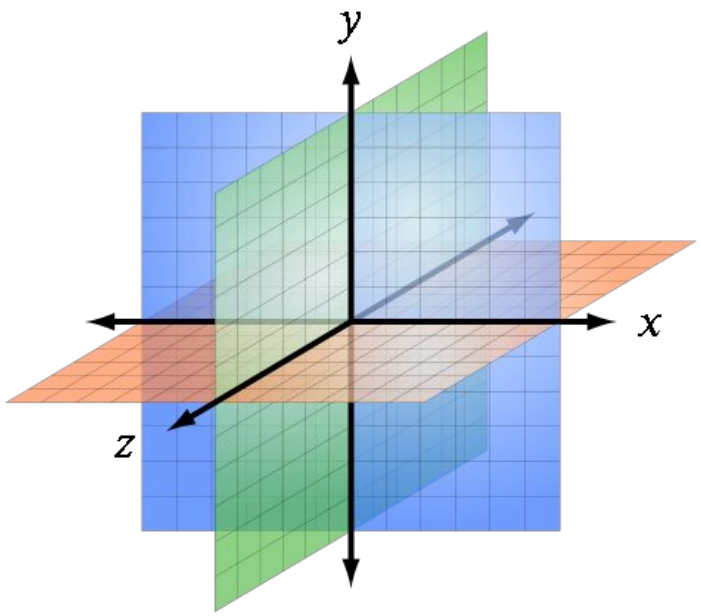
Dimensionality reduction with t-SNE

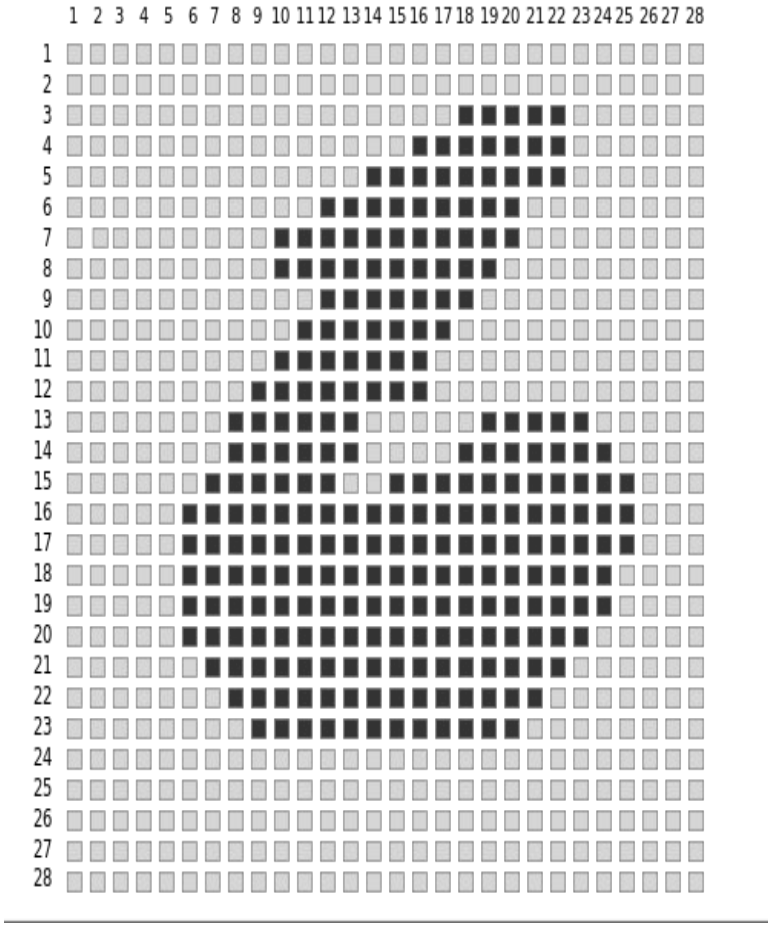
Zaur Fataliyev

Slides

- Dimensionality
- Dimensionality Reduction
- Taxonomy of dimensionality reduction techniques
- Principal Component Analysis
- PCA on MNIST
- Why manifold learning?
- Underlying idea of t-SNE
- Stochastic Neighbor Embedding
- Symmetric SNE
- t-Distribution
- t-Distributed Stochastic Neighbor Embedding
- Gradients of various types of SNE
- t-SNE Algorithm
- Results: MNIST
- Results: Olivetti
- Results: COIL-20
- Python Implementation of t-SNE
- References

Dimensionality





Dimensionality Reduction

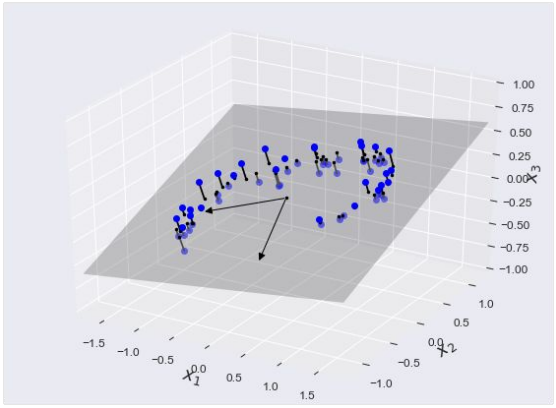
Dimensionality Reduction aims to map the data from the original dimension space to lower dimension space while minimizing information loss.

