Lecture Notes on Dimension Reduction

Edwige Cyffers

Contents

Contents

Section 1

Introduction

1.1 Motivation

Machine learning relies on the exploitation of large amounts of complex data. ChatGPT illustrates the massive scale of data involved: ChatGPT, for example, was trained on hundreds of billions of words from a wide range of internet sources. This is a consequence of our ability to collect data effortlessly, leading us to gather information with an unprecedented level of detail.

Instead of knowing from a paper agenda that you went to college in the morning, you could access your GPS position at each minute of the day. Instead of having the imprecise recording of palpitations, a smartwatch can report each heartbeat. For predicting weather, one can exploit the history of measurements, various sensors, and satellite images. Quantitative finance replaces a few macroeconomic indicators and discussions between traders with systematic analysis of prices and automated sentiment analysis...

The multiplication of data collection possibilities—such as the Internet of Things, the expansion of the internet, widespread use of smartphones, smart cities, and the unprecedented capability to process data with extremely efficient microprocessors—has created an environment where exploiting high-dimensional data is a plausible and reliable source of knowledge.

Dealing with high-dimensional data is relatively new, due to the emergence of new tools, leading us to mistakenly expect behaviors similar to the low-dimensional data we are accustomed to. Our intuition for high-dimensional data is lacking, and we often do not know how to interpret it by default. One of the goals of this lecture is to alert you to false intuitions you may have about high-dimensional data.

One could argue that all machine learning models are dimension reduction mechanisms: a classification task reduces the input to a single label; a regression task reduces it to a single real number. In more detail, a max-pooling layer in a neural network can be seen as a very basic dimension reduction technique. Why, then, study dimension reduction independently if machine learning models are capable of handling high-dimensional inputs?

Dimension reduction is driven by several key reasons, which we outline below:

Visualization Visualizing data on plots or charts can be very telling about its quality and uniformity. This is achieved by using dimension reduction techniques to represent each data point on the plot. However, it's important to consider what aspects this representation preserves and what insights cannot be derived from it.

Preprocessing For a statistical model to be effective, it needs a sufficient number of samples relative to the number of parameters it aims to fit. Reducing dimensions lowers the number of parameters, enabling faster learning. Sometimes, data that appears to be in a high-dimensional space actually exists in a lower-dimensional one. Applying additional constraints can help clean the data and deal with outlier points, which is particularly useful because, although we can interpolate data, extrapolating is more challenging.

Sobriety Dimension reduction becomes necessary when data transmission is limited by bandwidth, storage space is scarce, or when applying techniques that are computationally intensive.

The methods for reducing dimensions vary widely and are expected to evolve. This lecture series focuses on a mathematical understanding of the most established methods, providing the tools to comprehend other techniques encountered in real-world scenarios, with an emphasis on the underlying mathematical principles and outcomes. The lectures are organized as follows:

Algebra requirements and the curse of dimensionality We will start with the necessary mathematical foundation for upcoming lectures and discuss the curse of dimensionality to highlight common misconceptions about high-dimensional data.

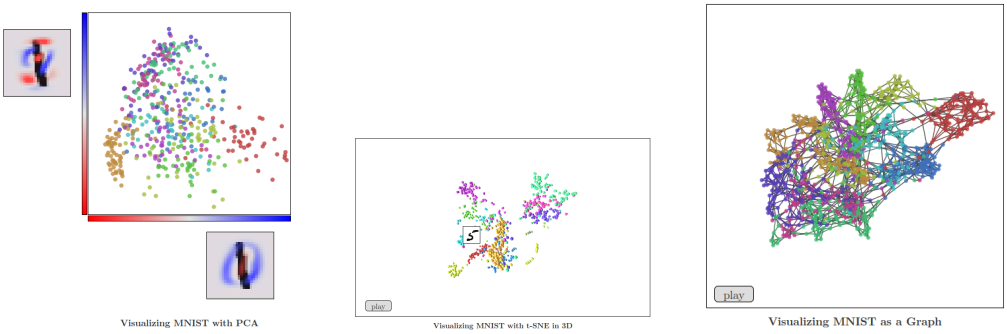
Principal Component Analysis PCA is a staple in many applications, offering clear mathematical backing. The aim is to thoroughly grasp its mathematical basis and learn how to implement it in Python. We will spend two lectures on PCA.

Manifold Learning Focusing on techniques like Locally Linear Embedding and t-SNE, this part will cover dimension reduction methods that aim to preserve local structures within the data.

Spectral clustering Graphs present unique challenges as examples of high-dimensional data. We'll explore how to structure data points into graphs and delve into spectral clustering.

Variational Autoencoders We will conclude with VAEs, where the embedded space results from a learning process.

A good example of visualization of the method that we will see during the lecture on the Mnist dataset can be seen in this blog post https://colah.github.io/posts/2014-10-Visualizing-MNIST/



(a) PCA (b) t-SNE (c) Graph

Figure 1. Visualization of MNIST dataset using different techniques done by Christopher Olah, you can run the animation online

1.2 Grading

The validation of the course is done in two equal parts:

- Quiz on the previous lectures at the beginning of each class.
- A small project by groups of 2 or 3 students.

1.3 Setting

Throughout the lecture, we consider a dataset with n samples, where the samples are in $\mathbb{R}^d$. Denoting X as the whole dataset, it can be seen as a matrix, where X_j^i is the j-th component of the i-th sample.

$$X = \begin{bmatrix} X^{1\top} \\ \vdots \\ X^{n\top} \end{bmatrix} \in \mathbb{R}^{n \times d}$$

The goal is to generate a smaller vector $Y^i \in \mathbb{R}^k$ to represent each sample X^i, where k is small compared to d.

Note that by default, a vector is always a column vector. This is why we use $X^\top$ to denote the row-vector. We can always represent data as a unidimensional vector, even when the data has a different natural shape. For instance, an image can have a shape $(32, 32, 3)$ but can be flattened. Imposing the coordinates to be floats remains general: in particular, integer and binary values can also be seen as real values. Categorical data can be encoded with binary values via one-hot encoding.

Given a dataset D with a categorical feature F, containing n unique categories $C = \{c_1, c_2, \ldots, c_n\}$, the one-hot encoding process is as follows:

1. Identify the unique categories within F, denoted as C.
2. For each category $c_i \in C$, create a binary vector V_i of length n, where V_i has a 1 at the position corresponding to c_i and 0s elsewhere.
3. Replace each sample's categorical feature with its corresponding binary vector V_i.

The resulting dataset will have its categorical feature transformed into n binary features, each indicating the presence or absence of a category with 1 or 0, respectively.

For example, encoding the categorical feature "Pet Type" with categories {"Dog", "Cat", "Horse"} would result in a binary vector representation for each category, such as $[1, 0, 0]$ for "Dog".

Below is a Python code snippet using the pandas library to perform one-hot encoding:

```python
import pandas as pd

# Sample dataset
data = {'Pet Type': ['Dog', 'Cat', 'Horse', 'Cat', 'Dog']}
df = pd.DataFrame(data)

# One-hot encoding
one_hot_encoded_df = pd.get_dummies(df, columns=['Pet Type'])
```

Now that we know our input data can be formatted as a single bidimensional matrix, let's define what a dimension reduction method is.

Definition 1
We call *embedding* such a function $\mathbb{R}^d \to \mathbb{R}^k$ that sends a sample to the smaller space and conserves or approximates some well-chosen properties.

Depending on the field and applications, the properties we are looking for differ. Hence, the goal of this lecture is also to provide good intuitions on what should be kept in mind when choosing a specific dimension reduction method.

In Machine Learning, samples are seen as trials of the unknown distribution we seek to approximate. Hence, we reuse probability concepts to describe the computations done on X.

