**Principal Component Analysis (PCA)**

This is a report to introduce Principal Component Analysis (PCA), a powerful tool in statistical learning to reduce the dimensionality of a dataset and visualize the data cloud in a comprehensive way. It is provided the mathematical background together with figures to visualize each step, and the application of PCA to our High Entropy Alloys images.

**Assumption:**

* In Machine Learning, the dimensionality of a dataset is given by the number of variables (features) characterizing the dataset. Technically speaking, the dimensionality is given by the number of columns of a dataset. When the dataset is composed by images rather than numerical data, we have to think to stretch an x image with pixels to a 1D array with entries. In this case, the dimensionality of the dataset is . **Figure 1** and **Figure 2** provide examples of a numerical dataset and an images dataset to visualize the dimensionality.

**Table

Description automatically generated**

**Figure 1.** This dataset is characterized by 14 variables (14 columns). Thus, the dimensionality of this dataset is .

**A picture containing graphical user interface

Description automatically generated**

**Figure 2.** A 256x256 image with 65536 pixels can be represented as a 1D array with 65536 entries. Considering images, the dataset is x65536 (65536 columns) and the dimensionality is .

* PCA is a dimensionality reduction technique, meaning that it allows to reduce a dataset of dimensionality in a dataset with dimensionality which keeps almost the same variance of the original dataset (the amount of retained variance depends on the number of Principal Components *m*, which varies from case to case).

**Graphical user interface, application, table, Teams

Description automatically generated**

**Figure 3.** A dataset with dimensionality is reduced to a dataset with dimensionality which keeps almost the same variance of the original dataset.

**General Motivations:**

* Summarize a large set of variables in a dataset with a smaller set of representative variables (i.e., Principal Components) that collectively explain most of the variability of the original set.
* Project the original large set of variables to directions (i.e., the Principal Components) along which the original data are *highly variable*.
* Use the smaller set of the Principal Components as predictors (i.e., variables) to fit a Machine Learning model in place of the original larger set of predictors. Technically speaking, the Machine Learning model is fitted with a dataset with a reduced number of columns to make the computation faster and also more efficient from a statistical point of view, because the variance of the variables is concentrated in variables, which are for this reason more meaningful from a statistical point of view. With PCA, the variance is the most important statistical variable. The mean is not considered because in many Machine Learning applications and in particular in PCA, the data are scaled using a Gaussian scaler, thus the mean of each variable is zero.
* Tool for the **visualization of the data cloud**. Suppose that we wish to visualize observations on a set of features as part of an explorative data analysis. We could do this by examining two-dimensional scatterplots of the data, each of which contains the observations on two of the features. However, there are scatterplots (for example, for there are 45 scatterplots. If is large it is not possible to look for all of them. In particular, most likely none of them will be informative since each variable contains just a small fraction of the total information present in the dataset. Thus, for the visualization purpose we aim to obtain a two-dimensional representation of the data that captures most of the information. PCA is a tool for doing this.
* Data that are close to each other in a scatterplot of the Principal Component system are data which have similar values of the features in the original system of the all and many variables. Using PCA it is possible to identify similar data analyzing the smaller set of the Principal Components rather than considering the larger set of the original variables.

**Figure 4** provides an example of how PCA can be used for the visualization of a data cloud. Each point in the data cloud represents an instance of the dataset (technically speaking a row of the dataset) with index 1,…,, and the horizontal and vertical coordinates are the values of first and second component for the instance (technically speaking columns 1 and 2 of the dataset transformed to the Principal Component system)

Chart

Description automatically generated

**Figure 4.** PCA as visualization tool. The PCs 1 and 2 are used to represent the data in the scatter plot. Rocks labeled as Hermitage (red points) have similar values of the first two components and they are close to each other in the Principal Component system. The rocks labeled as Carters have similar values of the first two components and different values compared to the rocks labeled as Hermitage.

The example is referred to a numerical dataset of different rocks with different characteristics. The rocks have different features, where each feature is represented as a column of the original dataset. Each rock is labeled according to two types of rock formation: Carters formation and Hermitage formation. **Figure 4** illustrates that rocks labeled as Hermitage (red points), have similar values of the first and second component and they are close to each other in the data cloud. Similarly, the rocks labeled as Carters (blue points) display similar values of the first two components and they are close in the Principal Components system. **Figure 4** illustrates only the first two components, but the same rule holds for the subsequent components.

