THINKFUL

Dimensionality Reduction

Non-linear methods

Isomap & LLE

Course

Objective

Gain an understanding of the necessity and the usefulness of non-linear dimensionality reduction methods, how they relate to data science work, and get a glimpse of the Kernel PCA method.

Curriculum

Module:

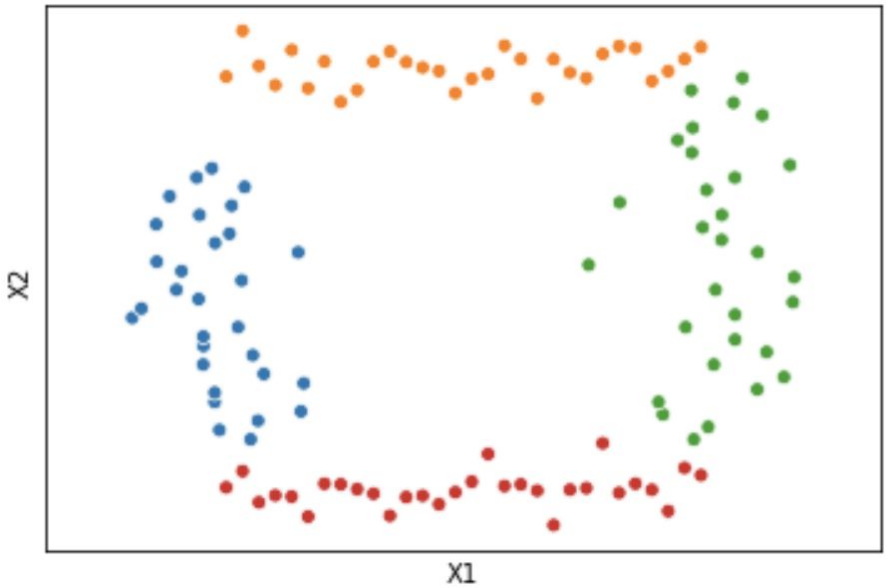
- ➔ Checkpoint 1 Title [linked]
- ➔ Checkpoint 2 Title [linked]

Agenda

- Warm Up
- Manifold Learning Embeddings & the Intuition behind Them
- Isomap
- LLE
- Assignment

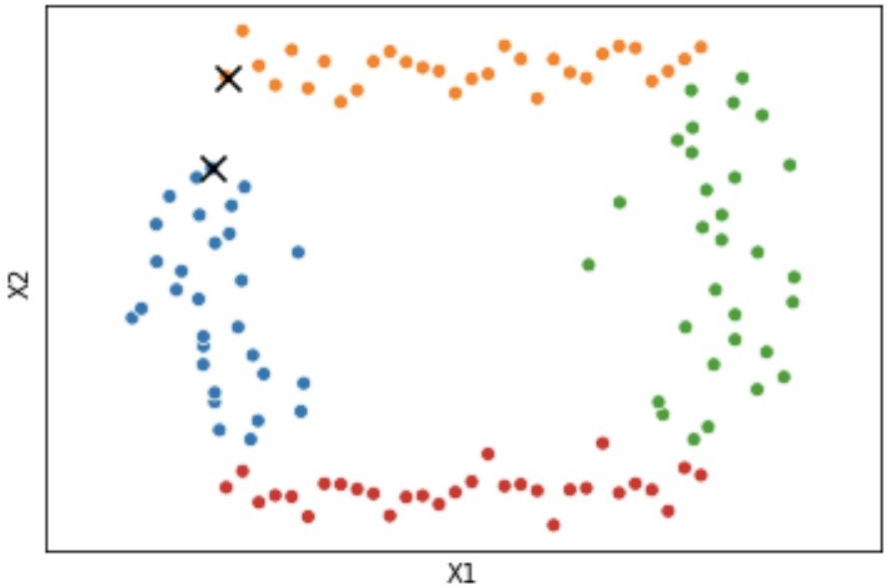
Warm Up

- Consider the input 2d data in the plot.



Warm Up

- Consider the input 2d data in the plot.
- If we were to perform dimension reduction down to 1d, should the highlighted points be close together or far apart in 1d?
- What's a justification for them being close together?
- What's a justification for them being far apart?



