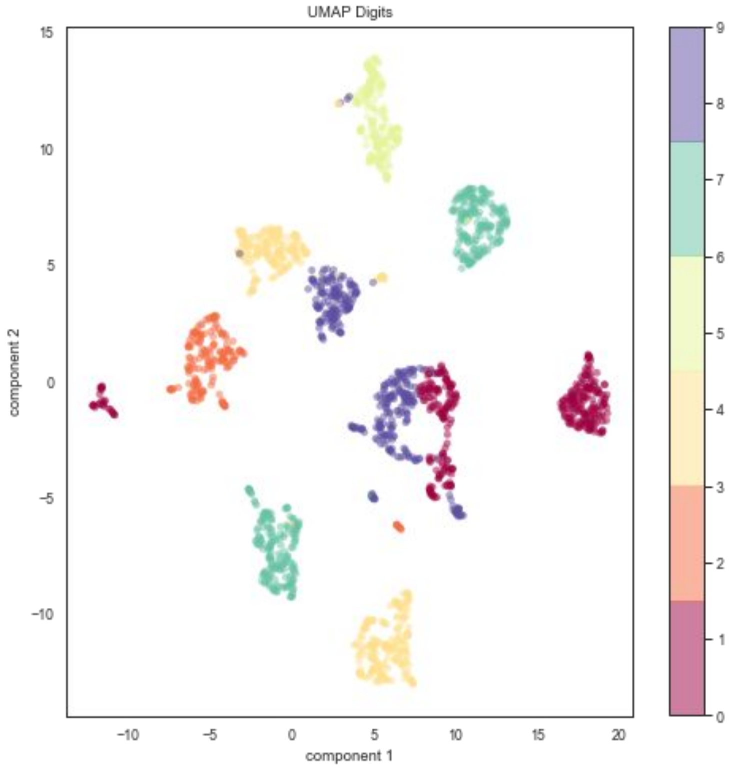
QUESTION

Would you use UMAP on a dataset with very high dimensionality (e.g. 100, 000 features)?

UMAP examples

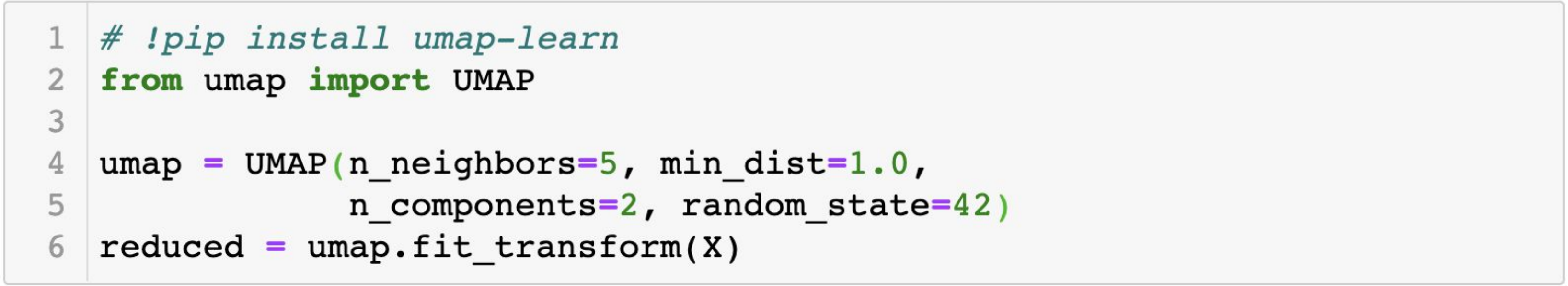
Fashion MNIST data embedded into two dimensions by UMAP visualised with Datashader





Applying UMAP in Python

- When applying, some parameters we can play with are:
  - `n_components`: number of dimensions to have in reduced space
  - `n_neighbors`: number of observations each point should be connected to in the graph
  - `min_dist`: how closely can points be placed together in the reduced space?
  - `metric`: the metric used for distance calculations (e.g. ‘euclidean’)



```python
# !pip install umap-learn
from umap import UMAP

umap = UMAP(n_neighbors=5, min_dist=1.0,
            n_components=2, random_state=42)
reduced = umap.fit_transform(X)
```