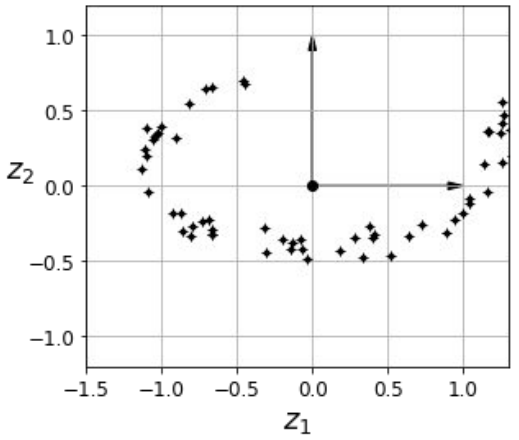
Reduce number of features for (un)supervised learning

- Feature selection or feature engineering
- Detecting intrinsic dimensionality

Lower computational demand

- Lower memory footprint
- Compression, scalability

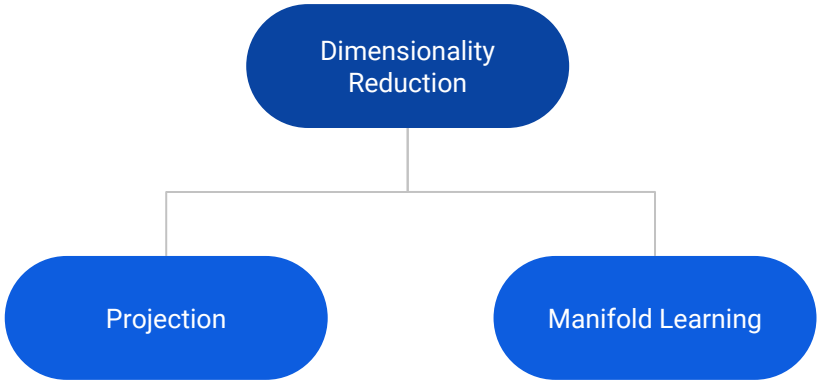




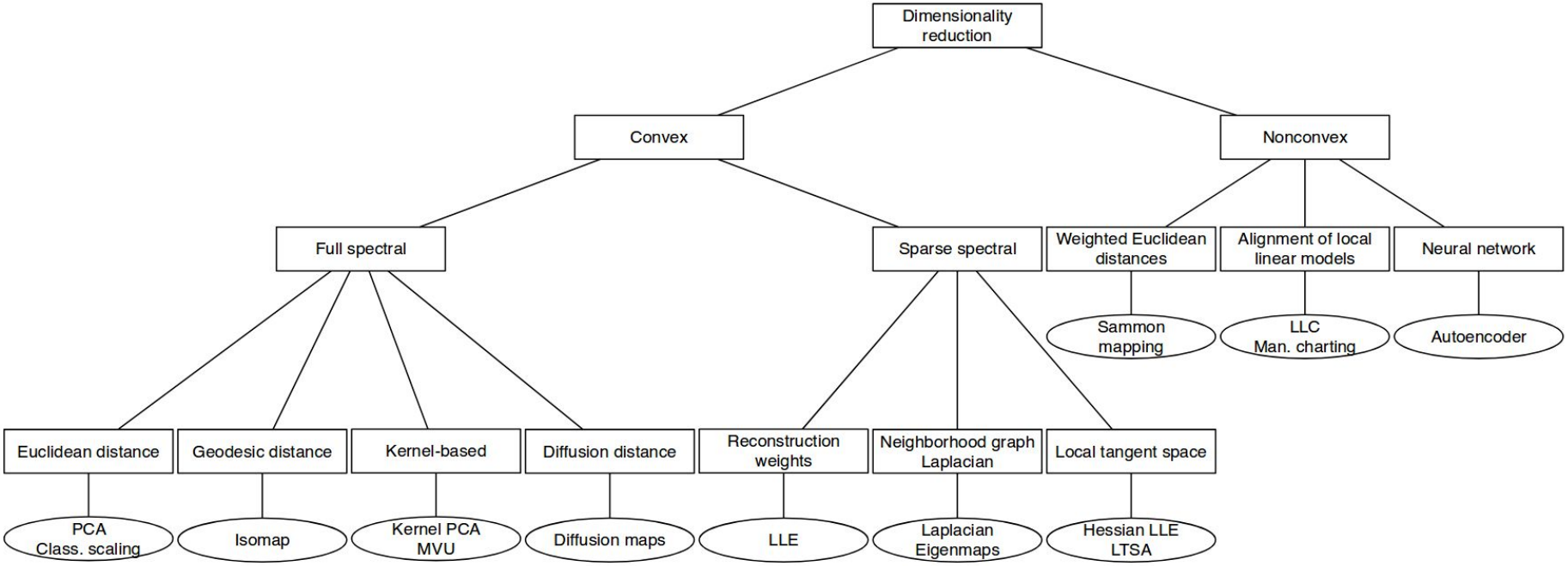
Dimensionality Reduction

There are many techniques for dimensionality reduction. They can be grouped into two general approaches.

- Projection: projecting high dimensional data into lower dimensional space.
  - Linear mapping
  - Examples: PCA, LDA, NMF
- Manifold Learning: modeling the manifold on which the training data lie
  - Nonlinear mapping
  - Examples: SNE, t-SNE, Autoencoder, Isomaps

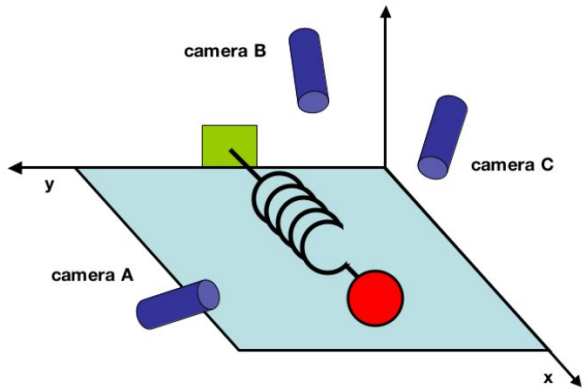


Taxonomy of dimensionality reduction techniques



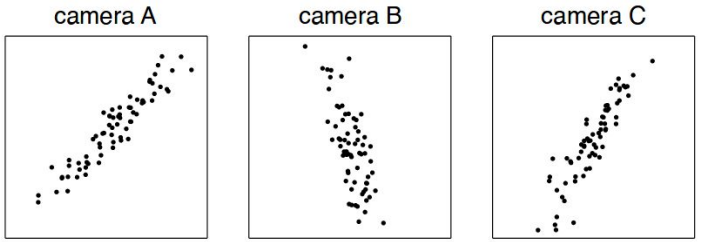
Principal Component Analysis (PCA)

1. **Normalize the data**
2. **Calculate the covariance matrix.**
3. **Find the eigenvectors of the covariance matrix.**
4. **Translate the data to be in terms of the components**.



