Unsupervised Learning

Sara Candussio

July 2023

1 Principal Component Analysis

This is the first Manifold Learning technique that we're going to see. It was suggested by Pearson in 1901 and the original paper has more than 15.000 citations; it's old, but still very useful.

It has many goals:

1. find the *projection* (or representation) *y of the data* in which the correlations between different variables (or features) is minimized;
2. (more related to Manifold Learning) learn the *dimension d* in which projecting the data and/or find the *function $f(\cdot)$* that does so, i.e. $f(X) = y$, where $X \in \mathbb{R}^D$ are the original features and $y \in \mathbb{R}^d$ are the projected ones. We want that $d << D$.

How do we do that?

Step 1: center the data
The first thing that we want to do is to *center (on zero) our data*, because the derivation with centered data is much easier and we don't lose any generality in doing this:

$$x_i^k = \tilde{x_i}^k - \mu = \tilde{x_i}^k - \frac{1}{N}\sum_{k=1}^{N} \tilde{x_i}^k$$

Step 2: covariance matrix computation
The aim of this step is to *understand how the variables of the input data set are varying from the mean with respect to each other*, or in other words, to see if there is any relationship between them.
We study this quantity because variables can be highly correlated in such a way that they contain redundant information. *If the information is redundant, we are "allowed" to get rid of some less useful variables because the remaining ones carry their same message.*
In order to identify these correlations, we compute the covariance matrix.

The covariance matrix is generally expressed in a form like the following one:

$$\begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{pmatrix} \text{ where } Cov(x,y) = \frac{1}{N}\sum_i (x_i - \mu_x)\cdot(y_i - \mu_y)$$

Since our goal (look at **Step 1**) is *to minimize the correlation between different features in the projected space*, we want to find a new set of coordinates (y_1, y_2) in which this set of points is uncorrelated.
This, mathematically speaking, is equivalent to ask that the covariance matrix of the new set of coordinates y_i is a diagonal matrix.
This is true because, statistically speaking, $Cov(x, x) = Var(x)$. Thus, all the correlations between different variables are asked to be null, except from the correlation of a variable within itself. The variance shows how much a certain variable is different from its mean, i.e. how much improper is its representation using its mean due to its variability. If we are considering a space on variables in which each variable carries a portion of the overall noise which is independent from the one carried by the other features, we are in front of a diagonal covariance matrix.

Given that C_{mn} is the element in position (m, n) of the covariance matrix of the original features x_i, we have that:

$$C_{mn} = Cov\{x_i\} = \frac{1}{N} \sum_i x_m^i x_n^i$$

(since data are centered as said in **Step 1**, $\mu_m = \mu_n = 0$).

The covariance of the starting set of coordinates can be expressed as $C = (C_{mn})_{m,n}$, and it is symmetric and positive semi-definite by construction.

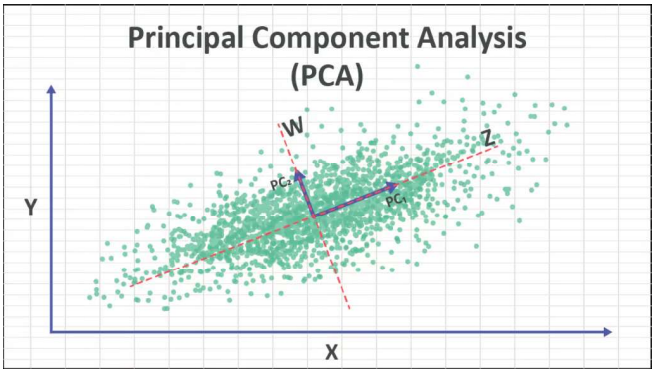
Step 3: eigendecomposition of the covariance matrix
The aim of this step is to construct linear combinations or mixtures of the initial features $x \in \mathbb{R}^D$, called principal components, such that the first one represents the main direction of the data that explains the maximum amount of variance, the second one represents the second direction of the data that explains the maximum amount of variance but with the extra condition of being orthogonal to the first component, and so on.