**Motivations for our application in the High Entropy Alloys (HEAs) dataset:**

* We are interested to use PCA as **visualization tool**, and not for the purpose of dimensionality reduction.
* We are not interested in the dimensionality reduction because our neural network performs semantic segmentation (i.e., pixel-wise prediction), so we need to keep the original dimensionality of 256x256 of the images.
* In our applications, we do not have labeled images (i.e., labeled HEAs nanoparticles), so the similarity of images is not referred to a specific label (as in the example of **Figure 4**).
* However, we know that the pixel intensity of the atomic columns depends on the structure of the HEAs. Thus, images with similar pixel values, represent HEAs with similar structures (i.e., similar column heights for the 5 elements). In our application the similarity is referred to the structure of the HEAs represented in the images.
* With PCA we can identify HEAs with similar structure by looking at the values of the principal components, and we can have insights about the distribution of our dataset for the point of view of column heights of the different elements.

**Nomenclature**

: number of data points in the dataset.

number of features in the dataset.

number of Principal Components used in the dimensionality reduction.

: data value for the data point and the feature

: features vectors 1,2, …, *p.*

: score for the data point and the component 1,2, …, *m.*

: score vectors for the component 1,2, …, *m.*

: loading vector for the component

1. **Calculation of the Principal Components**

Let’s call the vectors corresponding to the features as . Technically speaking, are the columns of the dataset (in Machine Learning they are called *feature vectors*).

A picture containing table

Description automatically generated

**Figure 5.** Each column of the original dataset is a feature vector with length , where is the number of observations (number of data points in the data cloud).

Before applying PCA, the dataset is scaled using a Gaussian scaler. Thus, the mean of each feature vector becomes 0, as it is illustrated in **Figure 6**.

**A picture containing bar chart

Description automatically generated**

**Figure 6.** Feature scaling using a Gaussian scaler. In many Machine Learning applications, including PCA, the dataset is scaled in order to keep balance among the features.

Each of the component is a linear combination of the features. The *first principal component* of a set of features is the normalized linear combination of the features:

(1)

that has the largest variance. The coefficients are called the *loadings* of the *first principal component*, and together they form the *loading vector .* The **first index** is referred to the **feature**, while the **second index** is referred to the **component** (i.e., the coefficient is the coefficient associated with the feature vector and the component , where and ). Considering all the observations , the calculation of the first component can be visualized as:

(2)

We refer to as the *scores* of the first principal component Technically speaking, is the first column of the transformed dataset with dimensionality , as it is represented in **Figure 7**.

Timeline

Description automatically generated

**Figure 6**. The first component is the first column of the transformed dataset with dimensionality .

The coefficients are calculated solving following *optimization problem*:

(3)

Since is the score of the first component , the *objective* of the optimization problem can be written as , which is the *variance* of the first component Thus, the objective of the optimization problem is to find the loadings which maximize the variance of the component. The target of PCA is the *variance*. We should remember that the *mean* is not taken into account because before applying PCA the dataset is scaled, so the mean of each feature vector with is 0 (i.e., ), thus the mean of the components with is also 0.

The *constraint* in the optimization problem is the normalization , to avoid arbitrary large values of the coefficients which results in an arbitrary large variance.

There is a specific *geometrical interpretation* of the first principal component. The loading vector defines a direction in the PCs space **where the data vary** **the most**. In other words, the first principal component is the component **with the highest variance**. If we project the data points onto this direction, the projected values are the first principal component scores themselves.

The calculation of the second component follow the same logic of the first component , but with the additional constraint that the direction of the loading vector of the second component is **orthogonal** to the direction of the loading vector of the first component. This constraint is essential to ensure that the second component is uncorrelated with the first component .

The calculation of the 3rd, 4th, …, up to follows the same logic. It is very important to remind that the variance of the components decreases from the first one. That is, . This notion is important to understand the criterion used to choose the number of components *m* to adopt in the dimensionality reduction, which is explained in the next section.

1. **How to choose the number of *m* components: the cumulative proportion of variance explained (PVE)**

When we reduce a dataset of dimensionality into a dataset with dimensionality , we aim to keep in the dataset with dimensionality almost the same variance of the original dataset with dimensionality . When reducing the dimensionality of a dataset, a certain amount of variance of the original dataset is lost. In general, the higher the number of components, the higher the retained variance.

From the point of view of the mathematical formulation, if we pick a number of components , where is the number of datapoints and the dimensionality of the original dataset, it is possible to keep the 100% of the variance. However, all the Python packages available to perform PCA, do not allow to use a number of datapoints for numerical stability purposes. Thus, in practical applications, it will always happen that and using PCs does not make sense because we would basically keep the same dimensionality of the original dataset. Thus, we will always choose , and consequently, the retained variance will always be < 100%. So how do we select the reasonable number of components *m*?

There is a threshold in the number of components over which the amount of the retained variance does not increase anymore, so it does not make sense to select an arbitrary high number of components. To find this threshold we look for a parameter called cumulative proportion of variance explained (cumulative PVE).