UP NEXT

Manifold Learning Embeddings & Intuition behind Them

Rationale of Manifold Learning

- Manifold learning aims to address non-linear relationships
- Manifold learning is a group of unsupervised learning models that attempt to describe datasets as low-dimensional manifolds (surfaces) embedded in high-dimensional spaces
- Most manifold learning models are stochastic in nature. Unlike deterministic models like PCA

Intuition of Manifold Learning

- 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; transforming the paper in 3D doesn't change the flat geometry of the paper.
- Manifold methods attempt to recreate the flat 2D piece of paper when presented with a 3D representation of it (unfolding origami).
- Just like in the warm-up, this means we have to address non-linear relationships.

UP NEXT

Isomap

What and Why?

- Isomap is short for **Iso**metric **Map**ping.
- The idea is to seek a lower-dimensional embedding which maintains geodesic distances between the higher-dimensional points
- This geodesic distance will be the number of ‘edges’ between 2 ‘nodes’ in a ‘graph’
  - Think Six Degrees of Kevin Bacon

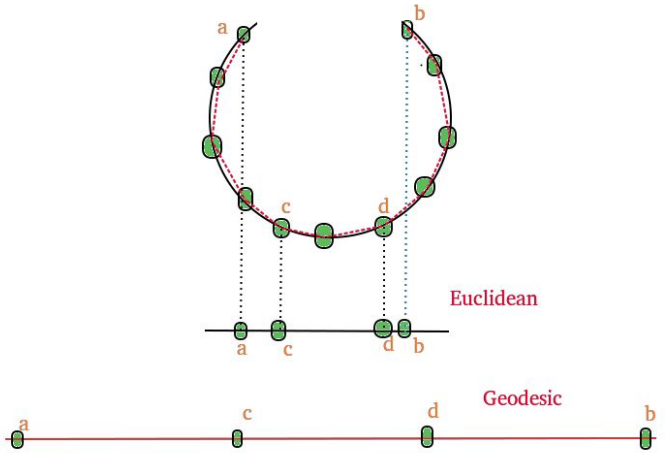


Image source: paperspace blog

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

Details of Isomap

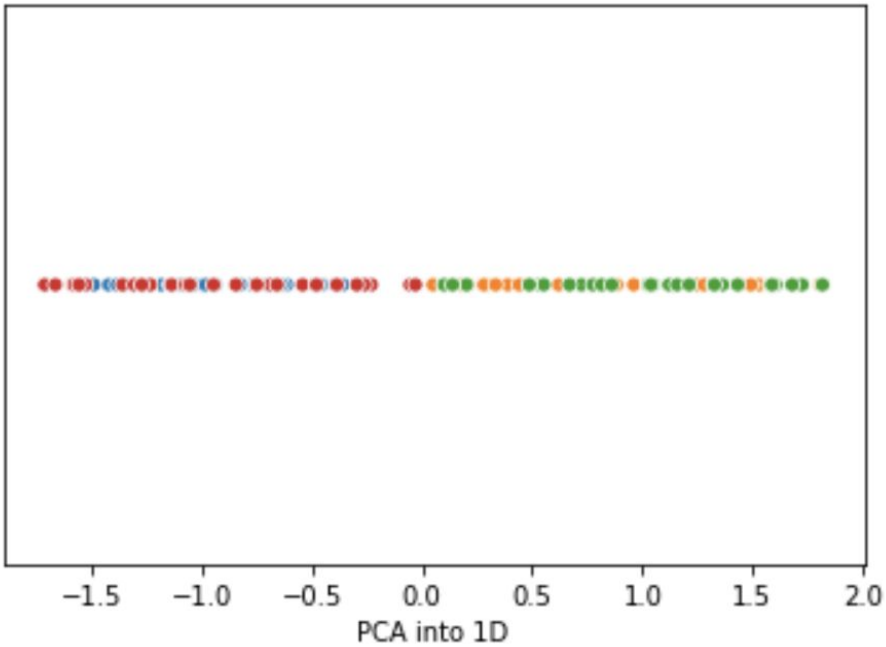
- Let’s apply the steps to the data from the warm-up
- If we build the neighborhood graph by connecting each point to its 6 closest neighbors we will have the graph shown here.
- Note, the 2 points highlighted in the warm-up are as far apart as possible via geodesic distance