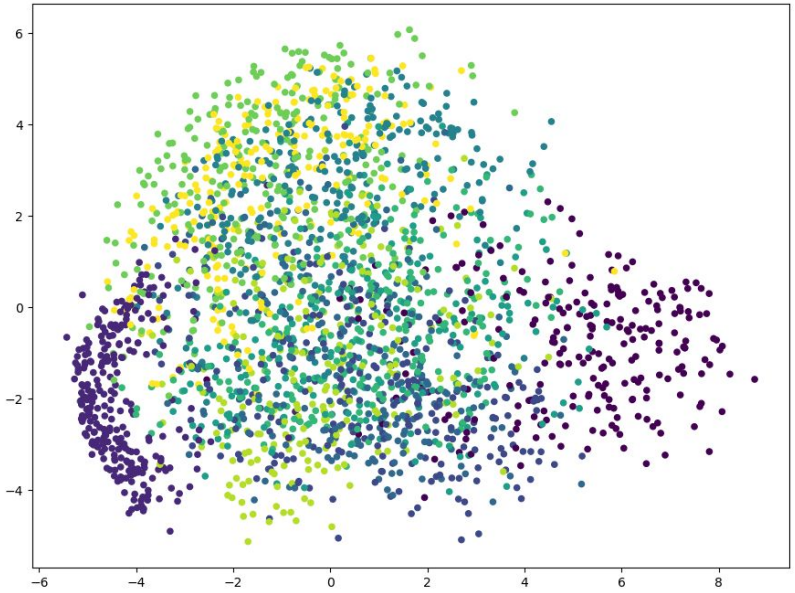

```
def pca(X=np.array([]), no_dims=50):
    """
    Runs PCA on the NxD array X in order to reduce its
    dimensionality to no_dims dimensions.
    """

    print("Preprocessing the data using PCA...")
    (n, d) = X.shape
    X = X - np.tile(np.mean(X, 0), (n, 1))
    (l, M) = np.linalg.eig(np.dot(X.T, X))
    Y = np.dot(X, M[:, 0:no_dims])
    return Y
```

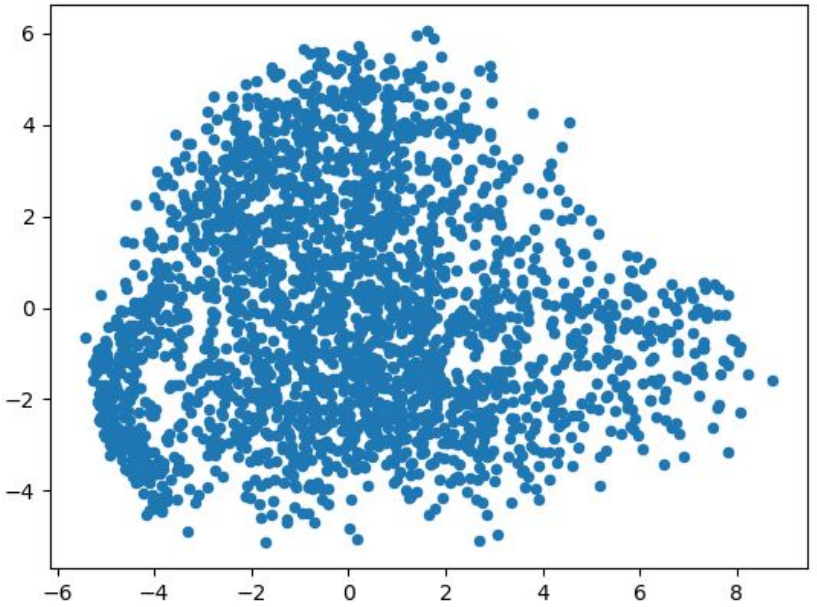


https://giphy.com/gifs/pca-Lyejb62QjQepG/fullscreen

PCA on MNIST



Visualization with labels



Visualization without labels

Why manifold learning?

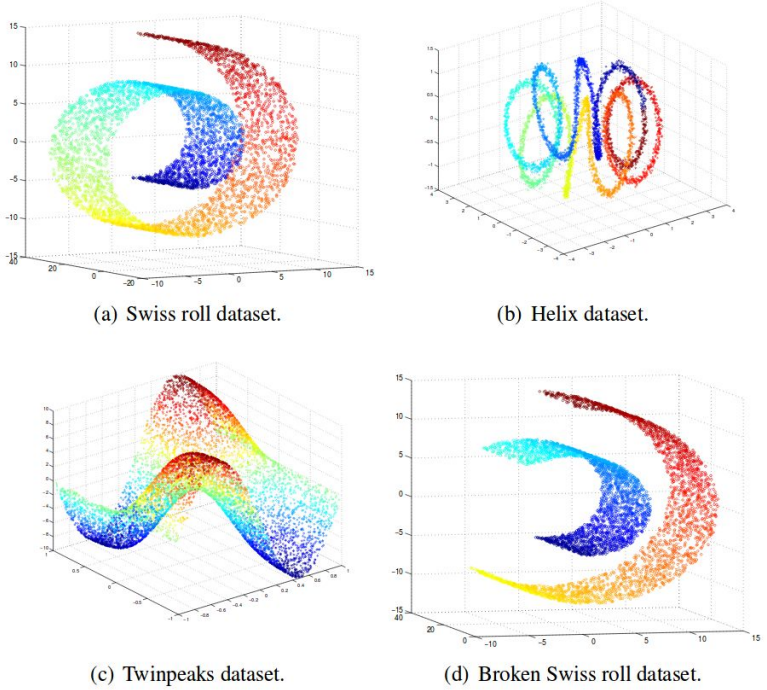
Why PCA fails to properly reduce dimensions of MNIST?

