Lecture 7 - Dimensionality Reduction

Machine learning models are well suited to approach problems with thousands to millions of features. This huge amount of features makes the training very slow and gives rise to solutions almost intractables or uninterpretable. This problem is known as the curse of dimensionality.

Simulon problems occurs when one attempts to deal with high geometrical dimensions or a huge amount of information. Our conventional is limited to a 3D world and its projections.

Thus, we cannot clearly imaginate the features of the 4th time dimension of special relativity or the 11 dimensions of string theory. The same with the 6N-dimensional phase space of a classical many particle system. In this case we have the problem with the 6-dimensions of momentum and space, needed to specify the state of a physical system and the N dimensional problem of an N-particle resulting in a good approximation for a dataset. This problem is also called "Curse's Demon".

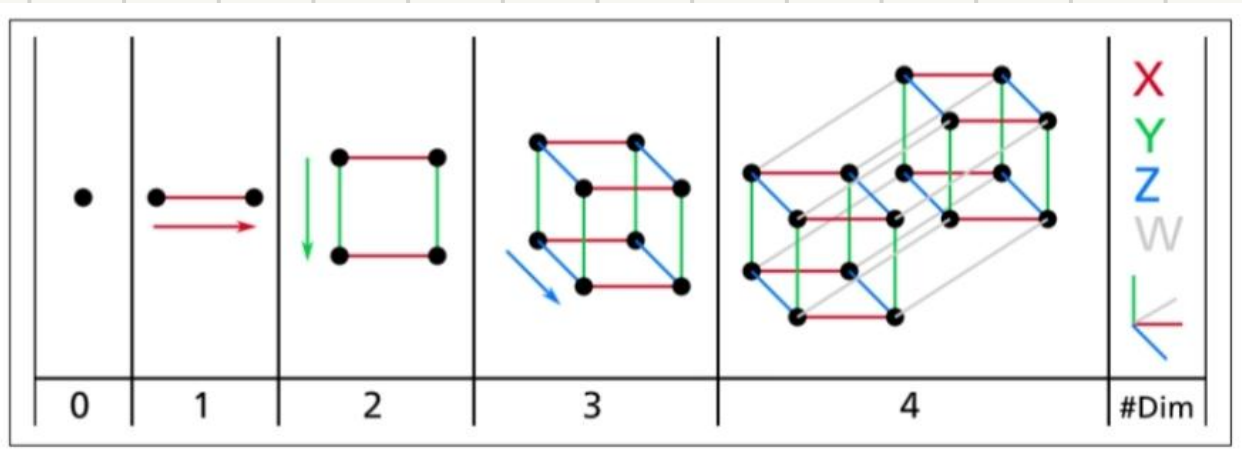


Figure 8-1. Point, segment, square, cube, and tesseract (0D to 4D hypercubes)[2]

Large dimensions mows the problem to be intractable not only due to computational complexity. But because due to our geometrical limitations, we can not take in account easily how basic facts scales with geometry:

- If you pick a random point in a unity square, only 0,4% of the points are at the very edge.

  However, in a 10^4-hyperspace, 99,9% of the points lies at the border.

- If you pick a point randomly in a unit square, the distance between those two points will be on average of 0.52. In a 3D unit cube, about 0.66. In a 10^6D unit cube, 408,25.

The counterintuitiveness of those basic facts rises from our limitations in interpret dimensions superior to 3. In theory, the solution to those problems, having only about ML, would be more datapoints and more computational power. However, the number of training instances to reach a given density grows exponentially with the number of dimensions. With 100 features, you would need more training instances than the number of atoms in the universe in order to each feature be about 0.1 of each other in a uniform universe.

ϕ Approaches to dimensionality reduction

There are two main families of approaches to dimensionality reduction: Projection and Manifold Learning.

A) Projection

In a real dataset, most of the points are not in a uniform spread. There are features that are almost constant and others that are very highly correlated. The result is that most of the samples of a training instance lies near a sub-space of a high dimensional space.