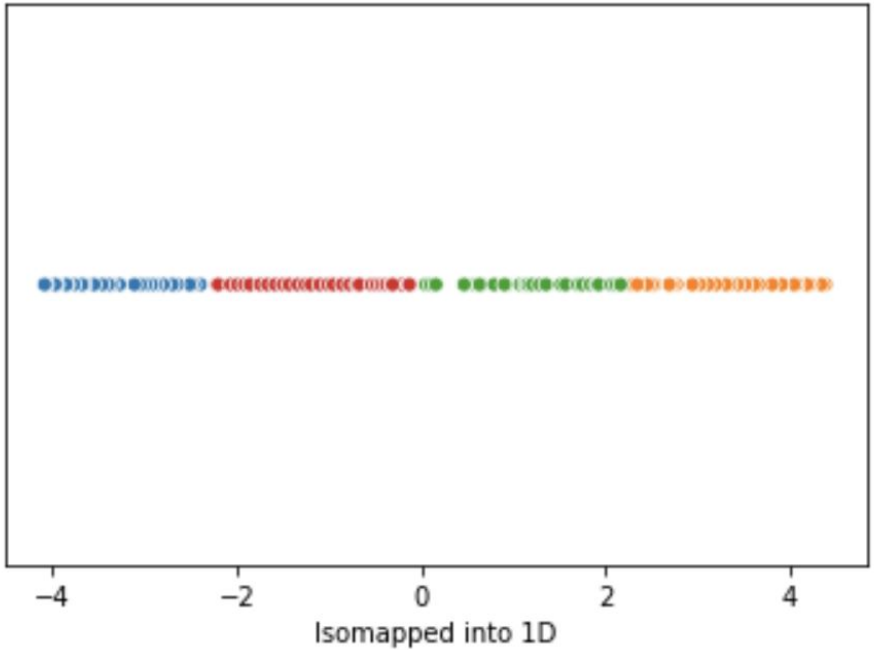
Details of Isomap

- With this graph, we can calculate the pairwise geodesic distance between every point and store this in a square matrix.
- We can then apply dimension reduction to this distance matrix to capture the non-linear information that it contains.

Details of Isomap

- Here are the results of applying both PCA and Isomap to our data.





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?

- "Isomap, LLE, and variants are best suited to unfold a single continuous low dimensional manifold" - `sklearn` docs

- "Real world" data is rarely a single perfectly continuous manifold
  - It might be multiple manifolds
  - It might have holes or gaps in the manifold(s)

- The method can also be sensitive to the number of neighbors selected

Isomap in Python

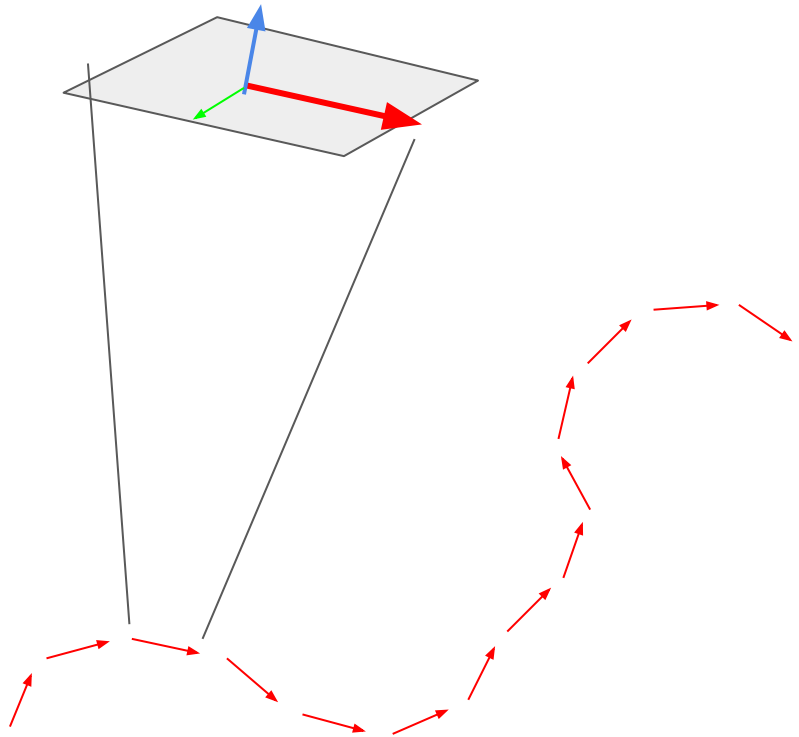
- We can apply `Isomap` with the familiar `sklearn` API
- We can set `n_neighbors` to influence how the neighborhood graph is built
- We can additionally set `n_components` just like in `PCA`

```
from sklearn.manifold import Isomap

embedding = Isomap(n_neighbors=36)
X_isomap = embedding.fit_transform(X)
```

