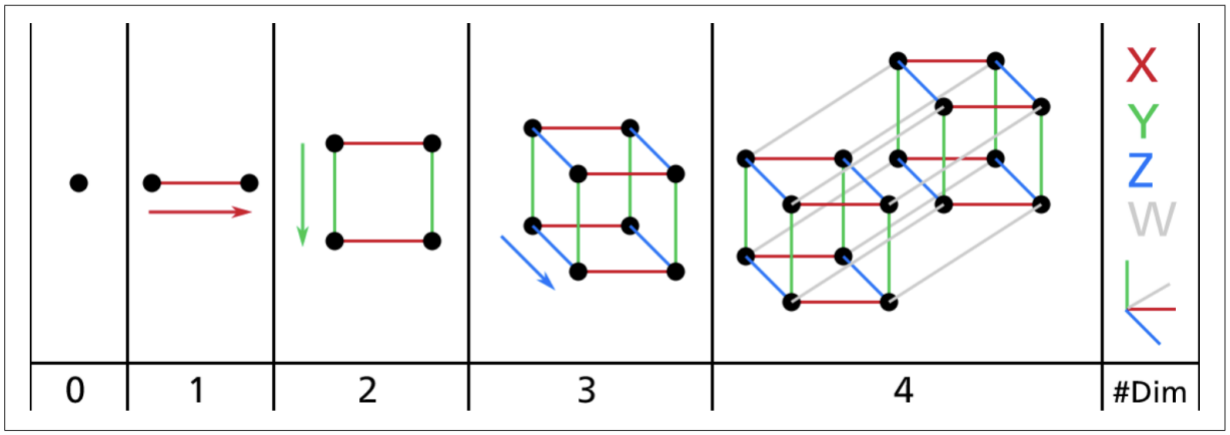
8. Dimensionality Reduction

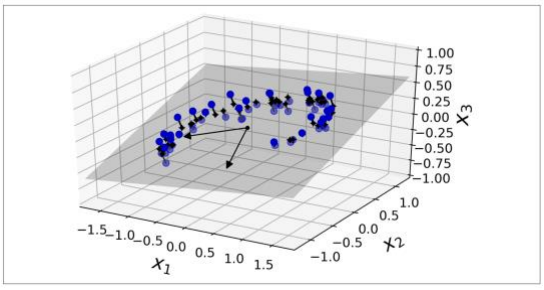
Curse of dimensionality

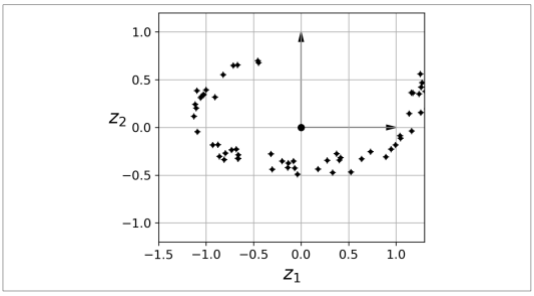
Too many features cause training and finding a good solution hard. There's a tradeoff between removing features to have a faster training and some information loss. Dimensionality Reduction can be helpful for Data Visualization.