It is also important to be able to switch from the statistical perspective to the algebraic formulation and the Python equivalent.

Example | As a first step in analyzing data, one wants to compute the mean of each feature on the samples. For the j-th feature, it corresponds to $\frac{1}{n}\sum_{i=1}^{n} X_j^i$.

This quantity can be computed in Python with:

```
X.mean(axis=0)
```

It corresponds to $\frac{1}{n}X^\top \mathbb{1}_n$ that returns the vector of size d where the j-th coordinate is $\mathbb{E}(X_j)$.

Exercise Give the expression of the matrix that should be applied and Python instruction to compute $\hat{X}$ the 0-mean matrix derived from X.

Answer In Python, the instruction is:

```
X_hat = X - X.mean(axis=0)
```

which corresponds to $(I - \frac{1}{n}\mathbb{1}_n\mathbb{1}_n^\top)X$. □

Section 2

Curse of Dimensionality

The expression *curse of dimensionality* was introduced in 1961 by Bellman. Bellman is famous for dynamic programming: you have/will see Bellman's equation in reinforcement learning.

The curse of dimensionality corresponds to the fact that adding a dimension dramatically increases the volume considered. Hence, in high dimensions, all points are isolated from each other. The concept of neighborhood you have from small dimensions does not scale to large dimensions, where interpolation is usually impossible because there is not enough data points to fill the space.

From privacy in machine learning (my field of research!), you can see the curse of dimensionality as the fact that all individuals are easily completely unique as soon as enough data is collected. A famous example is given by the Netflix prize. Netflix introduced a challenge to improve their algorithms to predict the rating of a film by a given user based on their previous ratings. In order to protect the anonymity of the customers, personally identifiable information such as username, credit card, gender, date of birth was removed. Only the list of the dates and ratings were shared. It was proven that in fact this information is already too precise: by crossing this database with the public ratings available on the MovieLens website, it is possible to re-identify users as soon as they rate four movies on both platforms, as was shown by researchers in 2008 [5]. We report one of their figures. Even mundane events such as watching some blockbusters make you unique, because other people won't see them exactly on the same days. This gives you an idea of how your datapoints are different from each other as soon as you have high-dimensional data.

We will see three different mathematical results to quantify this intuition:

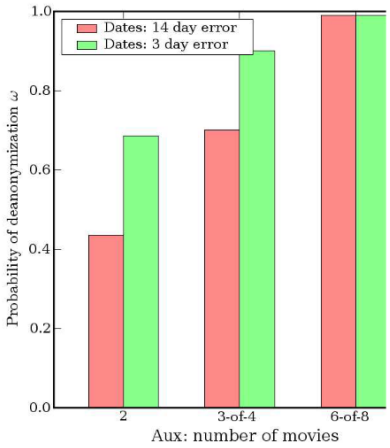


Figure 2. De-anonymization reported in [5]: adversary knows exact ratings and approximate dates

Number of points to fill the space We will show that for a given precision to estimate a function from existing samples, the number of samples grows exponentially with the dimension.

Volume concentrates on the surface We will see that the volume of a unit sphere is mainly due to the volume on the surface of the sphere, meaning that if points are drawn randomly in the space, the points contained in a sphere will be close to its surface.

Johnson-Lindenstrauss lemma We will see the intrinsic dimension of a set of points cannot exceed a given dimension depending on its cardinality, and that there exists an embedding in this dimension with little distortion.

Let's first consider that we want to learn a function from examples in $[0,1]^d$ and that the only regularity assumption is that the function is Lipschitz. For a given precision, we thus need a data point within distance ϵ to it. In particular, if we cut the unit cube into small cubes of edge length ϵ, we need at least one datapoint per distinct cube. The number of cubes is given by $(1/\epsilon)^d$. For example, for $\epsilon = 0.1$, we need at least 10^d data points to approximate the function correctly.

Let's now fix a small $\eta > 0$, what proportion of the sphere is close to the surface up to η? You know how to express the surface of the circle in 2D $V_2(r) = \pi r^2$ and the volume of the sphere in 3D $V_3(r) = 4\pi/3r^3$. More generally, we can express it as $V_d(r) = C_d r^d$. Thus, we can express the ratio of the volume of the complete sphere and the one reduced to $1 - \eta$

$$\frac{V_d(1) - V_d(1-\eta)}{V_d(1)} = \frac{1 - (1-\eta)^d}{1} \to 1$$

when d goes to infinity.

Let's now finish by discussing the Johnson-Lindenstrauss lemma.

Theorem 1 Johnson-Lindenstrauss lemma (1984)
Given $0 < \varepsilon < 1$, a set X of $n \in \mathbb{N}_{\geq 1}$ points in $\mathbb{R}^d$ ($d \in \mathbb{Z}_{\geq 0}$), and an integer $k > 4(\ln n)/\left(\varepsilon^2/2 - \varepsilon^3/3\right)$, there is a linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ to transform $(X^i)_i$ in the embedded $(Y^i)_i$ such that

$$(1-\varepsilon)\|X^i - X^j\|^2 \leq \|Y^i - Y^j\|^2 \leq (1+\varepsilon)\|X^i - X^j\|^2$$

for all $i, j \in \{1, \ldots, n\}$.

As you can see, the dimension of the embedding is independent from the original dimension of the data: what matters is only the space of constraints formed by the relationships among the datapoints. It also gives you an upper bound on how much we can fill the space with n datapoints: it does not exceed $\log(n)$ points. This recovers the idea that the number of datapoints needed to fill space grows exponentially with the number of dimensions, indirectly. Finally, note that ϵ must be small and, in particular, smaller than 1, so the term that dominates is ϵ^2.

The proof is based on the probabilistic method. It proves the existence of the embedding by demonstrating that its probability of existing is greater than 0. This method of proof was popularized by Erdős, who, for example, proved a lower bound on Ramsey numbers in 1947 (a combinatorics problem on the emergence of order in high-dimensional structures).

Proof Admitted result: Let's take uniformly at random a vector x on the sphere of dimension d. Let y be the vector of the k first coordinates of x. Then, for ϵ as defined above:

$$\mathbb{P}(\|y\|^2 \leq (1-\epsilon)\frac{k}{d}) \leq \frac{1}{n^2} \text{ and } \mathbb{P}(\|y\|^2 \geq (1+\epsilon)\frac{k}{d}) \leq \frac{1}{n^2}$$

Let's now consider a dataset X. We fix k unit random vectors and p the projection on these random vectors. We define f as the rescaled version of p: $f = \sqrt{\frac{d}{k}}p$.
Let's fix $u, v \in X$. We apply the previous lemma to the difference $u - v$. The probability of violation is $\frac{1}{n^2}$. As there are $\frac{n(n-1)}{2}$ pairs of vectors, upper bounding the union by the sum of probabilities, the probability of violation is less than $1 - \frac{1}{n}$, so f fulfills the constraint with probability greater than $1 - \frac{1}{n}$. Hence, on average, we need to draw n projections to find a good one. □

Note that the precise constant is not important. In particular, in practice, some other projections are more efficient than Gaussian and do not yield exactly the same constants. The presentation of the theorem is chosen to be coherent with the `scikit-learn` implementation. A detailed proof

can be found in [2].

Random projection might seem quite simple. In particular, it does not exploit information about the dataset; the projection is chosen independently of the datapoints. However, this technique is really useful as the preservation of the distance is generally good, while being extremely fast compared to other dimension reduction methods. Even from a visualization point of view, random projections are impressive. For an anecdote, several students mislabeled PCA and random projection results during the exam.

Section 3

Principal component Analysis

3.1 Deriving PCA

Principal Component Analysis (PCA) is the most important method of dimension reduction. Its popularity comes from several factors:

Linearity The transformation is linear, meaning that $Y^i = P^\top X^i$, where we can explicitly compute the matrix P.