UP NEXT

Locally Linear Embeddings (LLE) and Modified LLE (MLLE)



Details of LLE

- The LLE algorithm is as follows:

  1. Nearest Neighbors Search: Just like in Isomap, we'll connect points to their closest neighbors.

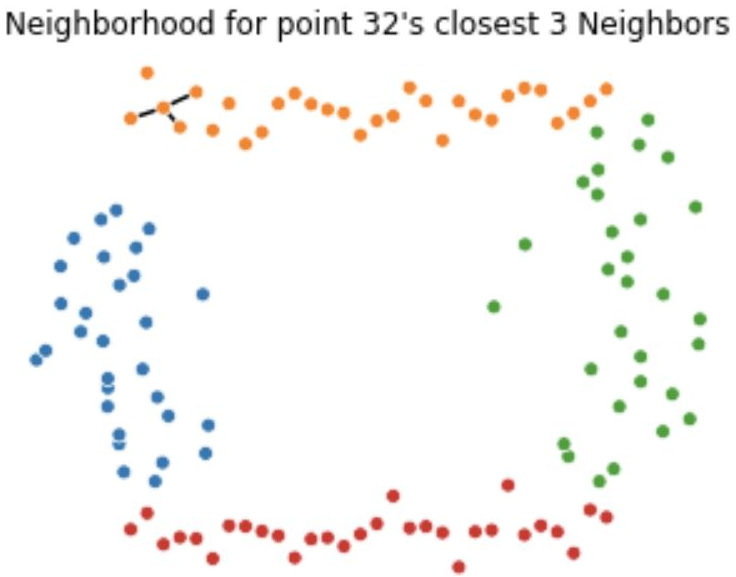
  2. Weight Matrix Construction: This is where LLE differs from Isomap. Instead of creating a matrix of geodesic distances, we'll attempt to capture 'locally linear' information about the data in the resulting matrix.

  3. Eigenvalue decomposition: Again, this is just like in Isomap. We essentially apply PCA to the matrix coming out of step 2.

Details of LLE

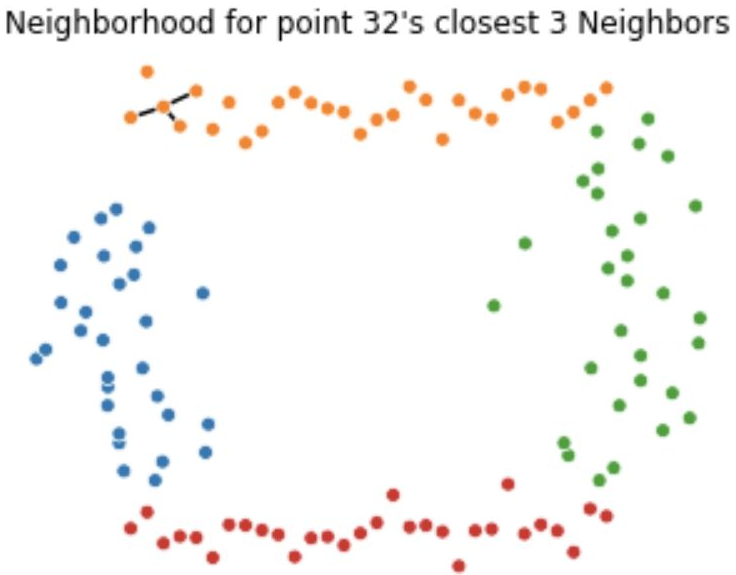
- Instead of viewing the full graph, we can consider a single observation of interest and it *n* closest neighbors

- To construct the weight matrix, we will create a linear model where the neighbors are the inputs and the observation of interest is the target.



