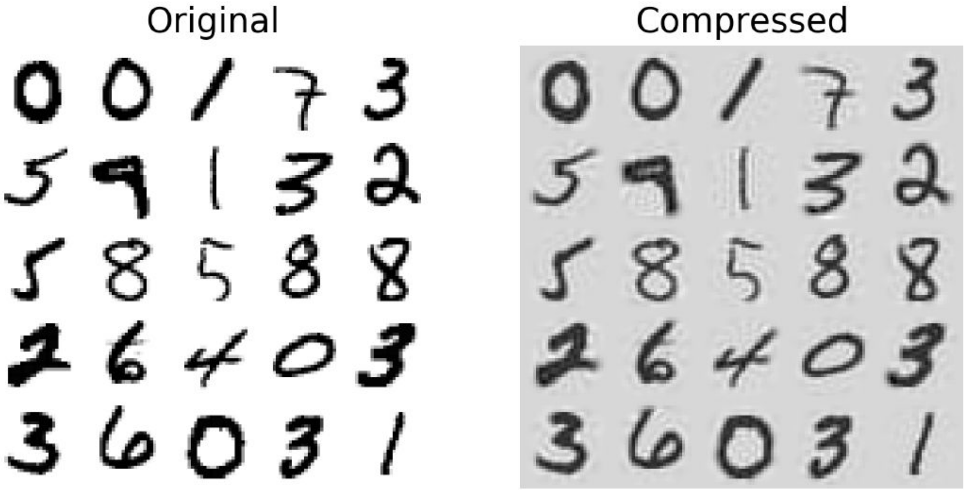
Dimensionality Reduction

Hands-on Machine Learning: Chapter 8

Dimensionality Reduction

Reduce number of features. Simplifying problems
Speed up training. Compression. Data visualization.



Curse of Dimensionality

Training instances grow exponentially with each dimension
Human intuition fails beyond 3D

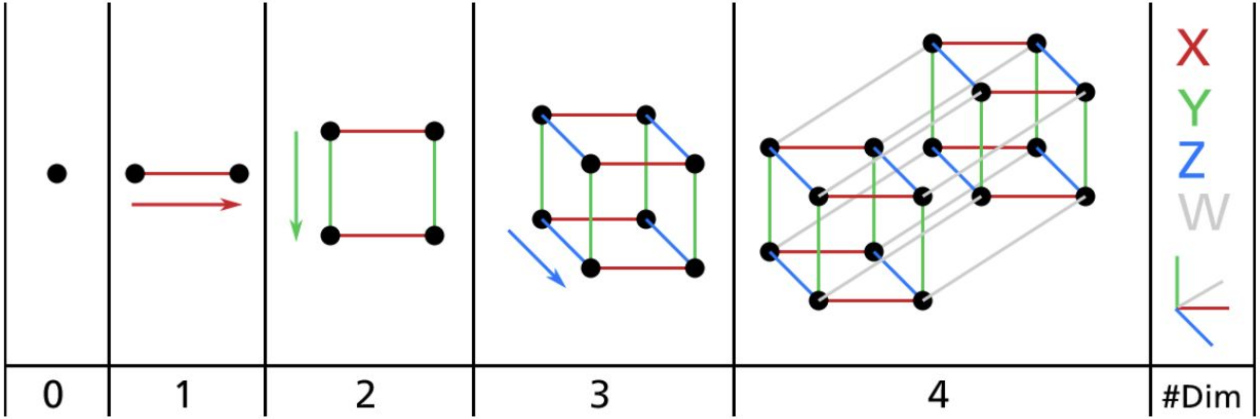
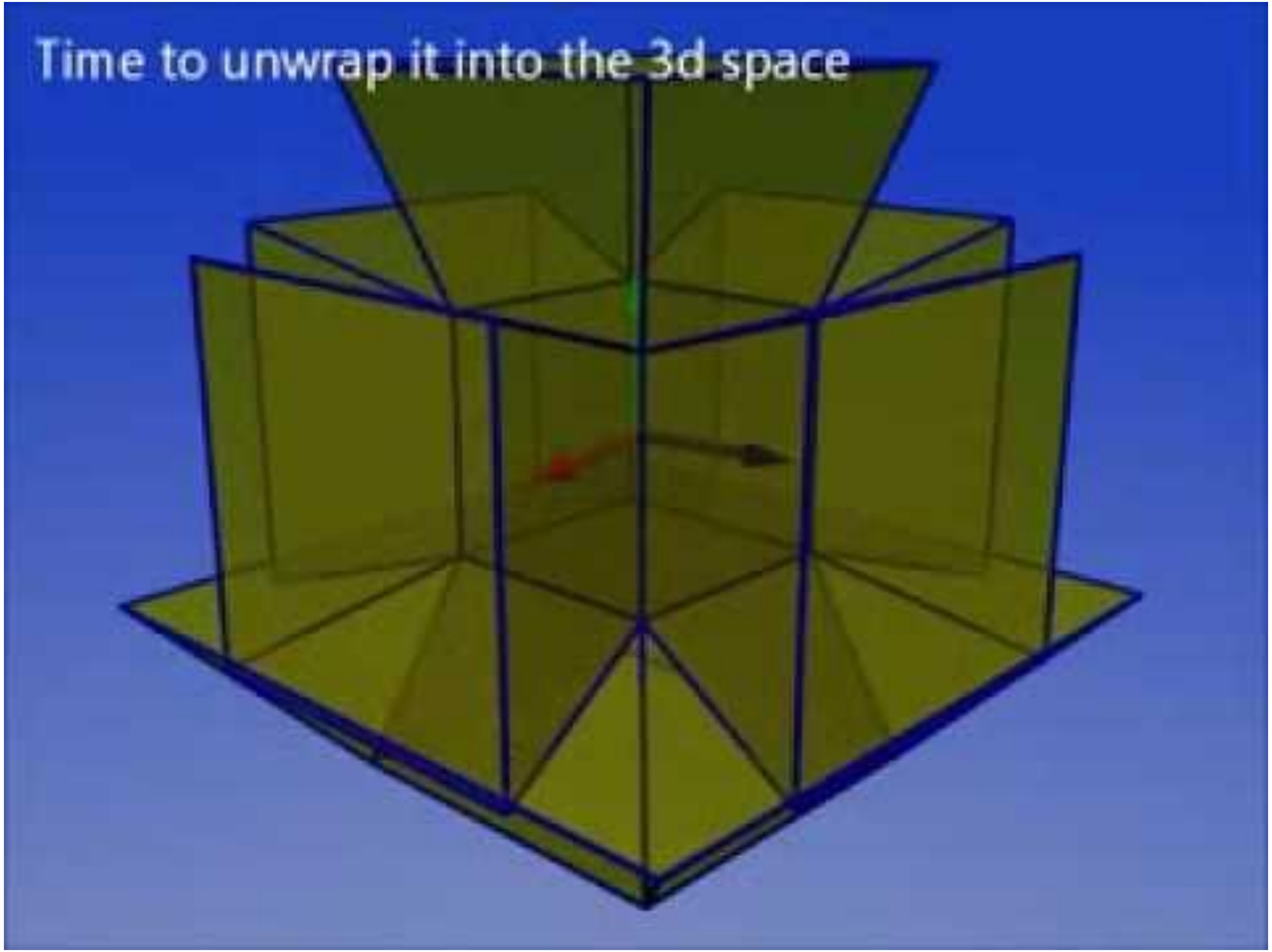


Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)[2]

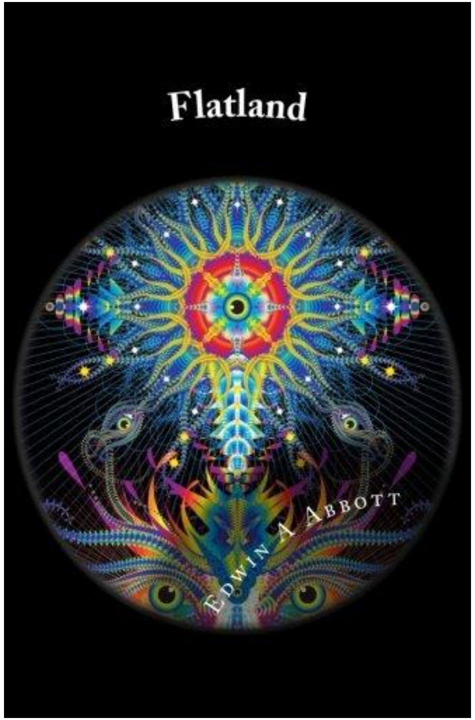

Time to unwrap it into the 3d space

Curse of Dimensionality

Flatland: A Romance of Many Dimensions
by Edwin Abbott 1884

- Square in 2-D polygon world
- Visited by 3-D sphere describing space
- Novella on math + philosophy, political satire, Victorian class structure, religious commentary

Sequel Flatterland
Alice in Wonderland meets The Phantom Tollbooth

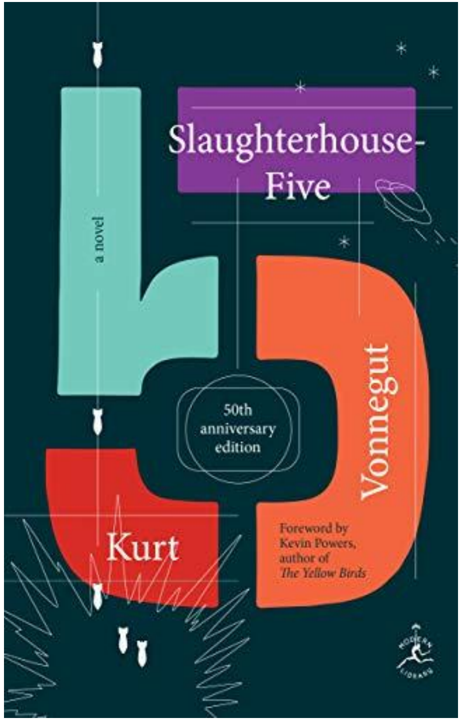


Curse of Dimensionality

Slaughterhouse-Five by Kurt Vonnegut 1969

- Autobiographical satire
- Historical fiction WWII firebombing of Dresden
- Abducted by aliens from Tralfamador living in 4th dimension