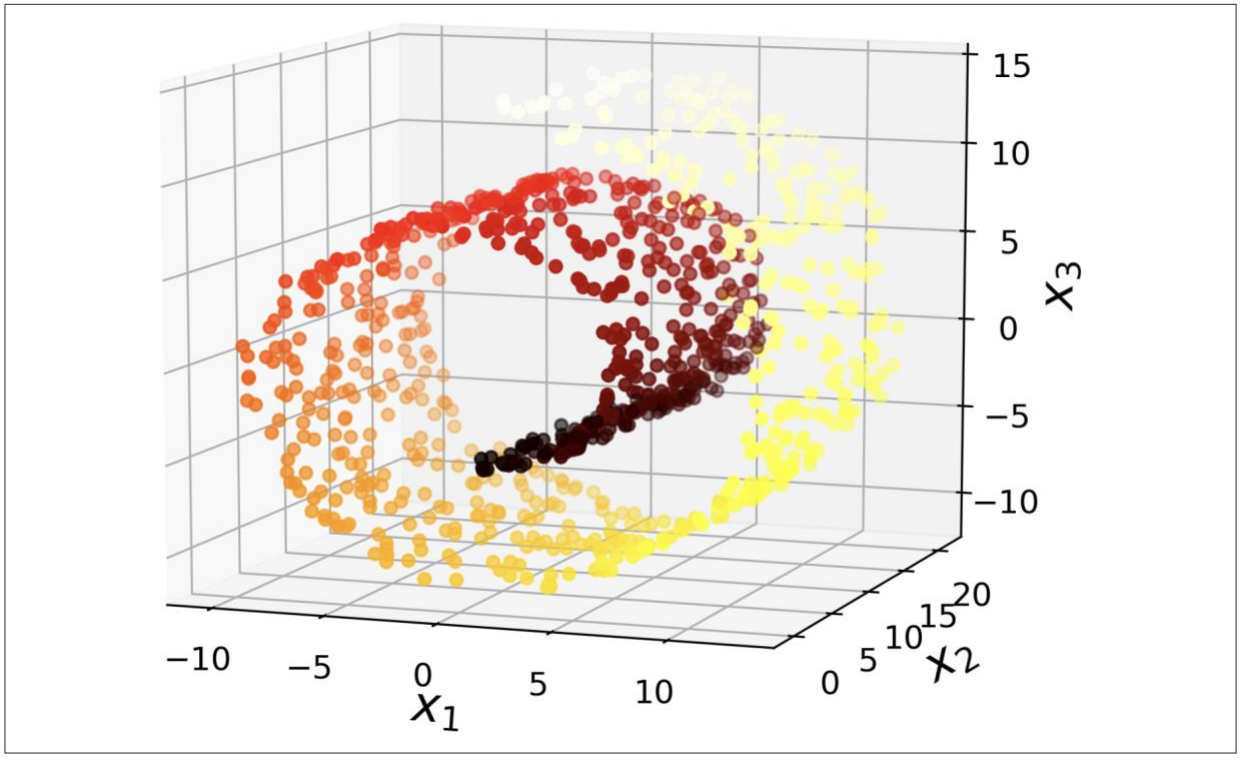
The more edges the easier it is to get close to an edge, and the more dimensions, the more space that is there between points on a hypercube. Then the problem arises when there are lots of features and the dataset hasn't got as many instances and sparsity in the dataset.

Main Approaches for Dimensionality Reduction

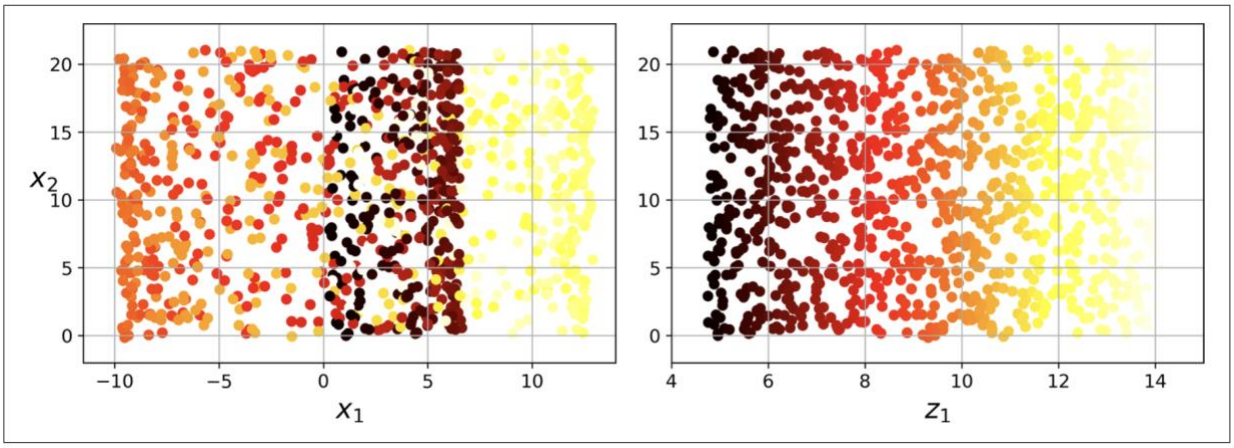




Finding the subspace that can represent the points perpendicularly, reducing the dimensionality creating 2 instead of 3 features.



