# Uniform Manifold Approximation and Projection

Victor Tuekam

Supervisor: Fabian Scheipl

# Outline

1. Introduction
2. Algorithm
3. Examples
4. Comparison
5. Application
6. Weaknesses
7. Conclusion

# Introduction

Working with high dimensional data is quite hard:

- Difficult to interpret
- Hard to visualize
- Computationally expensive
- Curse of dimensionality
- ...

What can we do ? => Use **Manifold learning** for Dimensionality Reduction

# Introduction

**Manifold Learning**: A class of algorithms for recovering a low-dimensional manifold embedded in a high dimensional ambient space [1].

Manifold Learning techniques can be divided into two groups:

- Linear (e.g., PCA)
- Nonlinear (e.g., ISOMAP, UMAP)

This presentation focuses on **Uniform Manifold Approximation and Projection (UMAP)**

# Introduction

**UMAP** is a nonlinear manifold learning technique, originally published in 2018 by McInnes et al.

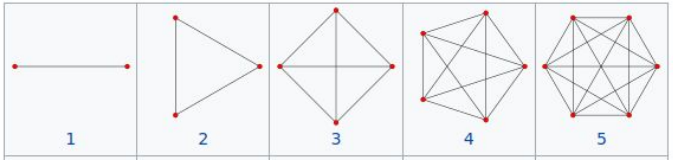
- It can be further categorized as a graph layout algorithm.
- It builds on ideas from Riemannian geometry and Algebraic topology.

# Algorithm

The UMAP algorithm consists of two phases:

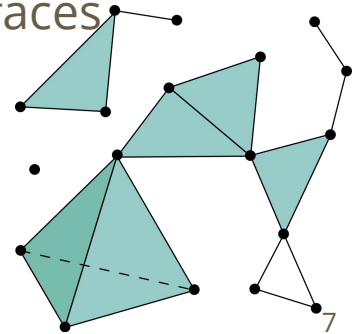
1. Build a weighted neighborhood graph of the topological space the data lives on.
2. Find a low dimensional representation that has a similar topological representation to the laid out graph.

# Algorithm (Phase 1)




**Some definitions:**

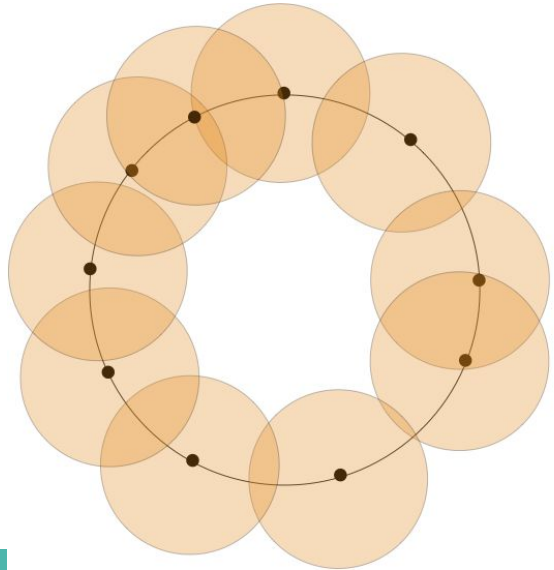
- **K-simplex**: A k-simplex is built by connecting k+1 points
- **Open cover**: An open cover is a family of open subsets of a space whose union is the whole space.
- **Simplicial complex**: A set of simplices glued together along faces

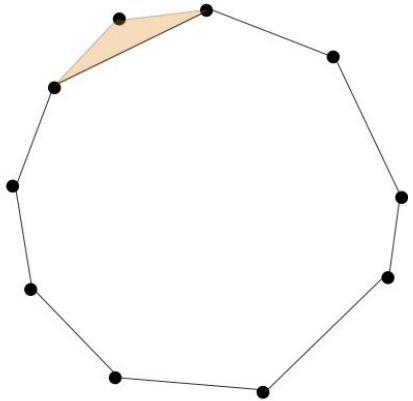


# Algorithm (Phase 1)

**Some definitions:**

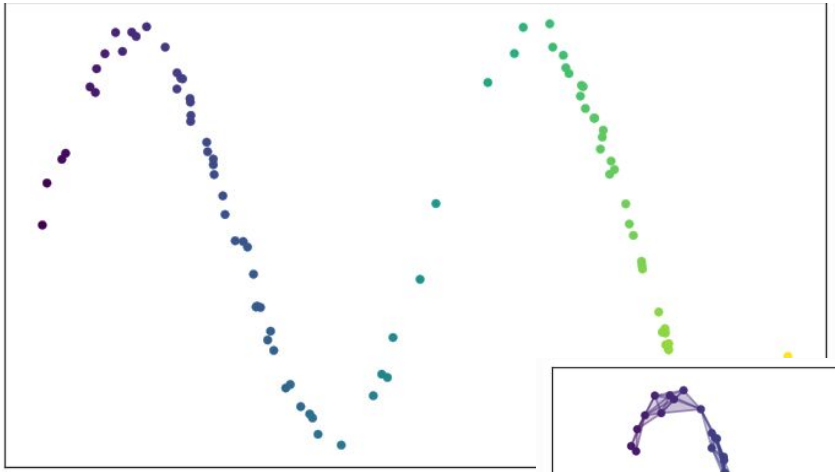
- **Čech complex**: A way of building a simplicial complex through the intersection of sets.