Force readers to look differently at the world



Projection

Map high-dimensional space into lower-dimensional subspace

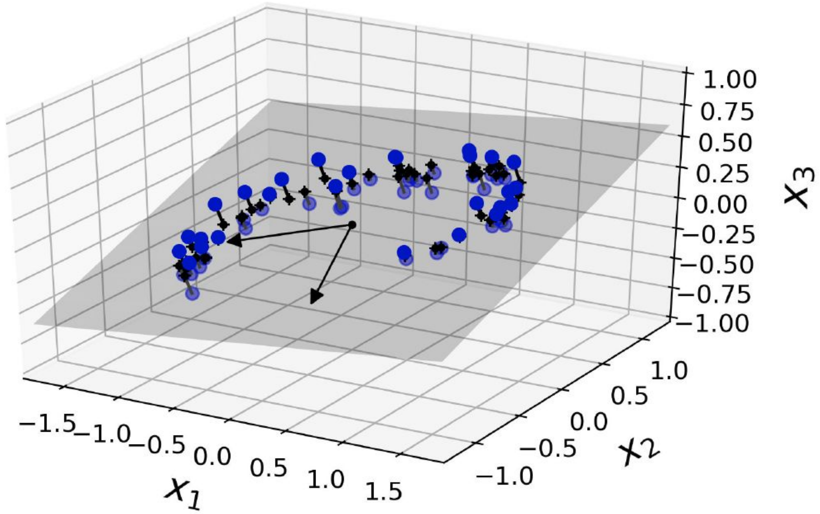


Figure 8-2. A 3D dataset lying close to a 2D subspace

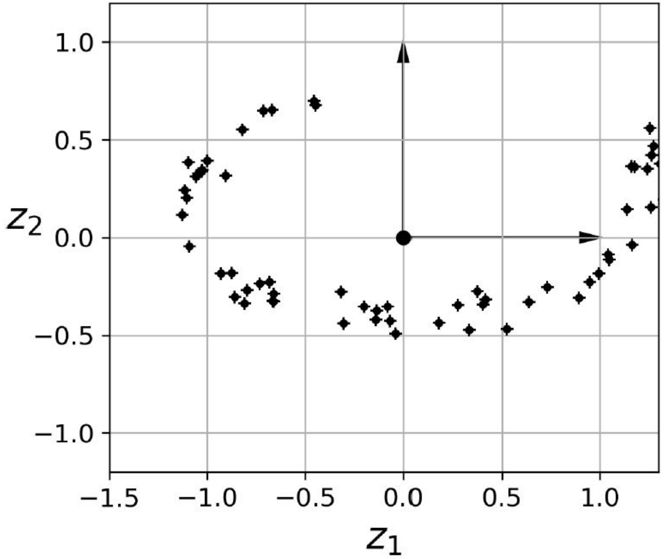


Figure 8-3. The new 2D dataset after projection

Manifold Learning

Swiss roll bents & twisted unrolled.

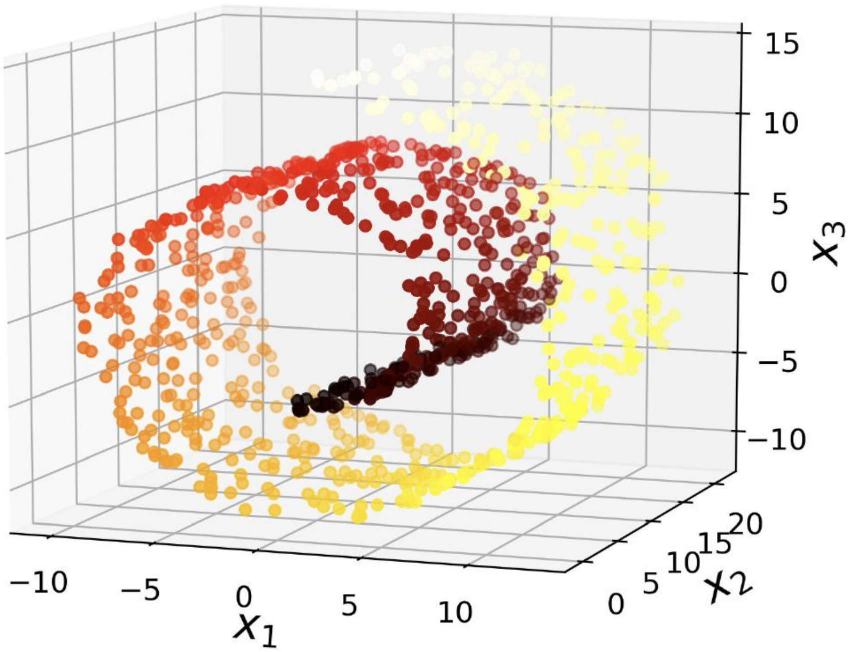
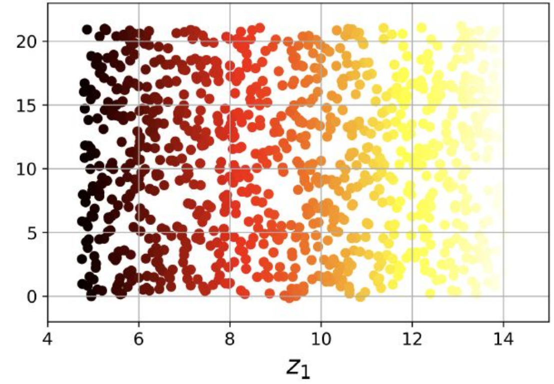


Figure 8-4. Swiss roll dataset



versus unrolling the Swiss roll

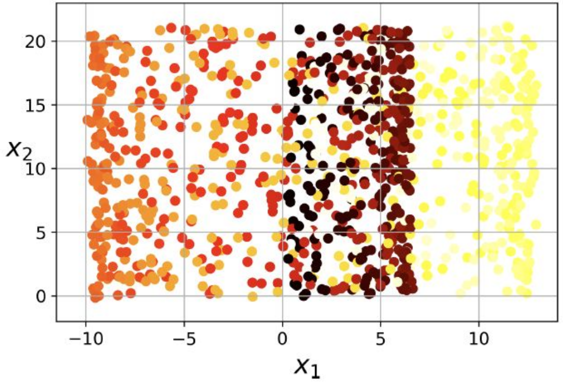


Figure 8-5. Squashing by projecting onto a plane

Manifold Learning

d-dimensional manifold is part of an **n**-dimensional space (where **d < n**) locally resembles a d-dimensional hyperplane