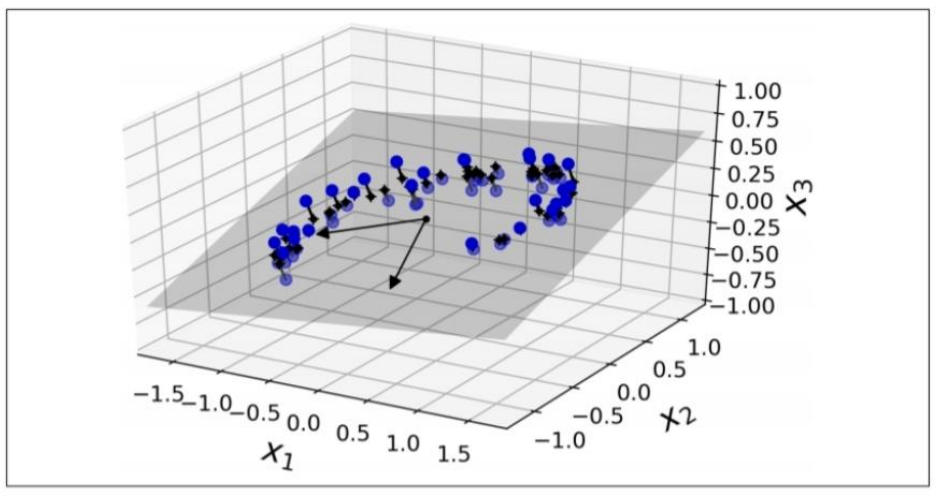


Figure 8-2. A 3D dataset lying close to a 2D subspace

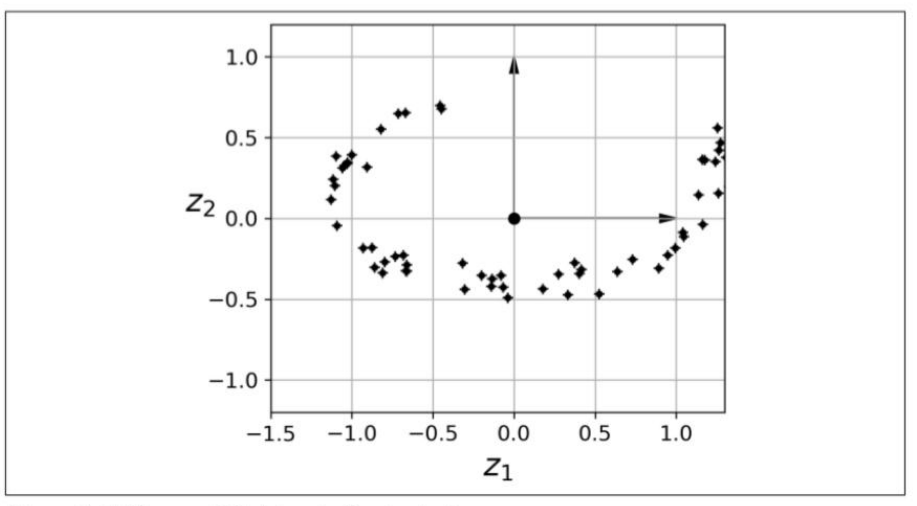
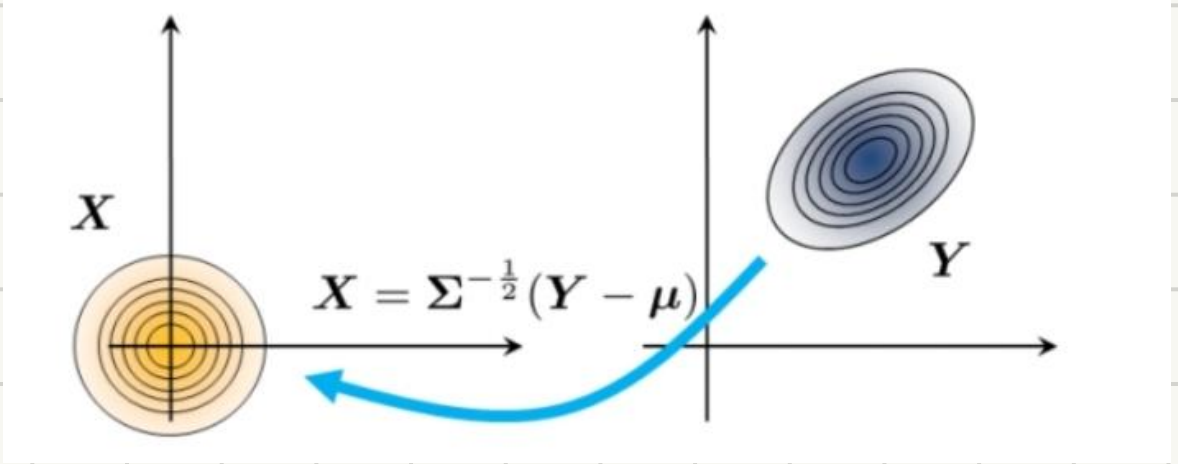
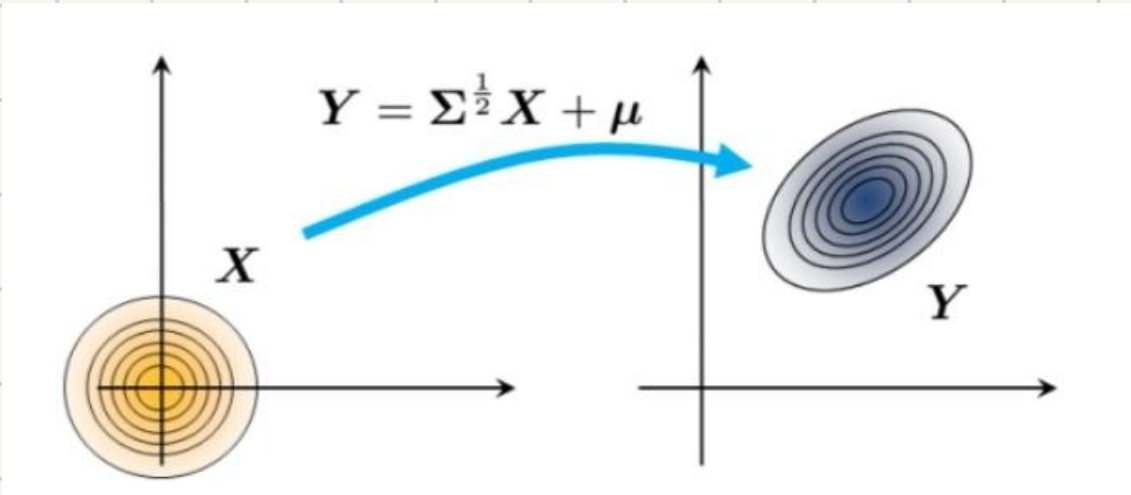


Figure 8-3. The new 2D dataset after projection

In the above picture, we have a 3D dataset, however most of the training instances lies on a hyperplane. One can project all points in this hyperplane and reduce the dimensionality to a 2D problem. Of course this is not the general case. Every dimensionality reduction comes with a loss of information. There are simple cases in which this loss is minimal because the information is easy to reconstruct or because there is a symmetry hidden in the data points.

