Dimensionality Reduction

Many machine learning problems involve thousands or even millions of features for each training instance. Not only do all these features make the training extremely slow , but they can also make it much harder to find a good solution . This problem is often referred to as curse of Dimensionality.

Not always it a good idea to reduce the dimension, but it makes the training faster.
Try on original data first and only then reduce Dimensionality.
It is also useful for visualization, reducing the dimensions can help plot a condensed view of the high dimensional training set.

Two main approaches to Dimensional reduction is Manifold learning and projection.

Curse of dimensionality solution is to increase the size of the training set to reach a sufficient density, but unfortunately the number of training instance required to reach a given density grows exponentially with the number of dimensions.

Anyone you know is probably and extremist in at least one dimension , example is how much sugar they put in their coffee if you consider enough dimensions

Projection

In most real-world problems, training instances are not spread out uniformly across all dimensions. Many features are almost constant, while others are highly correlated. As a result, all training instances lie within (or close to) a much lower-dimensional subspace of the high-dimensional space. This sounds very abstract.

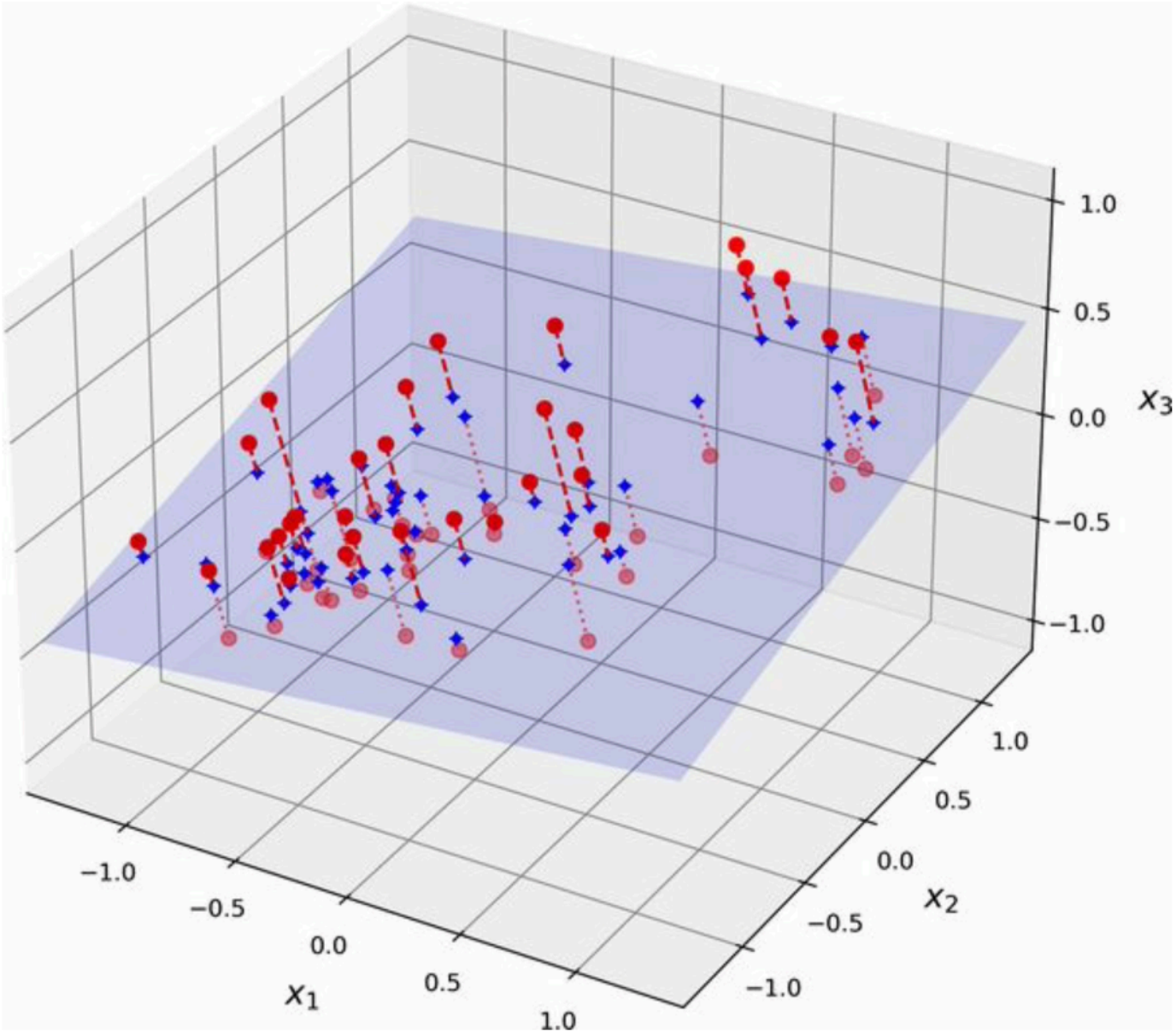

Figure 8-2. A 3D dataset lying close to a 2D subspace

Notice that all training instances lie close to a plane: this is a lower-dimensional (2D) subspace of the higher-dimensional (3D) space. If we project every training instance perpendicularly onto this subspace (as represented by the short dashed lines connecting the instances to the plane),
we get the new 2D dataset.

plane.

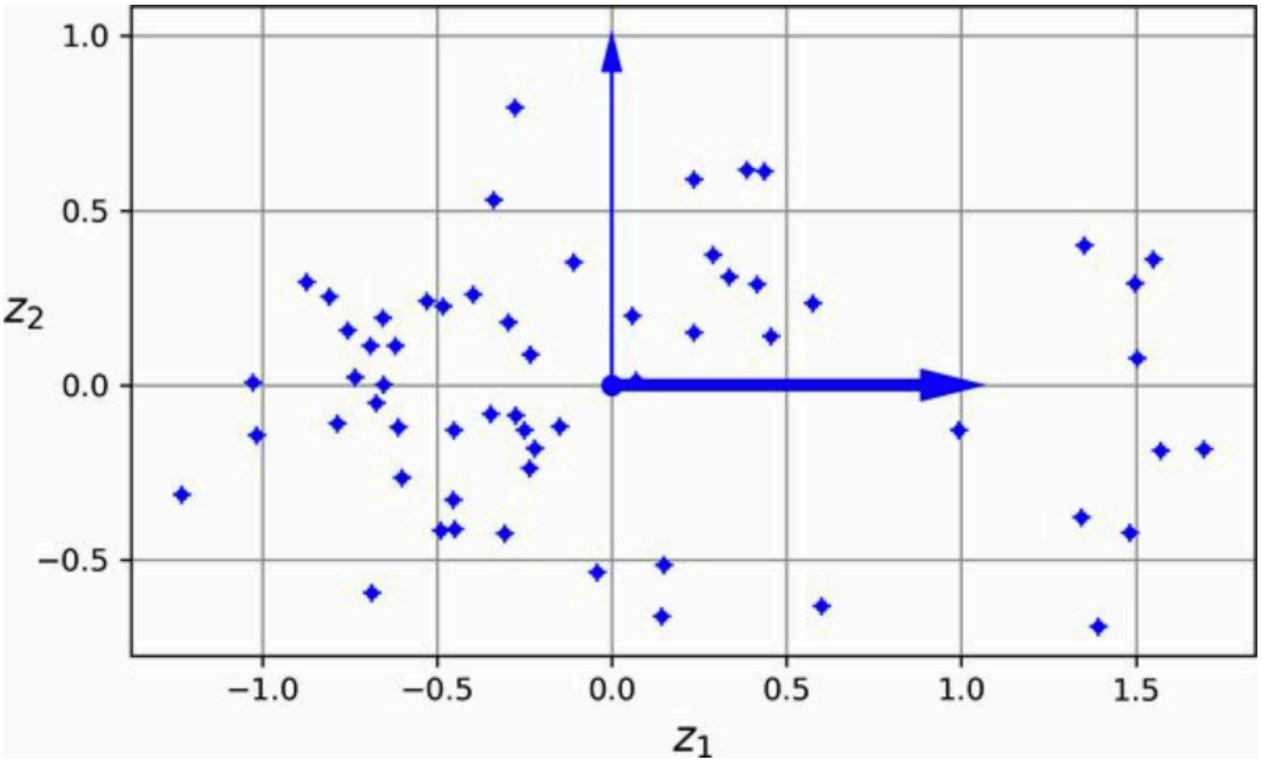

Figure 8-3. The new 2D dataset after projection

Manifold learning

projection is not always the best approach to dimensionality reduction. In many cases the subspace may twist and turn, such as in the famous Swiss roll toy dataset.

