THINKFUL

Dimensionality Reduction

t-SNE

Course

Objective

Gain an understanding of the necessity and the usefulness of t-distributed Stochastic Neighbor Embedding (t-SNE) DR method and its usefulness in data science

Curriculum

Module:

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

Agenda

- Warm Up
- Rationale for t-SNE
- t-SNE algorithm details
- Applying t-SNE
- Useful considerations about t-SNE
- Questions
- Assignment

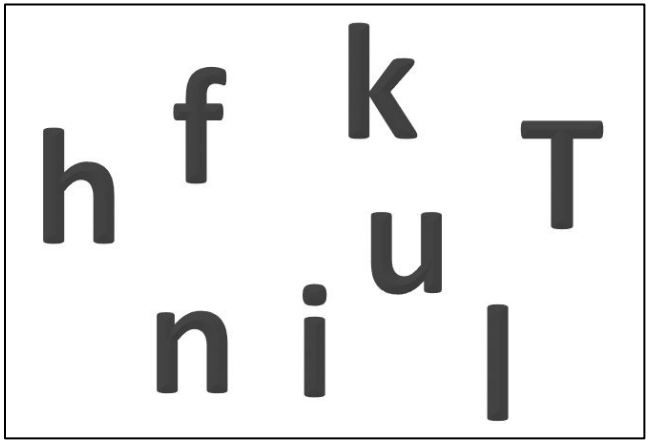
Warm Up

- Consider this input image of 3D text as a 2D manifold that we want to recover.
- Which dimension reduction output is ‘better’? Why?

Input

Dim Reduction A

Dim Reduction B

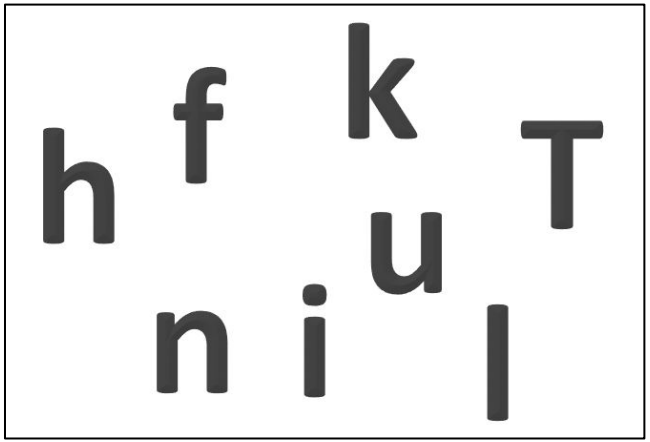


Warm Up

- Consider this input image of 3D text as a 2D manifold that we want to recover.
- Which dimension reduction output performs ‘better’? Why?
- How well do these options preserve the *local* structure of the 3D input? What about the *global* structure?

Dim Reduction A

Dim Reduction B



UP NEXT

Rationale for t-SNE

Why and what?

- High-dimensionality data is usually very difficult to depict in just 2 dimensions

- Oftentimes, real data is organized in fairly dense clusters that don’t work that well with other DR approaches
  - In the case of locally linear embeddings, we might be overly focused on local structure at the cost of potentially losing the global structure.

UP NEXT

t-SNE algorithm details

Algorithm overview

- t-SNE evaluates the pairwise similarity among all the data points using a similarity measure.
  - It does this first in the high dimensionality space and then in the reduced dimensionality space.
  - The similarity measure is parametric.

- The goal of t-SNE, is to optimize these pairwise similarities to be as similar as possible when compared between the higher and lower dimensions.
  - These pairwise similarities are viewed as a probability distribution (namely, the *t* distribution).
  - The goal is to make the lower dimensional version of this distribution appear as similar as possible to the original higher dimensional data.

QUESTION

How would you evaluate the similarity of two data points in a multi-dimensional space?

Algorithm details - similarity metric

- In t-SNE, we evaluate the distances for each pair of data points y_i and y_j, as joint probabilities that are given by the formula:

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

- The underlying distribution that is assumed, for these probabilities is the *t* distribution

- This metric has a useful property, namely the probabilities it yields are (almost) invariant to changes in scale of the map of points when it comes to points being very far apart. The fact that the *t* distribution is very similar to the Gaussian is the theoretical justification of this.

Algorithm details - optimization and stopping criterion

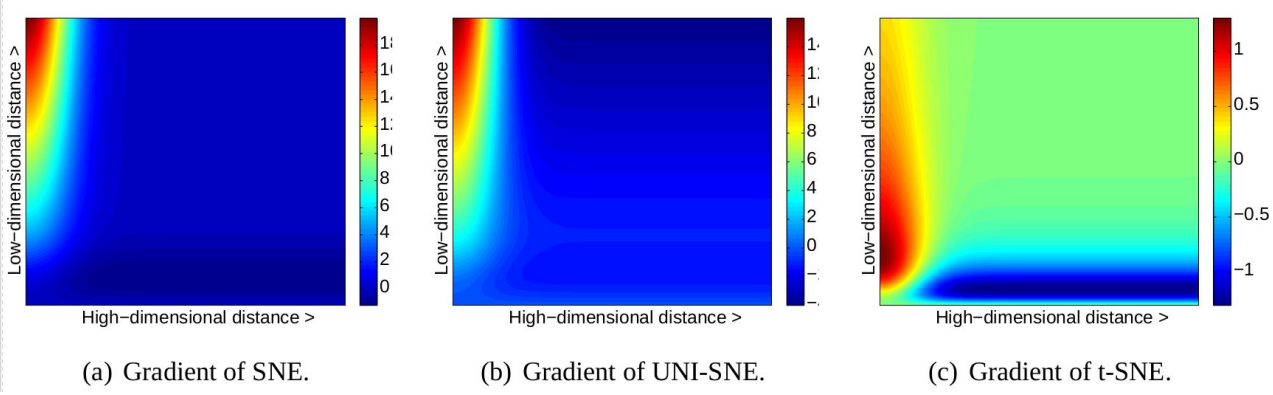
- For the optimization process, the cost function C and its gradient are used. The latter is given by the formula:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) \left(1 + \|y_i - y_j\|^2\right)^{-1}$$

- This gradient is particularly useful because it manages to get dissimilar data points to repel strongly.

Algorithm details - optimization and stopping criterion (cont.)

- t-SNE's advantage over similar methods is shown when we plot its gradient in relation to different high-dimensional distances (figure below)
- The low-dimensional representation (y) of t-SNE is updated for each step of the optimization process for a total of T steps. Once all steps are completed, y is not updated further and it is given as an output.
- The optimization process can be tweaked if needed to perform better



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

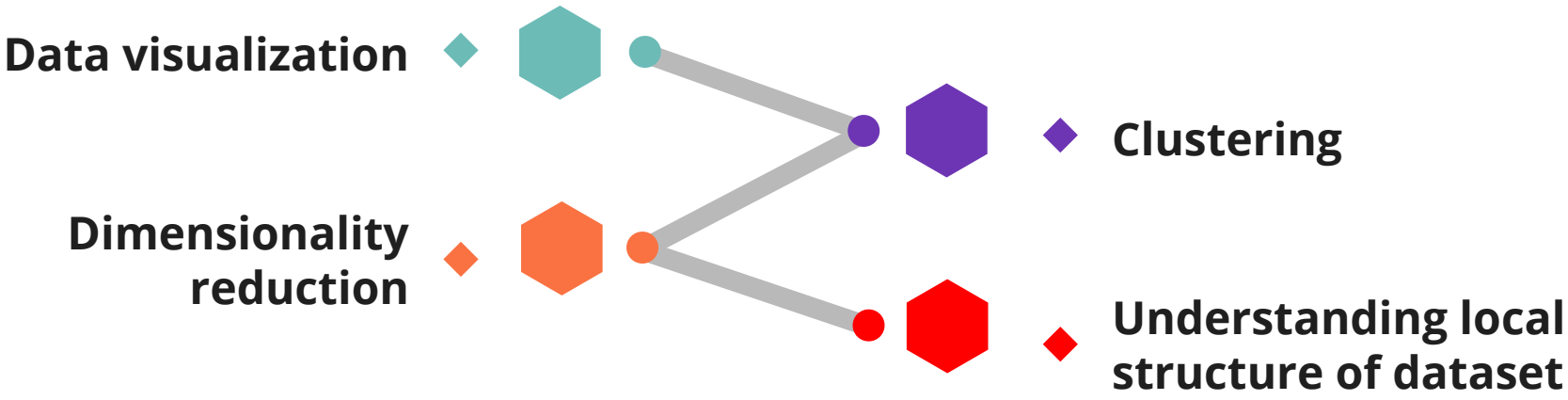
Algorithm details - tricks to improve perf.