A good example of information that can be recovered due to symmetry is the transformation property of a gaussian random variable.





Due to all informational content is stored in the properties of the covariance matrix Σ, we can easily reconstruct the the original data from a standard gaussian, using rotations and translations. Note that this is precisely the case of the hyperplane example.

However, a 1800 that twist and roll, like the Swiss roll dataset, may not have any clear symmetry and a projection may not be a good solution:

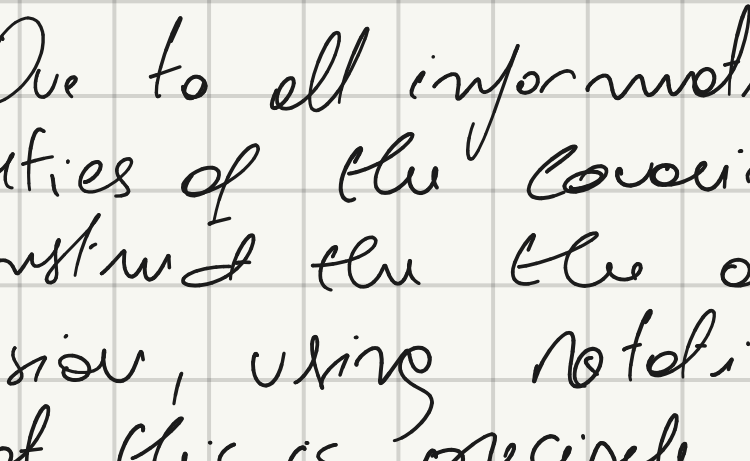


Figure 8-4. Swiss roll dataset

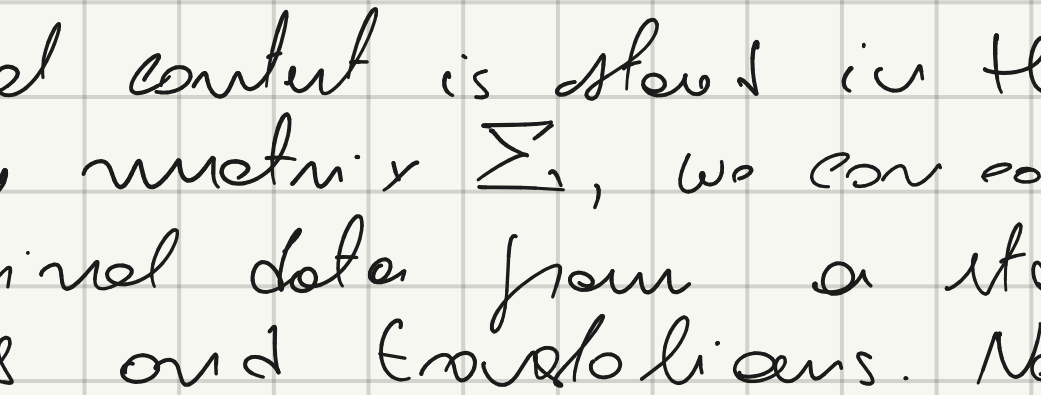


Figure 8-5. Squashing by projecting onto a plane (left) versus unrolling the Swiss roll (right)

Simple projections in a plane will squash different layers of the roll together and the information is lost.

B) Manifold Learning

The Swiss roll is a 2D manifold, i.e a shape that can be bent and twisted in a higher-dimensional space. A d-dimensional manifold is an object that is part of a n-dimensional space, $d < n$, that locally resembles a d-dimensional object.

In this case, a dimensionality reduction algorithm works by modeling the manifold on which the training instances lies. This is called manifold learning. This is a elaboration over the so-called manifold hypothesis, in which every dataset in a high dimensional space lies close to a low-dimensional manifold. This assumption can not be proven, but is empirically observed. Of course, we are also assuming that the task at hand will be simpler in this low-dimensional manifold. This does not always hold.