UP NEXT

Comparison with t-SNE and other DR methods

Advantages over t-SNE

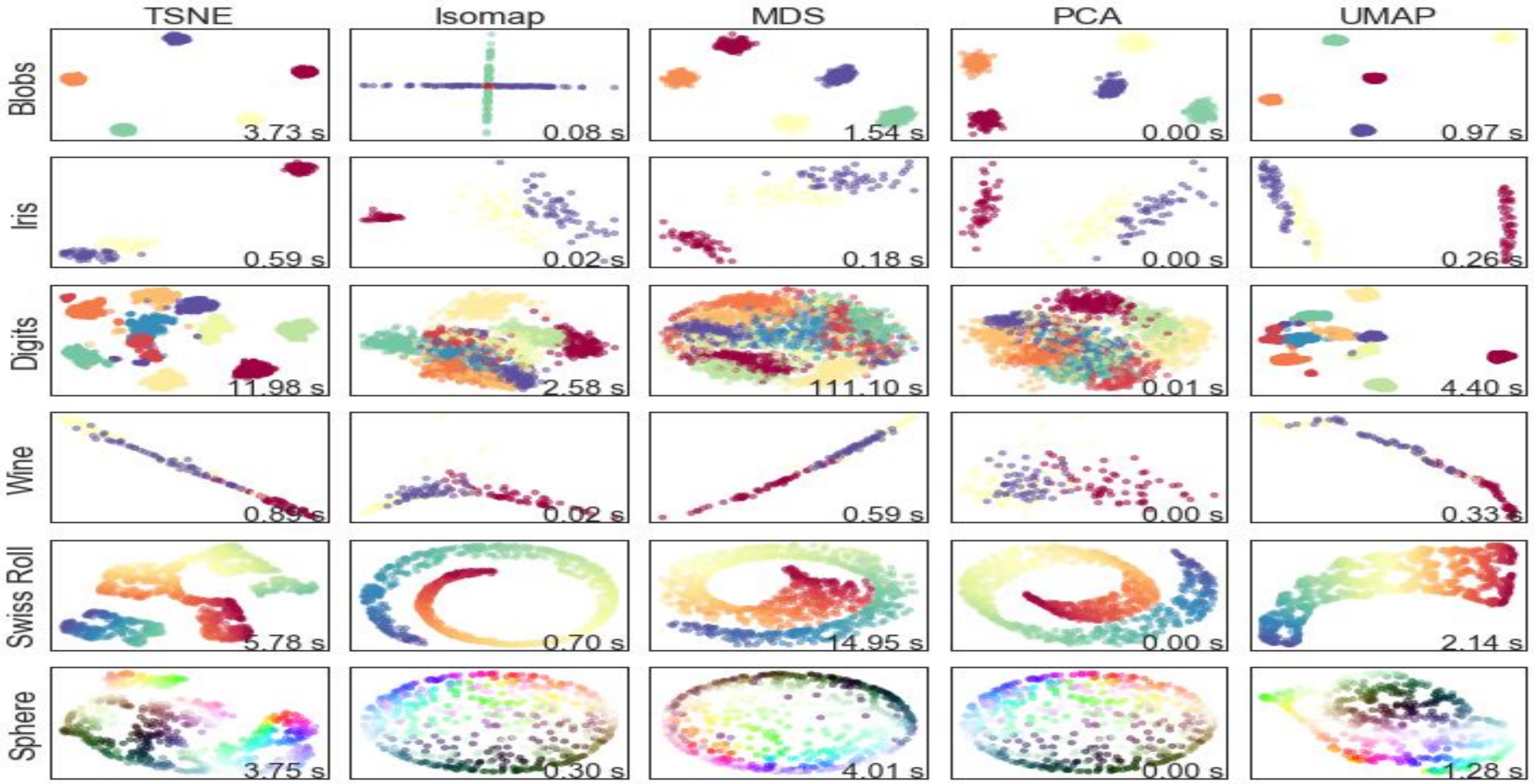
- UMAP can preserve the global structure of the dataset to some extent. This can make distances among outputted blobs / clusters more meaningful.

- UMAP is significantly faster than t-SNE & scales better.

- The projections (embedding values) of UMAP are more consistent between different runs of the algorithm, even if it is stochastic in nature, like t-SNE.

- UMAP supports various distance functions (e.g. correlation-based distance and cosine distance) instead of just standard distance metrics

Comparison with other methods



Summary

- The UMAP process can be thought of similarly to the t-SNE: optimize a lower dimensional representation to be similar to the higher dimensional input data.
    - UMAP's approach focuses on topology instead of joint probabilities
- UMAP scales better than t-SNE and several other manifold-based methods
- UMAP supports various distance functions, not just the standard ones
- UMAP can be considered for use as DR in a supervised learning task

Assignment

Dataset: Fashion-MNIST (Classification)
http://bit.ly/3b7ajLL
Given the data of the images of various fashion products, predictive analytics models are used to classify each image into one of the 10 categories.
1) Download the dataset
2) Apply the following DR methods to it: PCA, t-SNE, and UMAP for a reduction to 2 embeddings.
3) Experiment with different parameters (perplexity for t-SNE and number of neighbors and min-dist for UMAP). Make sure the number of iterations in both cases is sufficiently large (i.e. at least 1000)
4) Plot the reduced feature set in each case, using the optimal parameters
5) Note down your observations

THINKFUL

Thank You