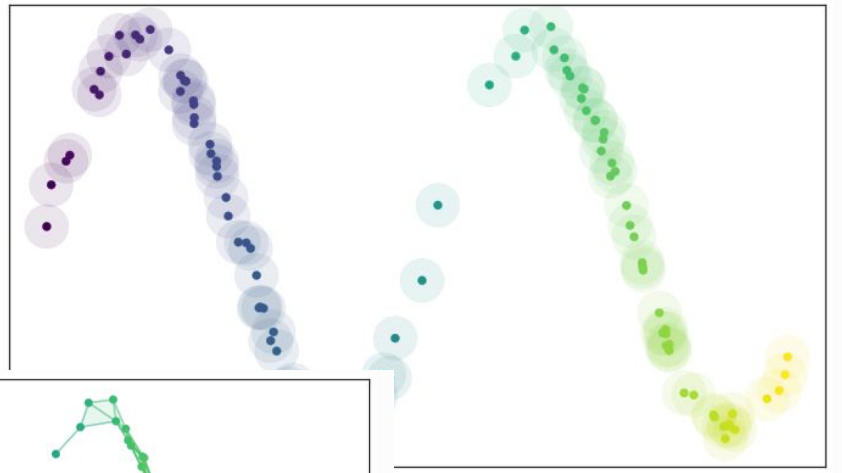


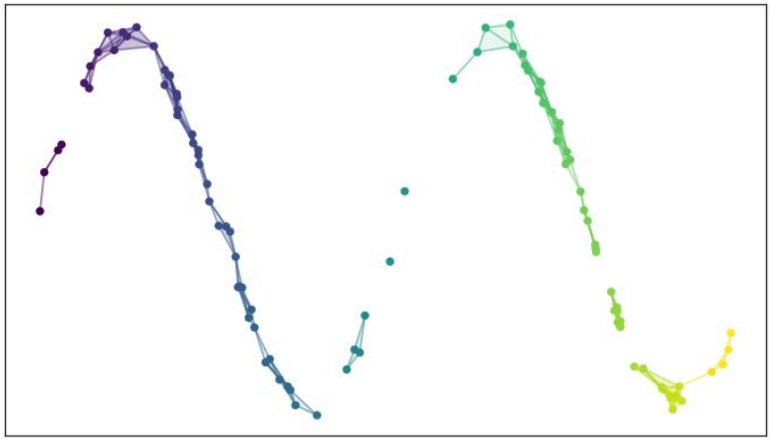
# Algorithm (Phase 1)

Data from some manifold (noisy sinusoidal)



cover (using unit balls)





Simplicial complex (through Čech complex).
This forms the neighborhood graph

Source: [2]

# Algorithm (Phase 1)

**Nerve theorem:**

$$Let\ U = \{U_i\}_{i \in I}\ be\ a\ cover\ of\ a\ topological\ space\ X.$$

$$If\ ,\ f\ or\ all\ \sigma \subset I,\ \bigcap_{i \in \sigma} U_i\ is\ either\ contractible\ or\ empty,$$

$$then\ N(U)\ is\ homtopically\ equivalent\ to\ X.$$

- Basically guarantees that building a simplicial complex as suggested in the previous slide, recovers the topological structure.
- The issue is that we have points from the space, not the total space.
- So what we produce is an approximation.

# Algorithm (Phase 1)

Practical issues:

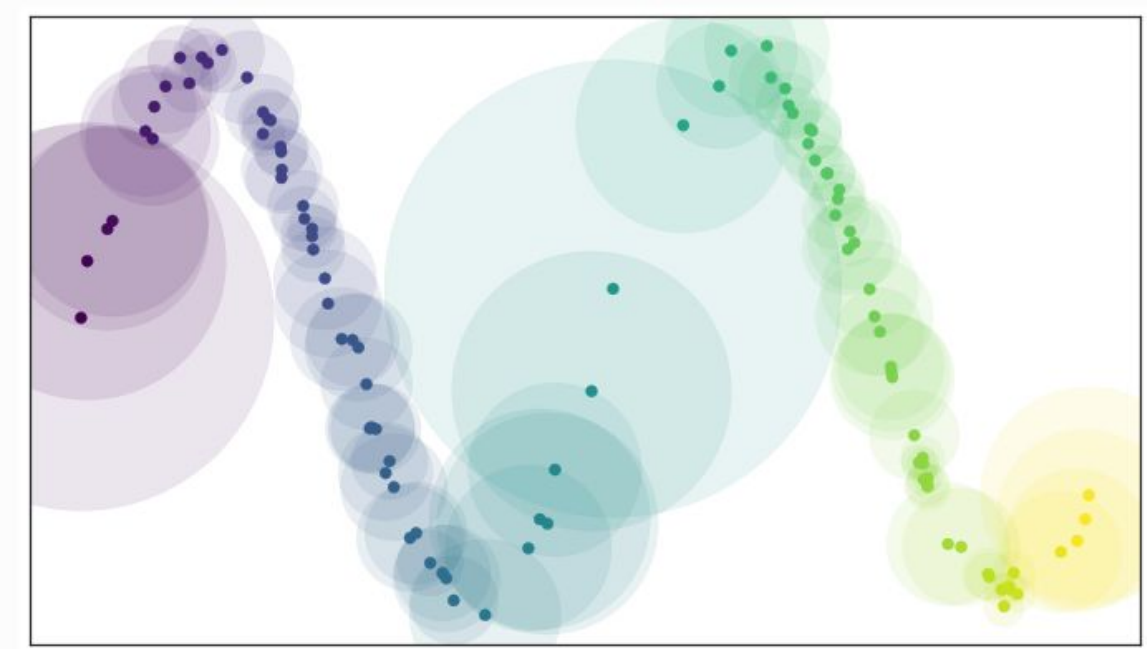
1. How do we choose the radius of the open ball ? (e.g., too high radius would lead to a few high dimensional simplices).
2. Data is generally unevenly distributed on the manifold.