Optimality When restricted to a given dimension k and linear transformation, PCA is optimal in maximizing the explained variance.

Simplicity The computation is easy to understand and relies on well-known algebra and has connections with statistical methods.

Orthogonality PCA uncorrelates the different axes.

This list of advantages might sound unclear to you at this point in the lecture. We will introduce PCA through the Rayleigh quotient on the empirical covariance matrix.

In all the following, to simplify the notation, we assume X to be centered. It is crucial to center the data to ensure the validity of the computations below.

From our data matrix X, we can easily compute the empirical covariance matrix $\Sigma = \frac{1}{n} X^\top X$ of size $d \times d$. This equality holds because the data is centered, so $\mathrm{Cov}(X_i, X_j) = \frac{1}{n} X_i^\top X_j$. This matrix Σ is symmetric and can thus be decomposed according to the spectral theorem:

$$\Sigma = PDP^\top$$

Let's now focus on finding a unique axis that would allow the data to spread as much as possible, which means maximizing the variance after transformation. Finding an axis is equivalent to finding a unit vector in the

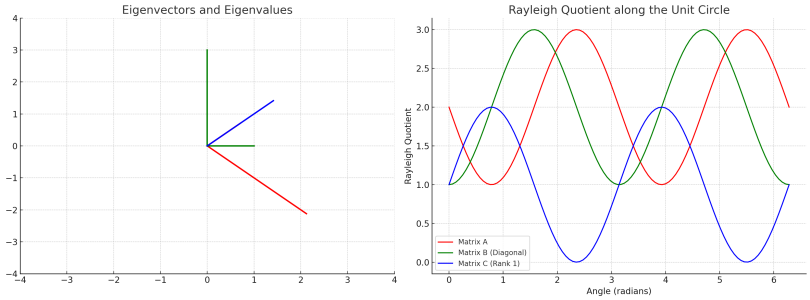


Figure 3. We plot the Rayleigh quotient of Matrix $A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$, Matrix $B = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$ and Matrix $C = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. You can see that A and C reach extremum for the same angle, because one of their eigenvectors is identical $(1/\sqrt{2}, 1/sqrt2)$ (you see only the blue line, the red one is below it). More generally, you see that the maximum corresponds to the largest eigenvalue and is obtained for the corresponding eigenvector.

direction of the axis. The maximality we aim for here is naturally captured by the Rayleigh quotient.

Definition 2 Rayleigh Quotient
The Rayleigh Quotient for a square matrix A and a nonzero vector $\mathbf{x}$ in $\mathbb{R}^n$ is a scalar defined as:

$$R(A, \mathbf{x}) = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}}.$$

The denominator, $\mathbf{x}^T\mathbf{x}$, is simply the dot product of $\mathbf{x}$ with itself, representing the squared Euclidean norm of $\mathbf{x}$.

Exercise Show that for a fixed matrix A, all the values taken by the Rayleigh Quotient can be reached by vectors of norm 1.

Answer | This follows from the fact that for every nonzero scalar λ, we have $R(A, \mathbf{x}) = R(A, \lambda\mathbf{x})$ due to the property of linearity. □

Let's represent the Rayleigh quotient for some matrices.

In this formulation, the axis $\mathbf{u}$ maximizing the variance after projection, namely $\mathrm{Var}(X\mathbf{u})$, is the one maximizing $R(\Sigma, \mathbf{u})$. Let's see how the decomposition provided by the spectral theorem allows us to further simplify this problem.

$$
\begin{aligned}
\max_{\|u\|=1} \operatorname{Var}(Xu) &= \max_{\|u\|=1} (Xu)^\top (Xu) && \text{(as } X \text{ is centered)} \\
&= \max_{\|u\|=1} u^\top X^\top X u \\
&= n \max_{\|u\|=1} \mu^\top \Sigma u && \text{(we recognize the Rayleigh quotient)} \\
&= n \max_{\|u\|=1} u^\top P D P^\top u && \text{(by the spectral theorem)} \\
&= n \max_{\|u\|=1} \sum_{i=1}^{d} \lambda_i \left(P_u^\top\right)_i^2 && \text{(via simple calculus)} \\
&= n \max_{\|v\|=1} \sum_{i=1}^{d} \lambda_i v_i^2 && \text{(as } P \text{ is orthonormal)} \\
&= n\lambda_1 && \text{(obtained for } v = e_1)
\end{aligned}
$$

So we see that the maximum is $n\lambda_1$. Reciprocally, this value is reached for the vector P_1, the eigenvector of Σ corresponding to the largest eigenvalue λ_1. Indeed, since eigenvectors are orthogonal:

$$
\left(P^\top P_1\right)_i = P_i^\top P_1 = \begin{cases} 1 & \text{if } i = 1, \\ 0 & \text{otherwise.} \end{cases}
$$

This computation proves that the axis maximizing the variance for projecting the data is given by the eigenvector associated with the largest eigenvalue.

Similarly, algebraic computations allow us to prove that the second direction to maximize the variance, while being orthogonal to the first direction, is exactly the one given by the eigenvector associated with the second largest eigenvalue, and so on. This means that for projecting the data points into k dimensions while maximizing the variance, we must project onto the first k eigenvectors.

Let's denote $P^{(k)} \in \mathbb{R}^{d \times k}$ the matrix composed of the first k columns of the matrix P. The PCA in k dimensions is the result of the projection by the matrix:

$$
Y^\top = P^{(k)\top} X^\top, \text{ i.e., } Y = X P^{(k)}.
$$

You should now understand the advantages listed above when introducing PCA. Note that we can easily add new points X' to an existing PCA, just by using the same projection matrix:

$$
Y' = X' P^{(k)}.
$$

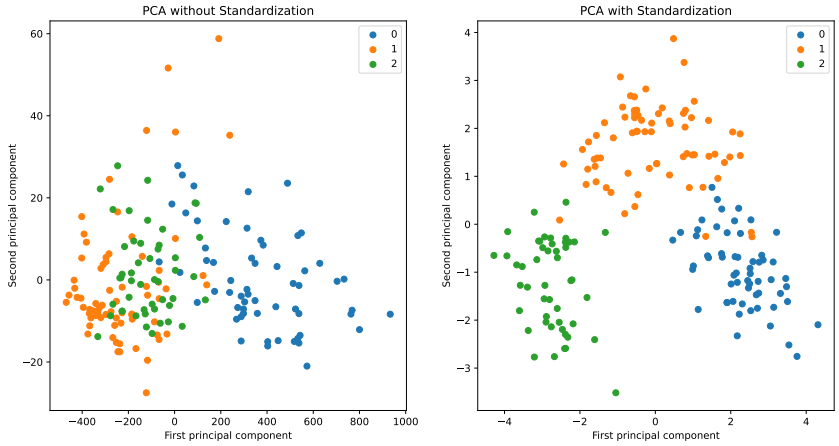


Figure 4. Comparison of PCA with and without normalization for the Wine dataset. Normalization allows better separation between classes because the axes are more informative.

For a given point in the transformed space, we can also easily transform the point back to the original space, just by seeing it as the linear combination of the eigenvectors: For a data point $y \in \mathbb{R}^k$, its inverse transform is given by $P^{(k)}y = \sum_{i=1}^{k} y_i P_i$.

Note that here we did all the computation by giving the same weight $\frac{1}{n}$ to the data points. This is what is done by default in machine learning, and it often makes sense when the data points have been acquired through the same mechanisms to give them the same importance. However, very similar computations can be derived with tailored weighting.

One motivation to downweight or even to remove some points from the dataset can be to remove outliers. Indeed, in case of outliers without preprocessing, PCA can be significantly influenced by a single point: in extreme cases, the first axis might become the axis going from the center of the cloud of points to the outlier. It is possible to compute the contribution to the axis of a data point, although we will not detail the method here.