The Swiss roll, d = 2 and n = 3 locally resembles a 2D plane, but it is rolled in the third dimension

Manifold assumption / hypothesis
Most real-world high-dimensional datasets lie close to a much lower-dimensional manifold

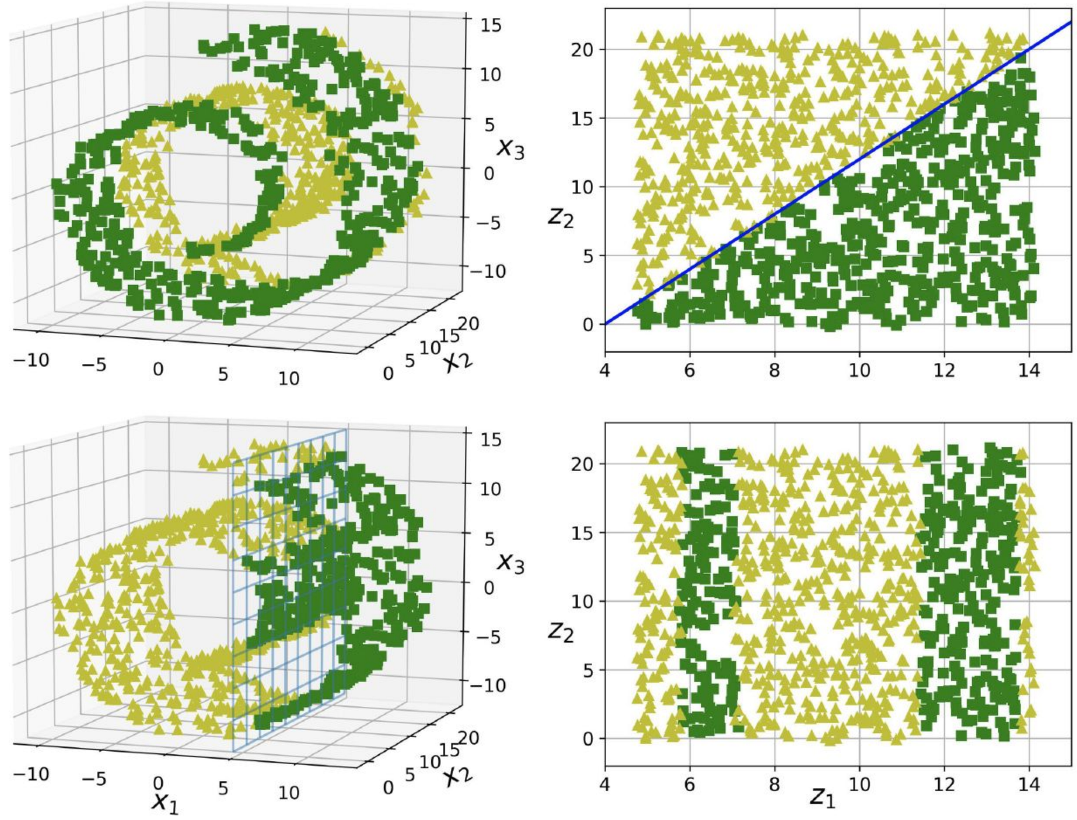
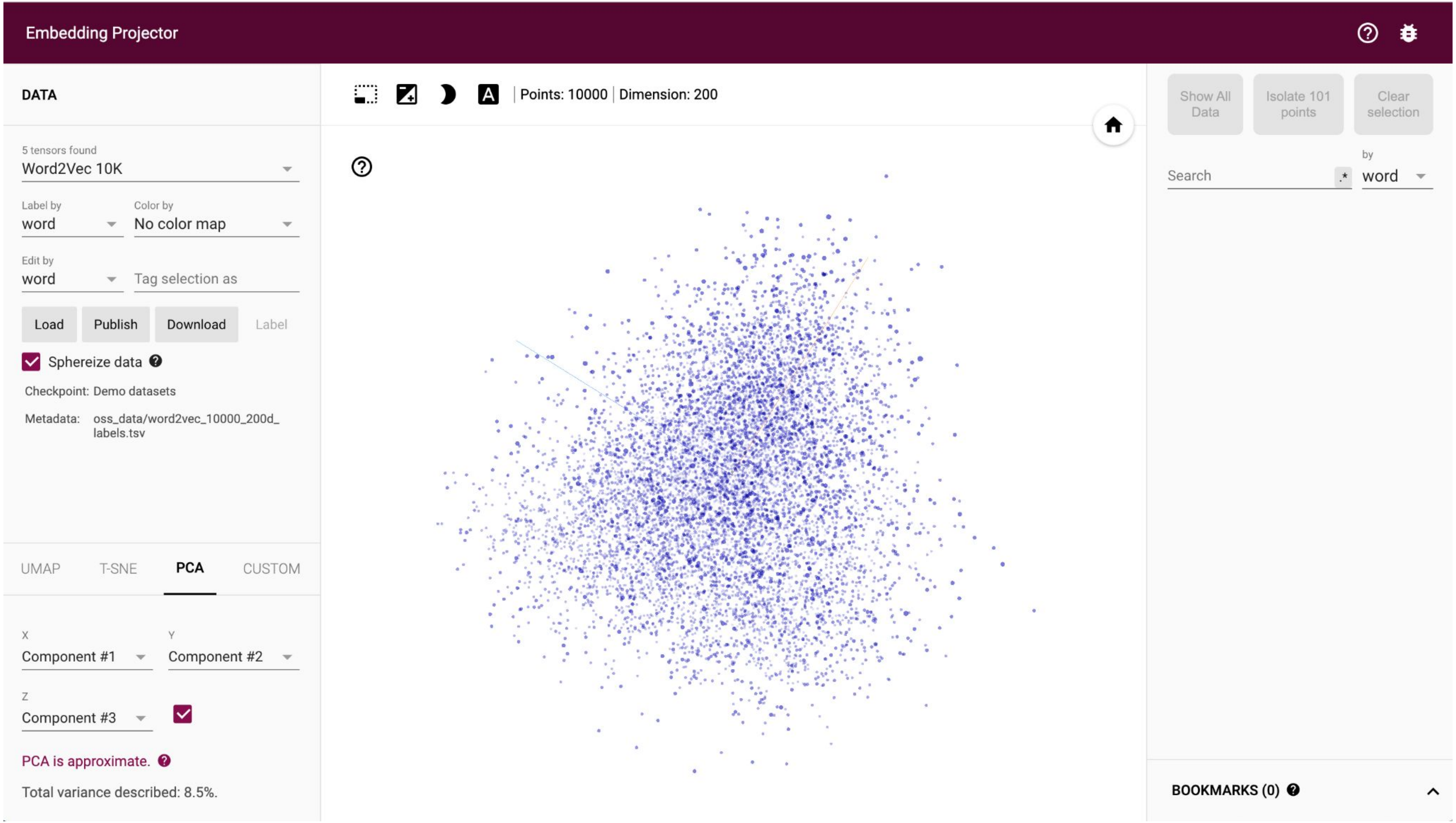