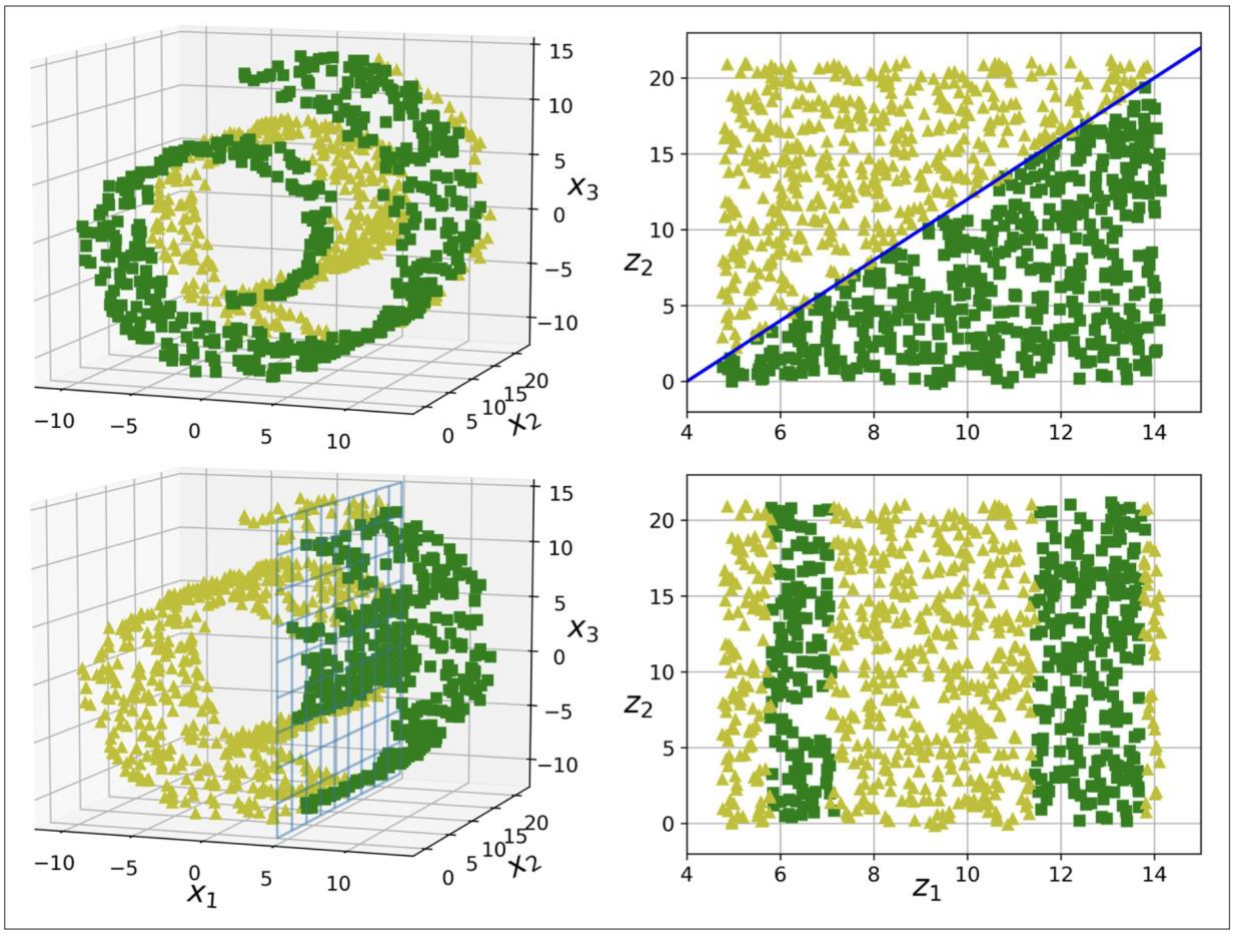
In this case projecting this into a plane would yield different results. The first is form under and the right one is the Swiss roll unpacked. This is called Manifold.