**Solution:**

1. Assume even data distribution on the manifold
2. Give its own unique distance metric such that the balls are unit radii, such that each ball has the same amount of **k**-nearest neighbors => finding a good **k** is generally easier.
3. Manifold is locally connected => no point is completely isolated.

# Algorithm (Phase 1)

Open balls of radius = 1 using locally varying distance metrics. Source [2]



# Algorithm (Phase 1)

- Distances are now incompatible
- Distances between the points should form the weight of our graph
- We might then have up to two edges (directed) between two points (nodes)
- These are combined into a single edge with a single weight using a fuzzy union.
- Essentially $a + b - a * b$ , where a and b are the directed edge weights between nodes A and B.

# Algorithm (Phase 2)

- We have the weighted neighborhood graph
- Now we want a lower dimensional representation with a similar topological structure.
- This is an optimization problem using a particular loss function.

# Algorithm (Phase 2)

- UMAP uses the cross entropy loss:

$$\sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right)$$

$w_h$: *edge weight from high dimensional representation*

$w_l$: *edge weigh from low dimensional representation*

- Spectral embedding techniques are used to get a good initialization of the lower dimensional representation.

# Algorithm: Other UMAP variants

**Parametric UMAP**:  In phase 2, it uses a neural network to learn the relationship between the neighborhood graph and the low dimensional embedding.

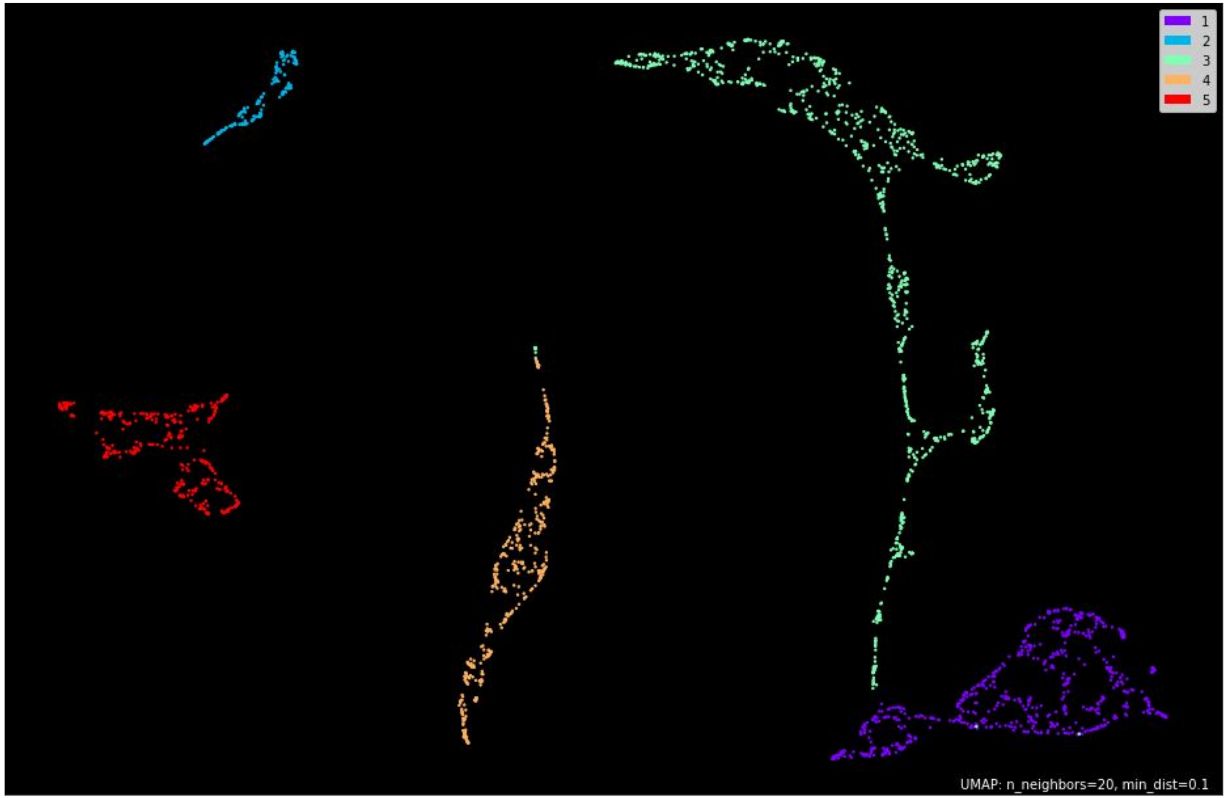
**DensMap:** Provides better preservation of the relative local density information of the data

# Algorithm (Some Hyperparameters)

- **n_neighbors:** number of neighbors used to compute the neighborhood graph.
- **min_dist:** minimum distance allowed between point in the lower dimensional embedding.
- **n_components:** target embedding dimension.
- **metric:** metric used to compute distances in the ambient space.
- **n_epochs:** number of training epochs.