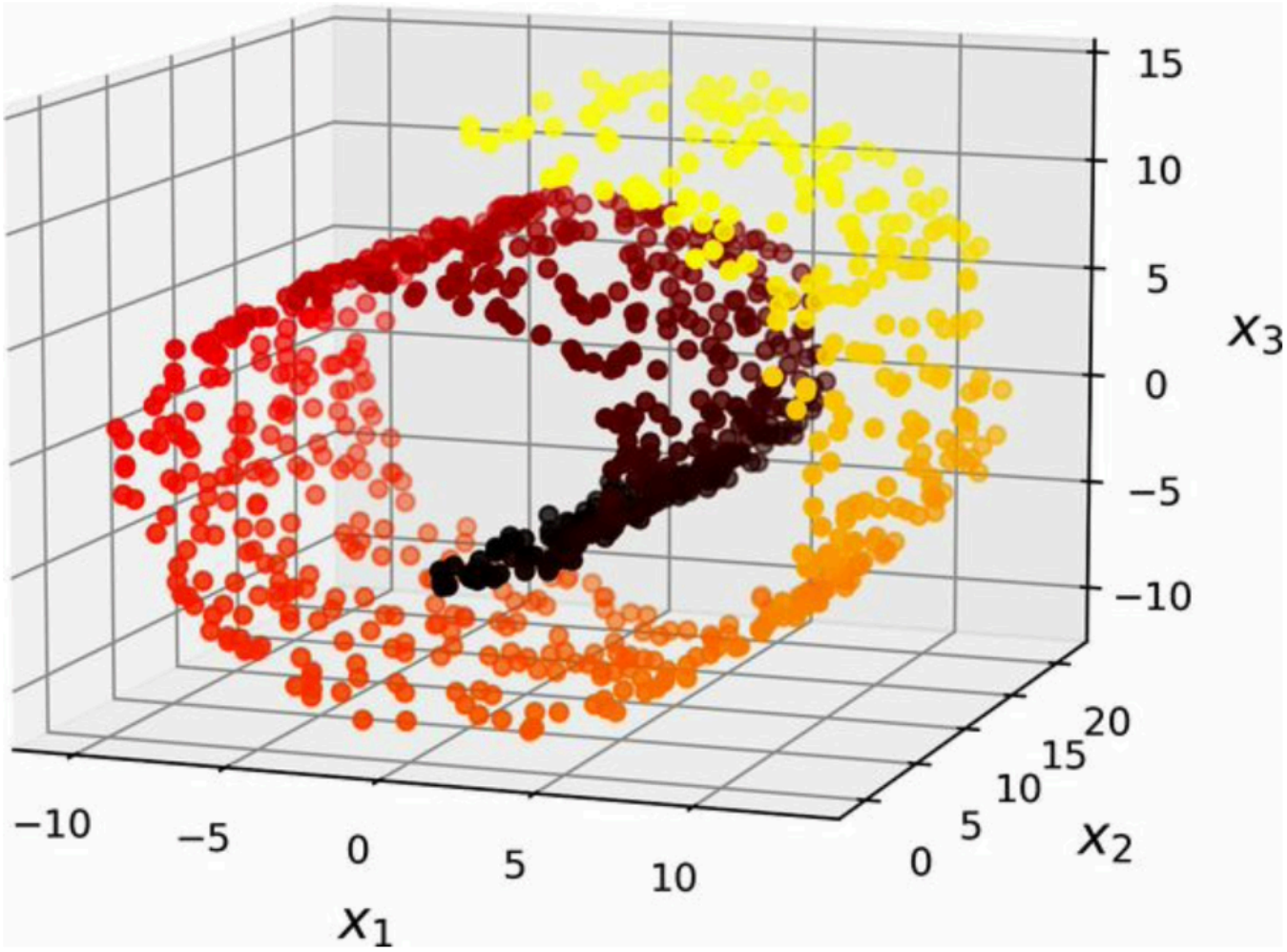

Figure 8-4. Swiss roll dataset

Projection this on the plane would squash layers of the Swiss roll together. Task is to unroll the Swiss roll to obtain the 2D dataset.