- PCA is good, but it is a linear algorithm, meaning that it cannot represent complex relationship between features

t-SNE is non-linear dimensionality reduction technique that has better performance. It is designed for visualization purposes.

Why not use Neural Networks?

- There is a dimensionality reduction technique based on Neural Network called Autoencoder!



(a) Swiss roll dataset. (b) Helix dataset.

(c) Twinpeaks dataset. (d) Broken Swiss roll dataset.

Good visualization

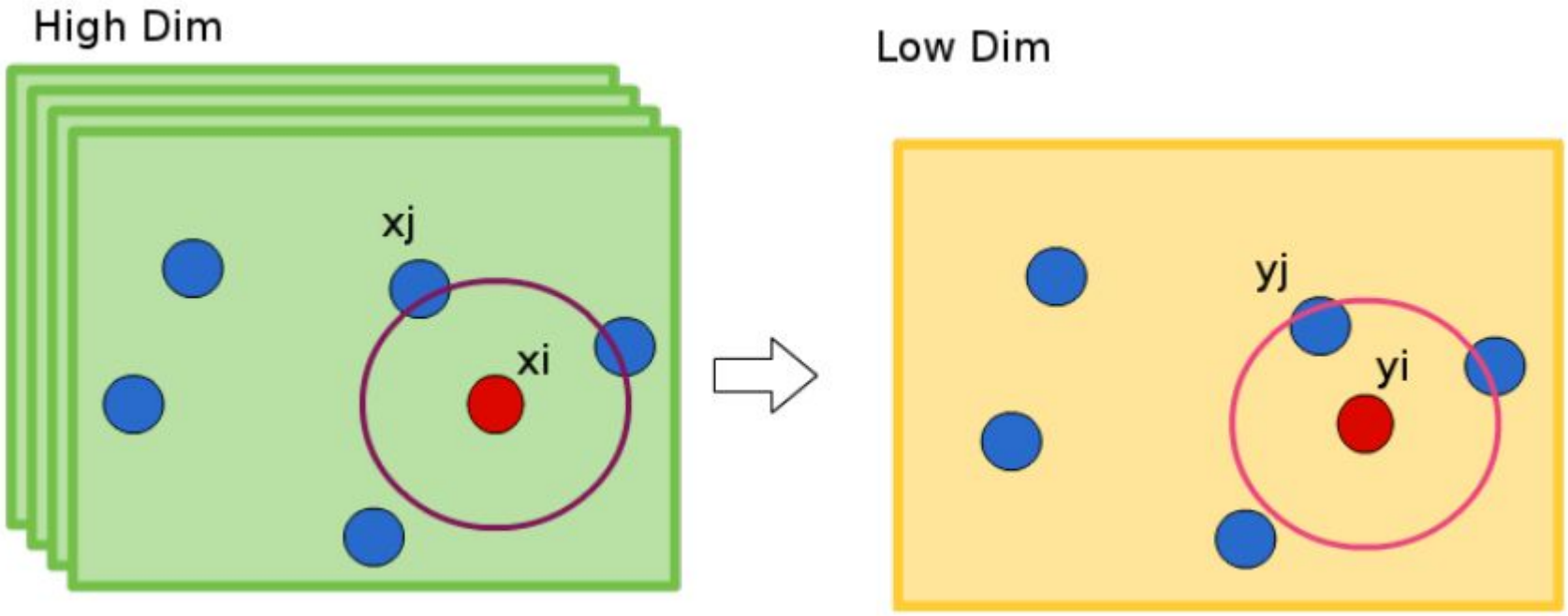
Patterns

- Discover natural clusters
- Linear relationships
- Visualize embeddings

Technical Requirements

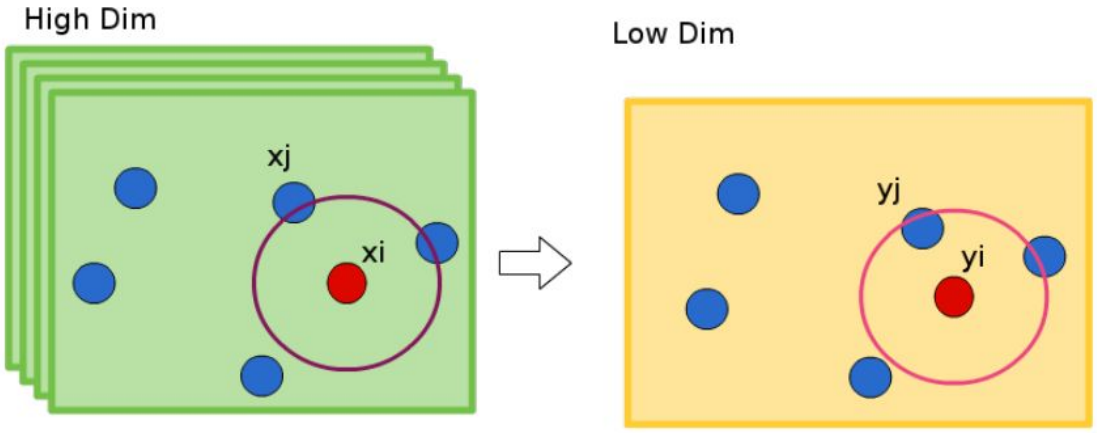
- Each high dimensional object is represented by a low-dimensional object
- Preserve the neighborhood
- Distant points correspond to dissimilar objects
- Scalability: large, high-dimensional data sets

Underlying idea of t-SNE



Stochastic Neighbor Embedding

Measure pairwise similarities between high-dimensional and low-dimensonal objects



