KIG4068 : Machine Learning

Week 9: Dimensionality Reduction

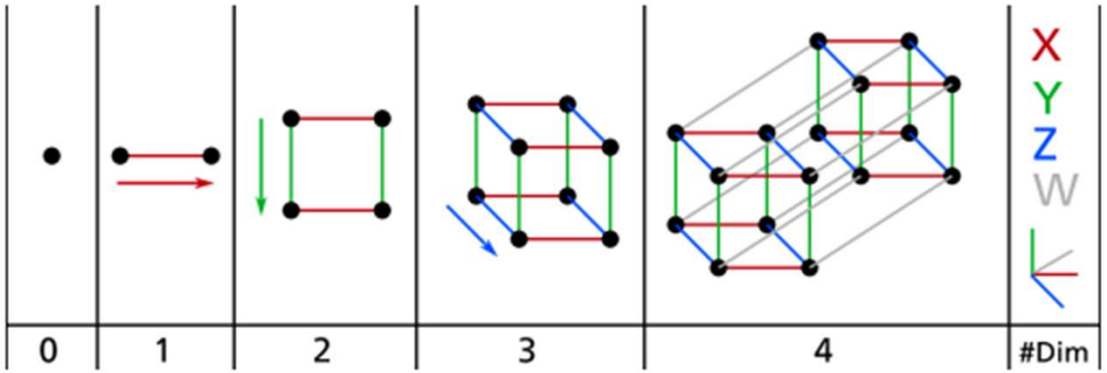
Semester 2, Session 2023/2024

Dimensionality Reduction

Curse of dimensionality

- High-dimensional datasets are often very sparse: most training instances are likely to be far away from each other.
- New instance will likely be far away from any training instance => harder to generalize well.
- High-dimensional datasets are prone to overfitting.
- More data? Data required to achieve given density of coverage grow exponentially with number of dimensions.