The *total variance* of a scaled dataset (thus assuming a zero mean) is defined as:

(4)

The variance explained by the *c* PC is defined as:

(5)

The PVE of the *c* PC is the proportion of the variance explained by the PC *c* respect to the total variance of the dataset and it is defined as:

(6)

The cumulative PVE for the component *c* where is simply the sum of the PVE from the component 1up to the component *c*. The reasonable number of components to pick in the PCA is the number of components for which the curve of the cumulative PVE starts to converge. The curve of the cumulative PVE starts to converge when increasing the number of components, there is not anymore improvement in the variance retained by the reduced dataset. Thus, increasing the number of components is not anymore necessary. The curve of the retained variance is called *scree plot* and the criterion to pick the number of components is called *elbow method*. **Figure 7** illustrates the scree plot that I obtained by applying PCA on our dataset of images of HEAs (number of images in the dataset: , dimensionality ).

A picture containing diagram

Description automatically generated

**Figure 7**. Scree plot obtained by applying PCA on the dataset of 5000 HEA images. With 1000 components we can retain ~90% of the variance of the original dataset according to the elbow method.

Comments about the scree plot represented in **Figure 7**:

* Using 1000 PCs, we can retain ~90% of the variance of the original dataset. We could also retain more of the original variance, for example if we use 4000 PCs we get an ~99% of the original variance. However, it does not make sense to use 4 times the number of PCs just to increase the explained variance by 9%. Thus, 1000 PCs is a reasonable trade-off.
* With PCA, we manage to reduce the dimensionality from 65536 (considering that we have a dataset of 256x256 images) to 1000, keeping the 90% of the original variance! This is an amazing result. However, we don’t use the dimensionality reduction in our problem because we are doing semantic segmentation (i.e., pixel-wise prediction), thus we need to keep 256x256 images.
* In general, the lower the dimensionality of the original dataset, the lower the number of components we need to keep. In this case we need 1000 components, which is itself a high number, but we need to remember that we start from a dimensionality of 65536. Thus, even if we have a dimensionality of 1000, we need to remember that we have already reduced the original dimensionality by a factor of ~65!.
* The number of PCs is an *hyperparameter* which need to be tuned to achieve the desired accuracy of the model. In Machine Learning, the tuning of the hyperparameters is performed with a method called cross-validation.

Once the number of PCs *m* is selected, we can apply an *inverse transformation* to transfer back from the system of the PCs to the original system. Since we have accepted to retain a lower amount of variance of the original dataset, it is not possible to come back to the exact same values of the original dataset, but with a retained variance of 90% we obtain a good approximation:

(7)

is value of the feature *j=*1,2,3, …,*p* for the datapoint *i=*1,2,3,…,*n* in the original dataset with dimensionality *p*, is the score of the component *c* for the datapoint *i* and is the loading of the component *c* associated with the feature *j*. Eq. 7 is simply the inverse transformation of the transformation we have performed in Eq. 2. The process is represented in **Figure 8**, where it is provided the example for the value

A picture containing graphical user interface

Description automatically generated

**Figure 8**. Inverse transformation to transfer from the reduced dataset with dimensionality to the original dataset with dimensionality *p.*

**Figure 9** illustrates the inverse transformation of one HEA image when PCA is applied with a different number of components. In particular, using different numbers of components leads to a different degree of approximation of the reconstructed image, which is related to the cumulative PVE associated with the number of PCs. The inverse transformation is applied on an image picked randomly from the dataset.

A picture containing qr code

Description automatically generated

**Figure 9**. Reconstruction of the original image using different number of components. Increasing the number of components, we obtain a more reliable approximation of the original image.

**Figure 9** shows the reconstruction of the original image of the HEA using different numbers of PCs. It is evident that increasing the number of components allows to obtain a more precise approximation of the original image. This is due to the fact that increasing the number of components, we retain more variance of the original system, as it is illustrated in the scree plot of **Figure 7**. In particular, using 1000 components we obtain a very good approximation of the original image, since 1000 components retain ~90% of the variance.

1. **PCA as visualization tool: the projector**

PCA can be used to visualize the data cloud in 2D or 3D scatterplots of the data in the PC system. The 3D scatterplot is available in Tensorboard (a visualization platform for Tensorflow) and it is called projector. There are main advantages of visualizing scatter plots in the PC systems.