Figure 8-6. The decision boundary may not always be simpler with lower dimensions


Embedding Projector
DATA
5 tensors found
Word2Vec 10K
Label by
word
Color by
No color map
Edit by
word
Tag selection as
Load
Publish
Download
Label
Sphereize data
Checkpoint: Demo datasets
Metadata: oss_data/word2vec_10000_200d_labels.tsv
UMAP
T-SNE
PCA
CUSTOM
X
Component #1
Y
Component #2
Z
Component #3
PCA is approximate.
Total variance described: 8.5%.
Points: 10000
Dimension: 200
Show All Data
Isolate 101 points
Clear selection
Search
by
word
BOOKMARKS (0)

PCA - Principal Component Analysis

Identifies hyperplane closest to data, then projects onto it

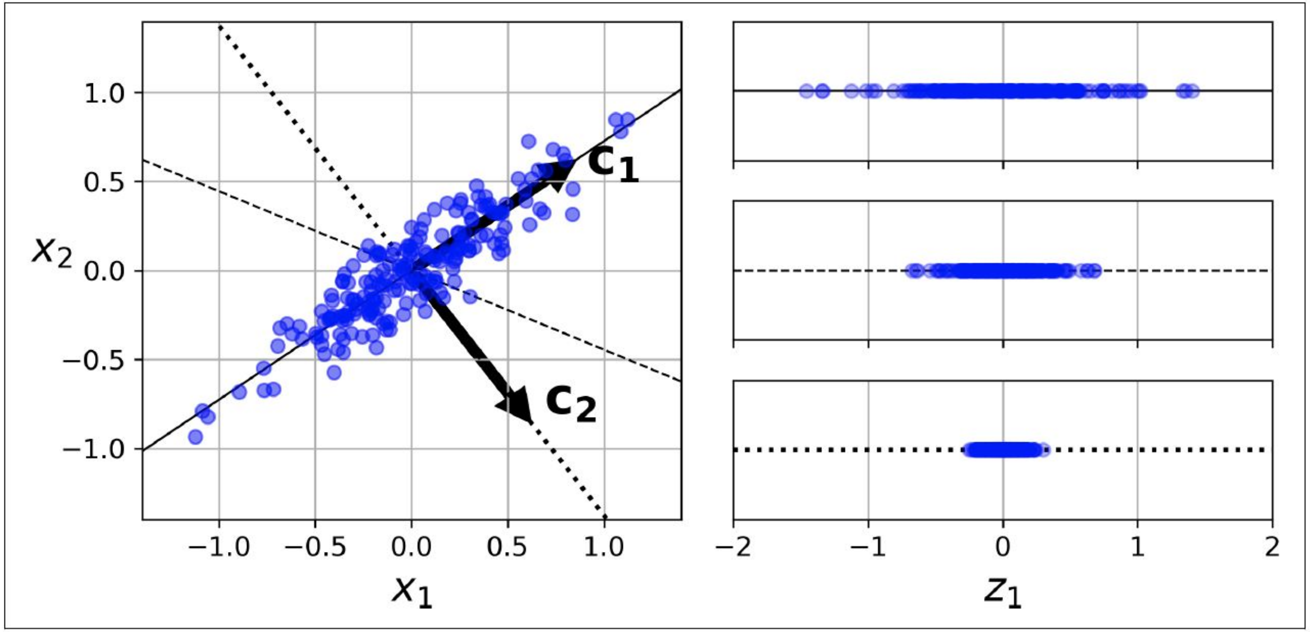


Figure 8-7. Selecting the subspace onto which to project

PCA - Explained Variance Ratio

```
X_centered = X - X.mean(axis=0)
U, s, Vt = np.linalg.svd(X_centered)
c1 = Vt.T[:, 0]
c2 = Vt.T[:, 1]

W2 = Vt.T[:, :2]
X2D = X_centered.dot(W2)

from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)

>>> pca.explained_variance_ratio_
array([0.84248607, 0.14631839])
```

Equation 8-1. Principal components matrix

$$\mathbf{V} = \begin{pmatrix} | & | & & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ | & | & & | \end{pmatrix}$$