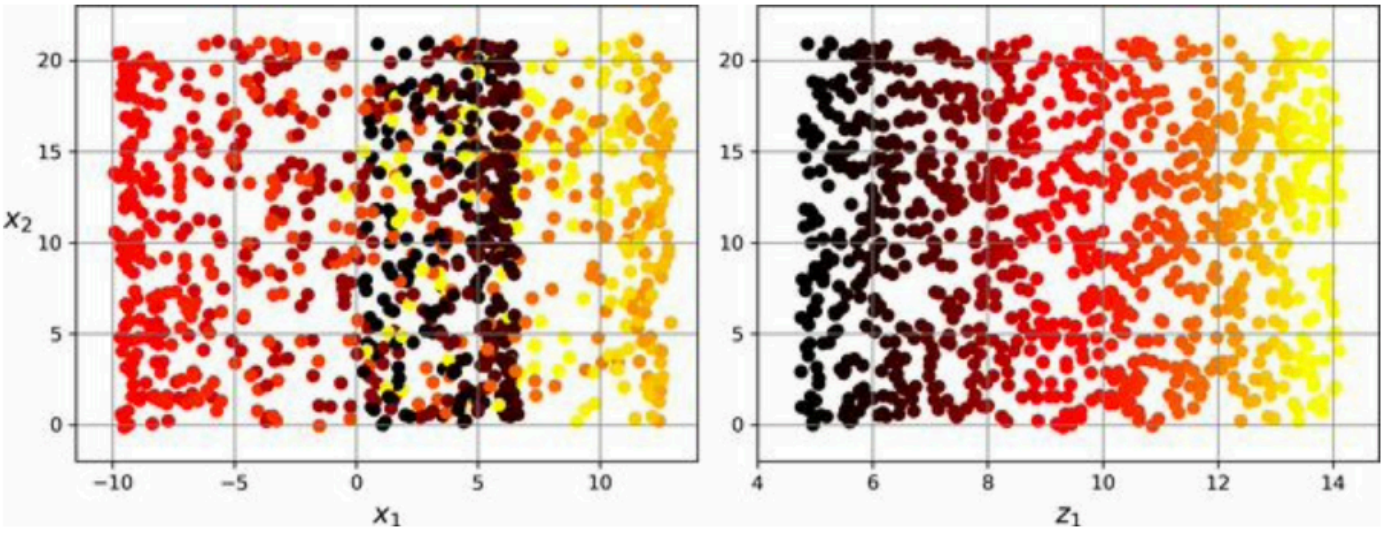

Figure 8-5. Squashing by projecting onto a plane (left) versus unrolling the Swiss roll (right)

A 2D manifold is a shape that can be bent and twisted in a higher dimensional space.
generally, a d-dimensional manifold is a part of an n-dimensional space (where d < n) that locally resembles a d-dimensional hyperplane. In the case of the Swiss roll,
d = 2 and n = 3: it locally resembles a 2D plane, but it is rolled in the third dimension.

Many dimensionality reduction algorithms work by modeling the manifold on which the training instances lie; this is called manifold learning. It relies on the manifold assumption, also called the manifold hypothesis, which holds that most real-world high-dimensional datasets lie close to a much lower-dimensional manifold.

It will speed up the training process but not always lead to a better solution.

PCA

Principal compound analysis is the most popular technique, 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.

On the right is the result of the projection of the dataset onto each of these axes. As you can see, the projection onto the solid line preserves the maximum variance (top), while the projection onto the dotted line preserves very little variance (bottom) and the projection onto the
dashed line preserves an intermediate amount of variance (middle).