Note that asking the direction of the principal components (i.e. the projected features) to be orthogonal is the same of asking the features to be uncorrelated. In this way we can get rid of the last principal components, because approximately the most of the information within the initial variables is squeezed or compressed into the first components.

The idea is 10-dimensional data gives you 10 principal components, but PCA tries to put maximum possible information (within the initial variables) in the first component, then maximum remaining information in the second and so on.

As it is easy to imagine, principal components don't have any real meaning since they are built as linear combination of other variables, hence they are not easy to interpret.

Geometrically speaking, principal components represent the directions of the data that explain a maximum amount of variance, that is to say, *the lines that capture most information of the data.* The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more information it has.



The *eigenvectors* of the covariance matrix are actually the *directions* of the axes where there is the most variance (most information) and that we referred to as principal components. The *eigenvalues* are simply the *coefficients* attached to eigenvectors, which give *the amount of variance carried by each principal component.*
By ranking your eigenvectors in order of their eigenvalues, highest to lowest, you get the principal components in order of significance.

Step 4: feature vector
The feature vector U is simply *a matrix that has as columns the eigenvectors of the components that we decide to keep* if we decrease the number of dimensions to d.
This makes it the first step towards dimensionality reduction, because if we choose to keep only d eigenvectors (components) out of D, the final dataset will have only d dimensions.

Step 5: new data set
In this last step *we use the feature vector to transform our data onto the new feature subspace.* We do so by dot product between the feature vector U and the initial data set X, after centering it (in other words, we multiply the mean-adjusted data set and the matrix of eigenvectors):

$$Y = U^T \cdot X \Rightarrow y_l^i = \sum_{j=1}^{N} U_{lj} \cdot x_j^i$$

where Y is a $N \times d$ matrix of the new data set, X is a $N \times D$ matrix of the initial data set, and U is a $D \times d$ matrix of the feature vector, where d is the number of dimensions of the new feature subspace (which is smaller than the original space), and D is the number of dimensions of the original data set.

1.1 Example of "done by hand" PCA:

For example, let's assume that we start from the following covariance matrix:

$$C = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1.09 \end{pmatrix}$$

in order to find the new covariance matrix, we need to diagonalize C into λ, i.e. solve the eigenvalue-eigenvector problem:

$$\lambda U = CU \iff \lambda = U^{-1}CU \quad where\ \lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

The solution may come from the Jacobi method, where we find U as the *rotation matrix*:

$$U = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}$$

What can be shown is that if we want to diagonalize C, we can take:

$$\tan(2\theta) = \frac{C_{12}}{C_{22} - C_{11}}$$

what we have is that $\tan(2\theta) = 20$, so $\theta \approx 0.7604$, hence:

$$U = \begin{pmatrix} 0.72 & 0.69 \\ -0.69 & 0.72 \end{pmatrix}$$

So now we have everything that we needed to find λ:

$$\lambda = U^{-1}CU = \begin{pmatrix} 1.946 & 0 \\ 0 & 0.144 \end{pmatrix}$$

But again, how do we find the representation y_i?

$$y_i = Ux_i \iff \begin{pmatrix} y_1^i \\ y_2^i \end{pmatrix} = \begin{pmatrix} 0.72 & 0.69 \\ -0.69 & 0.72 \end{pmatrix} \cdot \begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix} = \begin{pmatrix} 0.72 & 0.69 \\ -0.69 & 0.72 \end{pmatrix}$$

which is equivalent to:

$$\begin{cases} y_1^i = 0.72 \cdot x_1^i + 0.69 \cdot x_2^i \\ y_2^i = -0.69 \cdot x_1^i + 0.72 \cdot x_2^i \end{cases}$$

Note: starting from the coordinates (x_1, x_2), what we did in order to obtain the new uncorrelated coordinates (y_1, y_2), was rotating the space according to the matrix U.