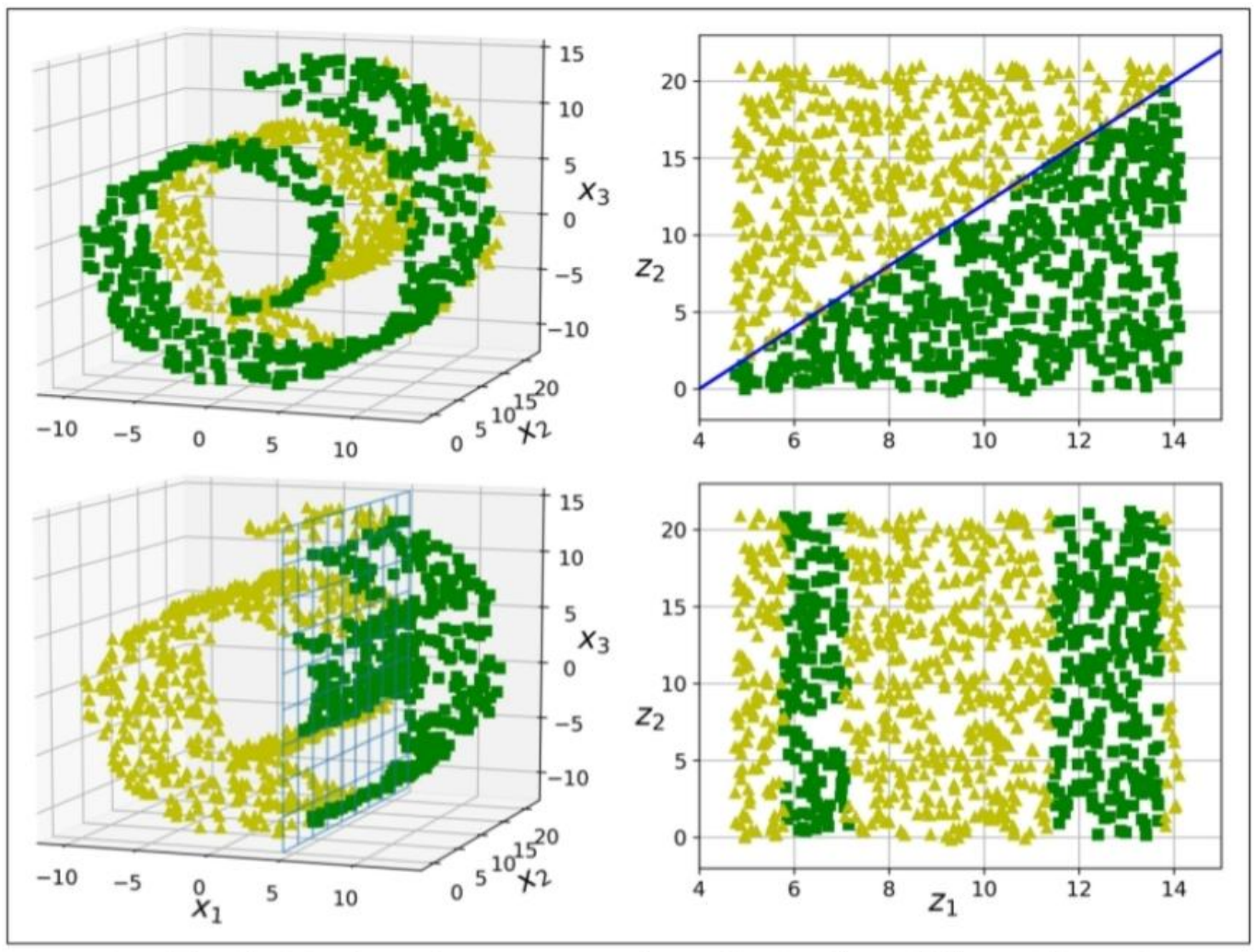


Figure 8-6. The decision boundary may not always be simpler with lower dimensions

Reducing dimensionality of your training set before training the model will usually speed up training but it may not always lead to better predictions.

Principal Component Analysis

In what follows, we are going to deduce one of the most used methods for dimensionality reduction. the principal component analysis. It is a projection method and lies it's roots in the algebric properties of the covariance matrix an it the geometrical properties of random vectors over an Euclidean space.

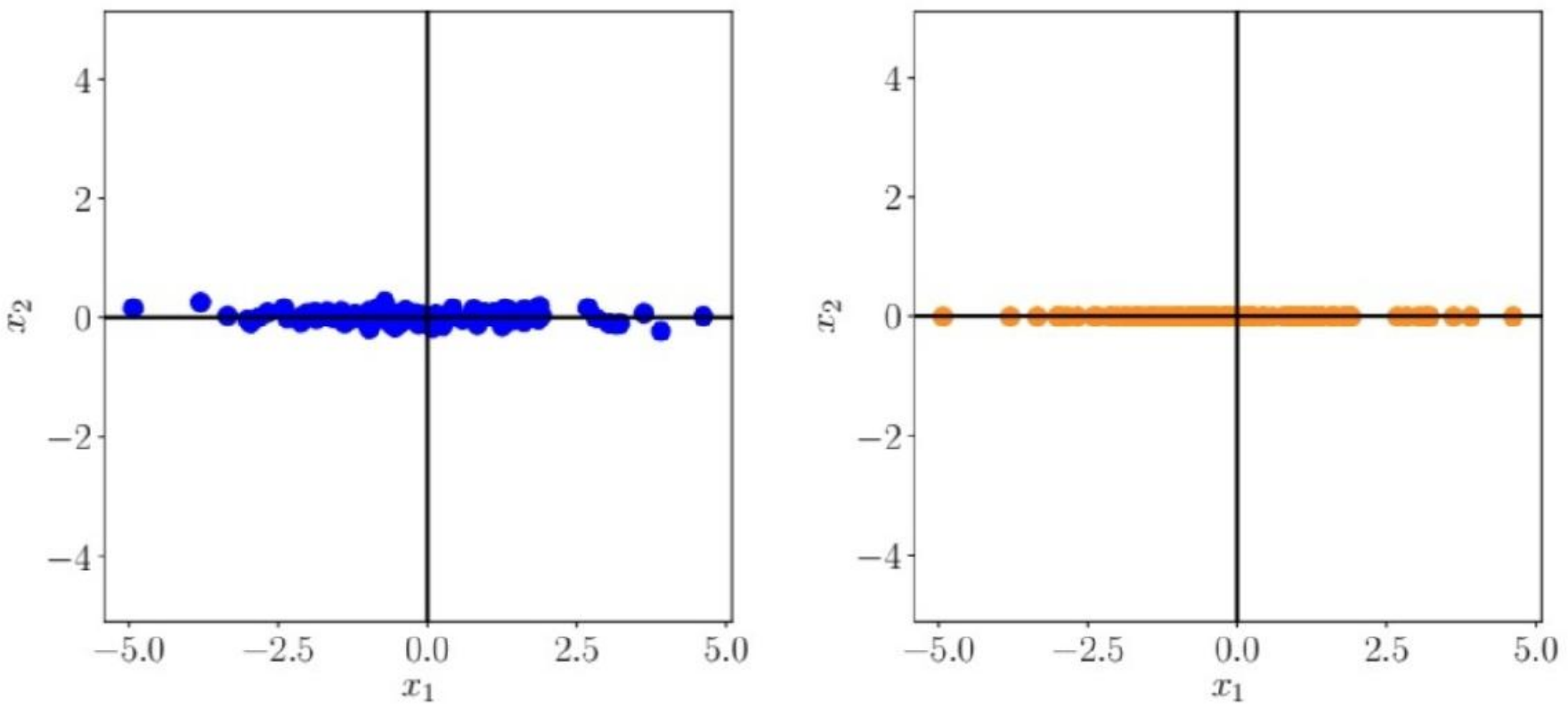
.