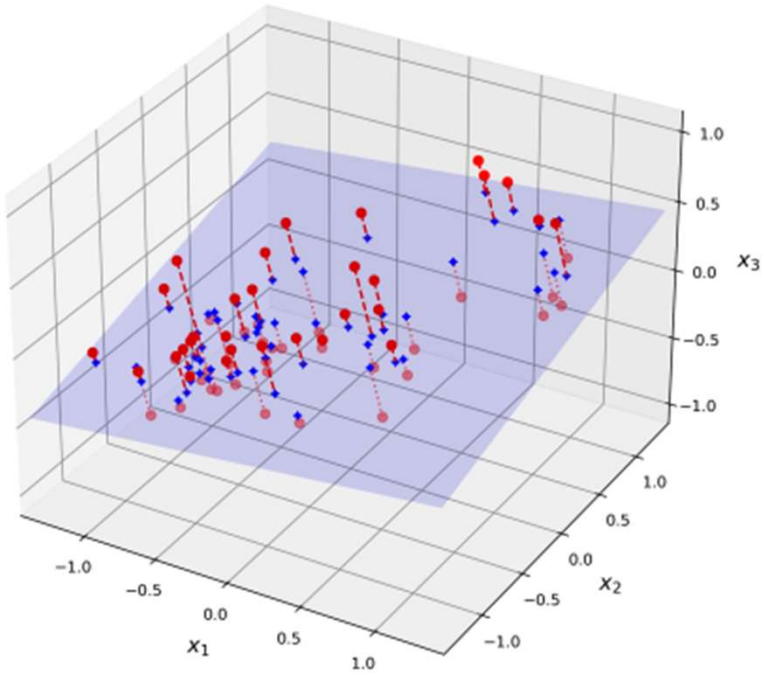
High-dimensional space



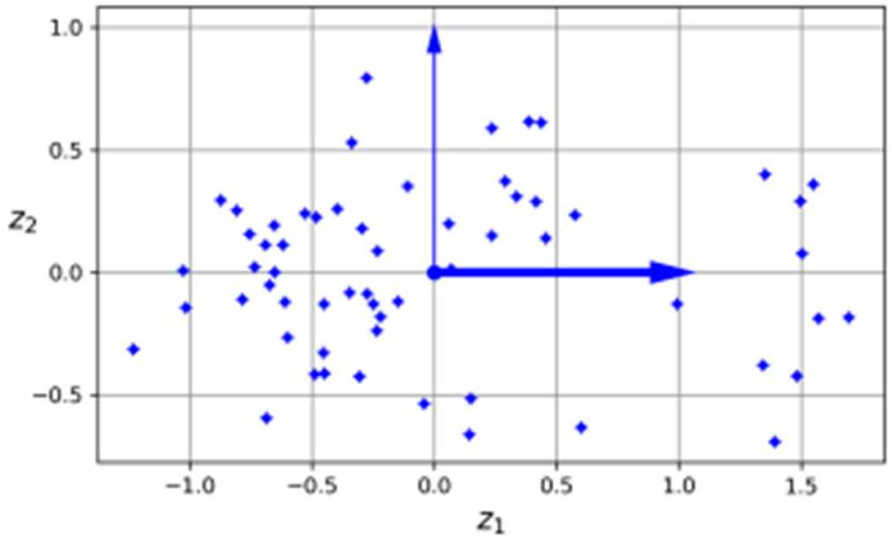
Point, segment, square, cube, and tesseract (0D to 4D hypercubes)[2]

Approaches for Dimensionality Reduction (Projection)

3D dataset lying close to 2D subspace

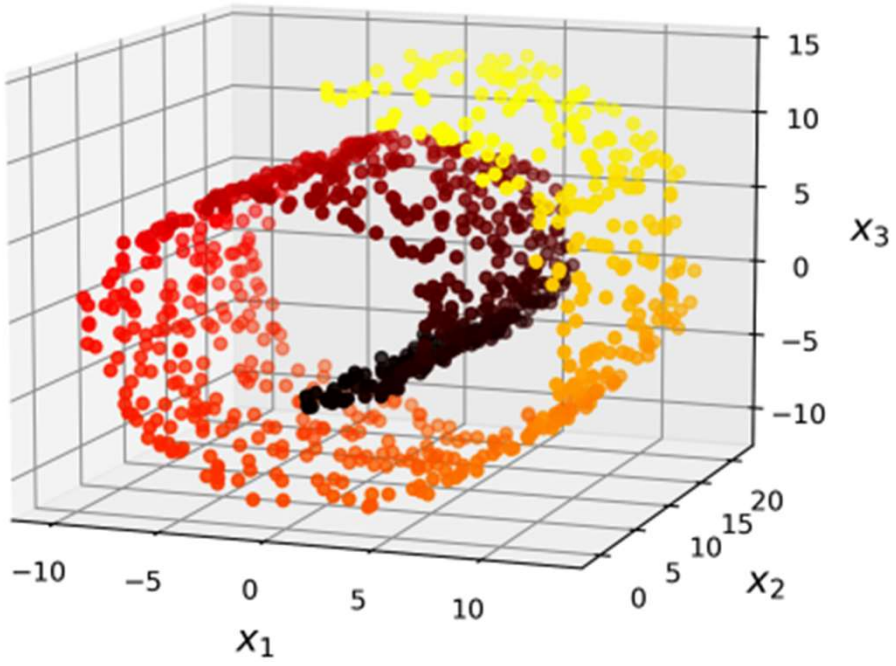


2D dataset after projection

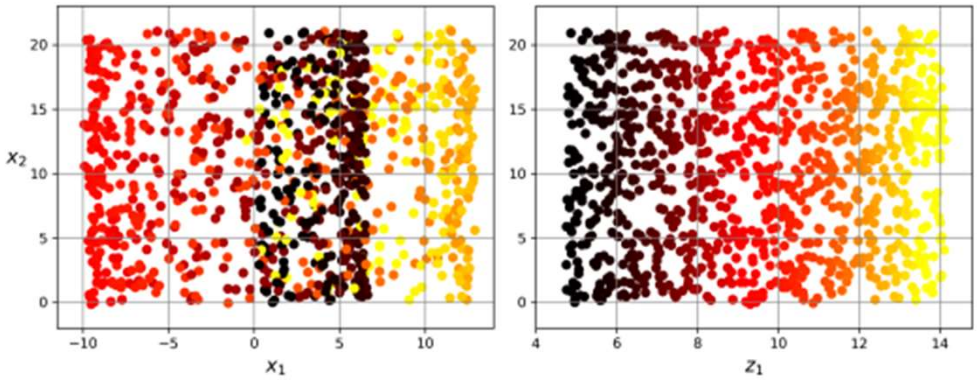


Approaches for Dimensionality Reduction (Manifold Learning)