Other introductions to PCA can be found in Chapter 12 of [1]. One could also have a look at [6].

3.2 Normalizing Features

The computation of the empirical covariance Σ relies on the data being centered. (Note that `sklearn` will center the data for you). If this is not the case, the computation loses its meaning, and the results will be highly unsatisfactory. However, many implementations go further in this direction by not only centering the data but also normalizing it, meaning they adjust the variance or standard deviation of each feature to be equal to 1. Unlike data centering, normalization is not a requirement of PCA.

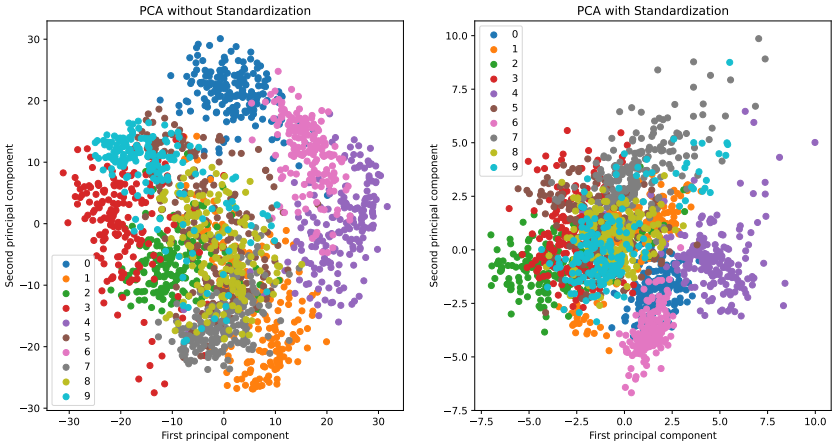


Figure 5. Comparison of PCA with and without normalization for the Digits dataset. The result without normalization offers better spans because all digits are already on an equivalent scale before transformation.

Why is normalizing features often used as a preprocessing step for PCA? Because the axes of PCA are chosen to maximize variance, normalizing ensures that all features will have the same importance in this variance calculation. In this specific case, the global variance is exactly d because each feature contributes 1 to the variance.

Enforcing the same scale across different variables is quite important when the features do not cover the same range (e.g., $X_1 \in [0, 1], X_2 \in [0, 10^4]$) to prevent the projection axes from aligning with the feature with the largest range (here, the main axis would be $\mathbf{e_2}$). An example of improvement via normalization can be seen in fig. 4. Conversely, when features already share the same scale (e.g., pixels, same physical unit...), normalization might be unnecessary or could even yield less favorable results than without normalization, as shown in fig. 5. It is also possible to compare the two results to decide if normalization was beneficial.

3.3 Proportion of explained variance

We have introduced PCA as a way to maximize the variance of the datapoints after transformation. It is thus quite logical to expect to quantify to what extend we succeed in this task.

After transformation, the new empical variance matrix is exactly the diagonal matrix D. Note that the matrix is diagonal, because PCA uncorrelates the dimension thanks to the fact that the decomposition is done in an orthogonal basis.

Hence, when keeping k axis, the variance of the transformed points is $\sum_{i=1}^{k} \lambda_i$. The total variance of the initial dataset is $\sum_{i=1}^{d} \lambda_i$, because it corresponds to the sum of the varaince of all the features. Note that this is

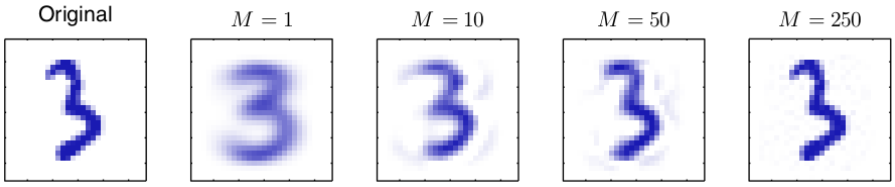


Figure 6. An example from the off-line digits data set together with its PCA reconstructions obtained by retaining M principal components for various values of M. As M increases the reconstruction becomes more accurate and would become perfect when $M = d = 28 \times 28 = 784$. Figure extracted from [1]

the trace, quantity that is invariant by change of basis, so the same of the variance of the variables of the initial data is the same that the one after transformation.

Definition 3 Explained variance ratio
When keeping k dimension in PCA, the explained variance ratio corresponds to the fraction of the variance that is kept after the transformation of the data

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

This quantity is thus between 0 and 1, and 1 is reached when we keep all the dimensions $k = d$. Note that keeping all the dimensions can still be useful in order to uncorrelates the features. You can easily see the level of precision by checking the explained variance ratio or by checking the inverse transformation of the PCA on some samples.

```
pca = PCA(n_components=k)
X_pca = pca.fit_transform(X)
print("Explained variance ratio:", pca.explained_variance_ratio_)
first_sample_pca = X_pca[0, :].reshape(1, -1)  # Reshape needed for a single sample
first_sample_original = pca.inverse_transform(first_sample_pca)
```

In most of use-cases of dimenison reduction, there is a trade-off between the will to keep k as small as possible and the will to keep information of the data. In the case of PCA, the Explained Variance Ratio gives a numerical indicator of the information kept. Some empirical rules have thus been developped around this quantity.

A common approach is to define a threshold, such as 90%. Then, one can choose k as the smallest integer for which the cumulative explained variance ratio exceeds this threshold. This technique is particularly useful for datasets with inherently low-dimensional structure, allowing for the selection of a relatively small k while still retaining most of the information.

However, this method may be overly optimistic for high-dimensional

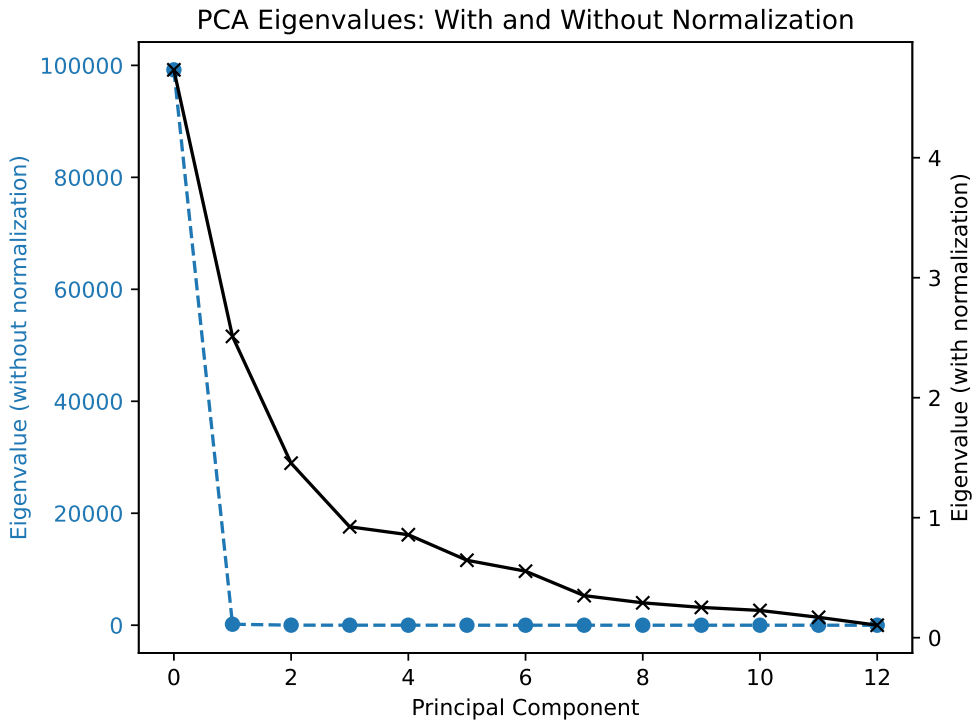


Figure 7. PCA eigenvalues in decreasing order for the Wine dataset. Note that here normalizing features is clearly more informative for this dataset.

data, as a significant portion of the variance may be lost when applying PCA. In such cases, examining the explained variance contributed by each component is more informative. The curve of explained variance can visually aid in determining an optimal k that balances accuracy and practicality. A useful heuristic is to identify an "elbow" in the plot, where the marginal gain in explained variance significantly decreases.