.

§ The PCA Problem

Principal component Analysis looks for projections $\vec{\tilde{x}}_n$ of datapoints $\vec{x}_n$, as similar as possible of the original, but with significantly lower intrinsic dimensionality.

This is in the heart of the so called feature extraction or feature selection



(a) Dataset with x_1 and x_2 coordinates.

(b) Compressed dataset where only the x_1 coordinate is relevant.

Figure 10.1 Illustration: dimensionality reduction. (a) The original dataset does not vary much along the x_2 direction. (b) The data from (a) can be represented using the x_1-coordinate alone with nearly no loss.

Suppose we have a vector $\vec{x}$, with $\dim(\vec{x}) = m$, and we want to transmit the encoded information with a vector $\tilde{x}$ with $\dim(\tilde{x}) = l$, and $l < m$. Of course, there will be a loss of information, however one might ask:

"There is a linear map $\hat{T}$ so that a compression $\tilde{\vec{x}} = \hat{T}\vec{x}$ is optimal, given a loss function?"

This loss function must be related with the (a) variance, (b) mean-squared-error, (c) geometry.

This $\hat{T}$ transformation has to posesses the property that some of its components has low-variance. The transformation with this property is called Principal Component Analysis.

Consider a i.i.d dataset $\mathcal{X} = \{\vec{x}_i\}_{i=1}^N$, $\vec{x}_i \in \mathbb{R}^D$ and

$$(1) \qquad \mathbb{E}[\mathcal{X}] = 0$$

and with a data-covariance matrix:

$$(2) \qquad \overleftrightarrow{S} = \frac{1}{N} \sum_{i=1}^{N} \vec{x}_i \, \vec{x}_i^{\,\dagger}$$

We claim that there is a transformation $\hat{T}$ that give us a compressed representation:

$$(3) \qquad \vec{z}_i = \hat{T} \vec{x}_i = \hat{B}^{\dagger} \vec{x}_i \in \mathbb{R}^M$$

where we define the projection operator:

$$(4) \qquad \hat{B} := [\vec{b}_1, \ldots, \vec{b}_M] \in \mathbb{R}^{D \otimes M}$$

with orthonormal columns:

$$(5) \qquad \vec{b}_i^{\,T} \cdot \vec{b}_j = \delta_{ij}$$

Our goal is to find a M-dimensional sub-space $U \subseteq \mathbb{R}^D$, $\dim(U) = M < D$, onto which we project the data. The projection is denoted as $\vec{\tilde{x}}_i \in U$, and their coordinate with respect to the basis $\{\vec{b}_i\}$ of U, by $\vec{z}_i$. We seek for projections $\vec{\tilde{x}}_i \in \mathbb{R}^D$, similar to $\vec{x}_i$ under a compression loss metric and can be efficiently described in the $\{b_i\}$ basis.

As an illustration, consider a dataset X, where we had subtracted the mean of all instances, so that:

$$(6) \quad \mathbb{E}[X] = M$$

$$(7) \quad X \longrightarrow X - M \Rightarrow \mathbb{E}[X] = 0$$

The projection over the orthonormal b-basis:

$$(8) \quad A = \vec{x}^T \cdot \vec{b} = \vec{b}^T \cdot \vec{x}$$

$$(9) \quad \mathbb{E}[A] = \vec{b}^T \cdot \mathbb{E}[\vec{x}] = 0$$

The variance of the projection:

$$(10)\quad \sigma^2 = E[A^2] - (E[A])^2$$

$$= E[\vec{b}^T \cdot \vec{x}\,\vec{x}^T \cdot \vec{b}]$$

$$= \vec{b}^T E[\vec{x} \cdot \vec{x}^T] \vec{b}$$

$$= \vec{b}^T \overleftrightarrow{R}\, \vec{b}$$

where $\overleftrightarrow{R}$ is the covariance matrix. The covariance function is symmetric, and is a property of the data set. Thus, the variance of the projection is a function of the orthonormal basis:

$$(11)\quad \varphi(\vec{b}) = \sigma^2$$

$$= \hat{b}^T \overleftrightarrow{R}\, \hat{b}$$

These vectors has to be chosen in a direction so that σ^2 is a extremum or at least stationary. Thus, this condition means that:

$$(12)\quad \varphi(\vec{b} + \delta\vec{b}) = \varphi(\vec{b})$$

(13) $\varphi(\vec{b}+\delta\vec{b}) = (\vec{b}+\delta\vec{b})^T \overleftrightarrow{R} (\vec{b}+\delta\vec{b})$