# Examples: World data

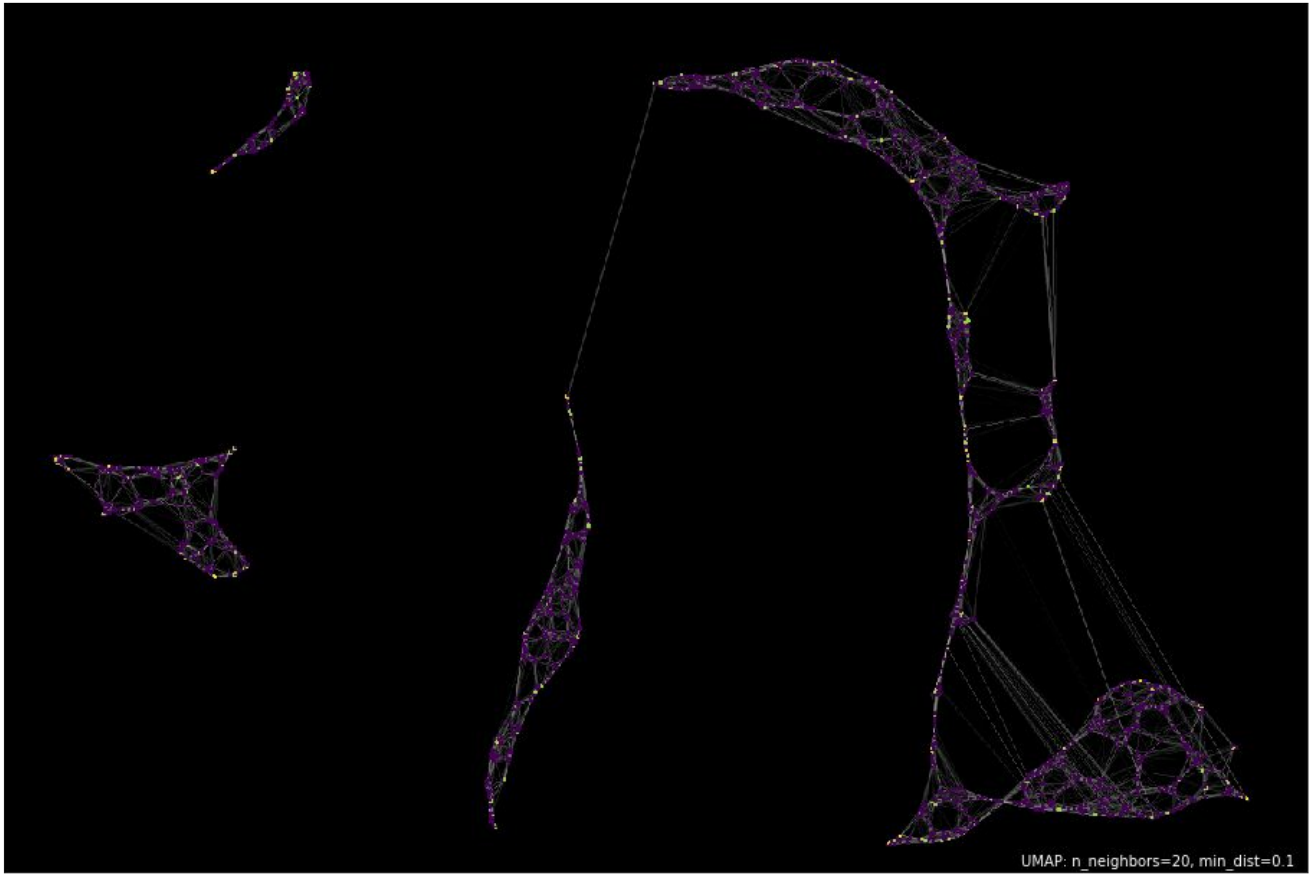
- 1: Africa
- 2: Australia
- 3: Eurasia
- 4: North america
- 5: south america



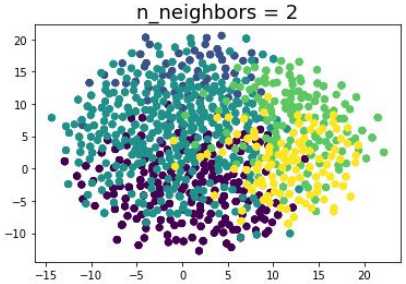

3d world data embedded into 2d space

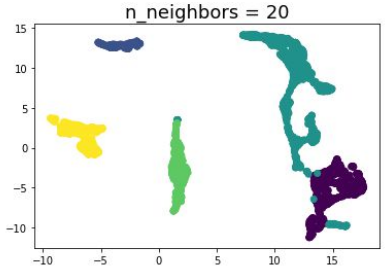
# Examples: World data

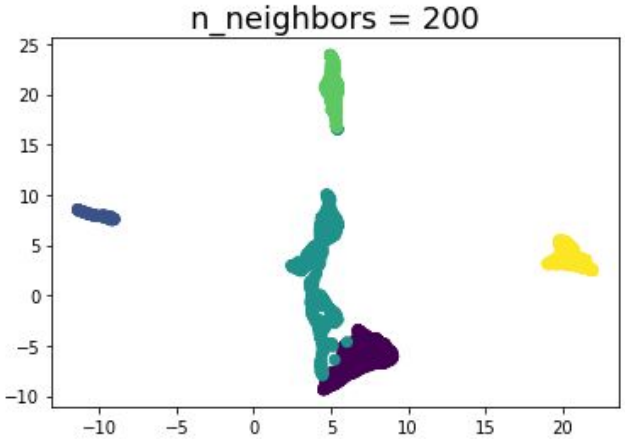
Manifold connectivity in embedding space

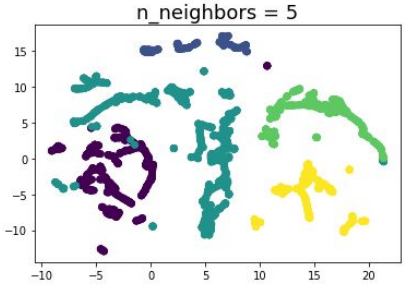


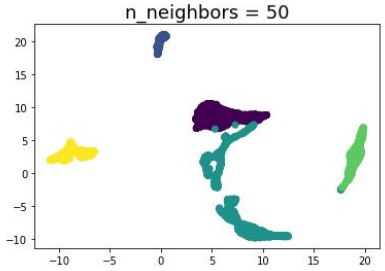
# Examples: World data (varying n_neighbors)