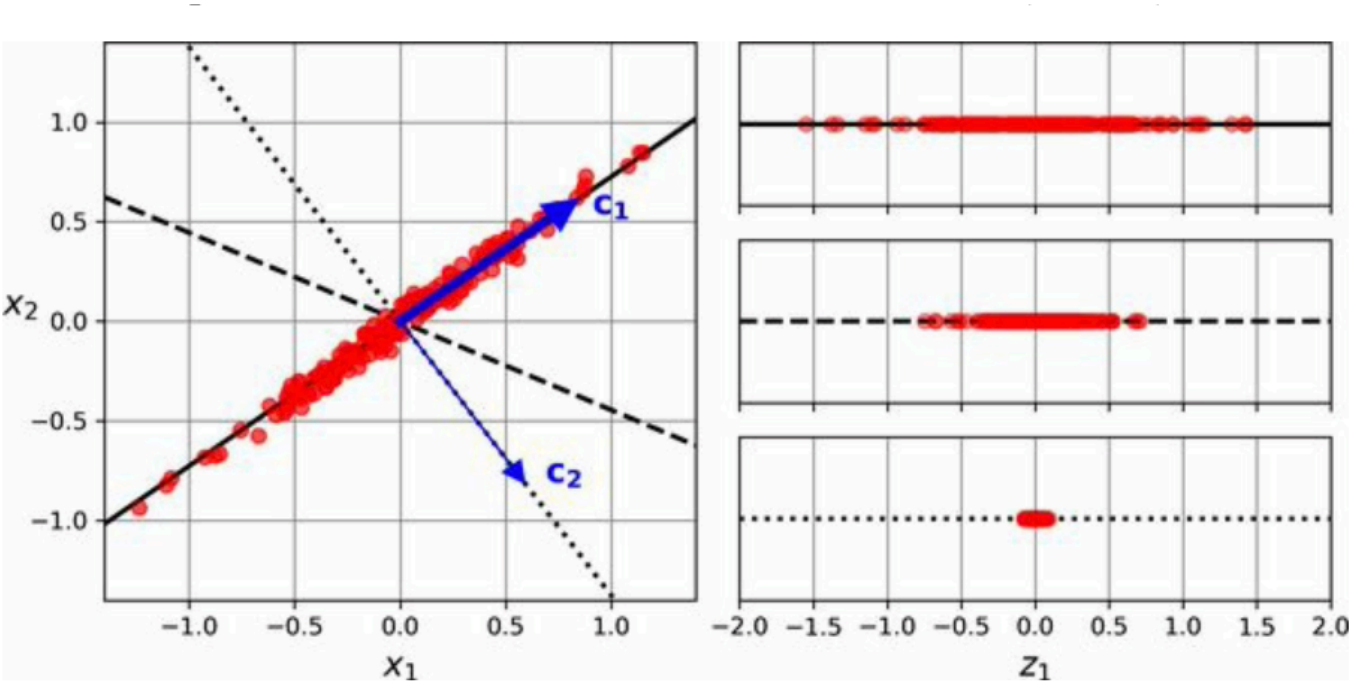

Figure 8-7. Selecting the subspace on which to project

It seems reasonable to select the axis that preserves the maximum amount of variance, as it will most likely lose less information than the other projections. Another way to justify this choice is that it is the axis that minimizes the mean squared distance between the original dataset and its
projection onto that axis.

PCA identifies the axis that accounts for the largest amount of variance in the training set. , it is the solid line. It also finds a second axis, orthogonal to the first one, that accounts for the largest amount of the remaining variance. In this 2D example there is no choice: it is the dotted line. If it were a higher-dimensional dataset, PCA would also find a third
axis, orthogonal to both previous axes, and a fourth, a fifth, and so on—as many axes as the number of dimensions in the dataset.

The ith axis is called the ith PC. PCA assumes the data is centered.
To find the PC of a training set a start matrix factorization is used called singular value decomposition.

Once you have identified all the principal components, you can reduce the dimensionality of the dataset down to d dimensions by projecting it onto the hyperplane defined by the first d principal components

Explained Variance Ratio

the explained variance ratio of each principal component, available via the explained variance ratio variable.

The ratio indicates the proportion of the dataset's variance that lies along each principal component.

Choosing the right number of dimensions - Instead of arbitrarily choosing the number of dimensions to reduce down to, it is simpler to choose the number of dimensions that add up to a sufficiently large portion of the variance—say, 95%.

Yet another option is to plot the explained variance as a function of the number of dimensions There will usually be an elbow in the curve,