Equation 8-2. Projecting the training set down to d dimensions

$$\mathbf{X}_{d\text{-proj}} = \mathbf{X}\mathbf{W}_d$$

PCA - Explained Variance Ratio

```
pca = PCA()
pca.fit(X_train)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1

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

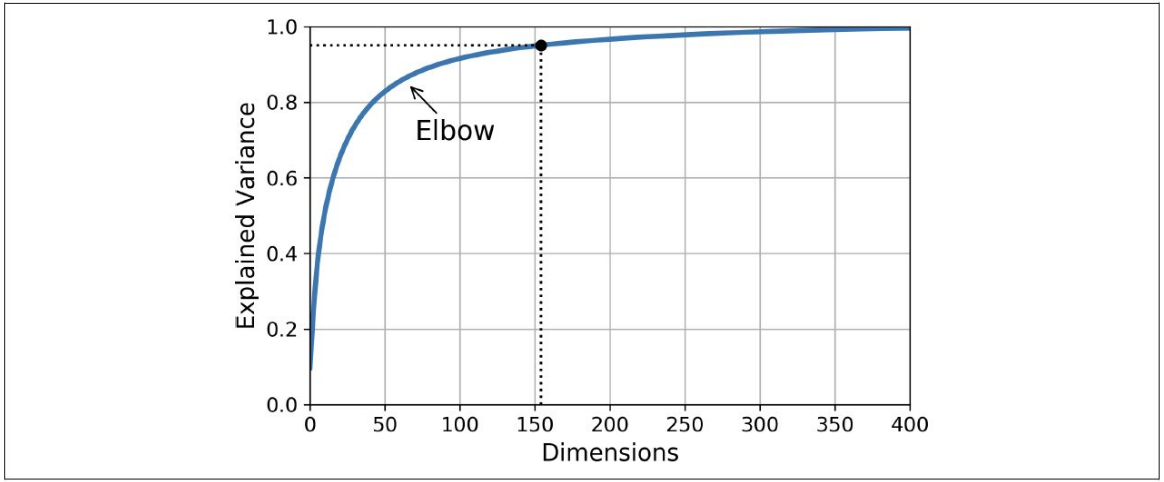


Figure 8-8. Explained variance as a function of the number of dimensions

PCA for Compression

```
pca = PCA(n_components = 154)
X_reduced = pca.fit_transform(X_train)
X_recovered = pca.inverse_transform(X_reduced)
```

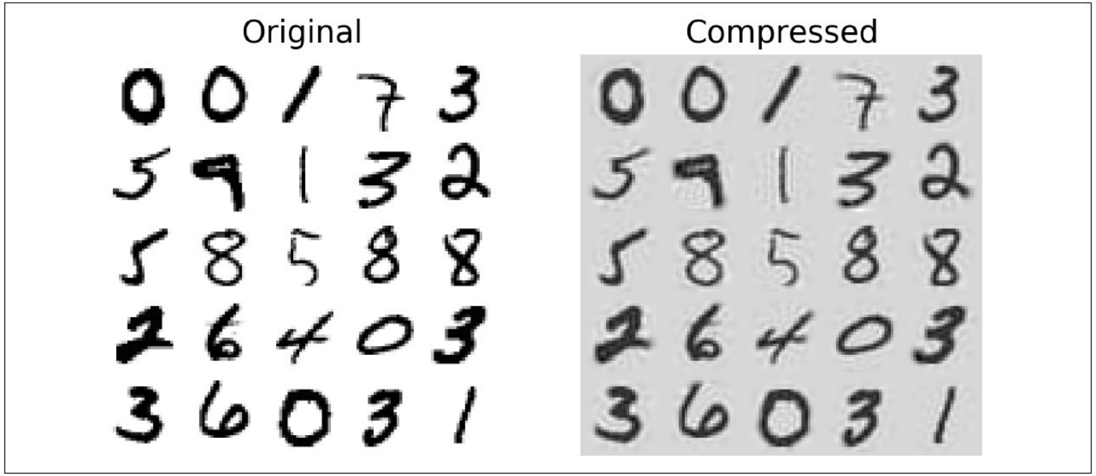


Figure 8-9. MNIST compression preserving 95% of the variance

Randomized & Incremental PCA

Randomized PCA

```
rnd_pca = PCA(n_components=154,
svd_solver="randomized")
X_reduced = rnd_pca.fit_transform(X_train)
```

Incremental PCA (IPCA)

```
from sklearn.decomposition import IncrementalPCA
n_batches = 100
inc_pca = IncrementalPCA(n_components=154)
for X_batch in np.array_split(X_train, n_batches):
inc_pca.partial_fit(X_batch)
X_reduced = inc_pca.transform(X_train)

X_mm = np.memmap(filename, dtype="float32",
mode="readonly", shape=(m, n))
batch_size = m // n_batches
inc_pca = IncrementalPCA(n_components=154,
batch_size=batch_size)
inc_pca.fit(X_mm)
```

kPCA - Kernel PCA