1.2 Small review of Lagrange multipliers

When we have an optimization problem with constraints, for example:

$$f(x,y) = 2x + 2y \; subject \; to \; x^2 + y^2 = 1$$

then we can rephrase the constraint as $g(x,y) = x^2 + y^2 - 1 = 0$.
Thanks to this, now the previous problem finds an equivalent formulation in the following one:

$$F(x,y,\lambda) = f(x,y) - \lambda g(x,y) = 2x + 2y - \lambda(x^2 + y^2 - 1)$$

and we want that:

$$\frac{\partial F}{\partial \lambda} = x^2 + y^2 - 1 = 0 \qquad \frac{\partial F}{\partial x} = 2 - 2\lambda x = 0 \qquad \frac{\partial F}{\partial y} = 2 - 2\lambda y = 0$$

that leads us to

$$x = \pm\frac{1}{\sqrt{2}} \qquad y = \pm\frac{1}{\sqrt{2}} \qquad \lambda = \sqrt{2}$$

1.3 A mathematical view of PCA

Goal: maximize $Tr(Cov\{y_i\}) = \frac{1}{N}\sum_{k=1}^{d}\sum_{i=1}^{N} y_k^i \cdot y_k^i$.

Important note: when we take the trace of a covariance matrix, we're just considering the elements in the diagonal, i.e. the variances. In order to obtain the optimal linear transformation of data, we need to obtain the coordinates in which the variance is max, so we have found our optimization problem.

What misses with respect to the Lagrange multipliers formulation is a constraint: we ask for the vectors of the transformation matrix U to be *normal*.

Given $U = \begin{pmatrix} u_{11} & ... & u_{1D} \\ ... & ... & ... \\ u_{d1} & ... & u_{dD} \end{pmatrix}$ *where* $U \in \mathbb{R}^{d \times D}$

We ask for

$$\sum_l u_{nl}^2 = 1$$

The sum of the rows squared *must be equal to* 1 because in this way we *avoid the arbitrarily trivial solutions* of multiplying by a huge number all the coordinates.

So

$$Tr(Cov\{y_i\}) = \frac{1}{N}\sum_{k=1}^{d}\sum_{i=1}^{N} y_k^i \cdot y_k^i$$

due to the fact that $y_k^i = \sum_{m=1}^{D} u_{km} \cdot x_m^i$, we have that:

$$Tr(Cov\{y_i\}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{D} \sum_{m=1}^{D} \sum_{k=1}^{d} u_{kl} \cdot x_l^i \cdot u_{km} \cdot x_m^i$$

and due to the fact that $C_{ml} = \frac{1}{N} \sum_{i=1}^{N} x_m^i \cdot x_l^i$, we can rewrite it as:

$$Tr(Cov\{y_i\}) = \frac{1}{N} \sum_{l=1}^{D} \sum_{m=1}^{D} \sum_{k=1}^{d} u_{kl} \cdot u_{km} \cdot C_{ml}$$

finally we have:

$$F = Tr(Cov\{y_i\}) - \sum_k \lambda_k \cdot \left(\sum_n u_{kn}^2 - 1 \right) = \frac{1}{N} \sum_{l=1}^{D} \sum_{m=1}^{D} \sum_{k=1}^{d} u_{kl} \cdot u_{km} \cdot C_{ml} - \sum_k \lambda_k \cdot \left(\sum_n u_{kn}^2 - 1 \right)$$

this entity is called *PCA loss* and it's what we maximize when we perform PCA, since it's equivalent to $Tr(Cov\{y_i\})$.

How do we maximize it? It's simple:

$$\frac{\partial F}{\partial \lambda_k} = 0 \; and \; \frac{\partial F}{\partial u_{kh}} = 0 \quad where \; \frac{\partial F}{\partial \lambda_k} = \left(\sum_n u_{kn}^2 \right) - 1 = 0$$

so we recover the already known constraint, and

$$\frac{\partial F}{\partial u_{kh}} = \sum_{k=1}^{d} \frac{\partial}{\partial u_{kh}} \left[\sum_{l=1}^{D} \sum_{m=1}^{D} u_{kl} \cdot u_{km} \cdot C_{ml} \right] - \sum_{k=1}^{d} \frac{\partial}{\partial u_{kh}} \lambda_k \cdot \left(\sum_n u_{kn}^2 - 1 \right) = 0$$

Let's consider the first term:

$$\frac{\partial}{\partial u_{kh}} \left[\sum_{l=1}^{D} \sum_{m=1}^{D} u_{kl} \cdot u_{km} \cdot C_{ml} \right]$$

we are going to consider different cases of it.