Manifold Learning

The Swiss roll is an example of a 2D manifold. Put simply, a 2D manifold is a 2D shape that can be bent and twisted in a higher-dimensional space.

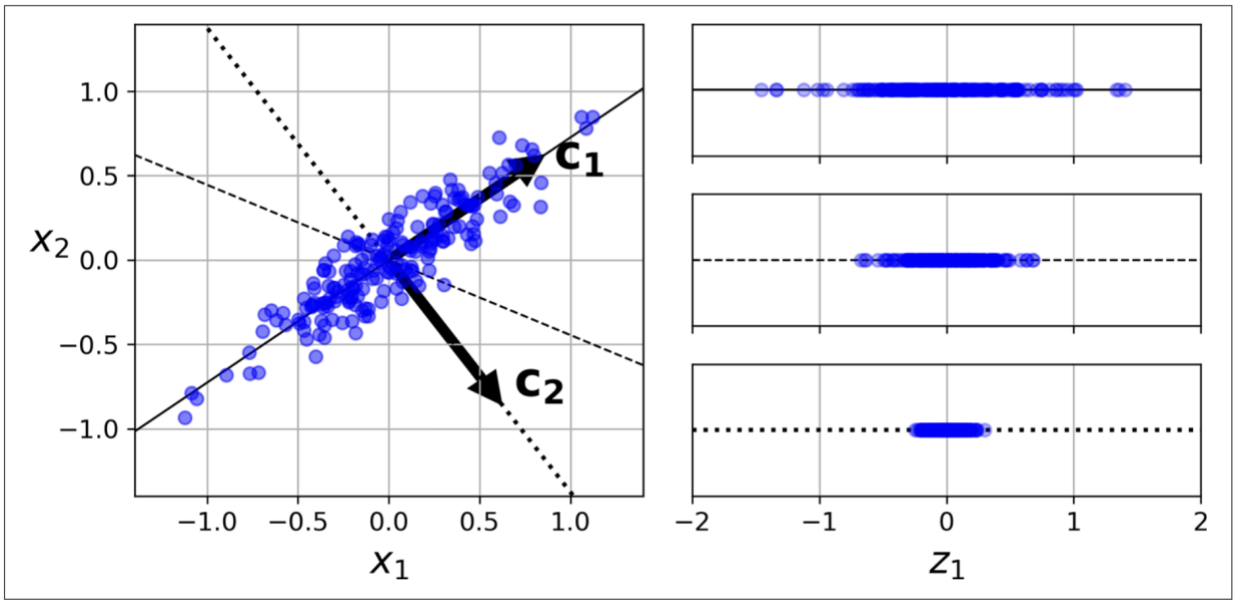
Once again, think about the MNIST dataset: all handwritten digit images have some similarities. They are made of connected lines, the borders are white, and they are more or less centered. If you randomly generated images, only a ridiculously tiny fraction of them would look like handwritten digits. In other words, the degrees of freedom available to you if you try to create a digit image are dramatically lower than the degrees of freedom you would have if you were allowed to generate any image you wanted. These constraints tend to squeeze the dataset into a lower-dimensional manifold.