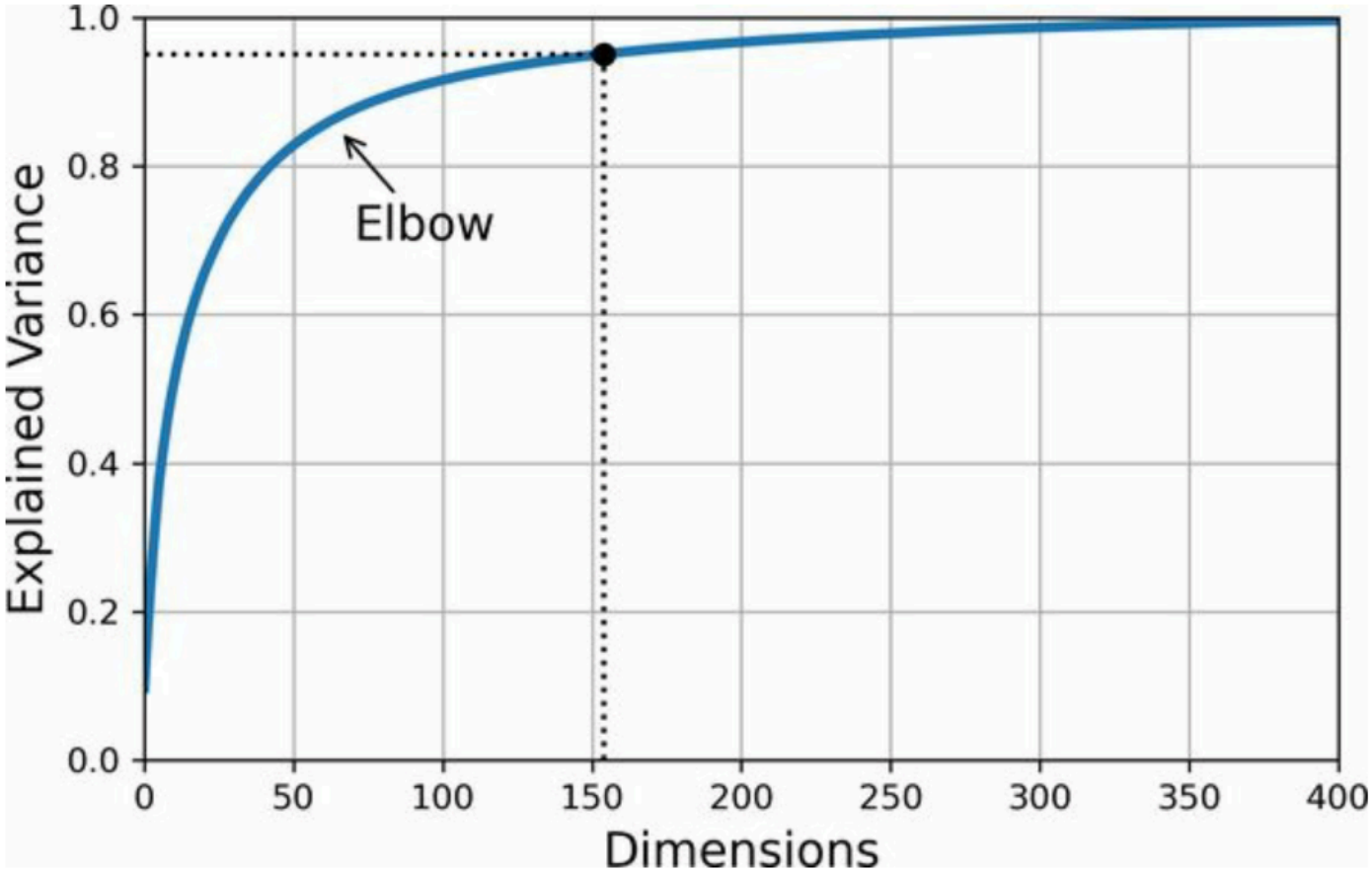

Figure 8-8. Explained variance as a function of the number of dimensions

PCA for Compression

It is also possible to decompress the reduced dataset back to 784 dimensions by applying the inverse transformation of the PCA projection. This won't give you back the original data, since the projection

lost a bit of information , but it will likely be close to the original data. The mean squared distance between the original data and the reconstructed data (compressed and then decompressed) is called the reconstruction error.

Randomized PCA

If you set the svd_solver hyperparameter to "randomized" , Scikit-Learn uses a stochastic algorithm called randomized PCA that quickly finds an approximation of the first d principal components. Its dramatically faster than full SVD when d is much smaller than n.

Incremental PCA

One problem with the preceding implementations of PCA is that they require
he whole training set to fit in memory in order for the algorithm to run. Fortunately, incremental PCA (IPCA) algorithms have been developed that allow you to split the training set into mini-batches and feed these in one mini-batch at a time. This is useful for large training sets and for applying PCA online.