$= \vec{b}^T \overleftrightarrow{R} \vec{b} + \vec{b}^T \overleftrightarrow{R} \delta\vec{b} + \delta\vec{b}^T \overleftrightarrow{R} \vec{b} + \delta\vec{b}^T \overleftrightarrow{R} \delta\vec{b}$

Up to 1st order of perturbation $= \vec{b}^T \overleftrightarrow{R} \vec{b} + 2\delta\vec{b}^T \overleftrightarrow{R} \vec{b}$

Using (12): $\delta\vec{b}^T \overleftrightarrow{R} \vec{b} = 0$, and we recover (13).

This constraint gives us a clue about the direction of the projection over the base:

(14) $\|\vec{b}+\delta\vec{b}\| = 1 \Rightarrow (\vec{b}+\delta\vec{b})^T(\vec{b}+\delta\vec{b}) = 1$

again up to 1st order in perturbation theory, we have:

(15) $\delta\vec{b}^T \cdot \vec{b} = 0$

Thus the variations has to be orthogonal to the basis vectors! This means that only transformations that the Euclidean norm remains unitary are admissible, and these ones are in the same direction of b:

(16) $\max_{\hat{b}} \hat{b}^T \overleftrightarrow{R} \hat{b}$

subject to $\|\hat{b}^T \cdot \hat{b}\| = 1$

Using the method of Lagrange multipliers:

$$(17)\quad \mathcal{L}(\hat{b},\lambda) = \hat{b}^T \overleftrightarrow{R} \hat{b} + \lambda(1 - \hat{b}^T\hat{b})$$

$$(18)\quad \frac{\partial \mathcal{L}}{\partial \hat{b}} = \hat{b}^T \overleftrightarrow{R} - \lambda \hat{b}^T = 0 \Rightarrow \hat{b}^T \overleftrightarrow{R} = \hat{b}^T \lambda$$

$$(19)\quad \frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \hat{b}^T\hat{b} = 0 \Rightarrow \hat{b}^T\hat{b} = 1$$

Resulting in a boundary value problem:

$$(20)\quad \overleftrightarrow{R}\,\hat{b} = \lambda \hat{b}$$

$$(21)\quad \hat{b}^T\hat{b} = 1$$

Let the $\{\lambda_i\}_{i=1}^m$ eigenvalues ordered in decrescent order:

$$(22)\quad \lambda_1 > \cdots > \lambda_m$$

and we have a set of basis vectors:

$$(23)\quad \hat{B} = [\hat{b}_1, \ldots, \hat{b}_m]$$

we have:

$$(24)\quad \overleftrightarrow{R}\,\hat{B} = \hat{B}\,\Lambda$$

where

(25) $$\Lambda = \text{diag}[\lambda_1, \ldots, \lambda_m]$$

the dataset can be represented as:

(26) $$\overleftrightarrow{R} = \hat{B} \hat{\Lambda} \hat{B}^+$$

and by means of the spectral theorem:

(27) $$\overleftrightarrow{R} = \sum_i \lambda_i \hat{b}_i \hat{b}_i^+$$

Principal component Analysis and the eigen decomposition of the covariance matrix are basically the same:

(28) $$\phi(\hat{b}_i) = \lambda_i$$

From this development, we see that every single vector in the feature space can be represented as a linear combination of the eigenvectors of the covariance matrix

(29) $$\vec{x} = \sum_i a_i \hat{b}_i$$

$$= \begin{bmatrix} \hat{b}_1 \\ \vdots \\ \hat{b}_m \end{bmatrix} \begin{bmatrix} a_1, \ldots, a_m \end{bmatrix}$$

where the a_i's are called principal component of the feature vector.