PCA

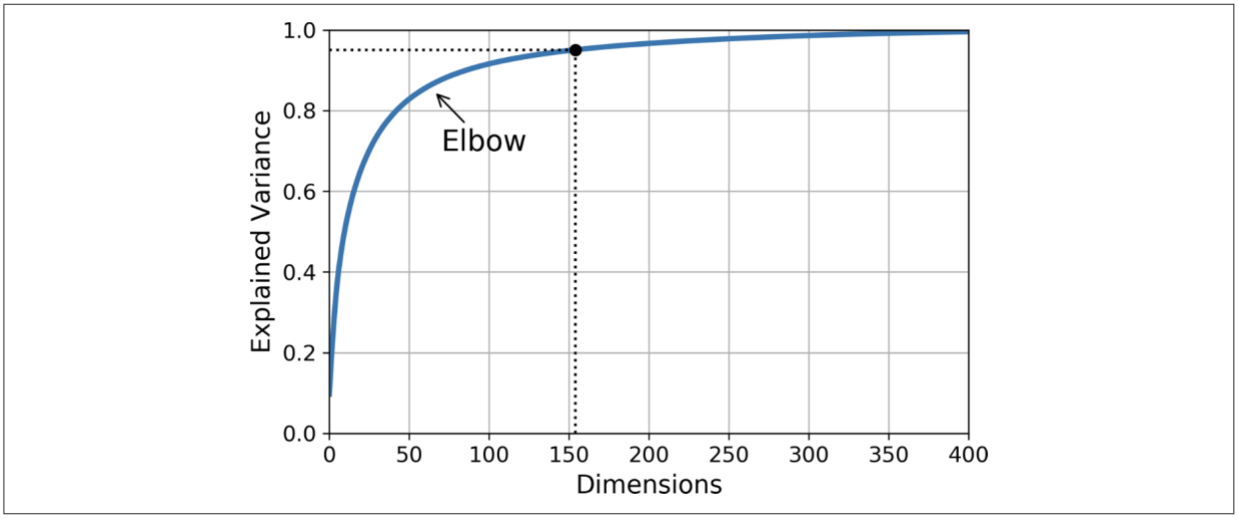
Principal Component Analysis (PCA) is by far the most popular dimensionality reduction algorithm. First it identifies the hyperplane that lies closest to the data, and then it projects the data onto it .

Before you can project the training set onto a lower-dimensional hyperplane, you first need to choose the right hyperplane. PCA assumes that the data is centered.

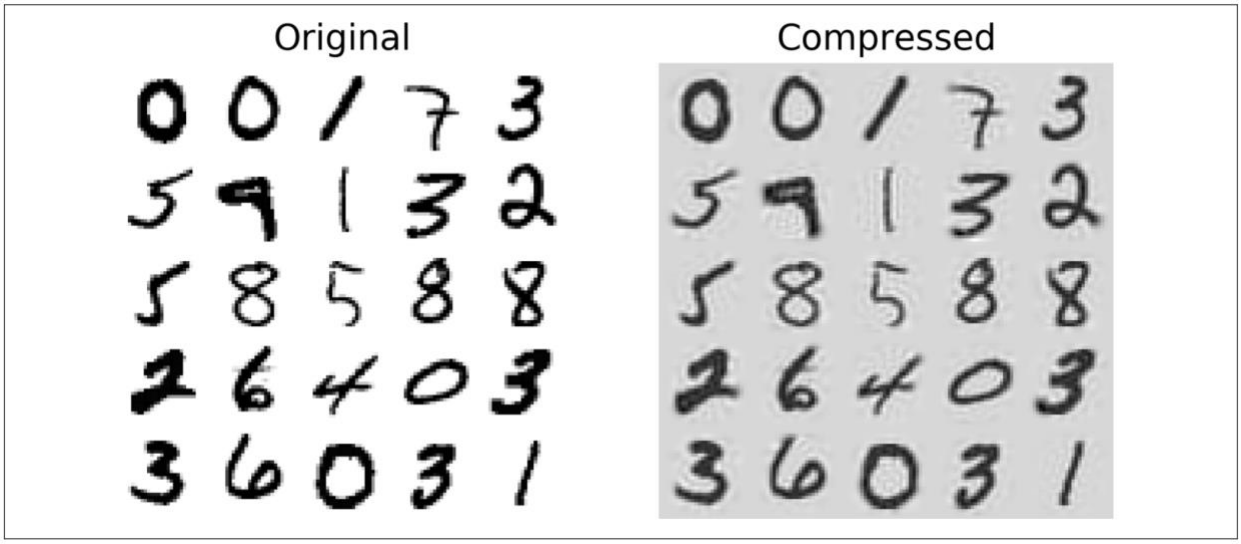


When doing PCA you can check the variance that lies within each feature of PCA.

For deciding how many dimensions to pick in the variance/dimensions tradeoff, we can plot the function of the number of dimensions and see where is the elbow.



