Chapter 8. Dimensionality Reduction

Purpose of dimension reduction:

- *Curse of dimensionality*: large number of features → training slow, difficult to find a good solution.
- Data visualization.

Main approaches:

- Projection
- Manifold Learning

Techniques:

- PCA
- Kernel PCA
- LLE

The Curse of Dimensionality

High-dimensional datasets are at risk of being very sparse. Of course, this also means that a new instance will likely be far away from any training instance, making predictions much less reliable than in lower dimensions, since they will be based on much larger extrapolations. In short, the more dimensions the training set has, the greater the risk of overfitting it.

In theory, one solution could be to increase the size of the training set. Unfortunately, in practice, the number of training instances required to reach a given density grows exponentially with the number of dimensions.

Main Approaches for Dimensionality Reduction

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 actually lie within (or close to) a much lower-dimensional subspace of the high-dimensional space.

Manifold Learning

A d-dimensional manifold is a part of an n-dimensional space (where $d < n$) that locally resembles a d-dimensional hyperplane. (Swiss roll, $d = 2,\ n = 3$)

Manifold Learning: modeling the manifold on which the training instances lie.

Manifold assumption (*manifold hypothesis*): most real-world high-dimensional datasets lie close to a much lower-dimensional manifold. (often empirically observed)

Another implicit assumption: the task will be simpler if expressed in the lower-dimensional space of the manifold.

Reducing the dimensionality of your training set before training a model, will definitely speed up training, but it may not always lead to a better or simpler solution; it all depends on the dataset.

PCA

Principle Component Analysis is the most popular dimensionality reduction algorithm. First, identifies the hyperplane that lies closest to the data; Then, projects the data onto it.

Preserving the Variance

Select the axis that

- preserves the maximum amount of variance.
- minimizes the mean squared distance between the original dataset and its projection onto that axis.

Principle Components

PCA identifies the axis that accounts for the largest amount of variance in the training set, also finds a second axis, orthogonal to the first one, that accounts for the largest amount of remaining variance.

i^{th} *principal component* (PC): unit vector of the i^{th} axis.

Singular Value Decomposition (SVD)

$$\mathbf{X} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^T$$

Principle component matrix ($n \times n$)

$$\mathbf{V}^T = \begin{pmatrix} \vdots & \vdots & & \vdots \\ c_1 & c_2 & \cdots & c_n \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

WARNING

> PCA assumes that the dataset is centered around the origin.

Projecting Down to d Dimensions

Projecting the training set down to d dimensions

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

where $\mathbf{W}_d$ contains the first d principal components

Explained Variance Ratio

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

Choosing the Right Number of Dimensions

Choose the number of dimensions that add up to a sufficiently large portion of the variance (e.g., 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, where the explained variance stops growing fast. You can think of this as the intrinsic dimensionality of the dataset.

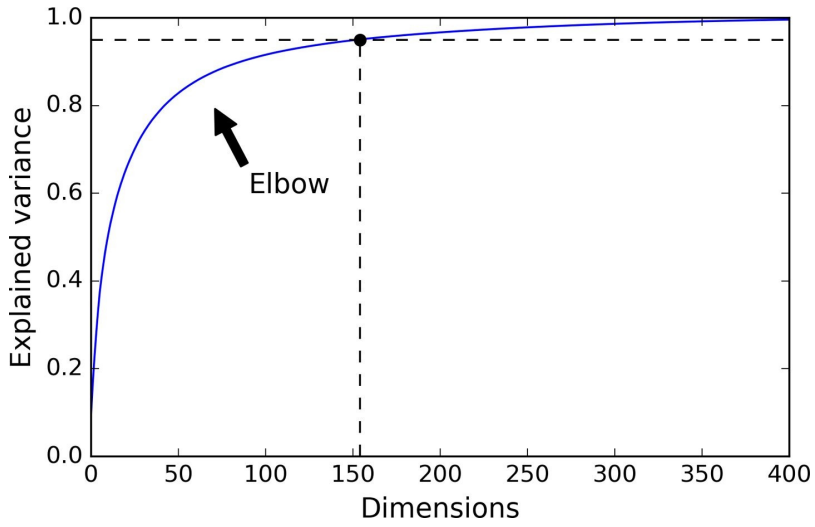

Figure 8-8. Explained variance as a function of the number of dimensions

PCA for Compression

Reconstruction error: the mean squared distance between the original data and the reconstructed data (compressed and then decompressed).

PCA inverse transformation, back to the original number of dimensions

$$\mathbf{X}_{recovered} = \mathbf{X}_{d-proj} \cdot \mathbf{W}_d^T$$

Incremental PCA

Problem: preceding implementation of PCA is that it requires the whole training set to fit in memory in order for the SVD algorithm to run.

Incremental PCA (IPCA): split the training set into mini-batches and feed an IPCA algorithm one mini- batch at a time.

Randomized PCA

A stochastic algorithm that quickly finds an approximation of the first d principal components. Computational complexity is $O(m \times d^2) + O(d^3)$, instead of $O(m \times n^2) + O(n^3)$, so it is dramatically faster than the previous algorithms when d is much smaller than n.

Kernel PCA

Kernel PCA (kPCA): the kernel trick can be applied to PCA, making it possible to perform complex nonlinear projections for dimensionality reduction. It is often good at preserving clusters of instances after projection, or sometimes even unrolling datasets that lie close to a twisted manifold.