Classic Swiss Roll Dataset



Projection (left) vs Unrolling (right)



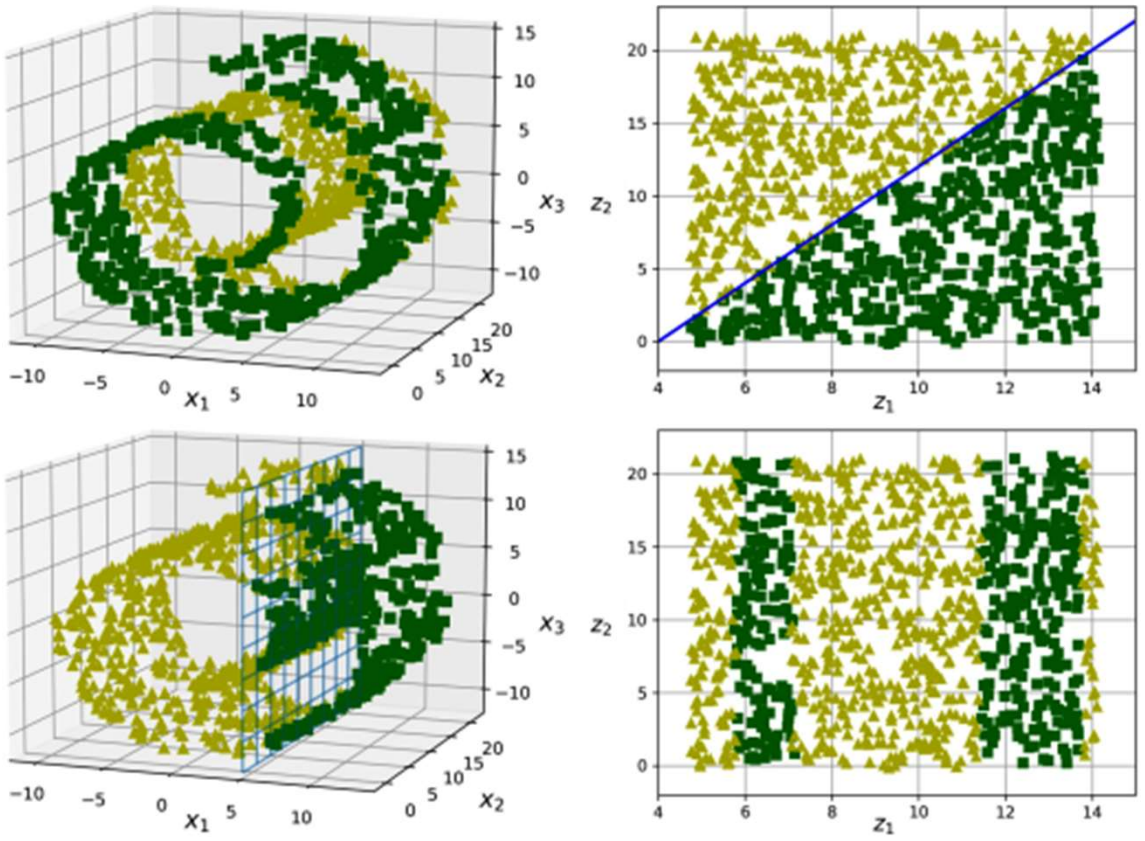
- Left: simply projecting onto a plane would squash different layers of the Swiss Roll together
- Right: ‘unroll’ the Swiss Roll to obtain the 2D dataset instead

Approaches for Dimensionality Reduction (Manifold Learning)

With manifold learning you try to learn the lower dimensional space that "best" represents the higher dimensional data.

- Relies on manifold hypothesis: most real-world high-dimensional datasets lie close to a much lower-dimensional manifold. Assumption is often observed empirically.
- Another implicit assumption: classification or regression task will be "easier" if expressed in the lower-dimensional space of the manifold. Does not always hold in practice.