Also, we can use PCA to compress datasets or images, saving space and training time:

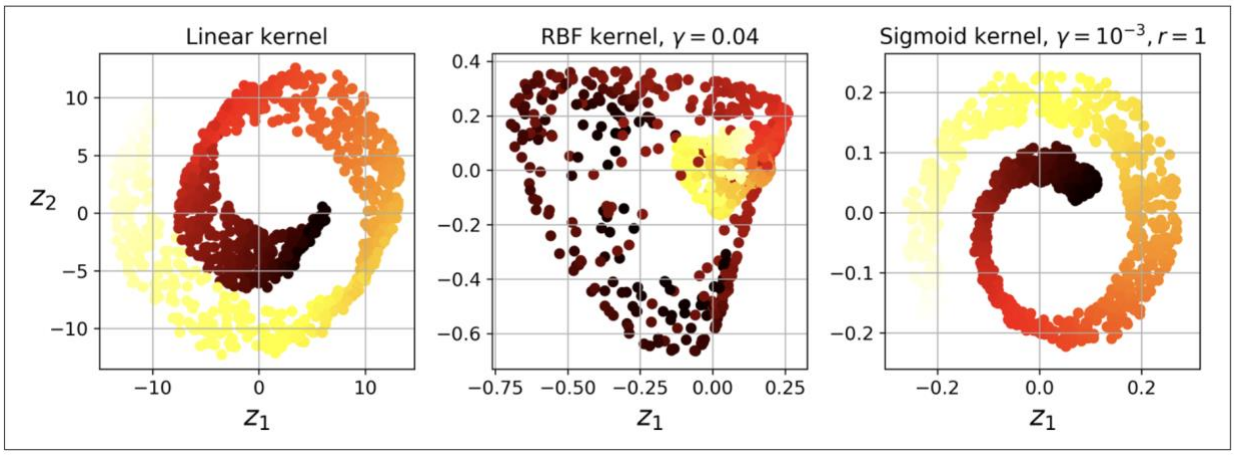


If we don't want to do PCA on batch (with all the data at once), then we can use Incremental PCA. They allow you to split the training set into mini-batches and feed an IPCA algorithm one mini-batch at a time. This is useful for large training sets and for applying PCA online (i.e., on the fly, as new instances arrive).

Kernel PCA

Application to PCA:

1. Nonlinear Dimensionality Reduction: PCA typically finds the principal components in a linear manner. However, by applying the kernel trick (similar to SVMs), PCA can be extended to perform nonlinear dimensionality reduction.
2. Kernel PCA: This technique uses a kernel function to implicitly map the input data into a high-dimensional feature space. It then performs linear PCA in this space. The result is that Kernel PCA can uncover complex, nonlinear structures in the data, which would not be possible with standard PCA.
3. Complex Nonlinear Projections: The kernel trick in PCA allows for the discovery of nonlinear relationships in the data, projecting it onto a lower-dimensional space while preserving important nonlinear relationships that exist in the higher-dimensional feature space.



In short:

1. **Vanilla PCA (Principal Component Analysis):**
    - Use Case: Ideal for datasets that are not too large and can fit comfortably in memory.
    - Features: It's the standard form of PCA and is great for linear dimensionality reduction.
    - Limitation: Not suitable for very large datasets as it requires the entire dataset to be in memory.
2. **Incremental PCA:**
    - Use Case: Best for large datasets that cannot fit in memory at once.
    - Features: Works by processing the data in mini-batches, making it more memory-efficient.
    - Limitation: Can be slower than vanilla PCA and might require tuning the batch size.