Selecting a Kernel and Tuning Hyperparameters

kPCA is an unsupervised learning algorithm. Dimensionality reduction is often a preparation step for a supervised learning task (e.g., classification), so you can simply use grid search to select the kernel and hyperparameters.

Another approach, this time entirely unsupervised, is to select the kernel and hyperparameters that yield the lowest reconstruction error.

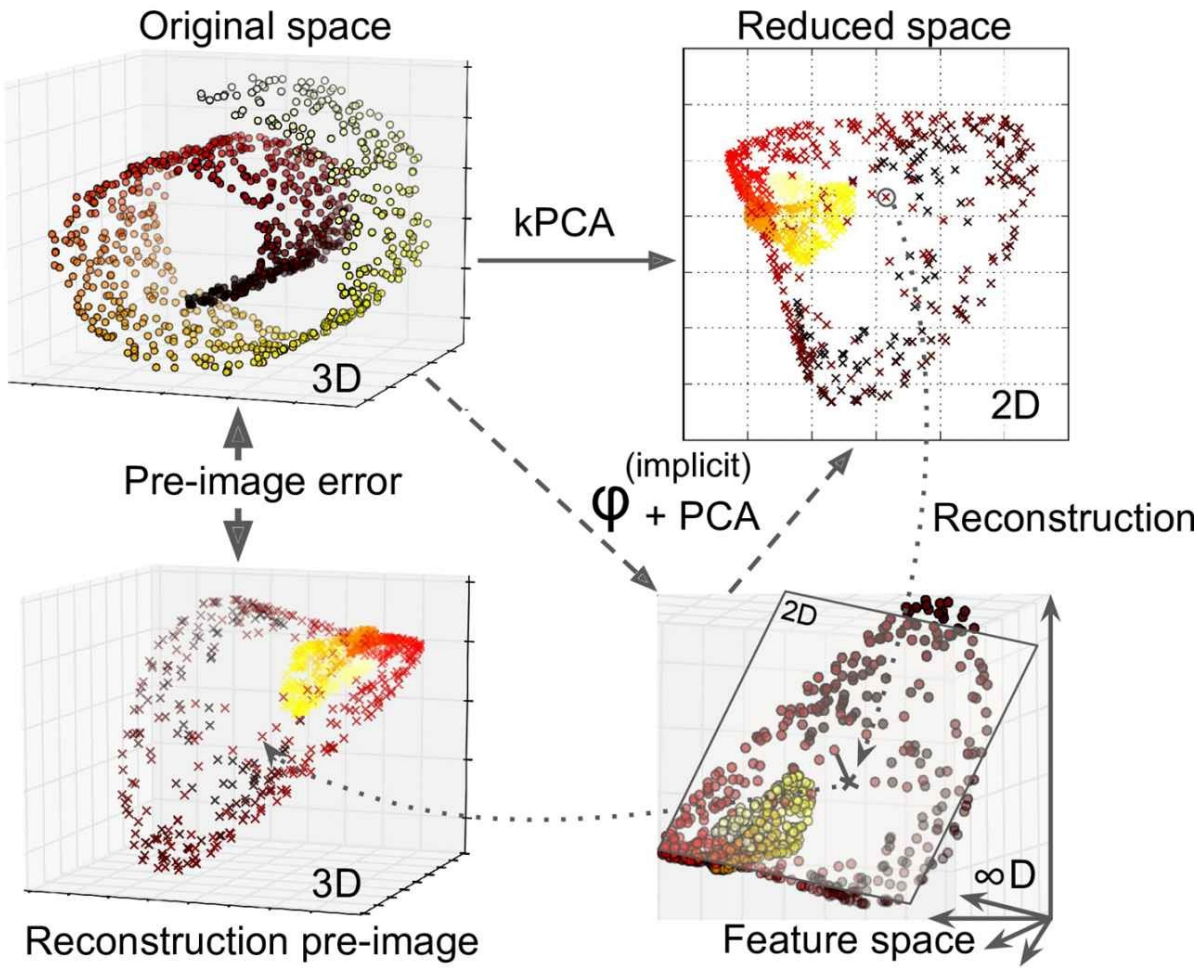

Figure 8-11. Kernel PCA and the reconstruction pre-image error

LLE

Locally Linear Embedding (LLE) is another very powerful *nonlinear dimensionality reduction* (NLDR) technique. A Manifold Learning technique that does not rely on projections.

First measuring how each training 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 makes it particularly good at unrolling twisted manifolds, especially when there is not too much noise.

The algorithm scale poorly to very large datasets.

Other Dimensionality Reduction Techniques

- Multidimensional Scaling (MDS) reduces dimensionality while trying to preserve the distances between the instances.
- 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.
- t-Distributed Stochastic Neighbor Embedding (t-SNE) reduces dimensionality while trying to keep similar instances close and dissimilar instances apart. It is mostly used for visualization, in particular to visualize clusters of instances in high-dimensional space (e.g., to visualize the MNIST images in 2D).
- Linear Discriminant Analysis (LDA) is actually a classification algorithm, but during training it learns the most discriminative axes between the classes, and these axes can then be used to define a hyperplane onto which to project the data. The benefit is that the projection will keep classes as far apart as possible, so LDA is a good technique to reduce dimensionality before running another classification algorithm such as an SVM classifier.