Approaches for Dimensionality Reduction (Manifold Learning)



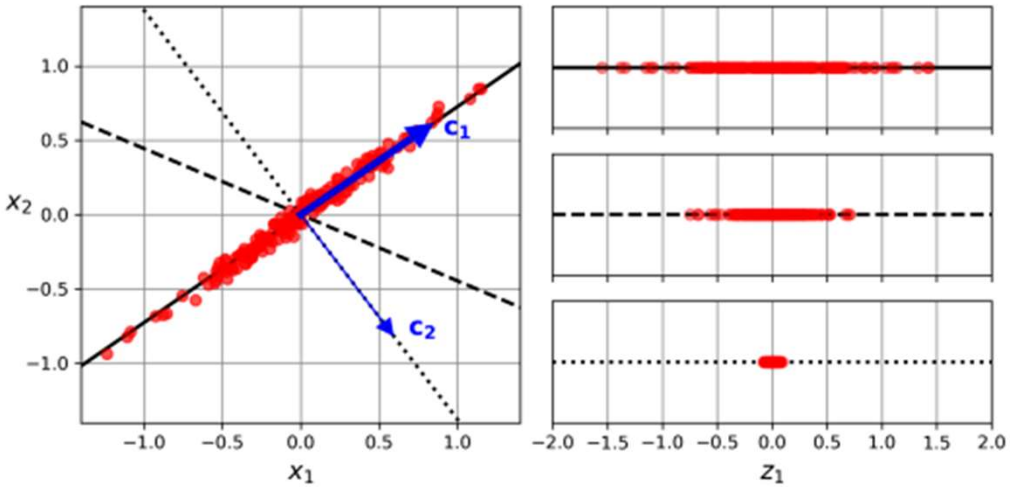
The decision boundary may not always be simpler with lower dimensions

Principal Components Analysis (PCA)

What is PCA?

- Find the lower dimensional hyperplane "closest" to the higher dimensional data.
- Project the data onto the lower dimensional hyperplane.
- "Closest" means preserves the most variation in the training data.

Selecting the subspace to project on



- Left – a 2D dataset with three different projection axes (hyperplanes)
- Right – Projection results:
  1. Top: Preserves max variance, retaining the most information
  2. Middle: Preserves an intermediate amount of variance
  3. Bottom: Preserves minimal variance, losing more information

Principal Components Analysis (PCA)

Principal components

- PCA identifies the axis that accounts for the largest amount of variance in the training set (solid line on the previous figure).
- PCA also finds a second axis, orthogonal to the first one, that accounts for the largest amount of remaining variance (dotted line on the previous figure).
- For an n-dimensional training set, PCA will find n principal components.
- Choosing d < n principal components allows you to project n-dimensional training set into a d-dimensional training set.

PCA in Scikit-Learn

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
X2D = pca.fit_transform(X)
```

To identify the ratio of the dataset's variance that lies along each principal component:

```
pca.explained_variance_ratio_
```

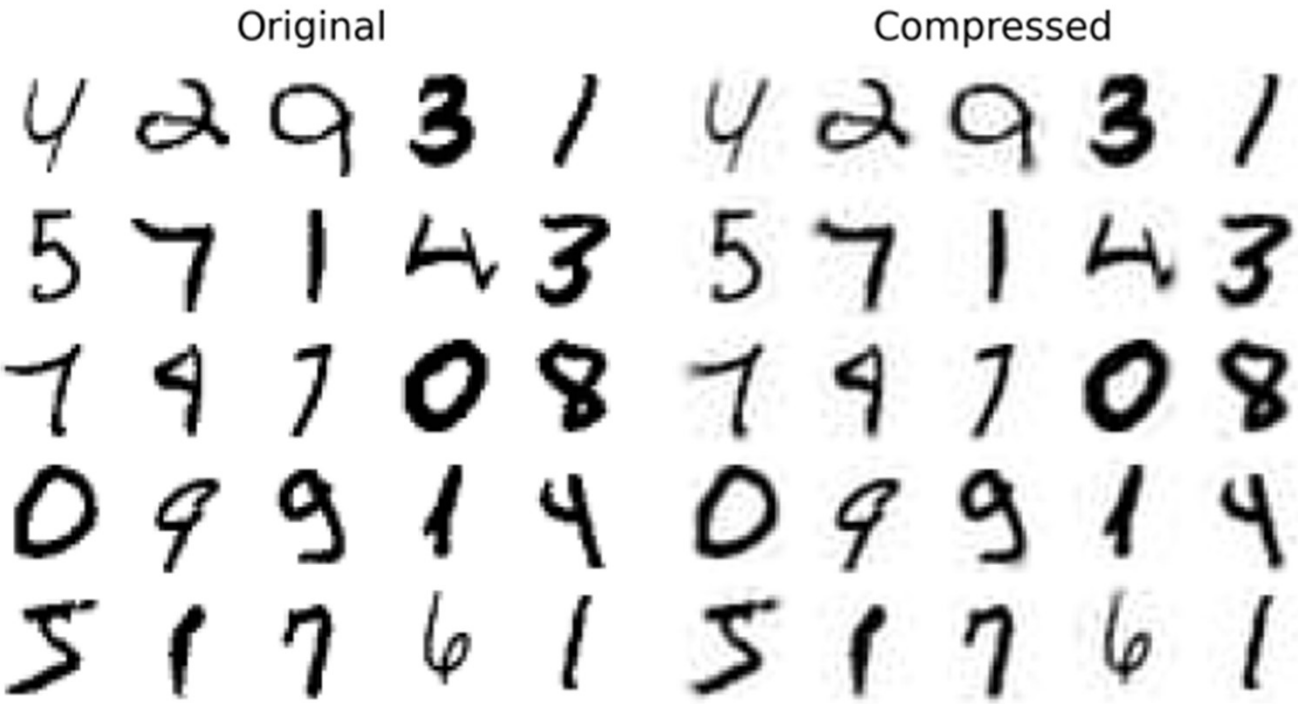
To choose the right number of dimensions, you can set n_components to be a float between 0.0 and 1.0, indicating the ratio of variance you wish to preserve:

```
pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X_train)
```

The actual number of components is determined during training, and can be extracted using:

```
pca.n_components_
```

PCA for Compression



MNIST compression that preserves 95% of the variance

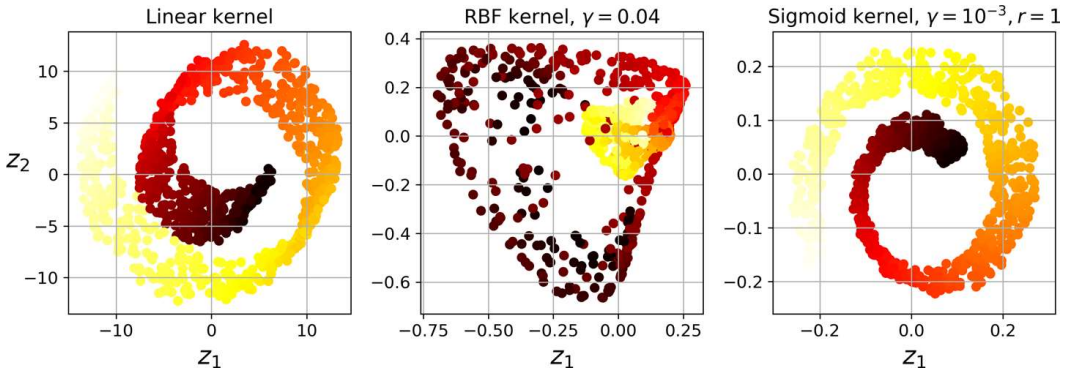
- Left – Some digits from original training set (784 features)
- Right – Projection results: Corresponding digits after applying PCA while preserving 95% of its variance (154 features). There is slight image quality loss, but the digits are still mostly intact. The dataset is now less than 20% of its original size, and only lost 5% of its variance.

Kernel PCA (kPCA)

What is kPCA?

- Can combine the "kernel trick" with PCA to perform complex non-linear projections.
- Good at preserving clusters of instances after projection, or sometimes even unrolling datasets that lie close to a twisted manifold.
- Compute and memory intensive; doesn't scale to large datasets.

Reducing swill roll to 2D with various kernels




```python
from sklearn.decomposition import KernelPCA

# Linear kernel
kpca_linear = KernelPCA(n_components=2, kernel='linear')

# RBF kernel
kpca_rbf = KernelPCA(n_components=2, kernel='rbf', gamma=0.1)

# Sigmoid kernel
kpca_sigmoid = KernelPCA(n_components=2, kernel='sigmoid',
                         gamma=0.01, coef0=1)
```

kPCA in Scikit-Learn

```python
import numpy as np
from sklearn.decomposition import KernelPCA

# Step 1: Apply Kernel PCA with all components
kpca_all = KernelPCA(kernel="rbf", gamma=15)
X_kpca_all = kpca_all.fit_transform(X)
eigenvalues = kpca_all.lambdas_