This approach is sometimes referred to as the "Elbow Rule" for its second heuristic. Identifying the elbow can be performed manually by inspecting the plot of eigenvalues. Alternatively, it can be defined mathematically as the point of concavity change in the curve. This involves calculating the finite difference $\delta_i = \lambda_i - \lambda_{i+1}$ for each eigenvalue, followed by the difference of these differences, $\Delta_i = \delta_i - \delta_{i+1}$. The elbow is then identified where Δ_i becomes negative for the first time.

Exercise In fig. 7, how many features should be kept? What is the sum of all the eigenvalues in the normalized version?

Answer | Here, the Elbow rule indicates to keep 4 dimensions. You can see that also give around 80% of the total explained variance. In the normalized version, the total variance is always equal to the number of dimensions, so here it is equal to 13. □

Section 4

Multidimensional Scaling

Multidimensional Scaling (MDS) is another statistical technique aimed at visualizing the level of similarity of individual cases of a dataset. The main goal of MDS is to position each object in a low-dimensional space such that the between-object distances are preserved as well as possible. It corresponds thus to a very intuitive objective: try to keep distances during the transformation. It is indeed an old method, popularized by Kruskal in 1964 [4] thanks to a Fortran implementation.

As seen in the beginning of the lecture, it is impossible to keep all dimensions equal after reduction primarily due to the curse of dimensionality. High-dimensional spaces have properties that do not manifest in low-dimensional spaces, such as the volume of the space increasing exponentially, leading to data becoming sparse.

Hence, MDS try to minimize the level of distorsion. Let's note d_{ij} the distance between X^i and X^j, and δ_{ij} the distance between Y^i and Y^j. MDS try to minimize the following loss function:

$$S(Y) = \sum_{i<j} (d_{ij} - \delta_{ij})^2 \tag{4.1}$$

To minimize this function S, gradient descent can be used. I refer you to the other courses to define Gradient Descent. By iteratively updating the positions of points in the low-dimensional space in the direction that most reduces the stress, gradient descent moves towards a configuration of points that best preserves the original high-dimensional distances.

It exists however situations where the problem admits a closed solution. this is the case for Euclidean distance, where the problem boils down to PCA, as we will show now.

$$\begin{aligned}
S(Y) &= \sum_{i<j} \Big(\|X^i - X^j\|^2 - \|Y^i - Y^j\|^2 \Big)^2 && \text{per definition} \\
&= \sum_{i<j} \Big(\|X^i\|^2 + \|X^j\|^2 - \|Y^i\|^2 - \|Y^j\|^2 + 2\langle X^i, X^j\rangle - 2\langle Y^i, Y^j\rangle \Big)^2 && \text{developping} \\
&= C + 2 \sum_{i<j} \Big(\langle X^i, X^j\rangle - \langle Y^i, Y^j\rangle \Big)^2 && \text{by fixing total variance}
\end{aligned}$$

Here, to continue the calculus, we introduce the Gram Matrix

Definition 4 Gram Matrix
The Gram Matrix $G \in \mathbb{R}^{n \times n}$ is the matrix stocking the value of the scalar product of all the datapoints between them $G_{ij} = \langle X^i, X^j \rangle$. The Gram matrix is a symmetrix matrix that can be computed by $XX^\top$.

We thus observe that minimizing S is equivalent to minimize

$$\begin{aligned}
\mathcal{L} &= \sum_{i<j} \Big(G_{ij} - \langle Y^i, Y^j \rangle \Big)^2 \\
&= \sum_{i<j} \Big(G_{ij}^2 + \langle Y^i, Y^j \rangle^2 - 2 G_{ij} \langle Y^i, Y^j \rangle \Big) \\
&= C' - 2 \sum_{i<j} G_{ij} \langle Y^i, Y^j \rangle \text{ as total variance is fixed} \\
&= C' - 2 \sum_{i<j} Y^{i\top} G_{ij} Y^j
\end{aligned}$$

We recover exactly the same problem of maximizing the Rayleigh quotient that appears in PCA. We thus now that the solutions will given by taking the projection along the eigenvectors associated to the biggest eigenvalues.

But let's go further, what can we say on this eigenvectors?

Proposition 1
Show that the eigenvectors of the matrices AA^T and A^TA can be deduce from each other, given a matrix $A \in \mathbb{R}^{m \times n}$.

Proof Let's start by considering an eigenvector v of A^TA, corresponding to a non-zero eigenvalue λ. That is, we have:

$$A^TAv = \lambda v$$

Pre-multiplying both sides by A, we get:

$$A(A^TAv) = A(\lambda v)$$

Using the associative property of matrix multiplication, this becomes:

$$(AA^T)(Av) = \lambda(Av)$$

This shows that Av is an eigenvector of AA^T corresponding to the same eigenvalue λ. Note that Av cannot be the zero vector because if it were, then we would have $A^T(Av) = \vec{0}$, implying $\lambda v = \vec{0}$, which contradicts the assumption that λ is non-zero.
Similarly, if we start with a eigenvector u of AA^T corresponding to a non-zero eigenvalue λ, i.e.,

$$AA^T u = \lambda u$$

and pre-multiply both sides by A^T, we can show that $A^T u$ is an eigenvector of $A^T A$ corresponding to the same eigenvalue λ.
This demonstrates that for every non-zero eigenvalue, the eigenvectors of AA^T and $A^T A$ are related by the multiplication by A or A^T, indicating that they are essentially "identical" in the sense that they represent the same directions in their respective spaces. □

In particular, this property demonstrates that the solution to MDS for the Euclidean distance is directly obtained from PCA; it results simply from the linear transformation by $X^\top$.

Of course, MDS is typically not used for the Euclidean case, for which PCA is better suited, but rather in scenarios where the distance measure is more exotic. We will explore such examples in the lecture on local methods.

Section 5

Kernel PCA

In machine learning, it is often beneficial to learn classification by identifying a separating hyperplane, as done, for instance, in logistic regression. To accurately learn a classification via this technique, the data needs to be linearly separable. The well-known 'xor' example demonstrates that some configurations of points, even simple ones, cannot be separated by a single hyperplane alone. One way to overcome this difficulty is to transform the data into a higher dimension where it becomes separable, as illustrated in fig. 8.

Kernel methods constitute a rich mathematical field, glimpses of which may have been seen in SVM applications. We will not delve into its depths here, but interested readers can refer to Jean Philippe Vert's lectures [3]. The foundational idea is that datapoints can be represented in higher dimensions, even infinitely so, with a function where scalar products remain computable. This usually leads to intractable computations but remains feasible in certain cases, known as the kernel trick:

Proposition 2 Kernel trick, from [3]
Any algorithm for vectorial data that can be expressed solely in terms of dot products between vectors can be implicitly performed in the feature space associated with any kernel by substituting each dot product with a kernel evaluation.

In essence, the kernel trick involves transforming computations that are impossible or too costly in high dimensions into manageable operations, typically by expressing everything in terms of scalar products.