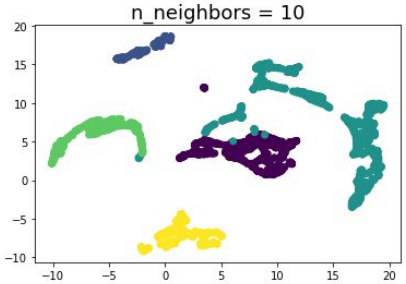


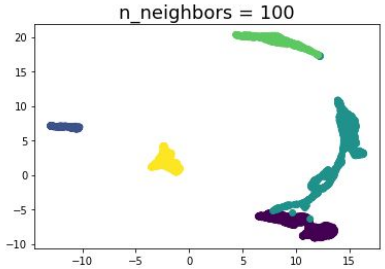



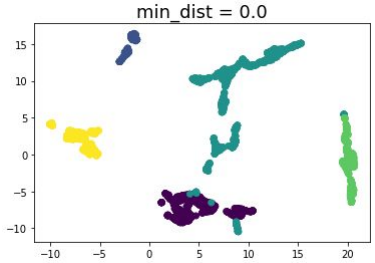



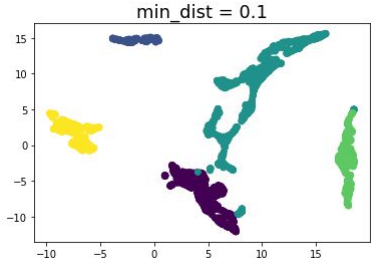



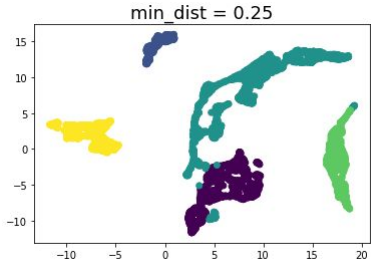



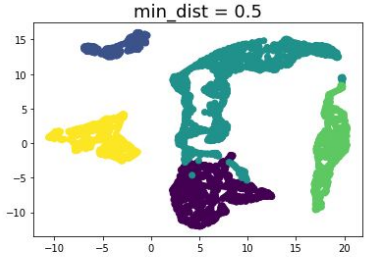

Tradeoff between global and local structure

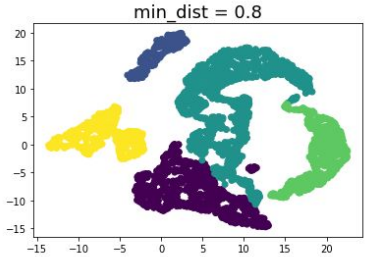
# Examples: World data (varying min_dist)

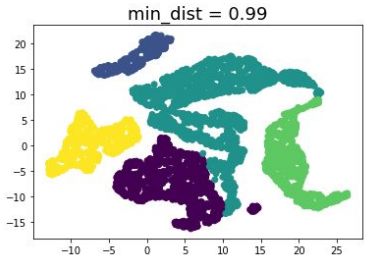


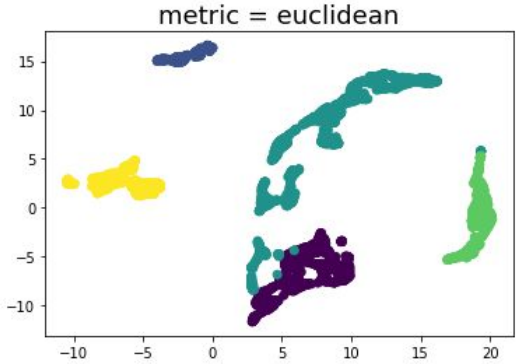



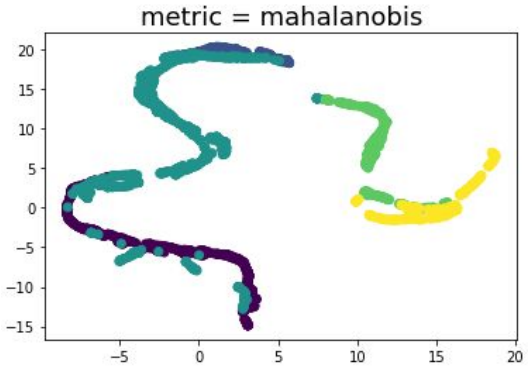



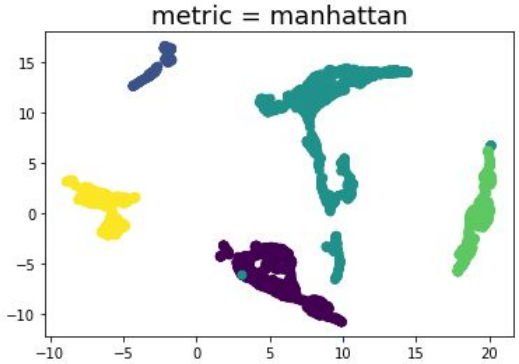



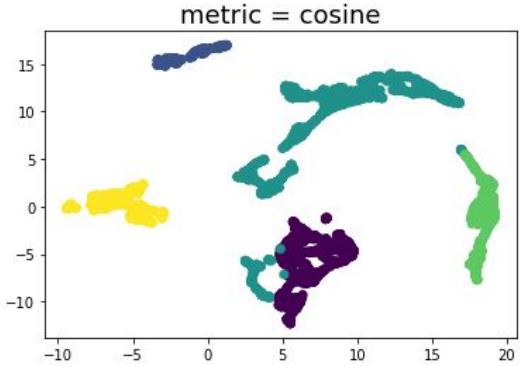

- Observe that as *min_dist* increases previously separated components become more and more connected.