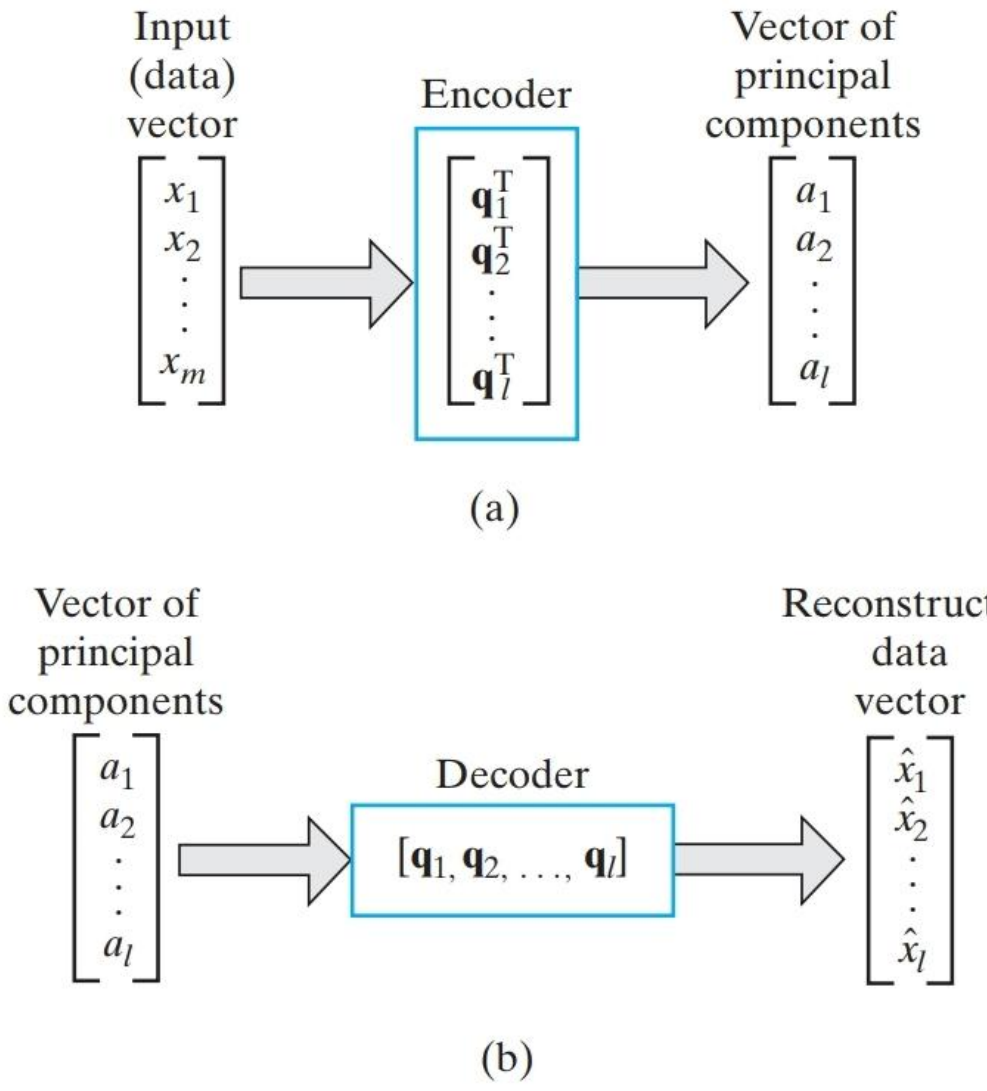


FIGURE 8.2 Illustration of two phases of principal-components analysis: (a) Encoding. (b) Decoding.

This approach is called maximal variance procedure.

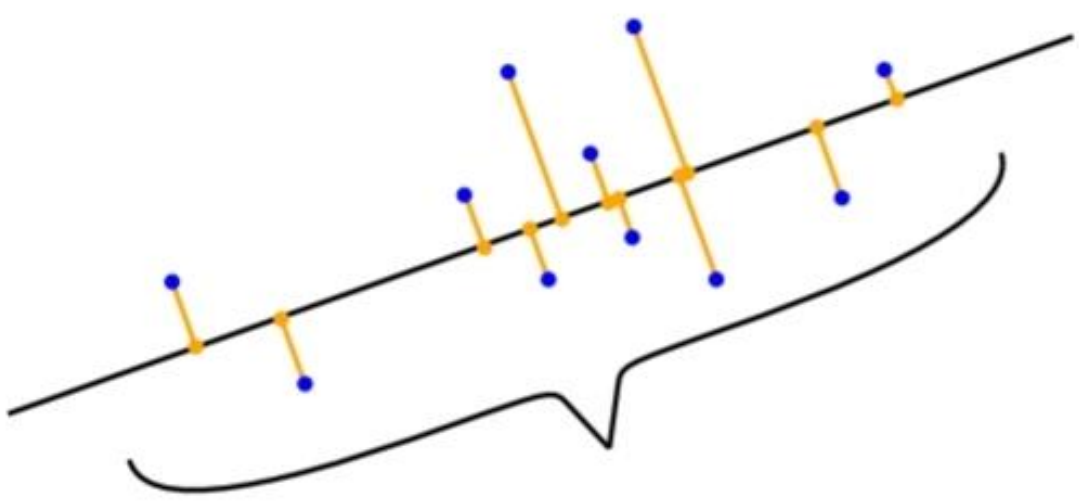


Figure 10.4 PCA finds a lower-dimensional subspace (line) that maintains as much variance (spread of the data) as possible when the data (blue) is projected onto this subspace (orange).

When we looks for a new basis where we can span the data set, truncating in a dimension $l < m$, retaining as much variance as possible.

The linear map that goes from the original data space $\mathbb{R}^m$ to $\mathbb{R}^l$ is called "encoder". And the way back transformation is called "decoder".

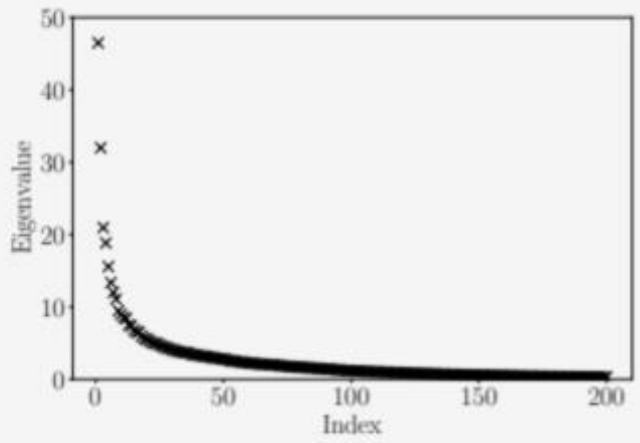
The general approach to dimensionality reduction is to select the first l biggest eigenvalues of the ordered eigenvalues and encode all information into these l basis vector.