```
from sklearn.decomposition import KernelPCA
rbf_pca = KernelPCA(n_components = 2, kernel="rbf", gamma=0.04)
X_reduced = rbf_pca.fit_transform(X)
```

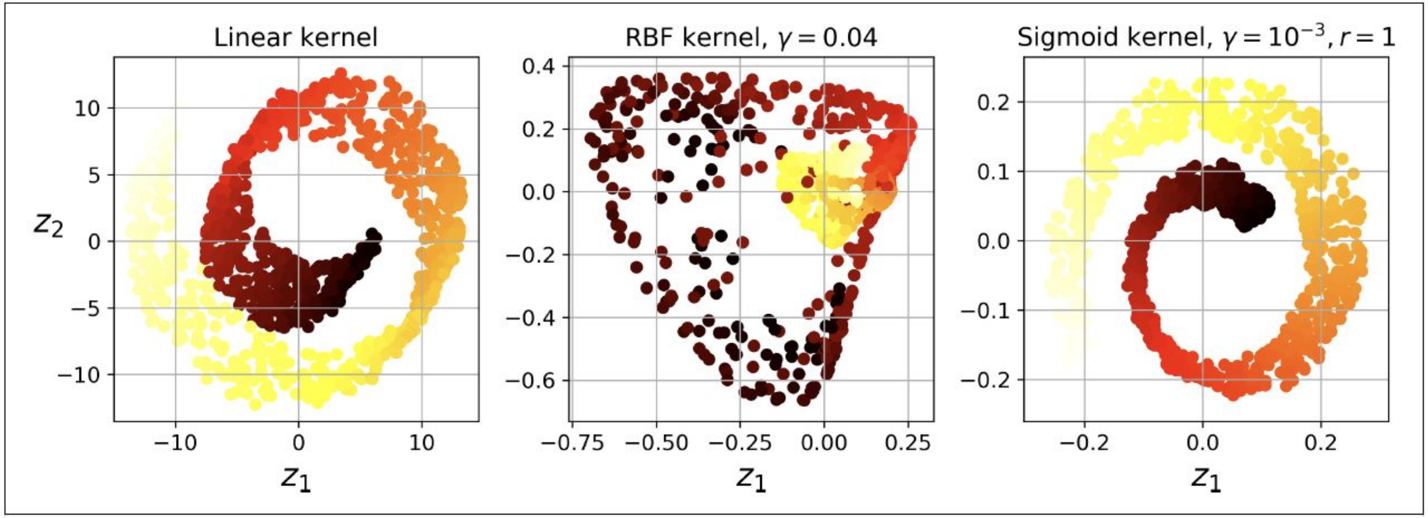


Figure 8-10. Swiss roll reduced to 2D using kPCA with various kernels

kPCA - Kernel PCA

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

clf = Pipeline([
("kpca", KernelPCA(n_components=2)),
("log_reg", LogisticRegression())
])
param_grid = [{
"kpca__gamma": np.linspace(0.03, 0.05, 10),
"kpca__kernel": ["rbf", "sigmoid"]
}]
grid_search = GridSearchCV(clf, param_grid, cv=3)
grid_search.fit(X, y)

>>> print(grid_search.best_params_)
{'kpca__gamma': 0.043333333333333335, 'kpca__kernel': 'rbf'}
```

kPCA - Kernel PCA

```
rbf_pca = KernelPCA(n_components = 2,
      kernel="rbf", gamma=0.0433,
      fit_inverse_transform=True)
X_reduced = rbf_pca.fit_transform(X)
X_preimage = rbf_pca.inverse_transform(X_reduced)

>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(X, X_preimage)
32.786308795766132
```

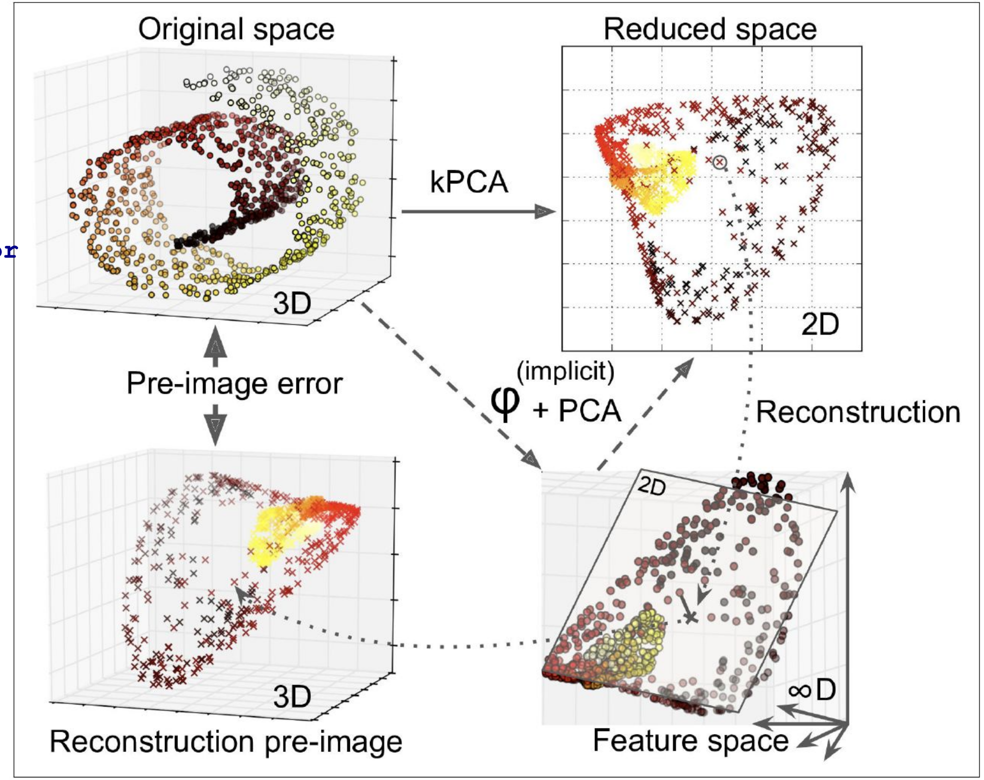


Figure 8-11. Kernel PCA and the reconstruction pre-image error

LLE - Locally Linear Embedding

NDLR Nonlinear dimensionality reduction technique

```
from sklearn.manifold import LocallyLinearEmbedd
lle = LocallyLinearEmbedding(n_components=2,
n_neighbors=10)
X_reduced = lle.fit_transform(X)
```