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

Stochastic Neighbor Embedding

Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities

- Similarity of datapoints in High Dimension

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

- Similarity of datapoints in Low Dimension

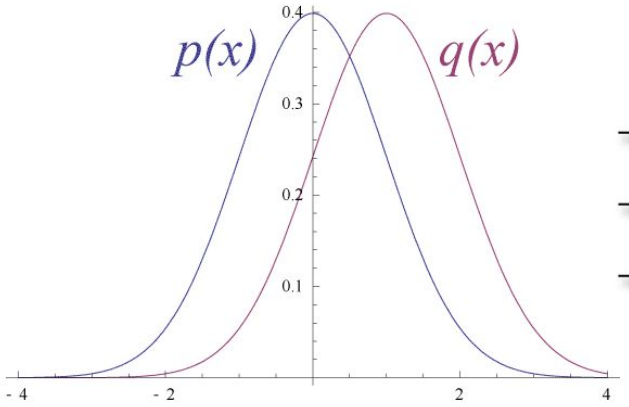
$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

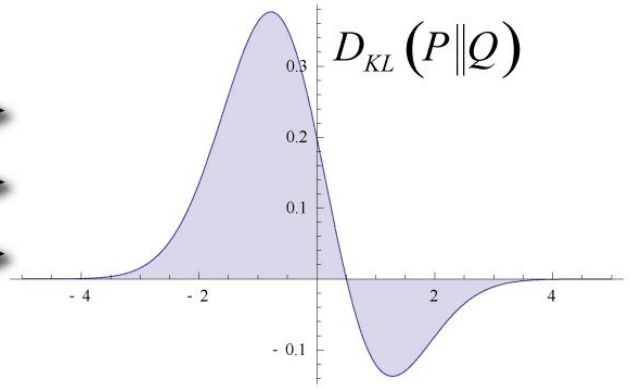
- Cost function

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

KL Divergence

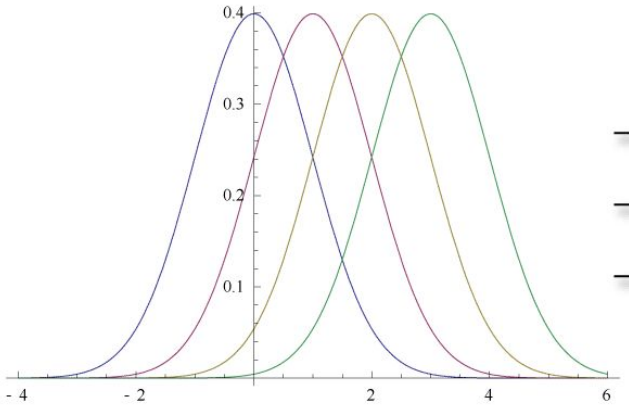
Measures the similarity between two probability distributions & it is asymmetric

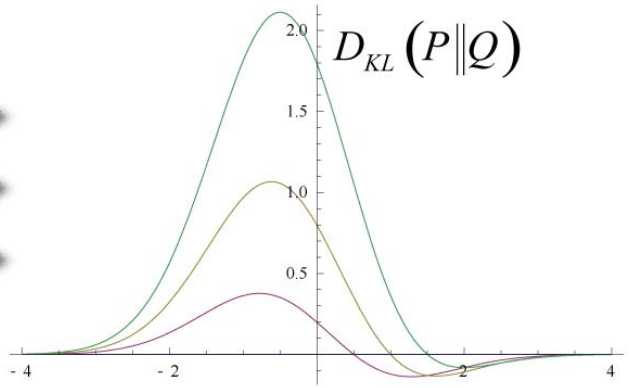




Original Gaussian PDF's

KL Area to be Integrated





$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i)\,\log\frac{P(i)}{Q(i)}.$$

Stochastic Neighbor Embedding

Gradient has a surprisingly simple form

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

The gradient update with momentum term is given by

$$Y^{(t)} = Y^{(t-1)} + \eta \frac{\partial C}{\partial y_i} + \beta(t)(Y^{(t-1)} - Y^{(t-2)})$$

Derivation of Gradient is given in paper [1]

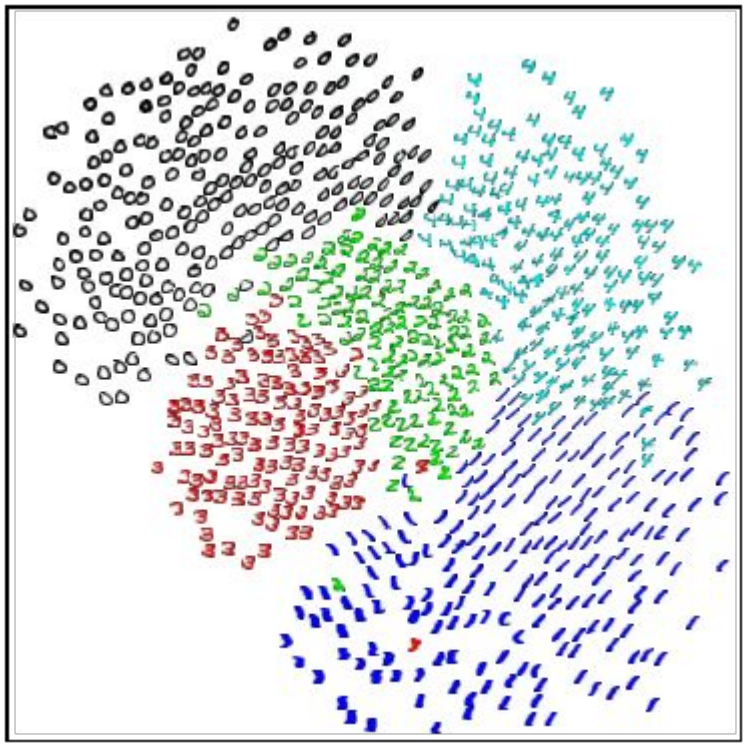
Stochastic Neighbor Embedding

The result of running the SNE algorithm on 3000 256-dimensional grayscale images of handwritten digits.

Pictures of the original data vectors xi (scans of handwritten digit) are shown at the location corresponding to their low-dimensional images yi as found by SNE.

The classes are quite well separated even though SNE had no information about class labels. Furthermore, within each class, properties like orientation, skew and strokethickness tend to vary smoothly across the space.