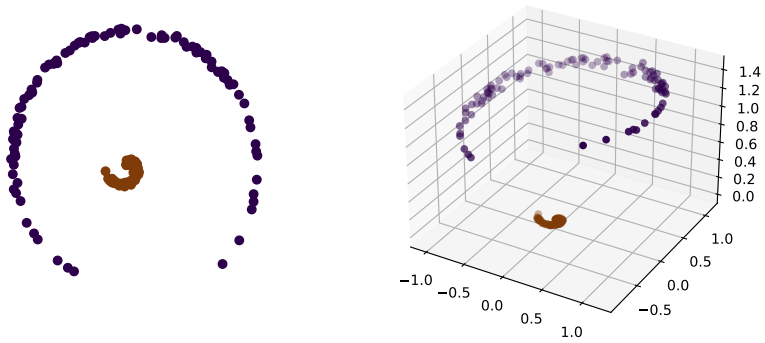


Figure 8. An example of non-separable classes that become separable by transforming the datapoint with the transformation $(x_1, x_2) \to (x_1, x_2, x_1^2 + x_2^2)$ on the right.

Kernel PCA exemplifies such a case. Introduced in 1997 [7], it involves transforming data with a kernel and performing PCA in the new space, which allows for identifying better axes than in the original space because, if the kernel is well-chosen, it captures more informative relationships between the points.

Consider using the traditional Gaussian kernel, where the datapoint X^i is represented by the function $f_{X_i}(x) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{X^i})^{\mathrm{T}}(\mathbf{x}-\boldsymbol{X^i})\right)}{\sigma\sqrt{(2\pi)^k}}$. Directly computing the empirical covariance matrix is impractical (it would be infinitely large). However, the scalar product between two datapoints is still well-defined:

$$\langle X^i, X^j \rangle = \exp\left(-\frac{\|X^i - X^j\|^2}{2\sigma^2}\right)$$

This means the Gram matrix can be computed, in this context, it is usually called the Kernel Matrix K. Then, as seen in MDS, we can determine the eigenvectors and thus obtain Kernel PCA.

A technical challenge arises with centering the data, which is not trivial to perform. It can be shown that it can be done by:

$$K_{ij} \leftarrow -\frac{1}{2}\left(K_{ij} - \frac{1}{n}\sum_l K_{il} - \frac{1}{n}\sum_l K_{jl} + \frac{1}{n^2}\sum_{l,m} K_{lm}\right)$$

Kernel PCA thus consisits it computing this centerel matrix, then by computing the k eigenvectors associated to the largest eigenvalues. Then, the

new points Y^i can be obtained through

$$Y^i = \langle X^i, P^{(k)} \rangle$$

as before, which means that we succeed to compute the projection despite not being able to compute the empirical covariance matrix.

Section 6

Algebra recap

6.1 General Matrix Manipulation

Let's consider a matrix $\mathbf{A}$ with coefficients A_{ij}, where i represents the row index and j represents the column index, and a vector $\mathbf{b}$. When we multiply the matrix $\mathbf{A}$ by the vector $\mathbf{b}$, each coefficient of the resulting vector, let's say $\mathbf{c}$, can be calculated as follows:

$$c_i = \sum_j A_{ij} b_j$$

where c_i is the i-th coefficient of the resulting vector, A_{ij} is the element of the matrix $\mathbf{A}$ at the i-th row and j-th column, and b_j is the j-th coefficient of the vector $\mathbf{b}$.

Example

Consider the matrix $\mathbf{A}$ of size 2×2 and a vector $\mathbf{b}$ of size 2×1:

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

The product $\mathbf{Ab}$ is:

$$\mathbf{Ab} = \begin{pmatrix} A_{11}b_1 + A_{12}b_2 \\ A_{21}b_1 + A_{22}b_2 \end{pmatrix}$$

Numerical Example:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

$$\mathbf{Ab} = \begin{pmatrix} 1 \cdot 5 + 2 \cdot 6 \\ 3 \cdot 5 + 4 \cdot 6 \end{pmatrix} = \begin{pmatrix} 17 \\ 39 \end{pmatrix}$$

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}$$

$$\mathbf{Ab} = \begin{pmatrix} 1 \cdot 7 + 2 \cdot 8 + 3 \cdot 9 \\ 4 \cdot 7 + 5 \cdot 8 + 6 \cdot 9 \end{pmatrix} = \begin{pmatrix} 50 \\ 122 \end{pmatrix}$$

Definition 5 transpose
The transpose of a matrix A, denoted by $A^\top$, is a new matrix whose rows are the columns of A and whose columns are the rows of A. Formally, if A is an $m \times n$ matrix, then $A^\top$ is an $n \times m$ matrix where each element $a_{ij}^\top$ of $A^\top$ is equal to a_{ji} of A.

Example

Consider the matrix

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

The transpose of A, $A^\top$, is obtained by swapping the rows and columns of A:

$$A^\top = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

Consider the matrix

$$B = \begin{pmatrix} 5 & 6 & 7 \\ 8 & 9 & 10 \end{pmatrix}$$

The transpose of B, $B^\top$, is:

$$B^\top = \begin{pmatrix} 5 & 8 \\ 6 & 9 \\ 7 & 10 \end{pmatrix}$$

Theorem 2
$(AB)^\top = B^\top A^\top$

Proof: Let A be an $m \times n$ matrix and B be an $n \times p$ matrix. Then, AB is an $m \times p$ matrix. The ij-th element of AB is given by the dot product of the i-th row of A and the j-th column of B. That is,

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

When we take the transpose of AB, the ij-th element of $(AB)^\top$ is the ji-th element of AB, which is:

$$(AB)_{ij}^\top = (AB)_{ji} = \sum_{k=1}^{n} a_{jk} b_{ki}$$

On the other hand, the ij-th element of $B^\top A^\top$ is given by the dot product of the i-th row of $B^\top$ and the j-th column of $A^\top$, which is:

$$(B^\top A^\top)_{ij} = \sum_{k=1}^{n} b_{ik}^\top a_{kj}^\top = \sum_{k=1}^{n} b_{ki} a_{jk}$$

Since the two expressions are equal, we have proven that

$$(AB)^\top = B^\top A^\top$$

Definition 6 Symmetric matrix
A **symmetric matrix** is a square matrix A that is equal to its transpose, i.e., $A = A^\top$. In other words, for every i, j, the elements of A satisfy $a_{ij} = a_{ji}$.

6.1.1 Eigenpairs

Definition 7 Eigenvector and Eigenvalue
An **eigenvector** of a square matrix A is a nonzero vector v such that when A is multiplied by v, the resultant vector is a scalar multiple of v. This scalar is known as an **eigenvalue** corresponding to the eigenvector v. Formally, this relationship is defined by the equation

$$A\mathbf{v} = \lambda\mathbf{v}$$

where A is the matrix, $\mathbf{v}$ is the eigenvector, and λ is the eigenvalue.

Example For the identity matrix scaled by 3, $3I$, every nonzero vector is an eigenvector with an eigenvalue of 3. This is because multiplying any vector by the identity matrix leaves the vector unchanged, and scaling the identity matrix by 3 simply scales the vector by 3. Formally, if

$$A = 3I = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$$

and $\mathbf{v} = \begin{pmatrix} x \\ y \end{pmatrix}$ is any nonzero vector, then

$$A\mathbf{v} = 3\mathbf{v}$$

showing that every nonzero vector is an eigenvector of $3I$ with an eigenvalue of 3.

Example Consider a diagonal matrix

$$D = \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix}$$

The eigenvectors are the standard basis vectors, and the eigenvalues are the diagonal elements d_1 and d_2. Specifically, the eigenvector corresponding to d_1 is

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

with eigenvalue d_1, and the eigenvector corresponding to d_2 is

$$\mathbf{v}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

with eigenvalue d_2.

Proposition 3
If $\mathbf{v}$ is an eigenvector of A associated with eigenvalue λ, and c is a nonzero constant, then $c\mathbf{v}$ is also an eigenvector of A associated with the same eigenvalue λ.