# Examples: World data (varying metric)



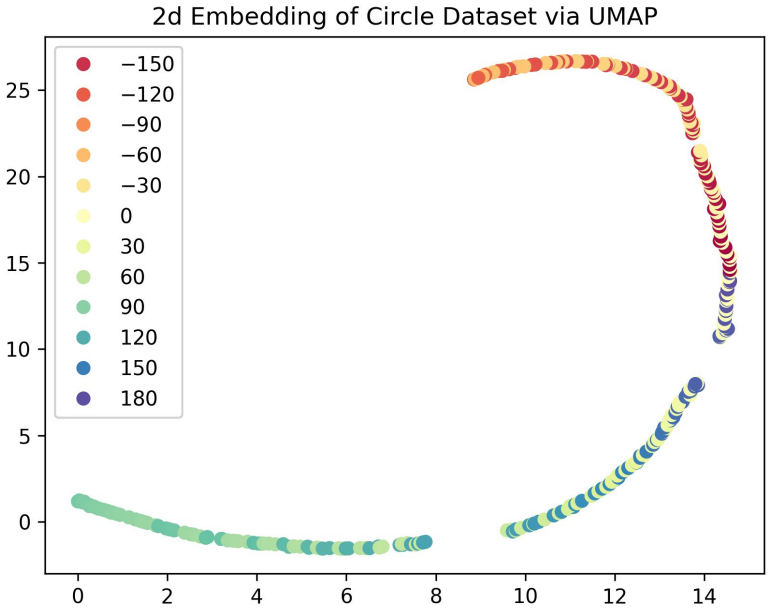



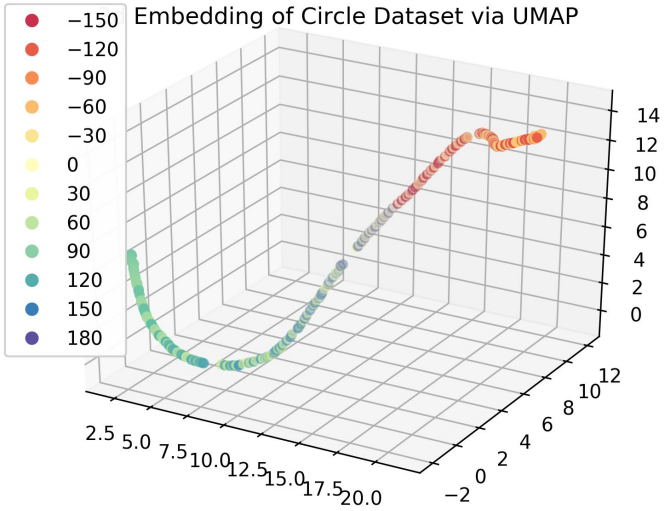



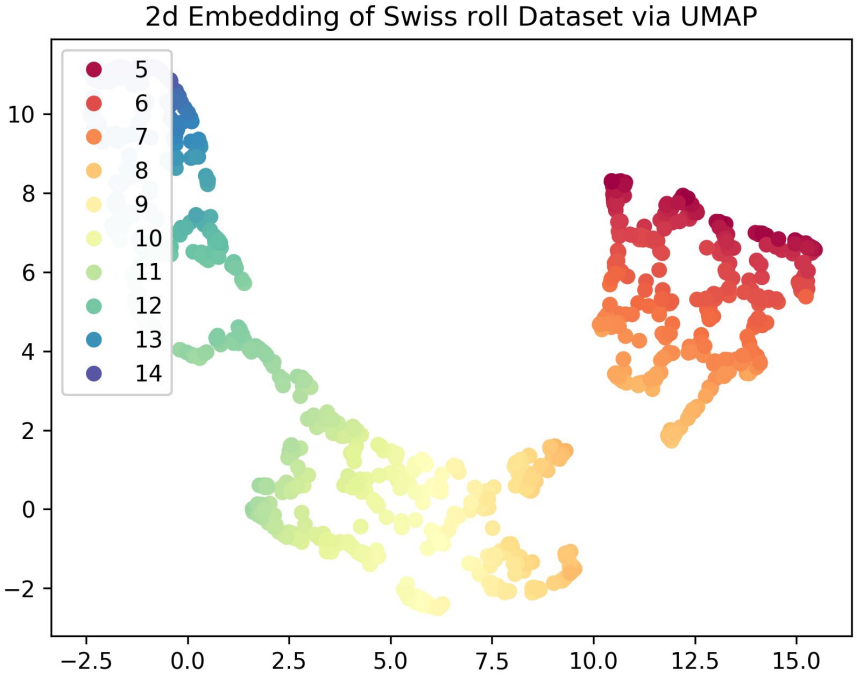

- Mahalanobis distance assumes a kind of normal spatial distribution. The embedding here is poor probably because this assumption doesn't hold for this dataset