Equation 8-4. LLE step 1: linearly modeling local relationships

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{m} \left(\mathbf{x}^{(i)} - \sum_{j=1}^{m} w_{i,j}\mathbf{x}^{(j)}\right)^2$$

$$\text{subject to } \begin{cases} w_{i,j} = 0 & \text{if } \mathbf{x}^{(j)} \text{ is not one of the } k \text{ c.n. of } \mathbf{x}^{(i)} \\ \sum_{j=1}^{m} w_{i,j} = 1 & \text{for } i = 1, 2, \cdots, m \end{cases}$$

Equation 8-5. LLE step 2: reducing dimensionality while preserving relationships

$$\widehat{\mathbf{Z}} = \underset{\mathbf{Z}}{\operatorname{argmin}} \sum_{i=1}^{m} \left(\mathbf{z}^{(i)} - \sum_{j=1}^{m} \widehat{w}_{i,j}\mathbf{z}^{(j)}\right)^2$$

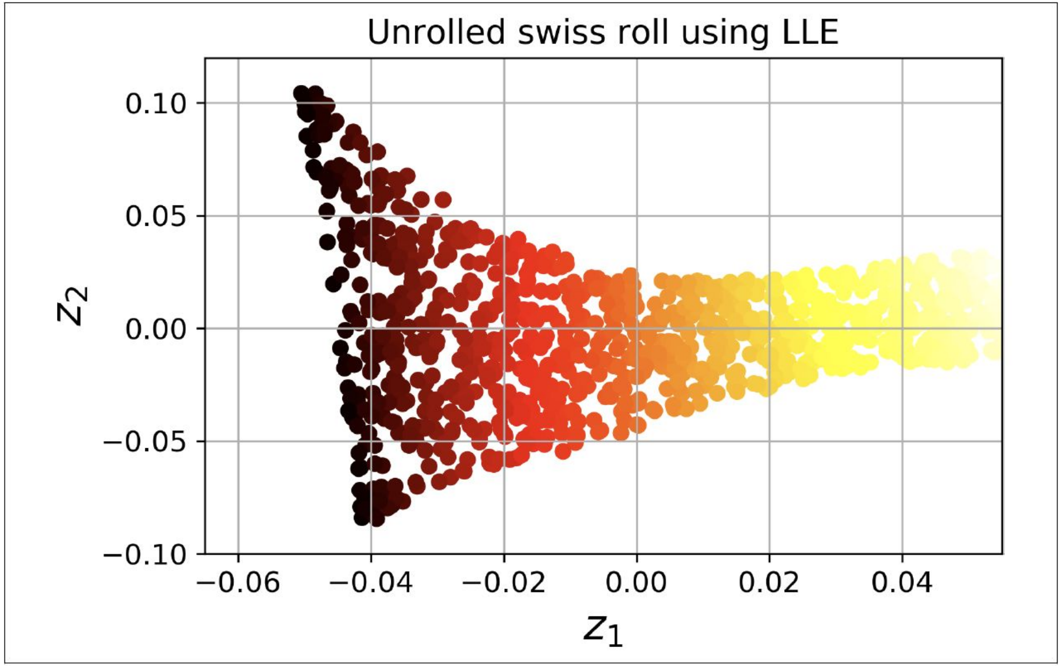


Figure 8-12. Unrolled Swiss roll using LLE

Other Techniques

- MDS Multidimensional Scaling
- Isomap
- t-SNE t-Distributed Stochastic Neighbor Embedding
- LDA Linear Discriminant Analysis

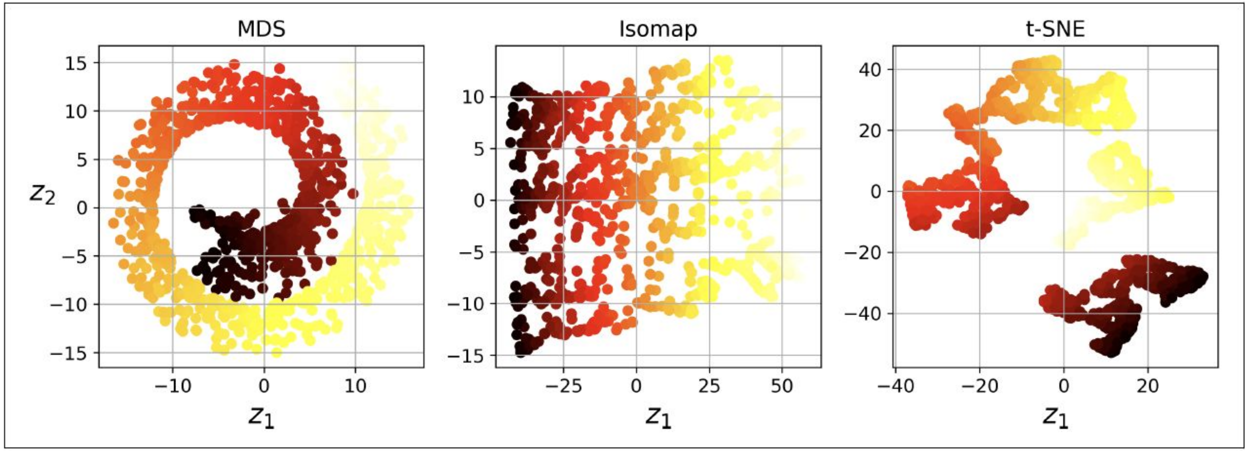


Figure 8-13. Reducing the Swiss roll to 2D using various techniques