The truncation error is given by:

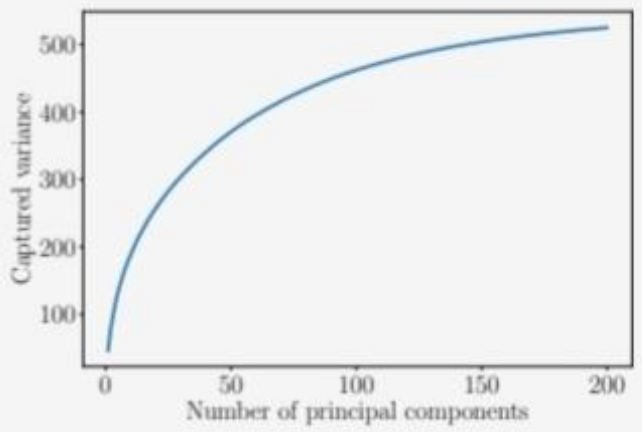
$$\hat{e} = \vec{x} - \vec{\tilde{x}} = \sum_{i=l+1}^{m} a_i \hat{b}_i \tag{30}$$

and the retained variance of the original data is given by:

$$\hat{v}^2 = \sum_{j=1}^{m} \sigma_j^2 = \sum_{j=1}^{m} \lambda_j \tag{31}$$



(a) Eigenvalues (sorted in descending order) of the data covariance matrix of all digits "8" in the MNIST training set.



(b) Variance captured by the principal components.

The dimension of the subspace is a tradeoff between complexity and retained amount of variance.

§ Eigenvector Computation

The space of principal components is a vector space formed by the eigenvectors of the covariance matrix:

$$\overleftrightarrow{R} = \frac{1}{N}\sum_{i=1} \vec{x}_i \cdot \vec{x}^T = \vec{\chi} \cdot \vec{\chi}^T \tag{32}$$

please, note that $\vec{\chi} \in \mathbb{R}^{N \otimes D}$, being N the number of observations and D the number of features, this is a huge rectangular matrix.

The eigendecomposition can be performed in two ways:

i) Directly from the matrix $\overleftrightarrow{R}$,

ii) Using the singular value decomposition.

the second approach is more reliable to our purposes:

$$\overleftrightarrow{\chi} = \underbrace{\overleftrightarrow{U}}_{D \otimes N} \underbrace{\overleftrightarrow{\Sigma}}_{D \otimes D} \underbrace{\overleftrightarrow{V}^T}_{D \otimes N \; N \otimes N} \tag{33}$$

where $\overleftrightarrow{U} \in \mathbb{R}^{D \otimes D}$ and $\overleftrightarrow{V}^T \in \mathbb{R}^{N \otimes N}$ are orthogonal matrices and $\overleftrightarrow{\Sigma} \in \mathbb{R}^{D \otimes N}$ is a matrix whose

whose non-zero entries are the singular values $\sigma_{ii} \geq 0$. It follows

$$\overleftrightarrow{R} = \frac{1}{N} \overleftrightarrow{X} \overleftrightarrow{X}^T \tag{34}$$

$$= \frac{1}{N} \overleftrightarrow{U} \overleftrightarrow{\Sigma} \underbrace{\overleftrightarrow{V}^T \overleftrightarrow{V}}_{\mathbb{1}_N} \overleftrightarrow{\Sigma}^T \overleftrightarrow{U}^T$$

$$= \frac{1}{N} \overleftrightarrow{U} \overleftrightarrow{\Sigma} \overleftrightarrow{\Sigma}^T \overleftrightarrow{U}^T$$

the columns of $\overleftrightarrow{U}$ are the eigenvectors of $\overleftrightarrow{R}$ and the eigenvalues λ_d of $\overleftrightarrow{R}$ are related with the singular values of $\overleftrightarrow{X}$ via:

$$\lambda_d = \frac{\sigma_d}{N} \tag{35}$$

thus, the maximum variance Principal component analysis is just the singular value decomposition of the $\overleftrightarrow{X}$ matrix.

In order to maximize the variance of the projected data, PCA chooses the columns of $\overleftrightarrow{U}$ as the eigenvectors of a M-dimensional space of the M-largest eigenvalues of $\overleftrightarrow{R}$.

This means that, we identify $\overleftrightarrow{B}$ as the projection of $\overleftrightarrow{U}$ into this M-dimensional vector space. The optimal low-rank approximation is the one obtained by the Eckart-Young theorem:

$$\overleftrightarrow{\tilde{X}}_M : \underset{\text{rank}(\overleftrightarrow{A}) \leq M}{\text{argmin}} \| \overleftrightarrow{X} - \overleftrightarrow{A} \|_2 \in \mathbb{R}^{D \times N} \quad (36)$$

where $\| \cdot \|_2$ is the spectral norm:

Thus, we obtain $\overleftrightarrow{\tilde{X}}_M$ by truncating the SVD representation of the top-M singular values:

$$\overleftrightarrow{\tilde{X}}_M = \underbrace{\overleftrightarrow{U}_M}_{D \times M} \underbrace{\overleftrightarrow{\Sigma}_M}_{M \times M} \underbrace{\overleftrightarrow{V}_M^T}_{M \times N} \in \mathbb{R}^{D \times N} \quad (37)$$

§ PCA Dimensionality Reduction Program

Given a arbitrary dataset, the PCA procedure runs the following steps:

1- Mean substraction

2- Standarization

3- Spectral factorization of the covariance matrix.

4- Projection.