# Step 2: Calculate cumulative variance
total_variance = np.sum(eigenvalues)
cumulative_variance = np.cumsum(eigenvalues) / total_variance

# Step 3: Find number of components for desired variance
n_components = np.argmax(cumulative_variance >= 0.95) + 1

# Step 4: Re-run Kernel PCA with the determined number of components
kpca = KernelPCA(n_components=n_components, kernel="rbf", gamma=15)
X_kpca = kpca.fit_transform(X)
```

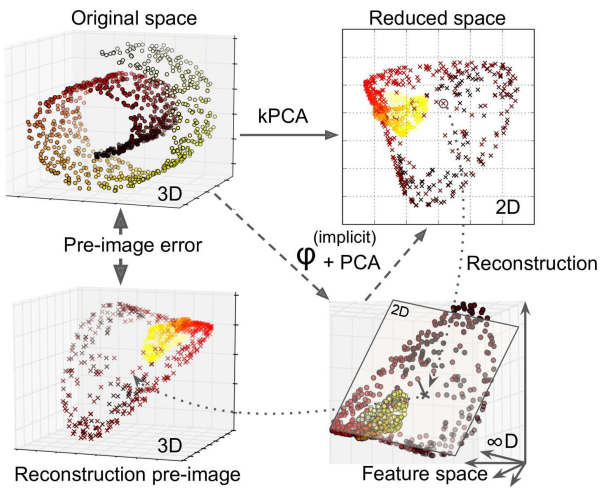
Assess the importance of each principal component

Kernel PCA (kPCA)

Selecting kernels and tuning hyperparameters

- If using kPCA as preprocessing step in classification/regression pipeline, then choose kernel and tune kPCA hyperparameters to max performance of the whole pipeline.
- Alternatively, choose kernel and tune kPCA hyperparameters to minimize reconstruction error.

kPCA and reconstruction pre-image error



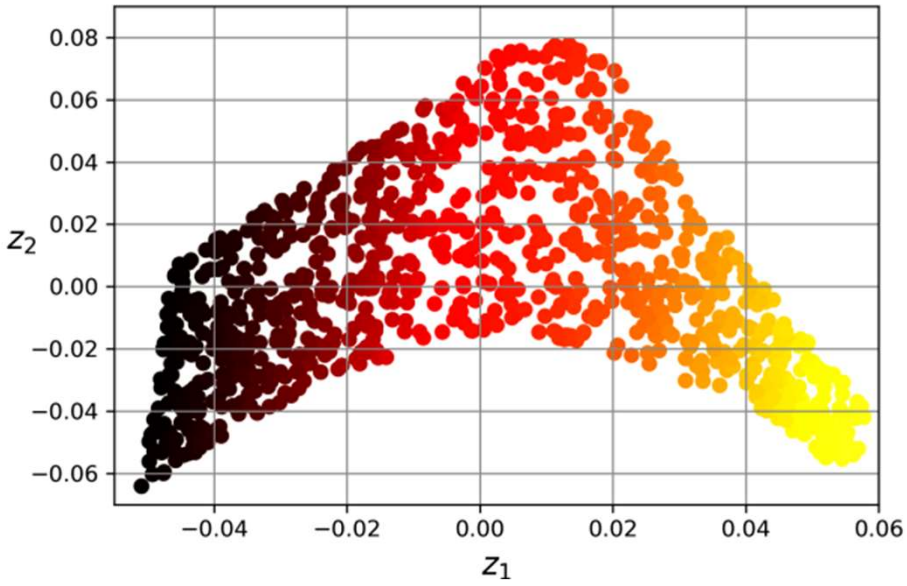
- Original space: Complex, non-linear relationships among the data points.
- Reduced space: After applying kPCA in reduced 2D space.
- Feature space: Visualize the result of 2D to 3D transformation, projecting onto a few key dimensions that capture most variance.
- Reconstruction Pre-image: Result of reconstruction showing how well the data can be mapped backed to its original space.

Locally-Linear Embedding (LLE)

What is LLE?

- Non-linear dimensionality reduction technique.
- Unlike previous algorithms, LLE doesn't rely on projections.
- Measures how each training instance linearly relates to its nearest neighbors, then looks for a low-dimensional representation of the training set that preserves those local relationships.
- Good at unrolling twisted manifolds, especially when there is not too much noise, but scales poorly to large datasets.

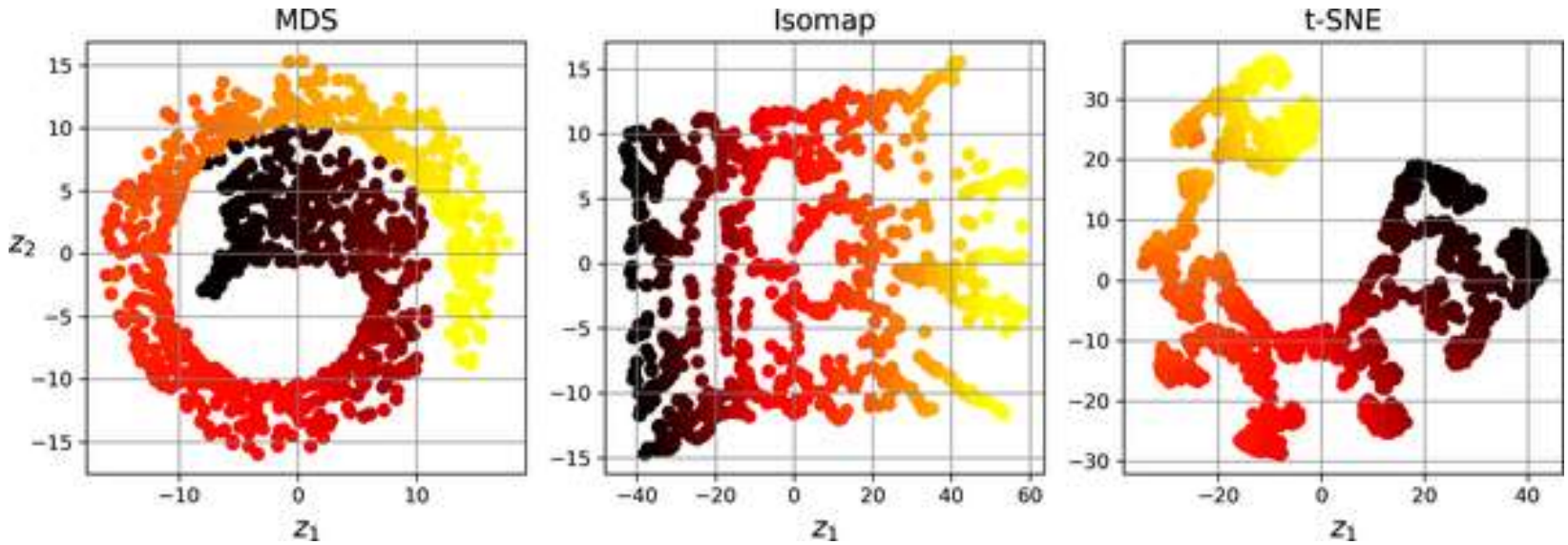
Unrolling the Swiss Roll with LLE