* First, the PCs incorporate the statistical content from all the features of the original dataset in *p* dimensions. Thus, in a 2D or 3D scatterplot the data are distributed by taking into account the values of all the features. A scatterplot in the original system allows to visualize the distribution of the data cloud by taking into account 2 features (if the scatter plot is 2D) or 3 features (if the scatter plot is 3D), which certainly are not sufficient to describe the data distribution, in particular when *p* is large.
* Then, since the total number of scatterplots we need to visualize to have insights of the data distribution is certainly lower. In applications where a low number of PCs is sufficient to retain almost all the variance of the dataset (for example 4 or 5 components), a single 2D scatterplot of the PC1 and PC2 or a single 3D scatterplot of the PC1,PC2 and PC3 are sufficient to have a very good understanding of the data cloud distribution. Increasing the number of components which are necessary to explain the variance of the dataset, the higher the number of the scatterplots we need to consider.
* Data points which are close to each other in the scatterplots in the PC system display similar values of the features in the original system. Thus, visualizing the data in such scatterplot allows to identify different population in the data cloud.

Since it could be not very intuitive to understand the scatterplot, I will present a step by step summary of how we arrive there. I will use our problem of HEA images as benchmark.

**Step 1**: Visualization of the dataset. We have 5000 images with size 256x256, which gives a dimensionality of .

Graphical user interface, application

Description automatically generated

**Figure 10**. Visualization of our dataset of 5000 images with dimensionality *p=*65536.

**Step 2**: Application of PCA using PCs (we saw that 1000 PCs are sufficient to retain the 90% of the variance, so we use this value). Before applying PCA, the dataset is scaled with a Gaussian scaler as illustrated in **Figure 6**.

A picture containing table

Description automatically generated

**Figure 11**. PCA applied to the dataset using *m=*1000 PCs.

**Step 3**: Select the PCs to represent in the scatterplot. I am going to illustrate a 2D scatterplot of the first two components, PC1 and PC2.

A picture containing chart

Description automatically generated

**Figure 12**. PC1 and PC2 are selected to be visualized in a 2D scatterplot. The coordinates of the datapoints (i.e., the images in the dataset) in the scatterplot are the corresponding values of the scores of the first and the second PCs.

**Step 4**: Visualization of the scatterplot of the PC1 and PC2.

Chart, scatter chart

Description automatically generated

**Figure 13**. 2D Scatterplot of the first and second components (PC1 and PC2).

Comments on the scatterplot:

* The large part of the datapoints are concentrated in the center of the distribution of the data cloud, while the points which are spread from the center look to be outliers. My interpretation of this distribution is that the dataset is created by modeling random HEAs with random positions of the atoms of the 5 elements. When we have a random distribution, the data are concentrated around the mean and spread toward the edges of the distribution in a low %. In this case, the images concentrated around the center (i.e., the mean) are ~93% of the data cloud, while the outlier are ~7%.
* From the scatterplot we can identify the outlier images and “clean” the dataset and focus our analysis on the main distribution.
* We can investigate the distribution of the column heights of the 5 elements for the images concentrated in the distribution and we can have insights how it is different from the distribution of column heights of the outlier images.
* We can apply PCA on the experimental images and see how they are positioned in the data cloud and get some preliminary considerations about the column heights: the experimental images will have column heights with similar values to the images which are close to them.
* Since we have 1000 components, we can visualize many scatterplots associated with different combinations of the components. However, it is important to remind that the first component is the one with the highest variance, then the second, the third, etc.. Thus, subsequent scatterplots will show less and less variance. **Figure 14** illustrates three scatterplots for subsequent components, for which the variance is reduced. **Figure 14 (a)** shows that PC1 spreads from -150 to 150, **Figure 14 (b)** shows that PC9 spreads from -120 to 120, **Figure 14 (c)** shows that PC19 spreads from -75 to 100.

Chart, scatter chart

Description automatically generated

**Figure 14**. Scatterplots for PC1&2 **(a)**, PC9&10 **(b)**, PC19&20 **(c)**.

If we visualize the scatterplots for the highest components, we realize that the variance explained is further reduced. For example, the range of the last components is between

-10 and 10 as it is illustrated in **Figure 15**.

Chart, scatter chart

Description automatically generated

**Figure 15**. Scatterplots for PC799&800 **(a)**, PC899&900 **(b)**, PC999&1000 **(c)**.

It is very important to not confuse the variance explained by the single components and the cumulative proportion of variance explained (cumulative PVE), the parameter used to assess the number of components *m* to adopt. The variance of a single component is simply its variance, and we know that the first component has the highest variance, then the second, the third, etc.. The cumulative PVE of component *c* is the sum of the PVEs from the component 1 up to the component *c*, and for this reason the cumulative PVE increases for higher components.

1. **References**

[1] An Introduction to Statistical Learning with Applications in R, G. James, D. Witten, T. Hastie, R. Tibshirani, Springer.