PROOF

$$A(c\mathbf{v}) = cA\mathbf{v} = c\lambda\mathbf{v} = \lambda(c\mathbf{v})$$

demonstrating that $c\mathbf{v}$ satisfies the eigenvector-eigenvalue equation for A and λ. □

Example Consider the matrix

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

The eigenvalues of A are $\lambda_1 = 3$ and $\lambda_2 = 1$, with corresponding eigenvectors

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

for λ_1, and

$$\mathbf{v}_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

for λ_2.

Example Consider the matrix

$$B = \begin{pmatrix} 4 & 1 \\ 0 & 3 \end{pmatrix}$$

The eigenvalues of B are $\lambda_1 = 4$ and $\lambda_2 = 3$, with corresponding eigenvectors

$$\mathbf{v}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

for λ_1, and

$$\mathbf{v}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

for λ_2.

A good visualization of eigenvectors can be found in https://www.youtube.com/watch?v=fNk_zzaMoSs

6.1.2 Invertible matrices as change of basis

In this section, we aim at giving the geometric intuition of the matrix multiplication. We first recap what is a basis and then use this notion to describe matrices as linear transformation of the canonical basis.

Definition 8 basis
A **basis** of a vector space is a set of vectors in that space that is both linearly independent and spans the entire vector space. In other words, a basis is a set of vectors where every vector in the space can be written as a unique linear combination of the basis vectors.

The **canonical basis** (or standard basis) for $\mathbb{R}^n$ consists of vectors where each vector has a 1 in one position and 0 in all other positions. For example, in $\mathbb{R}^2$, the canonical basis is given by

$$\mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \mathbf{e}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Exercise Show that the canonical basis is a basis.

Answer The canonical basis vectors are linearly independent, meaning that no vector in the set can be written as a linear combination of the others. This is because each vector has a 1 in a unique position which cannot be generated by any combination of the other vectors.
Furthermore, the canonical basis spans $\mathbb{R}^n$. Any vector $\mathbf{v} \in \mathbb{R}^n$ can be expressed as a linear combination of the canonical basis vectors. For a vector

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

we can write

$$\mathbf{v} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2 + \cdots + v_n\mathbf{e}_n$$

where each v_i is a scalar coefficient corresponding to the i-th component of $\mathbf{v}$. □

Exercise Consider the vectors $\mathbf{a} = (0, 2)$ and $\mathbf{b} = (2, 1)$. Show that these vectors form a basis for $\mathbb{R}^2$.

Answer To prove that the vectors $\mathbf{a}$ and $\mathbf{b}$ are linearly independent, we must show that the only solution to the equation

$$c_1\mathbf{a} + c_2\mathbf{b} = \mathbf{0}$$

is $c_1 = c_2 = 0$, where $\mathbf{0}$ is the zero vector.

Substituting the given vectors into the equation, we get

$$c_1(0,2) + c_2(2,1) = (0,0)$$

which leads to the system of equations:

$$\begin{aligned} 0c_1 + 2c_2 &= 0 \\ 2c_1 + 1c_2 &= 0 \end{aligned}$$

Solving this system, we find that the only solution is $c_1 = 0$ and $c_2 = 0$, proving that $\mathbf{a}$ and $\mathbf{b}$ are linearly independent.
To show that $\mathbf{a}$ and $\mathbf{b}$ span $\mathbb{R}^2$, we need to prove that any vector $\mathbf{v} = (x, y) \in \mathbb{R}^2$ can be expressed as a linear combination of $\mathbf{a}$ and $\mathbf{b}$.
That is, we need to find scalars c_1 and c_2 such that

$$c_1\mathbf{a} + c_2\mathbf{b} = \mathbf{v}$$

$$c_1(0,2) + c_2(2,1) = (x, y)$$

This equation translates to the system:

$$\begin{aligned} 2c_2 &= x \\ 2c_1 + c_2 &= y \end{aligned}$$

Given any $x, y \in \mathbb{R}$, this system has a solution for c_1 and c_2, indicating that any vector $\mathbf{v}$ in $\mathbb{R}^2$ can indeed be represented as a combination of $\mathbf{a}$ and $\mathbf{b}$.

□

When a matrix A is multiplied by one of the canonical vectors, the result is one of the columns of A. Specifically, multiplying A by $\mathbf{e}_i$ (where i is the index of the canonical vector) yields the i-th column of A. This can be formally written as:

$$A\mathbf{e}_i = \text{the } i\text{-th column of } A$$

This property illustrates how each column of a matrix A represents the image of the corresponding canonical basis vector under the linear transformation defined by A. In other words, the columns of A can be viewed as specifying the coordinates of the transformed canonical basis vectors in the new basis defined by the matrix A.

For example, consider a 2×2 matrix A and a vector $\mathbf{v}$ in $\mathbb{R}^2$:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = v_1\mathbf{e}_1 + v_2\mathbf{e}_2$$

6.2 Spectral Theorem

6.2.1 Theorem

Let's introduce first the notion of orthognoal matrices and symmetric matrices to be able to formulate the spectral theorem.

Definition 9
The **Euclidean norm** of a vector in $\mathbb{R}^n$, denoted by $\|\mathbf{v}\|$, is a measure of the vector's length (or magnitude) in the n-dimensional Euclidean space. For a vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, the Euclidean norm is defined as:

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

This norm is derived from the Euclidean distance formula, which calculates the distance between the origin of $\mathbb{R}^n$ and the point represented by $\mathbf{v}$.

Definition 10
An **orthonormal matrix** is a square matrix A whose columns are orthonormal vectors, meaning that their product scalar are zero if different and one otherwise.

Orthogonal matrices have several important properties:

Preservation of the Euclidean Norm For any vector $\mathbf{v}$ in $\mathbb{R}^n$, an orthogonal matrix A preserves the Euclidean norm of $\mathbf{v}$ upon transformation, i.e., $\|A\mathbf{v}\| = \|\mathbf{v}\|$. In particular, a unit vector is still a unit vector after transformation

Inverse Equals Transpose The inverse of an orthogonal matrix is equal to its transpose, i.e., $A^{-1} = A^\top$. This makes orthogonal matrices particularly easy to work with in computations.

Orthogonality of Eigenvectors The eigenvectors of an orthogonal matrix corresponding to different eigenvalues are orthogonal to each other.

With this defintion, we are ready to introduce the spectral theorem, a very important linear algebra that is useful to define PCA

Theorem 3 Spectral theorem
Every symmetric matrix A with real entries can be diagonalized by an orthogonal matrix Q. In other words, there exists an orthogonal matrix P and a diagonal matrix D such that $A = PDP^\top$. Furthermore, the diagonal elements of D are the real eigenvalues of A, and the columns of P are the orthonormal eigenvectors of A.

6.3 Empirical covariance matrix

Because we will have a probabilistic interpretation of the matrix, let's recap the concept of covariance.

Definition 11 Covariance
The **covariance** of two random variables A and B, denoted by $\text{Cov}(A, B)$, measures the degree to which the two variables vary together. It is defined as the expected value of the product of their deviations from their respective means:

$$\text{Cov}(A, B) = \mathbb{E}[(A - \mathbb{E}[A])(B - \mathbb{E}[B])]$$

Given n samples of the random variables A and B, the **empirical covariance** is calculated by:

$$\text{Cov}(A, B) = \frac{1}{n}\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})$$

where $\bar{A}$ and $\bar{B}$ are the sample means of A and B, respectively. Note that if the random variables are centered, there is no need to substract the mean,so the formula is simplified. If morever we define a vector $\mathbf{A}$ containing the n samples of A, the computation of the covariance boilds down $\text{Cov}(A, B) = \frac{1}{n}\mathbf{A}^\top\mathbf{B}$.