Not all points are shown: to produce this display, digits are chosen in random order and are only displayed if a 16 x 16 region of the display centered on the 2-D location of the digit in the embedding does not overlap any of the 16 x16 regions for digits that have already been displayed.



Symmetric SNE

- Minimize the sum of the KL divergences between the conditional probabilities

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

- Minimize a single KL divergence between a joint probability distribution

$$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

- The obvious way to redefine the pairwise similarities is

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

$$q_{ij} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq l} exp(-||y_l - y_k||^2)}$$

Symmetric SNE

Such that $p_{ij} = p_{ji}, q_{ij} = q_{ji}$, the main advantage is simplifing the gradient

$$\frac{\partial C}{\partial y_i} = 2 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

However, in practice we symmetrize (or average) the conditionals

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

Set the bandwidth σ_i such that the conditional has a fixed perplexity (effective number of neighbors) $Perp(P_i) = 2^{H(P_i)}$, typical value is about 5 to 50

t-Distribution

Use heavier tail distribution than Gaussian in low-dim space, we choose

$$q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$$

Then the gradient could be

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

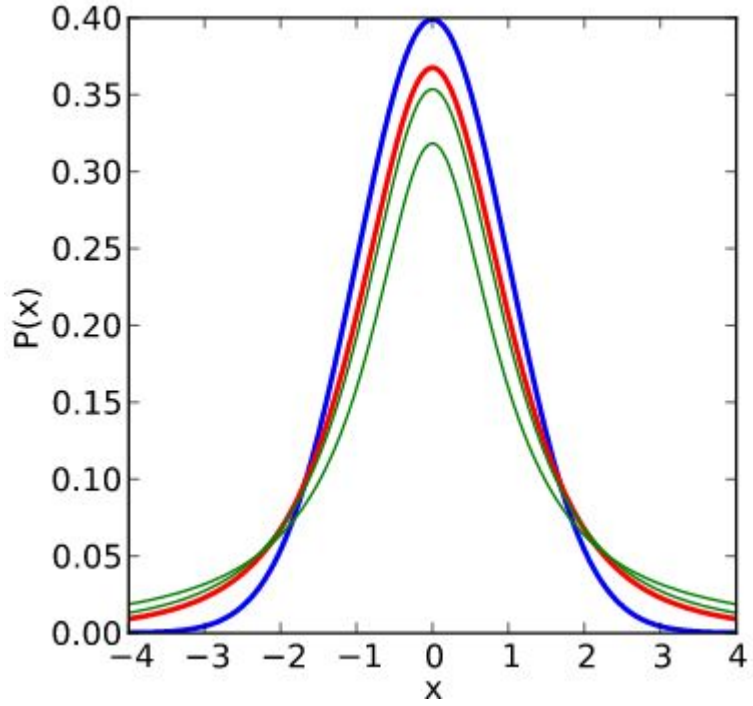
Why Student-t Distribution?

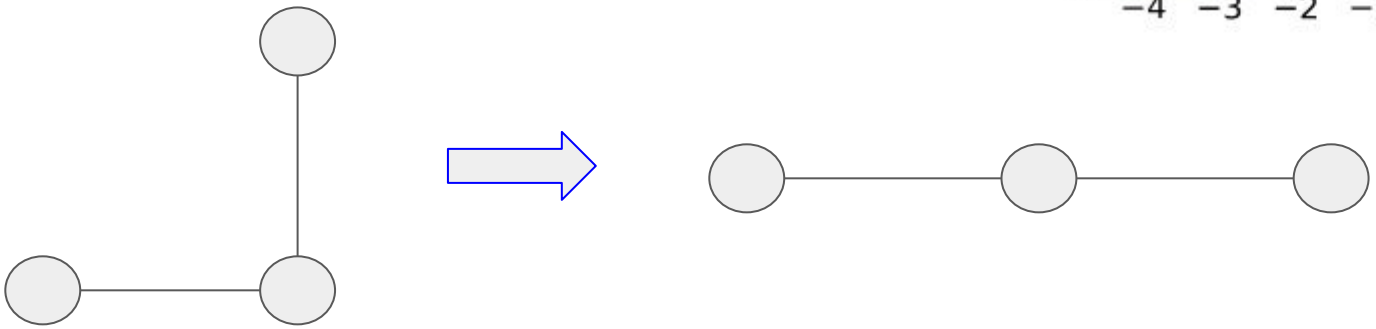
Why do we define map similarities as $q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$

Suppose data is intrinsically high dimensional

We try to model the local structure of this data in the map

Result: Dissimilar points have to be modeled as too far apart in the map!





t-Distributed Stochastic Neighbor Embedding

- Similarity of datapoints in High Dimension

$$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

- Similarity of datapoints in Low Dimension

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

t-Distributed Stochastic Neighbor Embedding

- Cost function

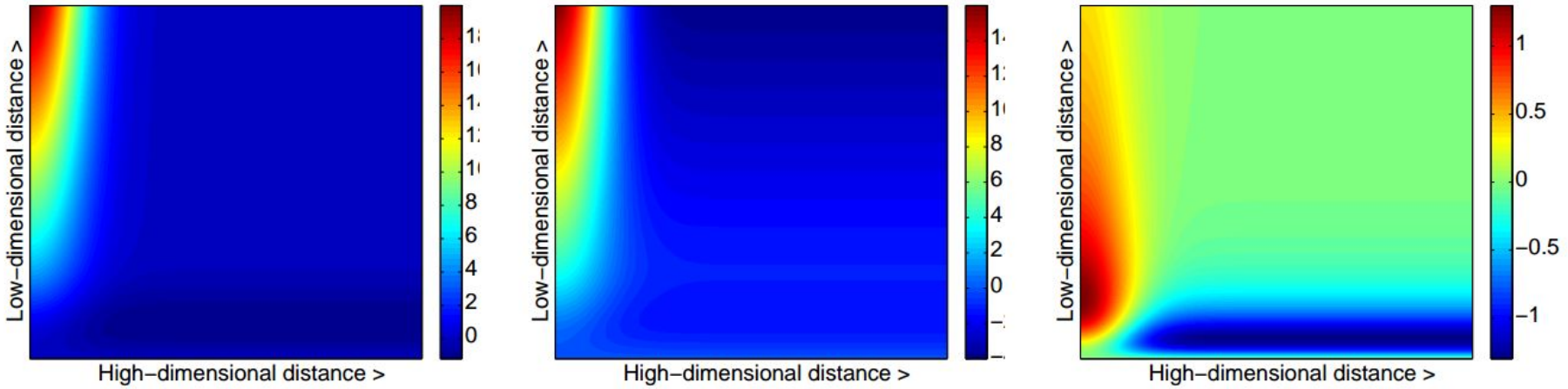
$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}$$