# Examples: Other datasets



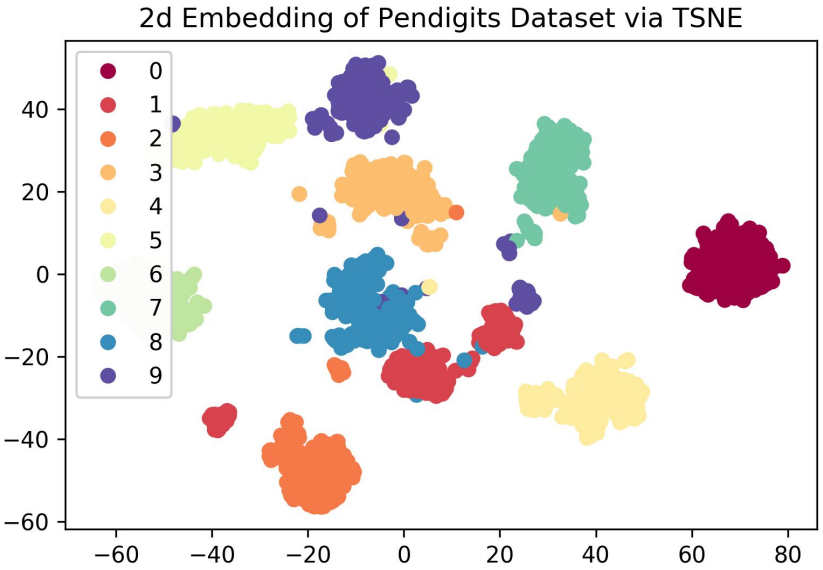



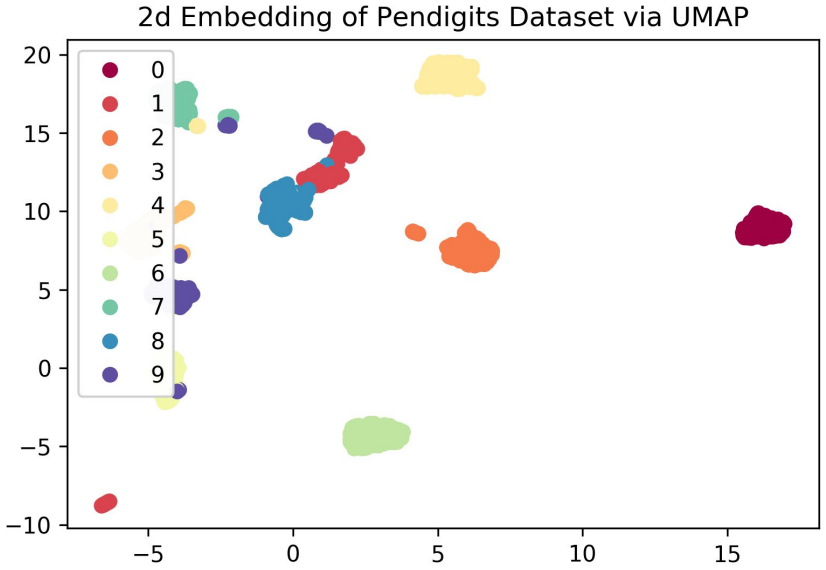
# Examples: Other datasets



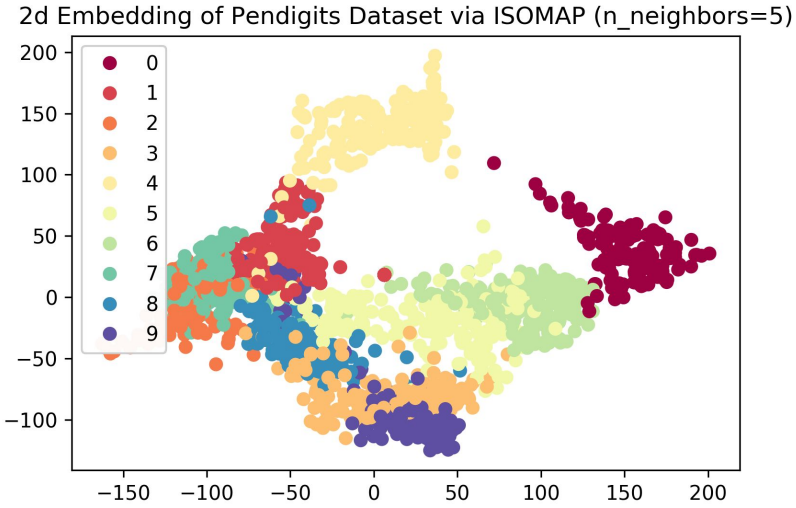
# Comparison (with TSNE)

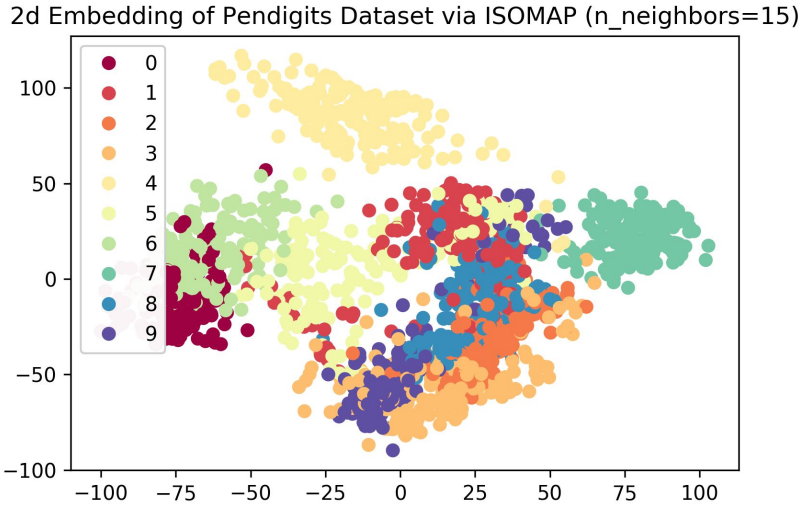
UMAP preserves more or the global structure
(clusters are better separated)