```
from sklearn.datasets import make_swiss_roll
from sklearn.manifold import LocallyLinearEmbedding

X_swiss, t = make_swiss_roll(n_samples=1000, noise=0.2, random_state=42)
lle = LocallyLinearEmbedding(n_components=2, n_neighbors=10, random_state=42)
X_unrolled = lle.fit_transform(X_swiss)
```

Other Approaches



- MDS: seeks to place each data point in a lower-dimensional space such that the distances between points in this space reflect their similarities as closely as possible.
- Isomap: a technique that compresses data into fewer dimensions while trying to keep the distances between data points similar to what they were in the higher dimensions.
- t-SNE: a method that reduces the complexity of high-dimensional data into 2 or 3 dimensions, aiming to keep similar data points close and dissimilar data points far apart in the reduced space.

MDS, Isomap and t-SNE in Scikit-Learn

```
# MDS
from sklearn.manifold import MDS
mds = MDS(n_components=2, random_state=42)
X_mds = mds.fit_transform(X)

#Isomap
from sklearn.manifold import Isomap
isomap = Isomap(n_neighbors=10, n_components=2)
X_isomap = isomap.fit_transform(X)

#t-SNE
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=300)
X tsne = tsne.fit transform(X)
```

*Perplexity - a key parameter that affects how the algorithm behaves, specifically in terms of balancing attention between local and global aspects of the data.