- if $l \neq h$ and $m \neq h$, it's 0 because we're differentiating constants with respect to u_{kh};
- if $l = h$ but $m \neq h$, it's $\sum_{m \neq h} u_{km} \cdot C_{mh}$ because the contribution will be 1 if $l = h$, 0 otherwise;
- if $l \neq h$ but $m = h$, it's $\sum_{l \neq h} u_{kl} \cdot C_{lh}$ (it's just the symmetric case);
- if $l = h$ and $m = h$, the whole term is

$$\frac{\partial}{\partial u_{kh}} \left[\sum_{l=1}^{D} \sum_{m=1}^{D} u_{kh}^2 \cdot C_{hh} \right] = 2 u_{kh} \cdot C_{hh}$$

Hence:

$$\frac{\partial}{\partial u_{kh}}\left[\sum_{l=1}^{D}\sum_{m=1}^{D} u_{kl}\cdot u_{km}\cdot C_{ml}\right] = \sum_{m\neq h} u_{km}\cdot C_{mh} + \sum_{l\neq h} u_{kl}\cdot C_{lh} + 2u_{kh}\cdot C_{hh} = 0$$

So we have that:

$$\sum_{m=1}^{D} u_{km}\cdot C_{mh} + \sum_{l=1}^{D} u_{kl}\cdot C_{lh} = 2\cdot\sum_{l=1}^{D} u_{kl}\cdot C_{lh}$$

While the second term can be more easily computed:

$$\frac{\partial}{\partial u_{kh}}\left[\lambda_k\cdot\left(\sum_n u_{kn}^2 - 1\right)\right] = 2\cdot u_{kn}\cdot\lambda_k = 0$$

So we can write the complete derivative as:

$$\frac{\partial F}{\partial u_{kh}} = \sum_{k=1}^{d} 2\cdot\sum_{l=1}^{D} u_{kl}\cdot C_{lh} - 2\cdot\sum_{k=1}^{d}\lambda_k\cdot u_{kh} = 0 \Rightarrow$$

$$\Rightarrow \sum_{k=1}^{d}\sum_{l=1}^{D} u_{kl}\cdot C_{lh} = \sum_{k=1}^{d}\lambda_k\cdot u_{kh} \Rightarrow \sum_{k=1}^{d}(CU)_{kh} = \sum_{k=1}^{d}\lambda_k\cdot u_{kh}$$

this is equivalent to the eigenvalue-eigenvector problem $CU = \lambda U$, so in order to solve the PCA we have to diagonalize!

This tells us that *if we want to maximize the trace of the covariance matrix of the projected data CU we have to choose the d highest eigenvectors of the covariance matrix of the original data C.* This is not mathematically proven, but it's quite intuitive.

What we have after the diagonalization is the *rotation and/or compression invertible matrix U*:

$$U = \begin{pmatrix} u_{11} & ... & u_{1D} \\ ... & ... & ... \\ u_{d1} & ... & u_{dD} \end{pmatrix}$$

and for each row vector $u_i = (u_{i1}, ..., u_{iD})$ we find the corresponding eigenvalue λ_i. The eigenvalues satisfy the following property: $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d$.

Since the eigenvectors come from a diagonalization, we surely know that they are orthogonal and normal (because we have imposed it through the constraint of Lagrange multipliers), thus they are orthonormal. This implies that $u = (u_1, ..., u_d)$ is a basis of $\mathbb{R}^d$.