- t-SNE can be quite slow, which is why there are a couple of tricks that can be applied to improve its performance:
  - **Early compression**: force map data points to stay close together at the beginning of the optimization process. Implemented using the L2 penalty to the cost function. The magnitude of this penalty is set by hand as a hyperparameter of the algorithm.
  - **Early exaggeration**: multiply all of the probabilities by a given number (e.g. 4) at the beginning of the optimization process. This way, optimization focuses primarily on the larger probabilities, forming clusters faster and making them tighter.

UP NEXT

Applying t-SNE

t-SNE applications



QUESTION

What kind of datasets would you use t-SNE on?

Applying t-SNE in Python

- When applying, some parameters we can play with are:
  - `n_components`: typically 2 since t-SNE is commonly used for viz
  - `perplexity`: this manages the local vs. global structure trade-off; lower values for perplexity lead to a more local view
  - `random_state`: the algorithm is stochastic, if we want consistent results we need to set a random seed

```
1 from sklearn.manifold import TSNE
2
3 tsne = TSNE(n_components=2, perplexity=50.0, random_state=42)
4 reduced = tsne.fit_transform(X)
```

UP NEXT

Useful considerations about t-SNE

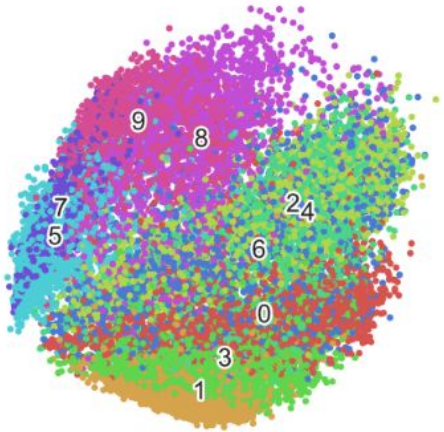
General consideration about t-SNE

- The algorithm is very sensitive to its initial parameters
- It is very slow when it is applied on larger datasets (more than 10000 data points)
- If you end up with several blobs, the distances between these blobs aren't very meaningful since the algorithm focuses more on local preserving local structure
- t-SNE is intended for data visualization and not for use in supervised learning (`sklearn`'s implementation doesn't even provide a `transform` method for us to apply to new data)

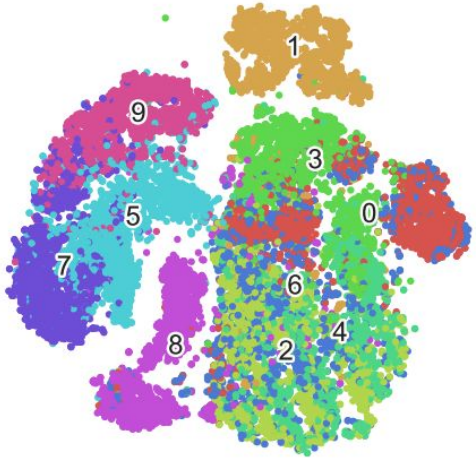
Comparison with other DR methods

- t-SNE can do a better job at reducing a complex dataset with non-linear structure

- For example, on the fashion MNIST dataset, PCA yields a single blob of data points while t-SNE preserves some of the structure of the original dataset



PCA, 2 Components



t-SNE, Perplexity=40

Pros and Cons of t-SNE

- Pros:
  - Often performs better than Isomap & LLE on ‘real world’ data
  - Great for visualizing higher dimensional data
  - Preserves data’s local structure well

- Cons:
  - Slow
  - Finding the right value of perplexity is non-trivial (esp paired with the speed issue)
  - Interpretability

Summary

- t-SNE performs dimension reduction using pairwise similarities measured in high-D & low-D. The algorithm optimizes the low-D representation of the data so that these similarities are as similar as possible.

- One of the main parameters in t-SNE is the perplexity parameter which effects how much importance is place on local structure

- In practice, t-SNE can work very well for visualization tasks, but it can be very slow.

Assignment

- Experiment with visualizing epileptic seizure data using PCA and t-SNE.
- Assignment
- Dataset

Thank You

THINKFUL

Dimensionality Reduction

T-SNE

Course

Objective

Gain an understanding of the manifolds transformations and the dimensionality reduction methods available that are based on them.

Curriculum

Module:

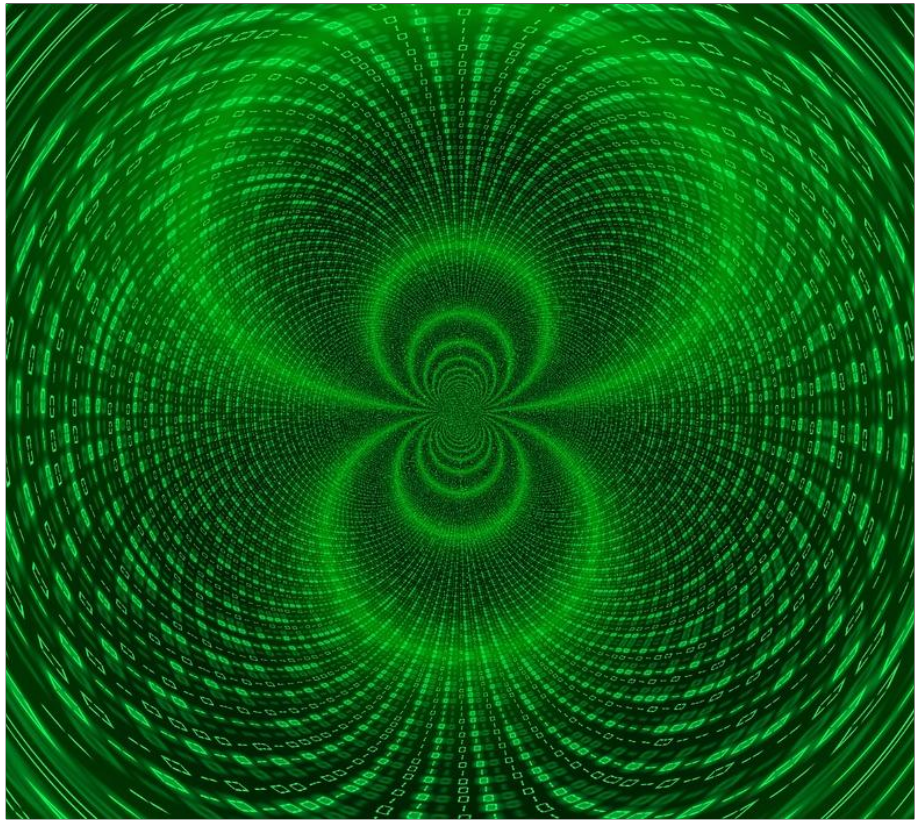
- ➔ Checkpoint 1 Title [linked]
- ➔ Checkpoint 2 Title [linked]

Agenda

- Warm Up
- Manifold Learning Embeddings & the Intuition behind Them
- Isomap
- MultiDimensional Scaling (MDS)
- Local Linear Embeddings (LLE)
- Modified LLE (MLLE)
- Summary
- Assignment

Warm Up

1. How could we warp the feature space so that it is alike to the small dimensionality space we are familiar with in our everyday lives?
2. Is it possible to transform a data set without having an analytical model of how much each feature contributes to a meta-feature, like in the case of PCA?