For high dimension data PCA can be very slow so considering projection sometimes may be a good option.

Random Projection

As its name suggests, the random projection algorithm projects the data to a lower-dimensional space using a random linear projection. This may sound crazy, but it turns out that such a random projection is actually very likely to preserve distances fairly well. It is simply fast and memory efficient and powerful technique.

LLE

Locally linear embedding (LLE) 8 is a nonlinear dimensionality reduction (NLDR) technique. It is a manifold learning technique that does not rely on projections, unlike PCA and random projection. In a nutshell, LLE works by first measuring how each training instance linearly relates to its nearest neighbors, and then looking for a low-dimensional representation of the training set where these local relationships are best preserved.

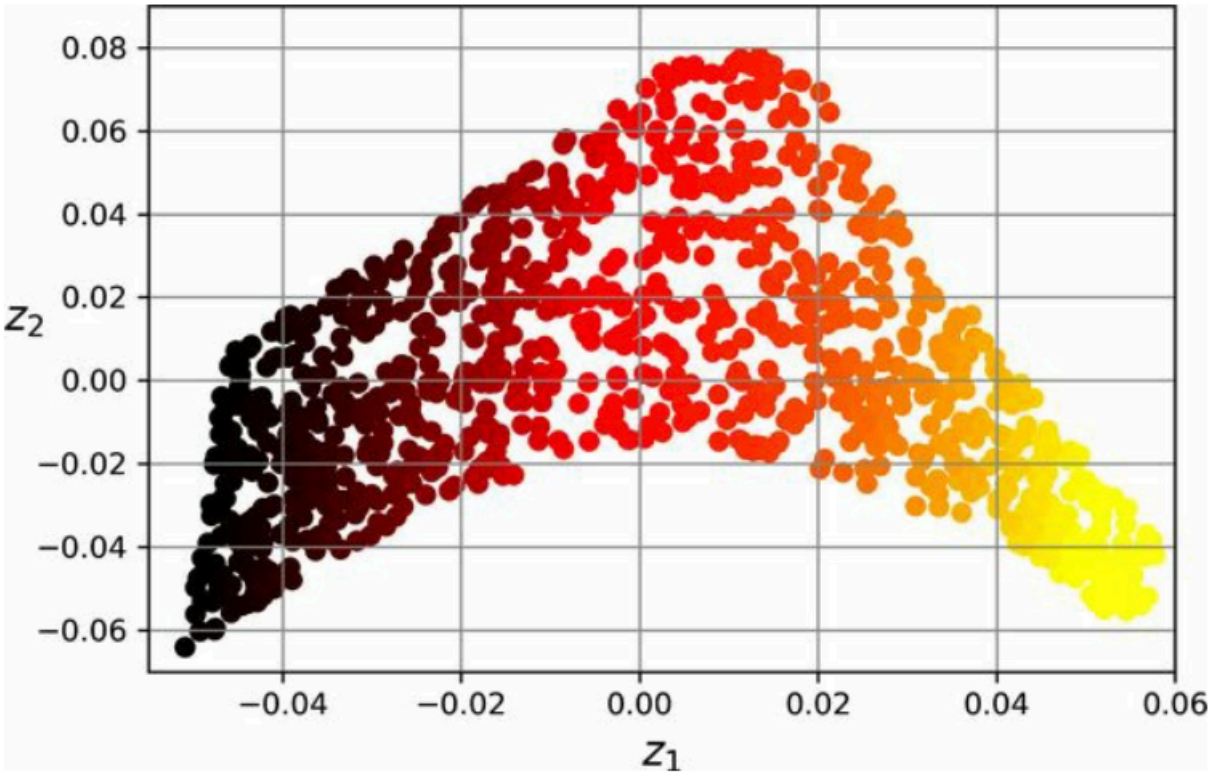

Figure 8-10. Unrolled Swiss roll using LLE

Other Techniques

1. sklearn.manifold.Isomap

Isomap creates a graph by connecting each instance to its nearest neighbors, then reduces dimensionality while trying to preserve the geodesic distances between the instances. The geodesic distance between two nodes in a graph is the number of nodes on the shortest path between these nodes.

2. sklearn.manifold.MDS

Multidimensional scaling (MDS) reduces dimensionality while trying to preserve the distances between the instances. Random projection does that for high-dimensional data, but it doesn't work well on low-dimensional data.