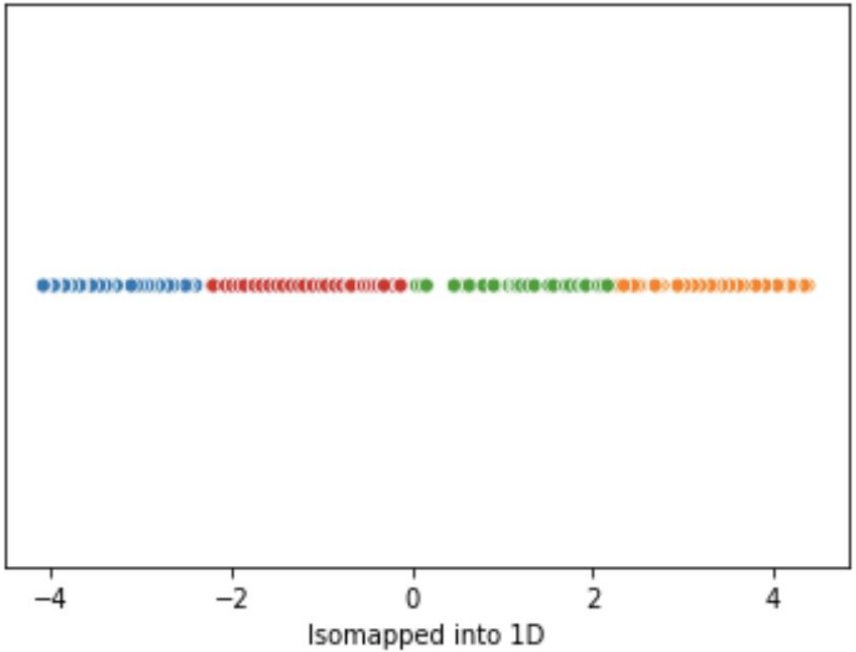
Details of LLE

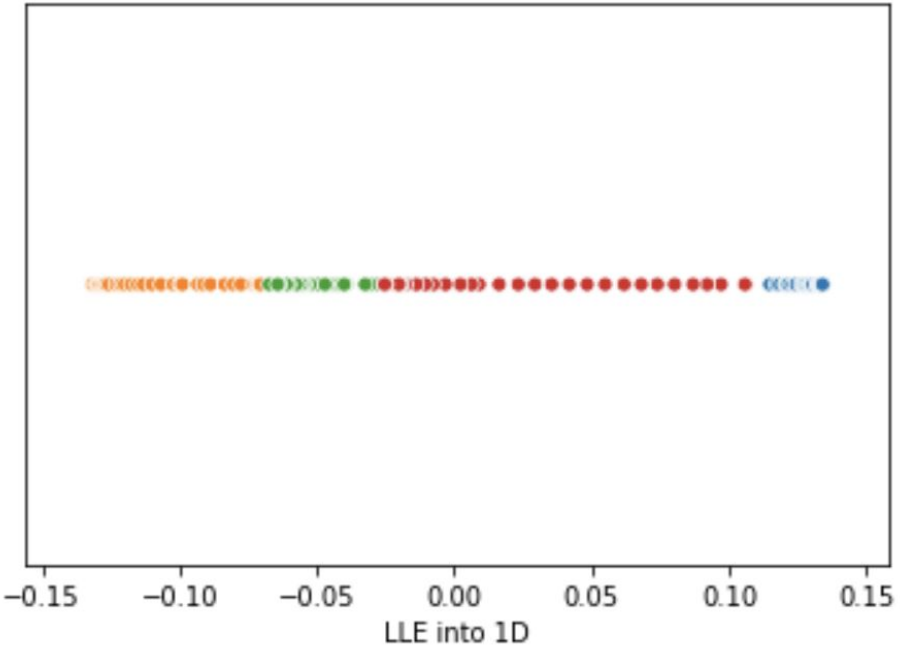
- We can then apply dimension reduction to this weight matrix that is holding this “locally linear” information.



Details of LLE

- Here are the results of applying both Isomap and LLE to our data.