High Level Agenda

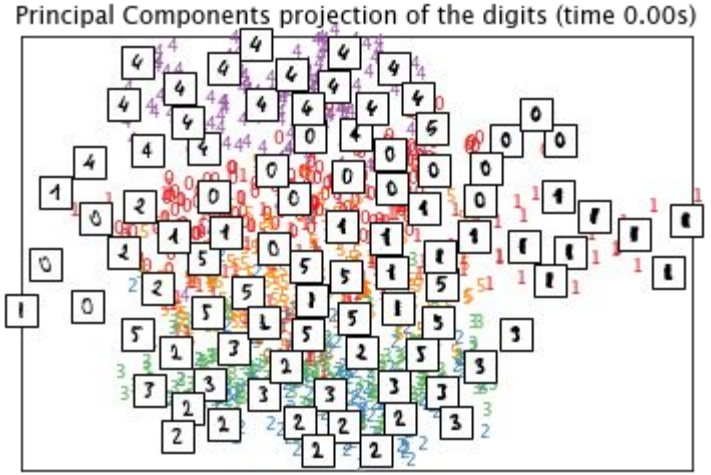
- Dimensionality Reduction Recap
- t-SNE introduction and motivation
- t-SNE versus PCA
- t-SNE algorithm
- Applying t-SNE
- t-SNE comparisons
- Assignment

Dimensionality Reduction Recap

- High-dimensional datasets can be difficult to visualize.
- While data in two or three dimensions can be plotted to show the inherent structure of the data, equivalent high-dimensional plots are much less intuitive.
- To visualize the structure of a dataset, dimensions must be reduced.
- PCA and other ***linear transformation methods*** can be used to reduce dimensions while maintaining the essential relationships between data points.

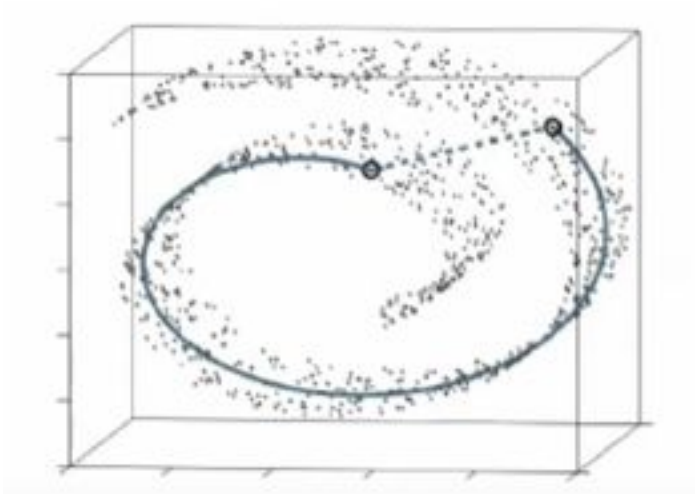
Dimensionality Reduction Recap

- PCA and other ***linear transformation methods*** can be used for dimensionality reduction.
- However, linear transformations fail to clearly separate data with nonlinear relationships.



Nonlinear Dimensionality Reduction Recap

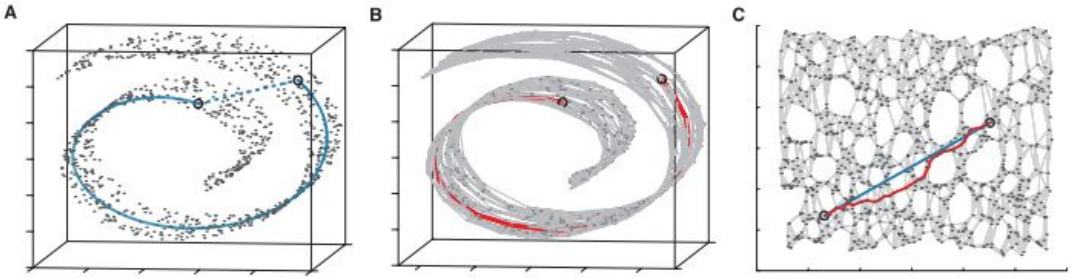
- The S-Curve data plot provides an illustration of high-dimensional data that lies on or near a low-dimensional non-linear manifold.
- Linear methods are unable to learn such non-linear relationships.



Source: https://www.quora.com/What-advantages-does-the-t-SNE-algorithm-have-over-PCA

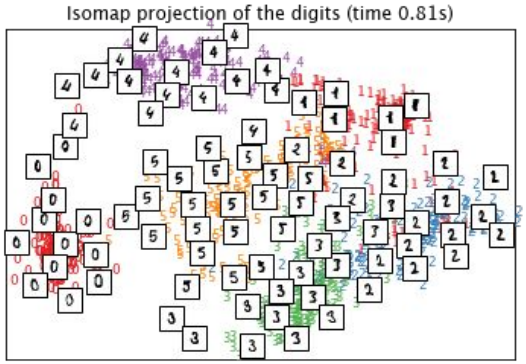
Nonlinear Dimensionality Reduction Recap

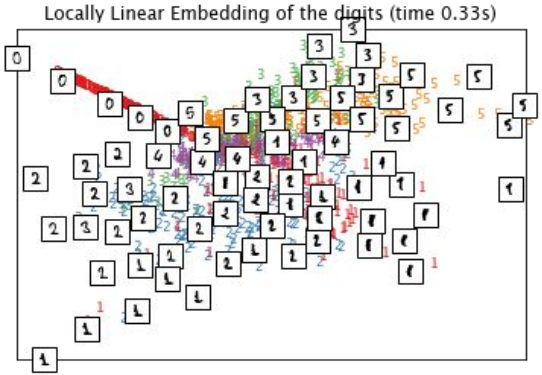
- Manifold learning methods like IsoMap, LLE, and MLLE attempt to generalize linear frameworks like PCA to be sensitive to non-linear structure in data.
- These methods emphasize *learning local data structure*, and infer larger data structure from the local structure.
- Manifold learning using IsoMap on the S-Curve is shown below.



Nonlinear Dimensionality Reduction Recap

- IsoMap and LLE tend to work well with synthetic “toy” datasets like the S-Curve, but often do not work well on “real world” datasets.
- Isomap can fail to cleanly separate clusters.
- LLE does a better job separating clusters, but the clusters tends to collapse in the center.
- Why do clusters collapse in the center when applying LLE?
- Can LLE be modified to “push apart” the clusters?





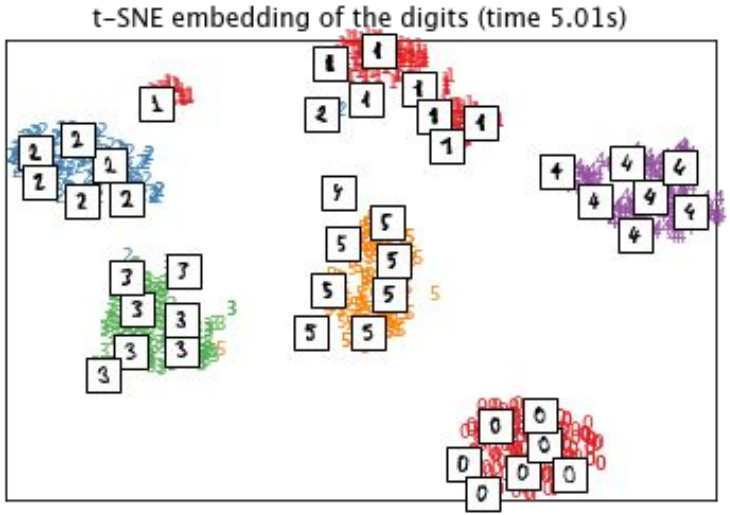
t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization into a low-dimensional space.