  - Large p_{ij} modeled by small q_{ij}: Large penalty
  - Small p_{ij} modeled by large q_{ij}: Small penalty
  - t-SNE mainly preserves local similarity structure of the data

- Gradient

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

Gradients of various types of SNE



(a) Gradient of SNE. (b) Gradient of UNI-SNE. (c) Gradient of t-SNE.

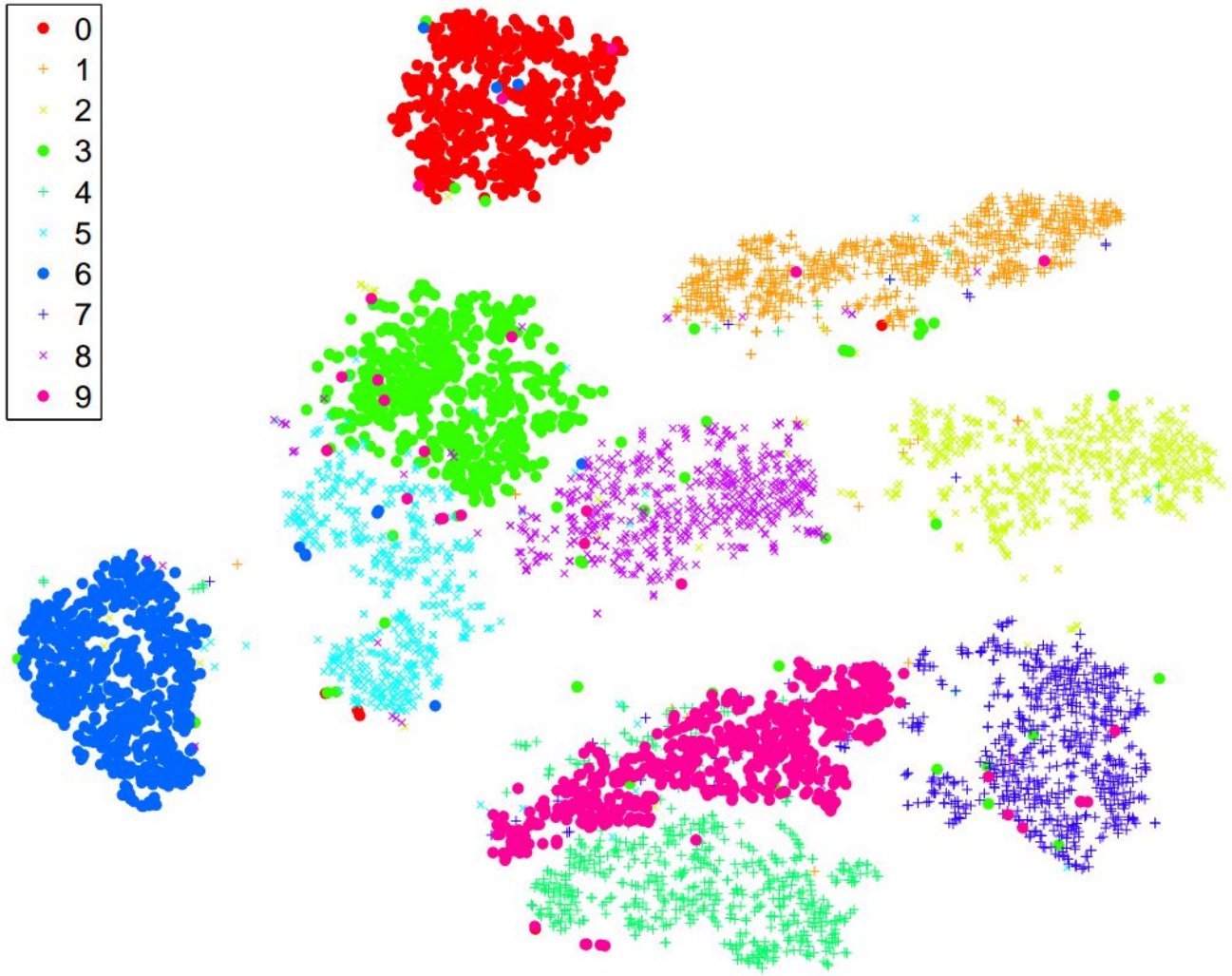
Gradients of three types of SNE as a function of the pairwise Euclidean distance between two points in the high-dimensional and the pairwise distance between the points in the low-dimensional data representation. Positive values of the gradient represent an attraction between the low dimensional data points yi and yj, whereas negative values represent a repulsion between the two data points.

t-SNE Algorithm

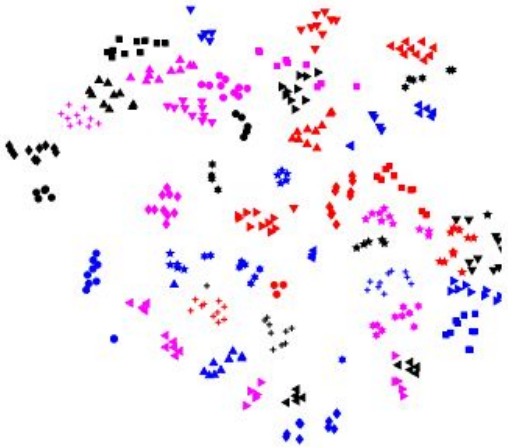
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=*1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