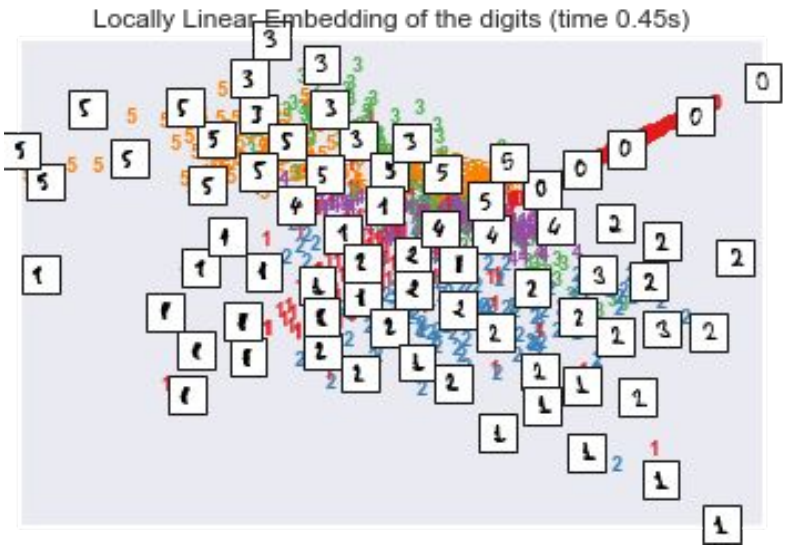


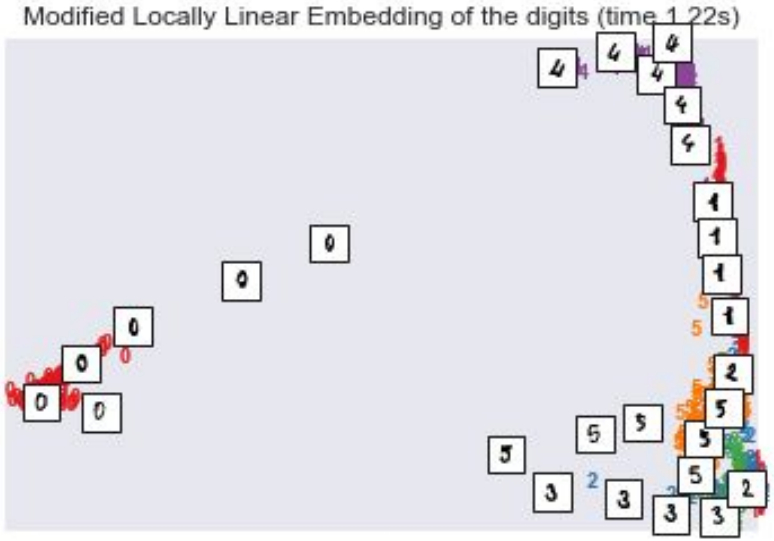
Details of MLLE

- There are some mathematical issues that arise based on LLE's processing (you can read more in `sklearn`'s docs here)
- To avoid these issues, `sklearn` has implemented the `reg` parameter. Alternatively, we can use `method='modified'` when using `LocallyLinearEmbedding()`
- The computational complexity of MLLE is higher than LLE and because of this, it will be slower to `fit`

MLLE Example

- MLLE does a better job at separating the various digits; however, it takes around 3x as long





When not to use (M)LLE

- Some of the same issues as Isomap: “Isomap, LLE, and variants are best suited to unfold a single continuous low dimensional manifold” - `sklearn` docs
- Unlike PCA, LLE is not suited to apply dimension reduction to new observations.
- The method inherently focuses on small local neighborhoods, what if we care about global structure?

Summary

Non-linear manifold methods can offer a potential advantage over linear methods depending on the structure of your data.

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

One of the limitations of the methods seen today is there extension to more 'real world' data.

We'll soon continue our discussion of manifold methods with the more commonly used *t*-SNE and UMAP algorithms.