- Models high-dimensional data points in a 2D or 3D space.
- *Similar* data points are modeled by nearby points, and dissimilar data points are modeled by distant points with high probability.
- Uses neighborhood graphs to capture the implicit structure of all of the data to influence the way in which a subset of the data is displayed.

t-SNE

- t-SNE produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map (reduced dimension space).
- *t-SNE does not preserve global data structure, so it is not suitable for clustering or feature selection as a precursor to machine learning.*



t-SNE was developed by Laurens van der Maatens and Geoffrey Hinton in 2008.

t-SNE versus PCA

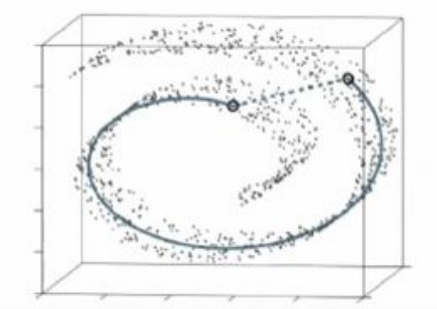
- PCA is a *linear* dimension reduction technique that seeks to maximize variance and preserves large pairwise distances.
- Things that are different end up far apart.
- PCA leads to poor visualization when dealing with non-linear manifold structures.

t-SNE versus PCA

- For high-dimensional data that lies on or near a low-dimensional, non-linear manifold it is usually more important to ***keep the low-dimensional representations*** of very similar data points close together.
- Typically not possible with linear mapping.

t-SNE versus PCA

- t-SNE preserves only small pairwise distances or local similarities.
- PCA preserves large pairwise distances to maximize variance.
- Similar, to local linear embedding (LLE), the differences in the PCA and t-SNE approaches can be illustrated with the Swiss Roll dataset.



Source:
https://www.quora.com/What-advantages-does-the-t-SNE-algorithm-have-over-PCA

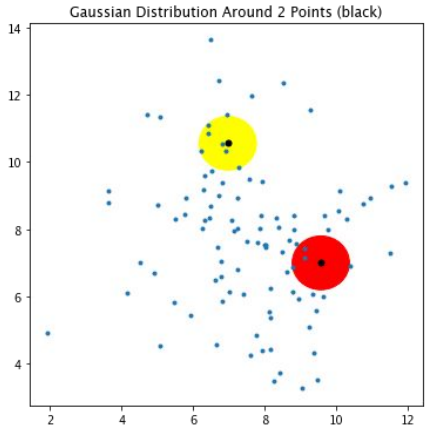
t-SNE Algorithm Summary

- t-SNE calculates a similarity measure between pairs of instances in a high dimensional space *and* in a low dimensional space.
- It then tries to optimize these two similarity measures using a cost function.

t-SNE - Step 1. Measure similarities between points in the high dimensional space

`t-SNE` starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities.

The similarity of datapoint x_j to datapoint x_i is the conditional probabiliy, $p_{j|i}$, that x_i would pick x_j as its neigbor if neighbors were picked in proportion to their probability density under a Guassian centered at x_i.



2D view of guassian around a point with $\mu = 8$ and $\sigma = 2.0$.

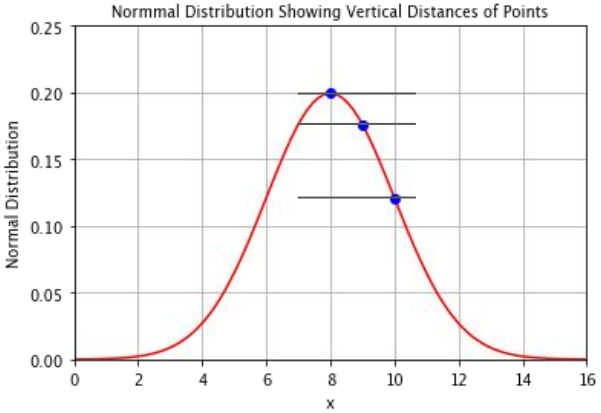
t-SNE - Step 1. Measure similarities between points in the high dimensional space

For nearby data points, $p_{j|i}$ is relatively high. For distant points, $p_{j|i}$ is infinitesimal. Observe the vertical distances between points plotted along a normal curve.

The conditional probability, $p_{j|i}$, is given by:

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

Where σ^2 is the variance of the Gaussian that is centered on datapoint x_i.



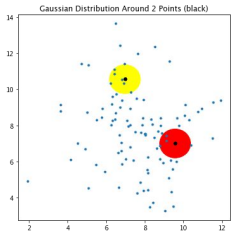
t-SNE - Step 1. Measure similarities between points in the high dimensional space

Points satisfy the symmetry rule by averaging their distances, i.e., the distance from i to j is the same as j to i. This also results in a simpler gradient.

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

The resulting distance measurements are stored in an NxN matrix of similarities, P.

Since we're only interested in pairwise similarities, the diagonal representing the value of $p_{i|i}$ is set to zero.



2D view of guassian around a point with $\mu = 8$ and $\sigma = 2.0$.

t-SNE - 1. Measure similarities between points in the high dimensional space

- The Gaussian distribution about $\boldsymbol{x}_i$ can be manipulated using perplexity, which measures how well a probability distribution predicts a sample.
- Perplexity influences the variance of the distribution (circle size) and consequently the number of nearest neighbors.
- A normal range for perplexity is between 5 and 50.

The perplexity of a discrete probability distribution p is:

$$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$$

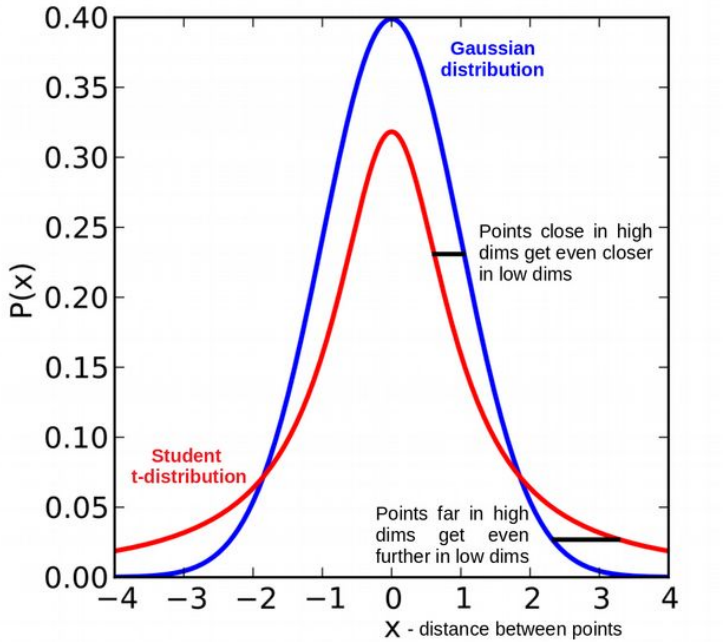
where $H(p)$ is the entropy (in bits) of the distribution and x ranges over events.

t-SNE - 2. Measure similarities between points in the low dimensional space.

A similar conditional probability $q_{i|i}$ is calculated for low-dimensional counterparts y_i and y_j, of the high-dimensional data points x_i and x_j.

To address the **crowding problem**, t-SNE uses a Student t-distribution rather than a Gaussian.

The heavy tails of the t-distribution overcome the crowding of points when embedding into low dimensions.

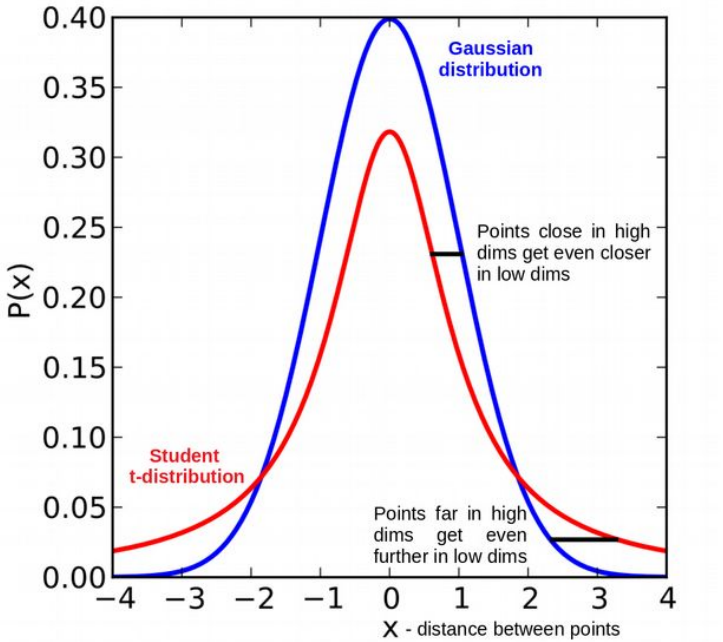


Source:
https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

t-SNE - 2. Measure similarities between points in the low dimensional space.