1.4 PCA algorithm

1. Center the data:
$$x_i^k = x_i^k - \frac{1}{N}\sum_{k=1}^{N} x_i^k$$
2. Compute the covariance matrix
$$C = \frac{1}{N}X^T X \in \mathbb{R}^{D\times D} \Rightarrow C_{ab} = \frac{1}{N}\sum_i x_a^i x_b^i$$
3. Find the eigenvalues and the eigenvectors of the covariance matrix C
4. Decide the projection dimension d
5. Sort the eigenvectors by decreasing eigenvalues and choose d eigenvectors with the largest eigenvalues to form a $D \times d$ dimensional matrix U (where every column represents an eigenvector)
6. Project data using $y_l^i = \sum_j U_{lj} \cdot x_j^i$ (i.e. $Y = U^T X$)

1.5 How to decide the projection dimension d

The first method can be intuitively explained as follows. Let's suppose to project the entire dataset in only one dimension, i.e. $y \in \mathbb{R}$. *We're projecting the dataset only with respect with the most important feature of it, discarding all the rest.* The portion of covariance of the initial dataset that this projection is able to explain is:
$$\frac{\lambda_i}{\sum_{i=1}^{D}\lambda_i}$$
in the previous example of this chapter, we have that:
$$\frac{\lambda_1}{\lambda_1+\lambda_2} = \frac{1.946}{1.946+0.144} = 93\%$$
i.e. the 93% of the covariance of the data is explained by the first component.

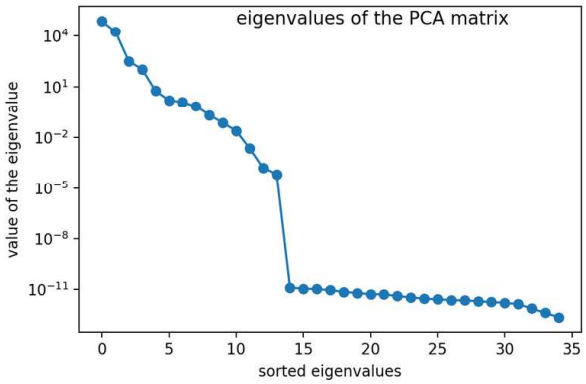
The first method consists in computing the *fidelity of the PCA*:
$$\frac{Tr(Cov\{y_i\})}{Tr(Cov\{x_i\})} = \frac{\sum_{i=1}^{d}\lambda_i}{\sum_{i=1}^{D}\lambda_i}$$
where d is the projected space dimension.

So now what we can do is try to determine d, i.e. the number of dimension in which we want to project data, in order to capture a certain percentage of the original covariance in the data (something that we choose):

$$\frac{Tr(Cov\{y_i\})}{Tr(Cov\{x_i\})} = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{D} \lambda_i} \geq \tau$$

where $\tau = (0.9, 0.95, 0.99, ...)$

The second method consists in plotting the *spectrum of the eigenvalues* and spot the gaps/elbows:



After the gap/elbow, almost all the dimensions have the same covariance (the plot goes to zero): even if we add one dimension, we do not obtain so much. In this way we're able to spot the minimum number of necessary dimensions.

1.6 Recap of the method

How it is the covariance matrix of the projected data? Diagonal.
Why is it diagonal? Because the projected data is uncorrelated.
Why we want it to be diagonal in first istance? We need a rotation and/or compression of the space that reduces (cancels) the covariance between the features.

To achieve this we try to diagonalize the covariance matrix C. The matrix U resulting from this process of diagonalization, provides us the desired rotation and/or compression matrix.
The idea is to have strongly correlated data arranged in a linear manifold. The goal was to find by diagonalizing the covariance matrix a new representation of the data, in which we the highest variance coordinate is the first one and so on.

If we choose to keep only the first d eigenvectors, we can reduce the dimensionality of the data from D to d and so we have both a rotation and a projection of the data. If instead the projected space is D-dimensional, the matrix U is just a rotation matrix.

1.7 PCA problems

The main problem is that when we do PCA, we are performing a linear transformation. This means that *if we're dealing with a non-linear manifold space, PCA is not useful to do a dimensionality reduction of the dataset*, but can still be important for other purposes.

The first problem derived from the fact that PCA is not suitable for non-linear transformation is that it is *not able to capture invariances*. Imagine to have a dataset in which we have densities, weights and volumes. Density is obviously just the ratio of the other two, but PCA will not be able to spot this.
The second derived problem is that PCA is *not scale-invariant*: a measure expressed in kilograms or in milligrams, the weight on the principal components vary a lot.