# Comparison (with ISOMAP)

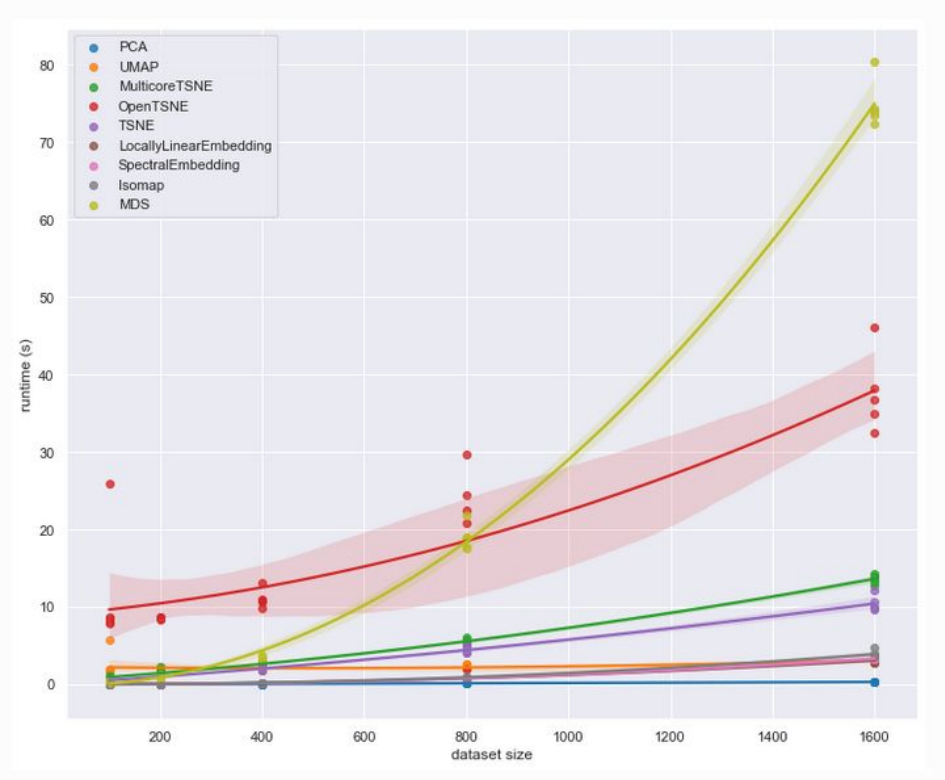


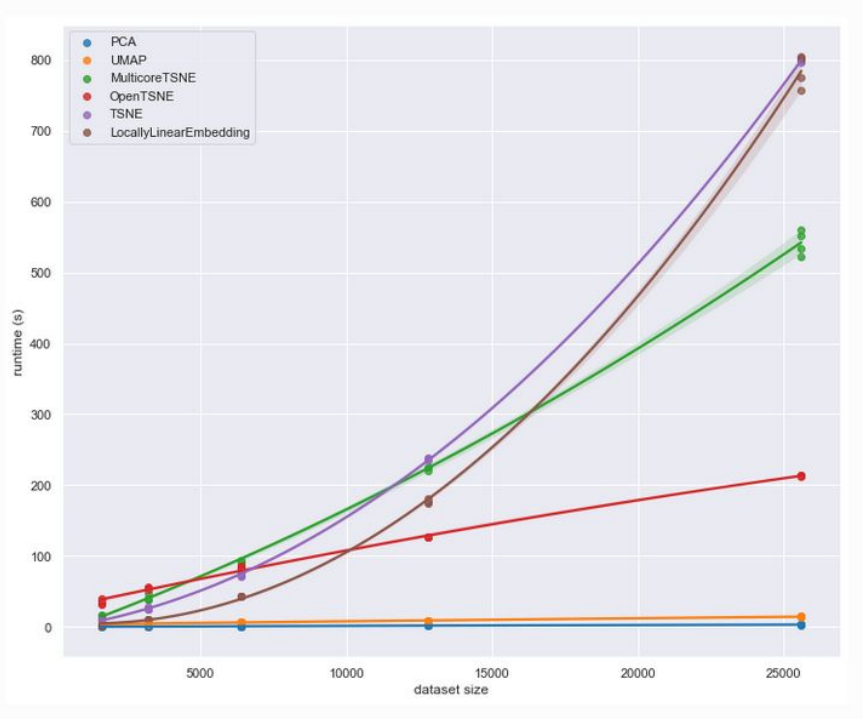




Very poor global structure preservation

# Comparison: Performance

Source: [3]



# Applications

- Dimensionality reduction
- Clustering
- Outlier detection
- Document embedding

# Weaknesses

- Dimension of the embedding are rather non-interpretable.
- UMAP would preserve noisiness in the data in the embedding.
- UMAP assumes local distances are more important than global distances.
- Due to its assumption of uniform data distribution, UMAP will put regions of high and low local densities on equal footing => use **DensMAP.**

# Conclusion

- UMAP is a nonlinear dimensionality reduction technique.
- It consists of two phases:
    a. Compute a graph representing your data
    b. Find a low dimensional representation of this graph by optimizing an objective function.
- UMAP is fast and scalable
- UMAP is most suitable for global structure preservation
- Some of its assumptions account for its weakness

# References

[1] Yunqian Ma, Yun Fu. Manifold Learning Theory and Applications, 2012.

[2] L. McInnes. How umap works - umap 0.5 documentation. URL https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

[3] L. McInnes. Performance comparison of dimensional reduction implementations. URL https://umap-learn.readthedocs.io/en/latest/performance.html