To employ the `Student t-distribution` with one-degree of freedom, the joint probabilities q_{ij} are defined as:

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



Source:
https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

t-SNE - 3. Set probabilities from the low-dimensional space Qij to reflect the probabilities of the high dimensional space Pij as similar as possible

If the map points y_i and y_j correctly model the similarity between the high-dimensional datapoints x_i and x_j, the conditional probabilities $p_{j|i}$ and $qj|i$ will be equal.

Motivated by this, `SNE` aims to find a low-dimensional data representation that minimizes the mismatch between $p_{i|j}$ and $q_{j|i}$.

t-SNE uses the KL-Divergence to measure the difference between the high and low distributions.

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

t-SNE - 3. Set probabilities from the low-dimensional space

Q to reflect the probabilities of the high dimensional space P **as similar as possible.**

The final free parameter is the variance σ_i in the Gaussian that is centered over each high_dimensional point x_i.

Any value for σ_i induces a probability distribution, P_i, over all datapoints. This distribution has an entropy which increases as σ_i increases.

`SNE` performs a binary search for the value of σ_i that prodces a P_i with a fixed perplexity specified by the user. The perplexity increases monotonically with the variance σ_i.

$$Perp(P_i) = 2^{H(P_i)}$$

$$H(P_i) = -\sum_j p_{j|i} log_2(p_{j|i})$$

t-SNE - 3. Set probabilities from the low-dimensional space

Q to reflect the probabilities of the high dimensional space P **as similar as possible.**

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

Gradient descent is used to minimize the cost function, C:

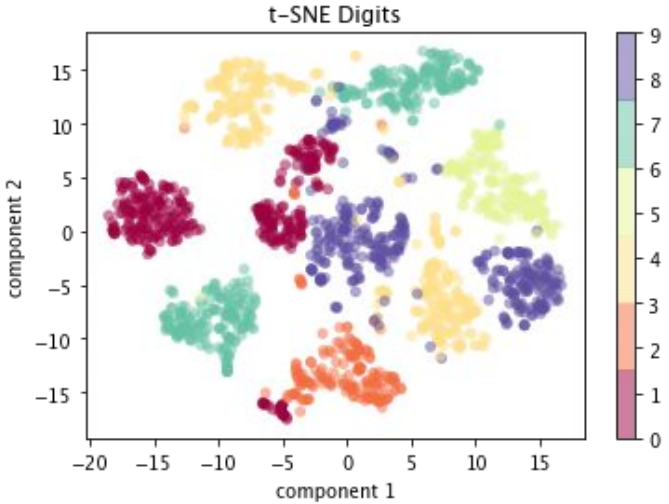
$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{i|j} - q_{i|j})(y_i - y_j)$$

Questions?

Using t-SNE with Scikit-learn

Key parameters:

- Perplexity: 5 to 50
- N_iter: > 1000 is typical.

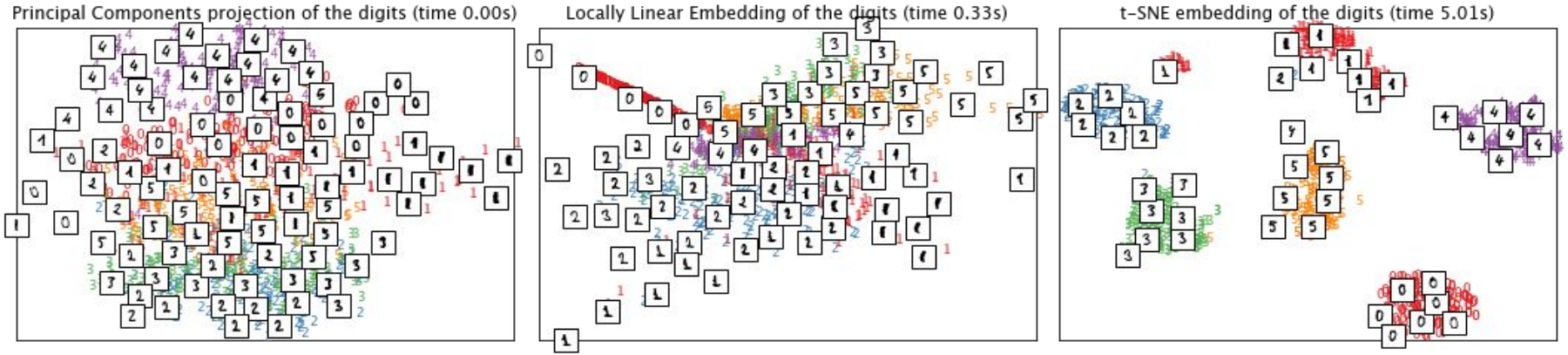



```
time_start = time.time()
tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=300)
tsne_results = tsne.fit_transform(X_mnist)
print('t-SNE done! Time elapsed: {} seconds'.format(time.time()-time_start))
```

```
[t-SNE] Computing 121 nearest neighbors...
[t-SNE] Indexed 70000 samples in 19.191s...
[t-SNE] Computed neighbors for 70000 samples in 6761.240s...
[t-SNE] Computed conditional probabilities for sample 1000 / 70000

[t-SNE] Mean sigma: 4.325675
[t-SNE] KL divergence after 250 iterations with early exaggeration: 95.929604
[t-SNE] KL divergence after 300 iterations: 5.003584
t-SNE done! Time elapsed: 7979.9682240486145 seconds
```

Comparing t-SNE Using Digits



Questions?

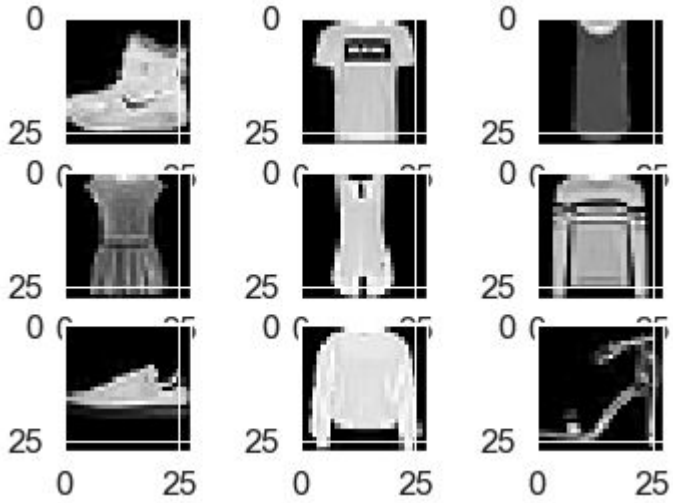
Fashion-MNIST

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

https://github.com/zalandoresearch/fashion-mnist

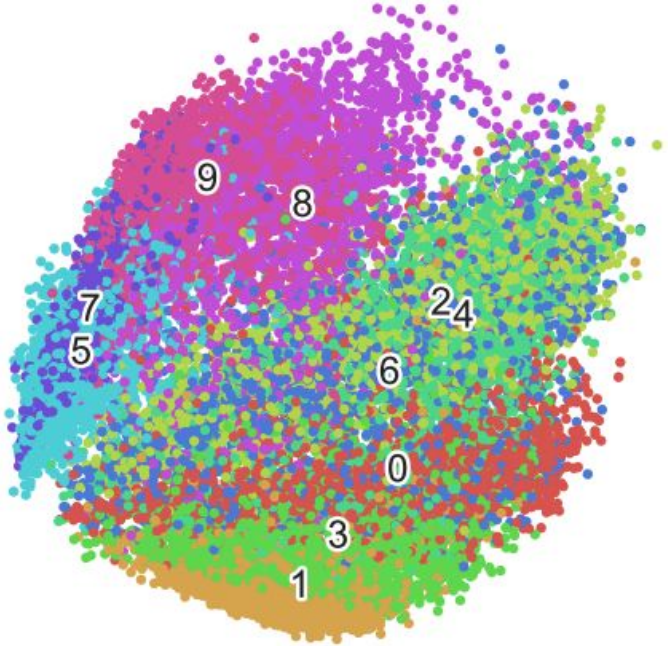
Each training and test example is assigned to one of the following labels:

- 0 T-shirt/top
- 1 Trouser
- 2 Pullover
- 3 Dress
- 4 Coat
- 5 Sandal
- 6 Shirt
- 7 Sneaker
- 8 Bag
- 9 Ankle boot

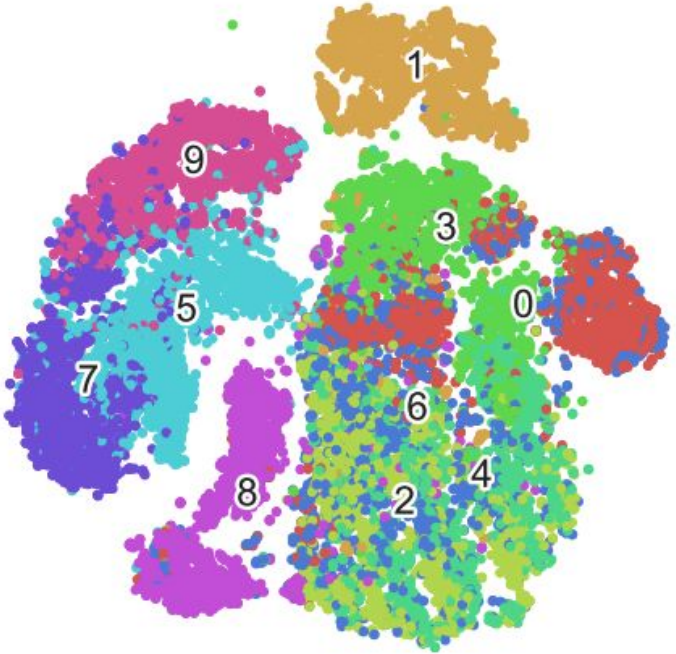


THINKFUL

Fashion-MNIST



PCA, 2 Components



t-SNE, Perplexity=40

t-SNE Summary - 1

- t-SNE is a nonlinear dimensionality reduction technique
- Well-suited for visualizing high-dimensional non-linear data in a low-dimensional space of two or three dimensions.
- t-SNE models high-dimensional data points in a two- or three-dimensional space in such a way that similar data points are modeled by nearby points and dissimilar data points are modeled by distant points with high probability.

t-SNE Summary -1

- t-SNE produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map (reduced dimension space).
- t-SNE does not preserve global data structure, so it is not suitable for clustering, or feature selection as a precursor to machine learning.

Recap

- Dimensionality Reduction Recap
- t-SNE introduction and motivation
- t-SNE versus PCA
- t-SNE algorithm
- Applying t-SNE
- t-SNE comparisons
- Assignment

Assignment 1

Assignment: PCA vs. t-SNE on Breast Cancer Data Set

Data: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

First, step through the tutorial showing how to use t-SNE on Fashion-MNIST

Perform the following steps without and then with standard scaling of the dataset.

1) Perform PCA analysis on the breast cancer dataset using 2 to 3 components.

2) Plot the first 2 principal components.

3) Apply t-SNE to the breast cancer dataset.

- You will have to experiment with perplexity (5, 50) and number of iterations (>1000).

4) Plot your t-SNE projections adjacent to your PCA plot and record your observations

5) Repeat steps 4 and 5 until you are satisfied with t-SNE being able to separate classes in the data set.

6) Record your optimal perplexity and iterations for t-SNE. Compare the PCA and t-SNE visualizations and record your results.

UP NEXT

Manifold Learning Embeddings & Intuition behind Them

Rationale of Manifold Learning

- Just like other non-linear DR methods, manifold learning aims to address the non-linear datasets
- Manifold learning is a type of unsupervised learning models that attempt to describe datasets as low-dimensional manifolds (surfaces) embedded in high-dimensional spaces
- Manifold learning models are usually stochastic in nature, meaning that every time you run them they yield a somewhat different result. This is in contrast to deterministic models like PCA and Kernel PCA

Intuition of Manifold Learning

- You can think of manifold learning as a way to generalize linear frameworks (e.g. PCA and ICA) so that they can pick up the non-linear aspects of a dataset
- Picture a sheet of paper. This is a 2D object that exists in a 3D world. It can be bent, rolled or stretched in two dimensions. We can think of this sheet as a 2D manifold embedded in 3D space. As a result, transforming the paper in 3D doesn't change the flat geometry of the paper. The latter is what we are looking for when performing DR and are referred to ML embeddings.
- All the aforementioned operations are akin to linear embeddings
- Bending, curling, or crumpling the paper, are different operations whereby the sheet remains a 2D manifold, but the embedding into the 3D space is no longer linear. This is the sort of ops explored in manifold learning.

UP NEXT

Isomap as an MLE method

What and Why?

- Isomap is short for **Iso**metric **Map**ping.
- It can be seen as an extension of Kernel PCA, in a way
- The idea is to seek a lower-dimensional embedding which maintains geodesic distances between all points
- If we were to apply the concept of geodesic distances to a more abstract plane, i.e. a graph, we'd end up with the following definition:
  - Shortest path: the particular edges that connect node A to node B in an optimal way
  - What is optimized in this case is the number of edges
  - The geodesic distance is that particular number (global minimum)
  - Usually the Floyd–Warshall algorithm is used for this

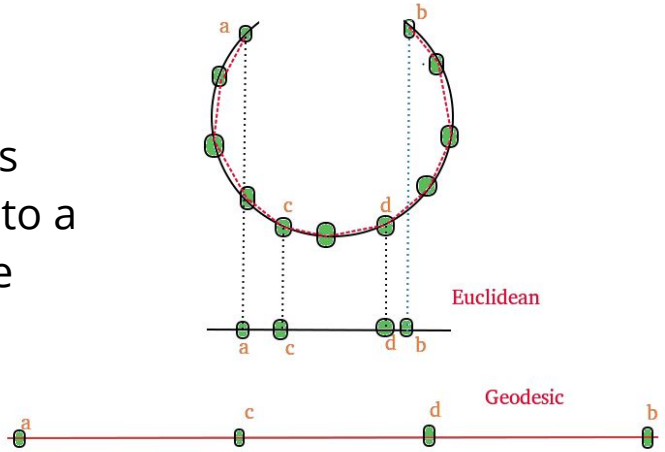


Image source: paperspace blog

Details of Isomap

- The Isomap algorithm is as follows:

  1. Neighbourhood graph: first we need to create a *neighborhood graph* and *adjacency matrix* from the dataset

  2. Dissimilarity Matrix: then we calculate the geodesic distances among the points, pairwise, using the neighbourhood graph

  3. Eigenvalue decomposition: finally, we square the distances and double center the dissimilarity matrix. Then we calculate the eigenvalues and their corresponding eigenvectors, like we do in PCA

QUESTION

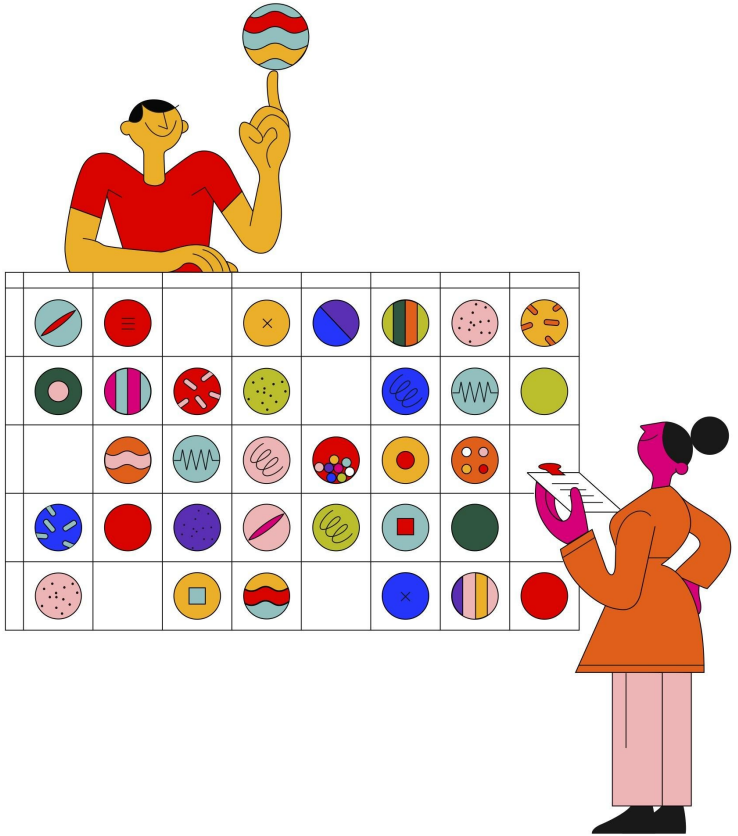
What do you expect to be the dominant shortcoming of Isomap?