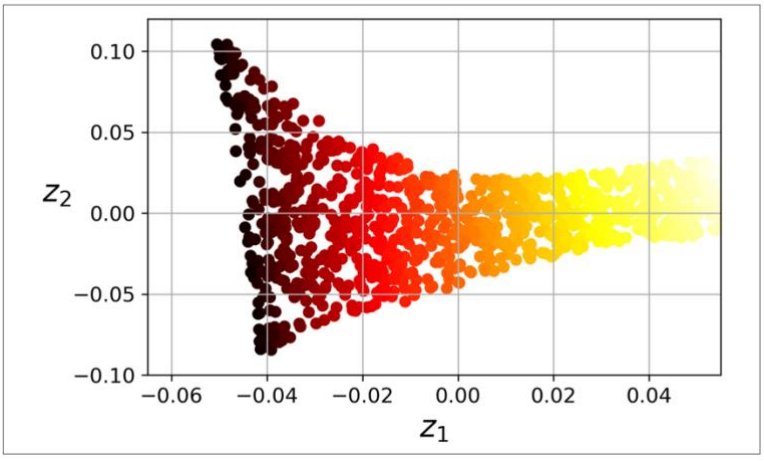
3. **Randomized PCA:**
    - Use Case: Useful when you have a large number of features, and you need to reduce dimensions quickly.
    - Features: Employs stochastic algorithms for approximating the first few principal components. It's faster than regular PCA on large datasets.
    - Limitation: The trade-off is that it provides an approximation, not exact results.
4. **Kernel PCA:**
    - Use Case: Ideal for nonlinear dimensionality reduction.
    - Features: Uses kernel methods to project data into a higher-dimensional space where linear separation (or linear PCA) is possible. Great for complex, nonlinear relationships.
    - Limitation: More computationally intensive and can be harder to interpret. Choosing the right kernel can also be challenging.

Selecting a Kernel and Tuning Hyperparameters

kPCA is an unsupervised learning algorithm, there is no obvious performance measure to help you select the best kernel and hyperparameter values. Despite this, DR is used usually before classification, and then using Grid Search to select the best kernel that fits the situation.

LLE

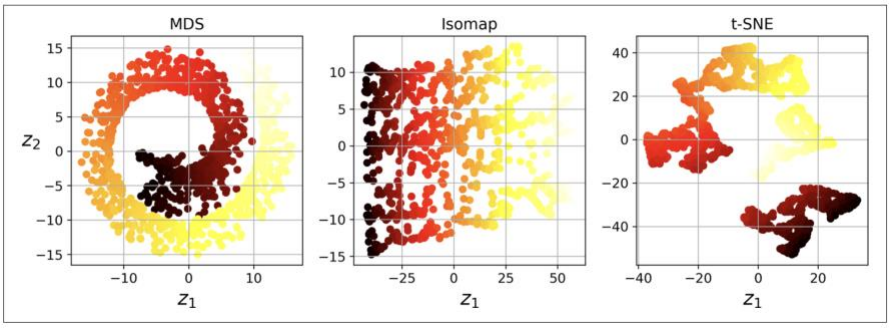
Local Linear Embedding, powerful nonlinear dimensionality reduction technique. This one doesn't rely on projection, it measures how each instance linearly relates to its closest neighbors (c.n.), and then looking for a low-dimensional representation of the training set where these local relationships are best preserved. This approach works nice when there is not much noise.



Other Dimensionality Reduction Techniques

1. **Random Projections:**
    - Randomly projects data into a lower-dimensional space using a random linear projection.
    - Despite its randomness, it tends to preserve distances between data points well. The effectiveness of this method depends on the number of data points and the target lower dimension, but not on the original dimension of the data.

2. **Multidimensional Scaling (MDS):**
    - Aims to reduce dimensions while maintaining the distances between data points. Its goal is to keep the original structure of the data in terms of distances, even in a space with fewer dimensions.



3. **Isomap:**
    - Builds a graph by linking each data point to its nearest neighbors, then reduces dimensions while trying to preserve geodesic distances (the shortest paths in the graph) between points. Particularly good at unfolding twisted and curved manifolds.

4. **t-Distributed Stochastic Neighbor Embedding (t-SNE):**
    - Focuses on reducing dimensions while keeping similar data points close and dissimilar points apart. Often used for visualizing data in 2D or 3D, especially useful in identifying clusters in high-dimensional data (like visualizing groups in image datasets).

5. **Linear Discriminant Analysis (LDA):**
    - Primarily a classification technique, but it learns axes that best separate different classes during training. These axes can then be used for dimensionality reduction. The projection maximizes class separability, making it an excellent preprocessing step for other classification algorithms, such as SVMs.