Definition 12 Covariance matrix
For a set of d random variables, the **covariance matrix**, denoted by Σ, is a $d \times d$ matrix where each element Σ_{ij} represents the covariance between the i-th and j-th variables. Formally,

$$\Sigma_{ij} = \text{Cov}(X_i, X_j)$$

Note that, by definition, the covariance matrix is symmetric. To have an unbiased estimator, one should, in fact, divide by $n - 1$ and not by n; this adjustment is known as the Bessel correction. The idea behind it is that because the mean is also empirical, the formula tends to underestimate the true variance. In this lecture, this coefficient has little importance (in particular, the reasoning often holds for any matrix proportional to the true empirical matrix), so we keep the n factor for simplicity.

The covariance is not bounded, you should not confuse it with the correlation between two variables. The mathematical relationship between covariance and correlation is formalized through the Pearson correlation coefficient, denoted as ρ, between two variables A and B. The Pearson correlation coefficient is defined as:

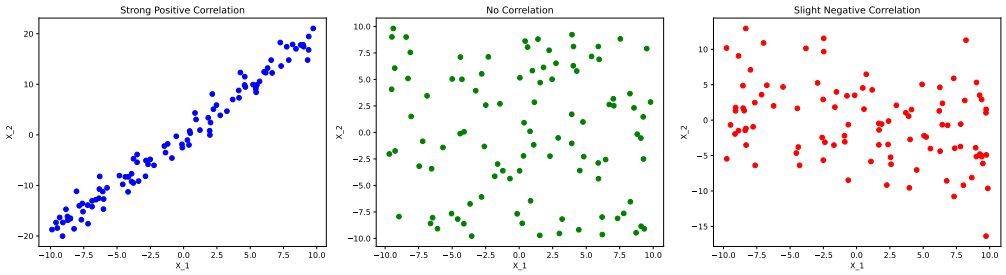


Figure 9. Strong Positive Correlation: The left plot shows a dataset where the variables have a strong positive linear relationship. This means that as X_1 increases, X_2 also increases in a predictable manner. The correlation coefficient for such a relationship is close to 1. No Correlation (Uncorrelated Variables): The middle plot displays a scenario where there's no apparent relationship between the variables, resulting in a correlation coefficient near 0. Slight Negative Correlation: The right plot illustrates a slight negative correlation between the variable, corresponding the a negative correlation, but not reaching -1.

$$\rho(A, B) = \frac{\text{Cov}(A, B)}{\sigma_A \sigma_B}$$

where σ_A and σ_B are the standard deviations of A and B, respectively. This formula effectively standardizes the covariance by the product of the standard deviations of the two variables, resulting in a dimensionless coefficient that ranges from $-1$to 1. A value of 1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship, and 0 implies no linear relationship between A and B. This standardized measure, unlike covariance, provides a scale-independent assessment of the strength and direction of a linear relationship between two variables.

Here is example of samples with various correlation type between variables fig. 9.

6.4 Singular Value Decomposition

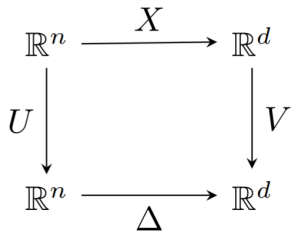


Figure 10. Schema of Singular Value Decomposition. for example, for the matrix $X = \begin{pmatrix} 3 & 2 & 7 \\ 2 & 0 & 0 \\ 4 & 5 & 5 \\ 6 & 8 & 4 \\ 1 & 4 & 9 \end{pmatrix}$ we would have $U = \begin{pmatrix} -0.42 & -0.37 & 0.64 & 0.09 & -0.51 \\ -0.05 & 0.17 & 0.61 & 0.26 & 0.73 \\ -0.47 & 0.17 & 0.06 & -0.85 & 0.19 \\ -0.57 & 0.67 & -0.19 & 0.39 & -0.17 \\ -0.52 & -0.60 & -0.42 & 0.22 & 0.37 \end{pmatrix}$, and $V = \begin{pmatrix} -0.41 & -0.57 & -0.71 \\ 0.53 & 0.49 & -0.70 \\ 0.74 & -0.66 & 0.10 \end{pmatrix}$ and $\begin{pmatrix} 17.31 & 0 & 0 \\ 0 & 6.35 & 0 \\ 0 & 0 & 2.45 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Definition 13 Singular Value Decomposition
The *Singular Value Decomposition* of a matrix $X \in \mathbb{R}^{n \times d}$ is a factorization of the form $X = U\Delta V^\top$, where:

- $U \in \mathbb{R}^{n \times n}$ is an orthogonal matrix whose columns are the eigenvectors of $XX^\top$.
- $V \in \mathbb{R}^{d \times d}$ is an orthogonal matrix whose columns are the eigenvectors of $X^\top X$.
- $\Delta \in \mathbb{R}^{n \times d}$ is a diagonal matrix with non-negative real numbers on the diagonal, known as the singular values of X, and zeros elsewhere.

The singular value decomposition provides a geometric interpretation of a matrix in terms of rotation, scaling, and another rotation. The matrices U and V represent rotations or reflections, while the matrix Δ represents scaling along the principal axes defined by U and V. This decomposition is particularly insightful for understanding the effect of X on the Euclidean space it acts upon, revealing how X stretches vectors (by Δ) and how it rotates or reflects these vectors (by U and V), as illustrated in fig. 10

Definition 14 Singular value
The *Singular values* of a matrix X are the diagonal entries of the matrix Δ in the singular value decomposition $X = U\Delta V^\top$.

Proposition 4
Let $X \in \mathbb{R}^{n \times d}$ be a matrix. The singular values of X are the square roots of the eigenvalues of the matrix $X^T X$. When noting $X = U\Delta V^\top$, the matrix U contains the eigenvectors of the matrix $XX^\top$ and V contains the eigenvectors of the matrix $X^\top X$.

Proof | Given the singular value decomposition $X = U\Delta V^T$, we can express $X^T X$ as follows:

$$X^T X = (U\Delta V^T)^T (U\Delta V^T) = V\Delta^T U^T U \Delta V^T.$$

Since U and V are orthogonal matrices, $U^T U = I$ and $V^T V = I$, where I denotes the identity matrix. Thus, we simplify the above expression to:

$$X^T X = V\Delta^2 V^T.$$

We recognize the decomposition given by the spectral theorem. In particular, this shows that $X^T X$ is similar to Δ^2, so they have the same eigenvalues. The eigenvalues of Δ^2 are the squares of the singular values of X, as Δ contains the singular values of X on its diagonal. Similarly, V is composed of the eigenvectors of $X^T X$. Similar computation showd the result for U □

Exercise | Given a matrix $X \in \mathbb{R}^{m \times n}$ and its singular value decomposition $X = U\Delta V^T$, demonstrate that each left singular vector u_i can be expressed in terms of the right singular vector v_i and the corresponding eigenvalue λ_i of $X^T X$ as $u_i = \frac{Xv_i}{\sqrt{\lambda_i}}$.

Answer | By multiplying by V, we have $XV = U\Delta$. By checking the i-th column, we have $XV_i = \sigma_i U_i$. the result follows from $\sigma_i = \sqrt{\lambda_i}$. □

References

[1] Bishop, C. M. (2007). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition.

[2] Dasgupta, S. and Gupta, A. (2003). An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65.

[3] Jean-Philippe Vert, Koji Tsuda, B. S. (2004). A primer on kernel methods.

[4] Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27.

[5] Narayanan, A. and Shmatikov, V. (2007). How to break anonymity of the netflix prize dataset.

[6] Rigollet, P. P. (2016). Statistics for applications.

[7] Schölkopf, B., Smola, A., and Müller, K.-R. (2006). *Kernel Principal Component Analysis*, volume 1327, pages 583–588.