When not to use Isomap

- We’ve already examined when non-linear DR methods like Isomap shine. But when is it that Isomap fails to deliver?
- Based on what we discussed previously, this happens when the manifolds are not sampled properly and as a result they contain gaps in them
- Beyond this, the method is sensitive to its initial parameters so failing to select the proper values for them can result to unhelpful outputs

UP NEXT

MultiDimensional Scaling (MDS) as an MLE method



Rationale of MDS

- MDS is a family of DR methods using manifolds
- In general, there are two types of MDS methods:
  a. Metric based MDS: these involve continuous variables.
  b. Non-metric based MDS: these involve ordinal variables.
- The metric based MDS which is what we'll focus on here makes use of the Majorization-Minorization (MM) optimization algorithm
- When optimizing a function f(), MM finds another function g() which is simpler (easier to optimize) and uses that as a guide. This function g() needs to have the following attributes:
  a. Optimizing g(x,xm) should be easier than f(x) .
  b. For any x , $f(x) \leq g(x,xm)$
  c. $f(xm) = g(xm,xm)$

Details of MDS - MM optimizer

- The Majorization-Minorization (MM) optimizer works as follows:

  1. Choose a random support point x_m

  2. Find the value x_{min} for which $g(x,x_m)$ minimizes: $x_{min} = \text{argmin}(g(x, x_m)$

  3. If $f(x_{min})-f(x_m) \approx e$ break the loop (e is a very small number given as a parameter of the algorithm)

  4. Set $x_m=x_{min}$

  5. Go back to step 2

Details of MDS - MM optimizer (cont.)

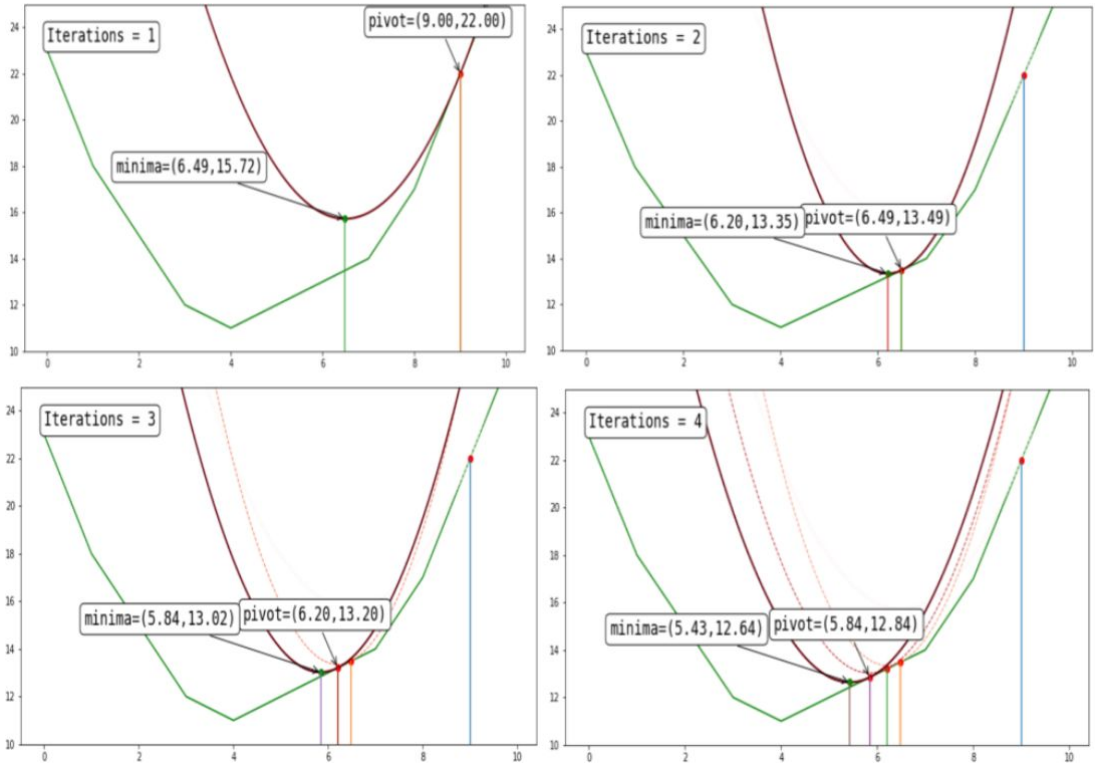


Image source: [paperspace blog]

Details of MDS - MDS algorithm

- The MDS algorithm is as follows:

  1. Create a dissimilarity matrix
  2. Choose a random point *Xm* as pivot
  3. Find the minimum of stress majorizing functions
  4. If $\sigma(Xm) - \sigma(Xmin) < e$ break else set $Xm=Xmin$ and go to step 2

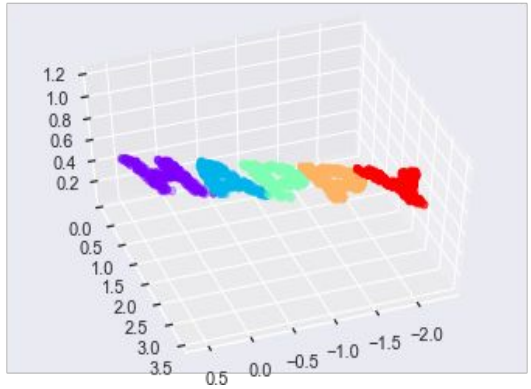
Clearly there are lots of similarities to the MM algorithm we saw previously

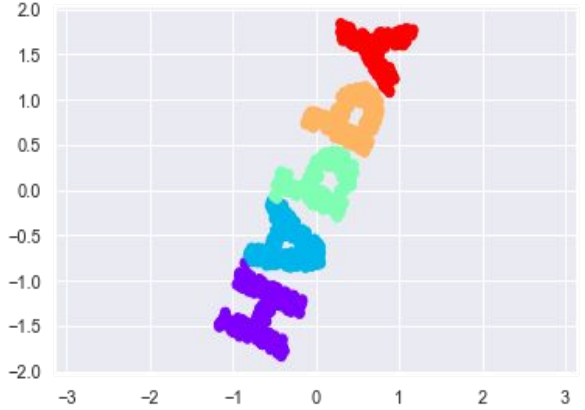
When not to use MDS

- Despite its strengths, MDS is not suitable as a DR method when:
  a. You need to use the MDS model on new data (if so, you'll need to fit it from scratch as there is not apply() option)
  b. Your computational resources are limited (it takes a lot of them for calculating the dissimilarity matrix and it needs to do that in every iteration!)

MDS Example

- Here is an example of MDS on 3D data (top figure) and the embedding it yields on a 2D plane (bottom figure)
- Notice how all the classes (colors) are preserved
- This is a simple example but it illustrates how the outputs of MDS can be quite useful (even if the data has been transformed making it look strange to us)





For every complex problem there is an answer that is clear, simple, and wrong.

H. L. Mencken

UP NEXT

Local Linear Embeddings (LLE) and Modified LLE (MLLE) as MLE methods

Rationale of LLE

- Instead of looking at all the distances, we can just focus on the ones of the nearby neighborhoods, preserving the local aspects of the dataset
- So, a new kind of similarity is applied, one that is obtained through the conjunction of small linear blocks
- These linear blocks can represent particular micro-features. Such micro-features can be the shape of a person’s nose, their mouths, or the presence of glasses, all of which remain unchanged in the different portraits of the same person
- LLE is preferable when the original dataset is intrinsically locally linear, possibly lying on a smooth manifold. Also, it’s a good choice when small parts of a sample are structured in a way that allows the reconstruction of a point given the neighbors and the weights.