Assignment

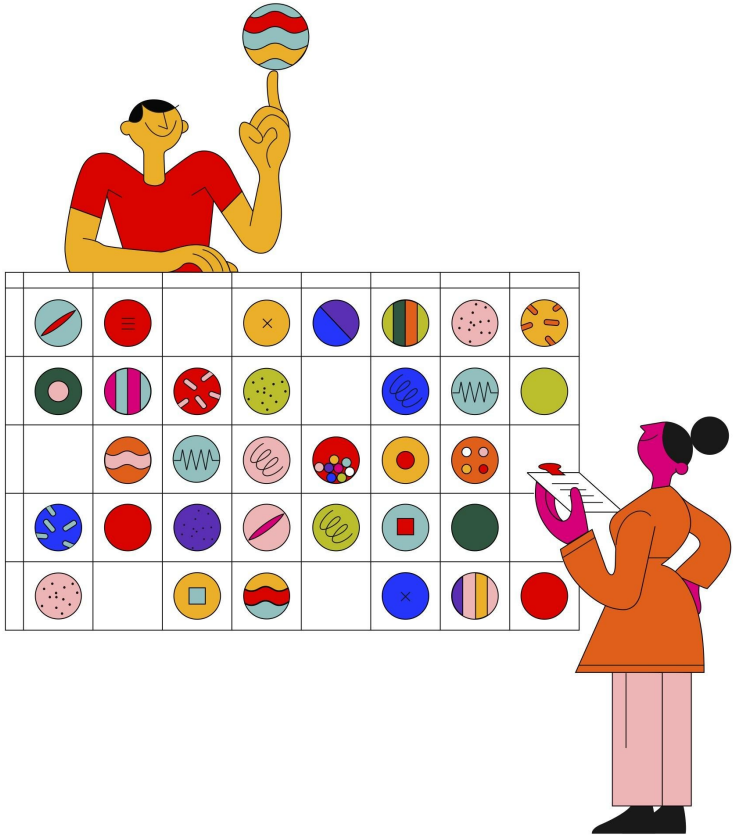
Dataset: Kaggle Red Wine Dataset available at Red Wine Quality
This dataset has 1599 data points with 12 features on wine quality. The target variable is wine quality and ranges from 0 to 10

In this assignment you need to do the following:
1) Load the the wine quality data set.
2) Fit PCA to all 12 components 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) Fit LLE with the same number of components and 30 neighbors, then repeat 4).
6) Record your observations and identify your top performing model. Does manifold learning improve predictive performance over PCA?

Thank You

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.)

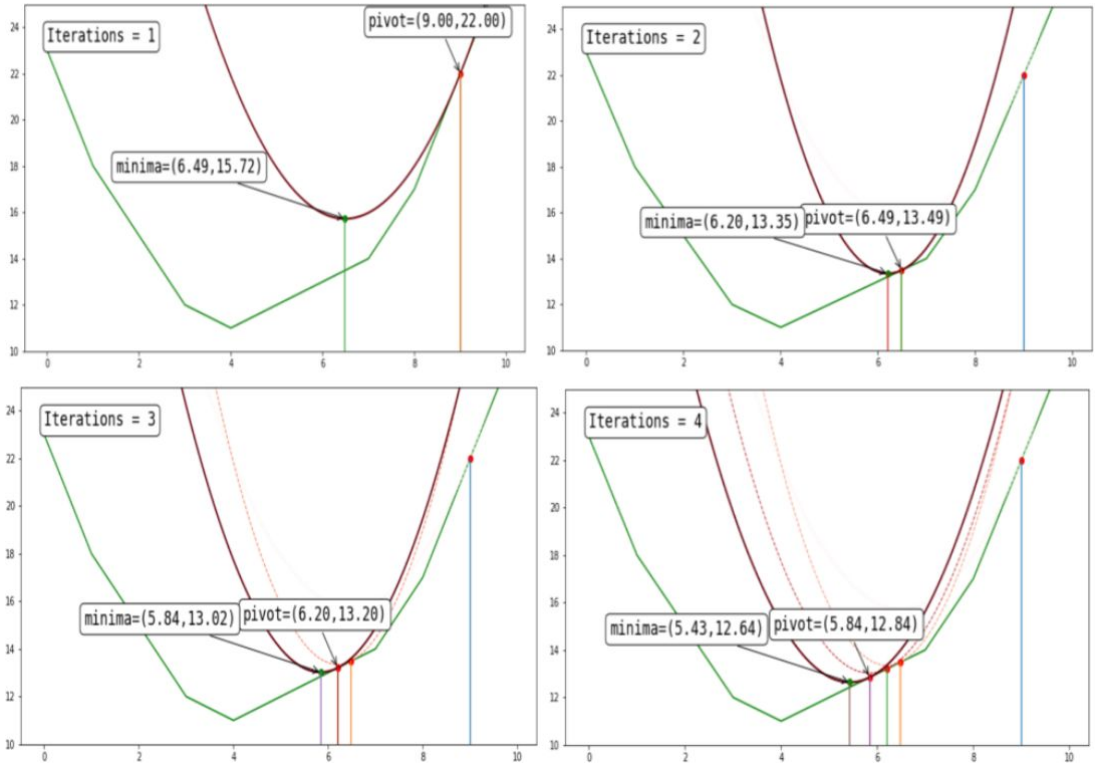


Image source: [paperspace blog]

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)