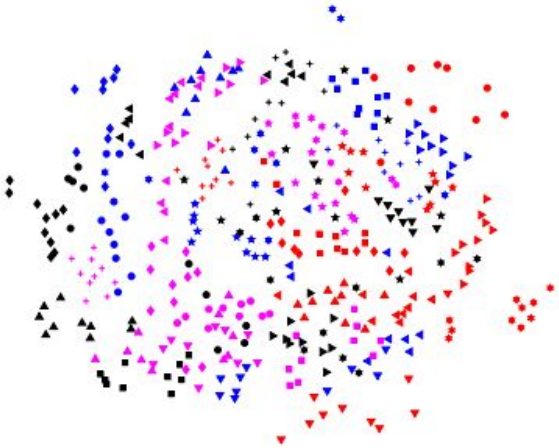
Results: MNIST



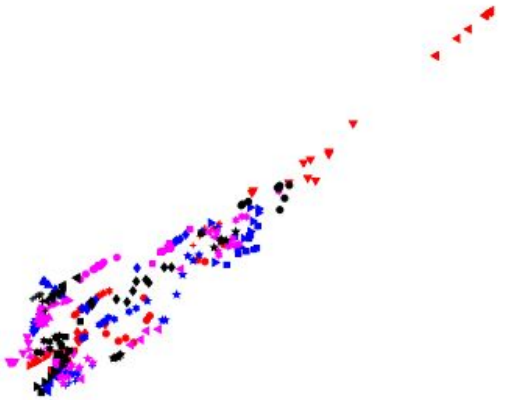
Result: Olivetti faces



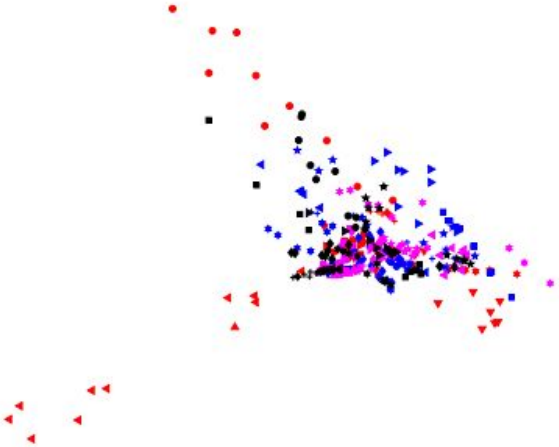
(a) Visualization by t-SNE.



(b) Visualization by Sammon mapping.



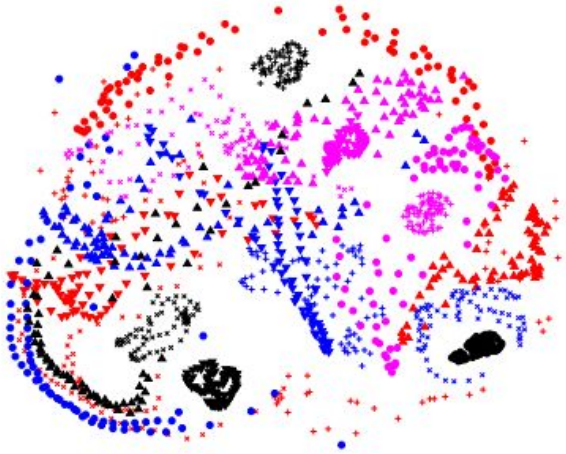
(c) Visualization by Isomap.



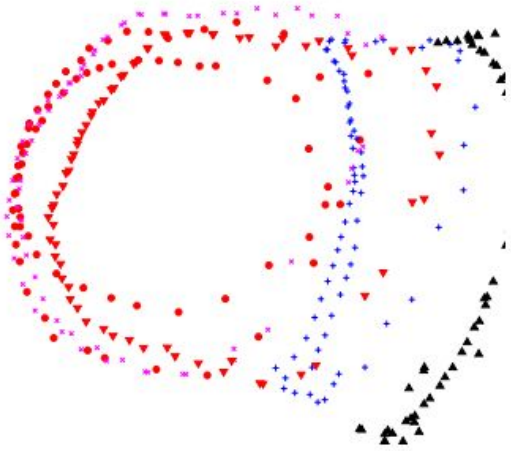
(d) Visualization by LLE.

Results: COIL-20

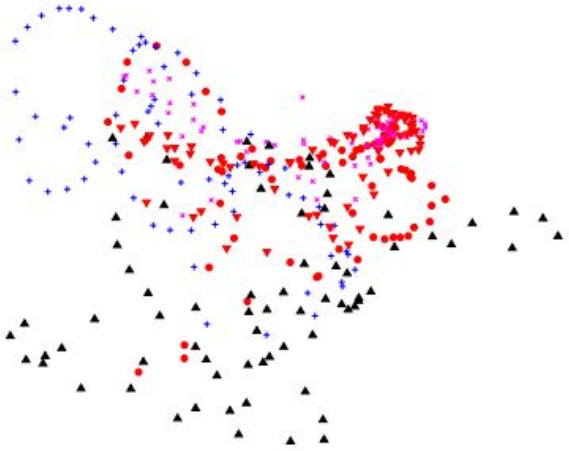
(a) Visualization by t-SNE.



(b) Visualization by Sammon mapping.



(c) Visualization by Isomap.



(d) Visualization by LLE.

Implementation

Let’s take a look at Python Implementation

References

1. “Visualizing Data using t-SNE”, L. Maaten, et. al.
2. “Stochastic Neighbor Embedding”, G. Hinton, et. al.
3. t-SNE implementations and other resources - https://lvdmaaten.github.io/tsne/
4. “A Tutorial on Principal Component Analysis”, J. Shlens
5. “Dimensionality Reduction: A Comparative Review”, L. Maaten
6. Google Tech Talks: https://youtu.be/RJVL80Gg3lA