Details of LLE

- Uses a greedy optimization process for finding progressively smaller subsets of features
- Makes use of a fairly fast data model (e.g. logistic regression)
- Ranks features based on the order of their elimination
- Runs until a given set of subfeatures is reached
- Aims at eliminating feature dependencies and collinearities
- Particularly useful for large feature sets having lots of redundant features
- Other dimensionality reduction methods can be used in tandem, but after the original feature set is reduced through RFE

Details of LLE (cont.)

- The neighborhood is assessed through the measurement of errors using the function:

$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum_j W_{ij} \vec{X}_j \right|^2$$

- LLE constructs a neighborhood preserving mapping using the embedding cost function:

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2$$

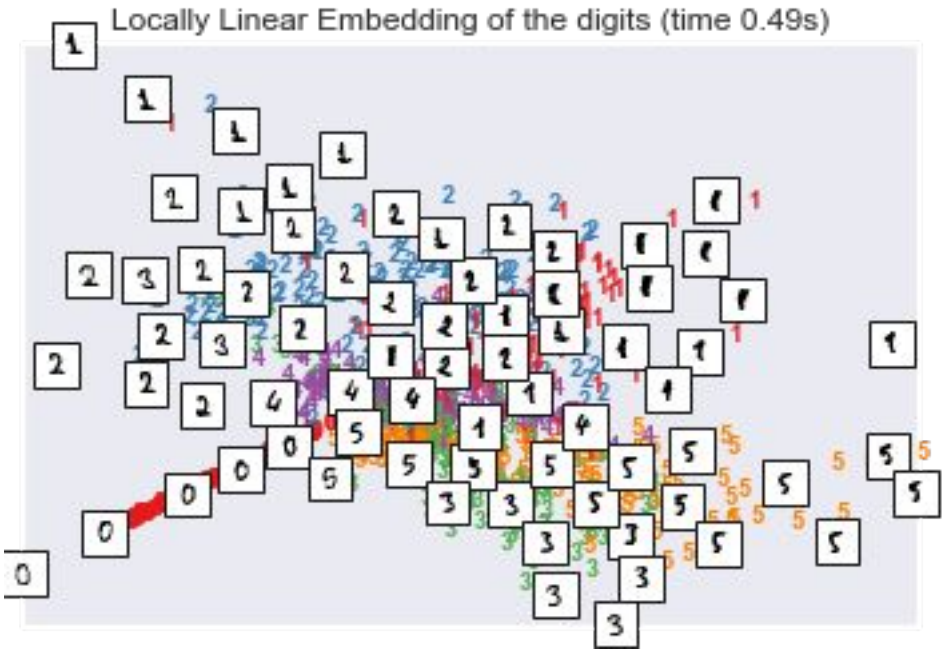
- Before applying the eigenvalue analysis, the covariances are calculated based on the various distances already calculated, using the formula:

$$C_{jk} = \frac{1}{2} \left(D_j + D_k - D_{jk} - D_0 \right)$$

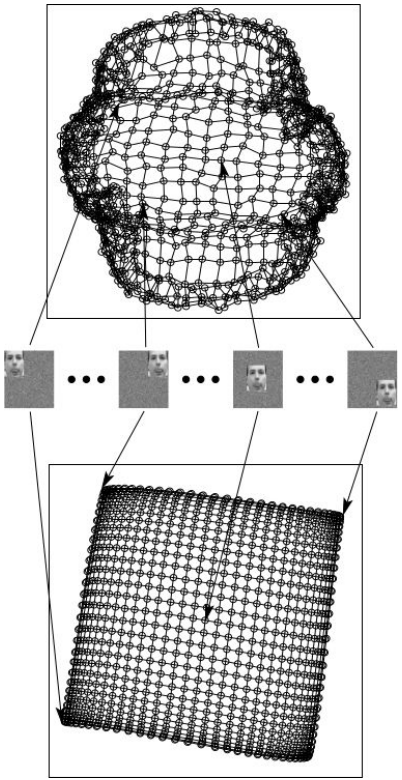
When not to use LLE

- Despite its strengths, LLE is not suitable as a DR method when:
  a. If you need to use LLE on new data (it doesn't have an internal model that you can reuse beyond the data it was originally applied on)
  b. When your data has regions of non-uniform sample densities (it doesn't handle them well at all)
  c. When you care about the global distances too, instead of just the local ones

LLE Examples



Digits image analysis using LLE



Face image analysis using PCA (above) and LLE (below)

Rationale of Modified LLE (MLLE)

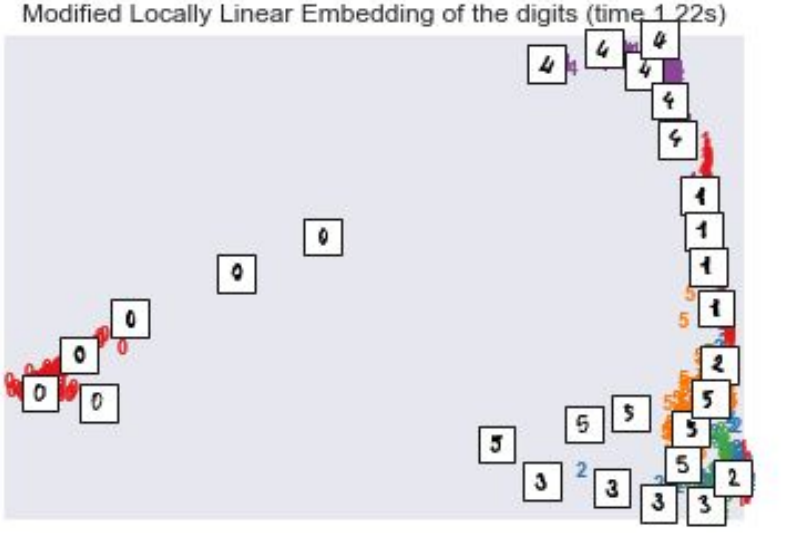
- LLE has issues such as the local weight matrix conditioning problem
- We need to be able to handle cases where there are linearly independent weight vectors in each neighborhood
- MLLE addresses these concerns and provides a generally better performance than LLE
- MLLE is more expensive computationally, however

Details of MLLE

- Its main difference to the standard LLE method is that does more work on the Weights Matrix construction part
- The complexity of MLLE on that part is O[DNk^3] + O[N(k-D)k^2], which is somewhat larger than that of the standard LLE: O[DNk^3]
- You can learn more about it and how it is used through Scikit Learn here

MLLE Example

- MLLE does a better job at separating the various digits (right figure) compared to LLE (left figure), in the digits images dataset, esp. for the zeros
- However, it takes around 3x as much time for the same data



Summary

Non-linear methods like the manifold-based ones we saw here offer an advantage over the linear ones as they're able to preserve nonlinear relationships in the data

However, it's best to explore the data with a method like PCA first before applying a non-linear method

For high-dimensional data from real-world sources, LLE often produces poor results, and IsoMap seems to generally lead to more meaningful meta-features

The *t-distributed stochastic neighbor embedding* (T-SNE) DR method, which is covered in the next lecture, seems to work well for highly clustered data

Assignment

Dataset: Kaggle Red Wine Dataset available at http://bit.ly/2QafOkA
This dataset has 1599 data points with 12 features on wine quality. The target variable is wine quality and ranges from 0 to 10

In this assignment you need to do the following:
1) Load the the wine quality data set.
2) Fit PCA to 12-components (max) and plot the cumulative sum of the 'pca.explained_variance_ratio_'
3) Identify the number of principal components to explain 90% of the variance.
4) Build a logistic regression model and record the accuracy.
5) Repeat steps 2 and 4 using LLE with the same number of components and 30 neighbors.
6) Record your observations and identify your top performing model. Does manifold learning improve predictive performance over PCA?

Note: you need to use the code notebook DimensionalityReduction_53_2_nonlinear_lle_mlle_iso

Thank You