Assume you have \( R \) samples of data each with \( C \) features in an \( R \times C \) matrix called \( X \). That is, each sample of \( C \) dimensional data is in a row. The matrix could represent any kind of data in which there are \( C \) features in each of \( R \) samples. This data could even be a matrix of gray levels of a picture where each row of gray level pixels is a point in \( C \) dimensional space. The data is usually correlated in some from one feature to some other feature (column to another) but not necessarily adjacent. That is, the data usually occupies some much lower dimension. Somewhere in the data are the essential features. In fact, the essential features may be a linear combination of the features given. If they are, we can find these combinations by using Principle Component Analysis (PCA). In fact, PCA can tell us the relative contribution from various orthogonal components so we can concern ourselves only with the most salient features. Sometimes reducing the dimensionality can be used to eliminate noise and help focus an algorithm on learning the important parts of the data.

1 The PCA Algorithm for Compression

Here is a recapitulation of the algorithm in the book but with more detail and some example data in detail. I will assume that some data such as the list of eigenvectors comes stored as one eigenvector per row. The algorithms can be adjusted to work with an assumption that eigenvectors are in columns, but that was not what I chose here because of the way C/C++ stores matrices. Be sure to check the code you are using to get the alignment of vectors correct.

1.1 Center the data.

This is done by computing the mean position of the \( R \) points in \( C \) dimensional space and then subtracting that mean from each row.

\[
X'[r] = X[r] - Mean(X) \quad \forall r
\]

Where \( X[r] \) the the \( r^{th} \) row and Mean returns the mean of all the rows.

If the data in different dimensions are on different scales like 0 to 10 inches vs 0 to 1000000 years then scaling with Z-score is advised.

\[
X'[r] = (X[r] - Mean(X))/Stddev(X) \quad \forall r
\]

This makes sure each dimension (column) is scaled within that dimension.
1.2 Compute covariance matrix.

compute the covariance matrix $M$ using the normalized data $X'$.

$$M = (1/R)X'^T \cdot X'$$

$M$ should be a $C \times C$ matrix. Beware that this is the biased covariance because we divided by $R$. The unbiased covariance divides by $(R-1)$. So know what your covariance routine is producing!

1.3 Compute eigenvalues and eigenvectors of $M$

The covariance matrix is symmetric and so the eigenvalues $W_i$ are all real numbers. Eigenvalues, $W_i$, and eigenvectors $V_i$, come out as pairs, one eigenvalue for each eigenvector and are often computed together in one routine. The eigenvectors indicate the direction of the variation. The eigenvalues indicate the amount of variation. When calling an eigenvector routine know if the eigenvectors are normalized or not. Their real purpose is to give a direction so they are often normalized. Also know if they are sorted by magnitude of eigenvalue. We will be concerned with the largest eigenvalues.

$$V = \text{eigenvector}(M) \quad W = \text{eigenvalues}(M)$$

$$V_i \in \mathbb{R}^C \quad \text{and} \quad W_i \in \mathbb{R}$$

I will assume the eigenvectors are stored as rows in an array.

1.4 Normalize the eigenvectors

The length of each eigenvector should be 1. This can be done without considering the eigenvalues because an eigenvector is really a direction and magnitude is not important. If they are not normalized for some reason you can normalize them.

$$V'[r] = \text{Normalize}(V[r]) \quad \forall r$$

1.5 Sort eigenvectors by eigenvalue

We will consider only the eigenvectors with the $k$ largest eigenvalues in magnitude. We can do this by sorting the eigenvalue/eigenvector pairs by eigenvalue magnitude. They may already be sorted. Check your documentation. Then taking the $K$ largest.

$$\hat{V} = \text{Max}_k(W, V')$$

1.6 Translate the normalized data

We now use the reduced set of eigenvectors to reduce the dimension of $X$. We do this by selecting the $k$ largest eigenvalues and their associated vectors and ignoring the rest. This makes a matrix of $k$ row vectors.

$\hat{V}$ is now a $k \times C$ matrix.
\[ X'' = X' \cdot \hat{V}^T \]

The resulting matrix \( X'' \) is \( R \times k \) in size rather than \( R \times C \). It is smaller! It is this reduced dimension matrix \( X'' \) that we could apply our machine learning algorithms to and see if they work better. This approach has been used in facial recognition for instance.

### 1.7 Recovering data from compressed data

Machine Learning doesn’t require you recover the image from its compressed form. It is generally just used as a first step in classifying. But to show how the compressed form is capturing key attributes of the original data we can recover the image, but it requires: the compressed image \( X'' \), the selected eigenvectors \( \hat{V} \) and the scaling information \( \text{Stddev}(X) \) and \( \text{Mean}(X) \).

First rotate the data back using the reduced eigenvector matrix:

\[ X^* = X'' \cdot \hat{V} \]

Then move the data back from where it was centered to its old position: Note the \( \text{Mean}(X) \) is the original mean.

\[ X^*[r] + \text{Mean}(X) \quad \forall r \]

or if Z-score

\[ (X^*[r] \ast \text{Stddev}(X)) + \text{Mean}(X) \quad \forall r \]

The result should now be \( R \times C \) matrix. But it is not exactly the same because \( k < C \) and this causes some data to be lost, but because we kept the \( k \) largest eigenvalues we kept most of the sources of variance in our data. Note that if \( k = C \) then we should get the exact data back, ignoring tiny errors in the arithmetic.

### 1.8 The Component Matrix

Component analysis tells you the “weight” of each of the original dimension’s involvement in the new axes. In this case multiply each of the eigenvectors by the square root of the corresponding eigenvalue. The Matrix of scaled eigenvectors is sometimes called the Component Matrix.

\[ V_i \sqrt{W_i} \quad \forall i \]

Values in each vector that are near 1 are strongly influences and values near 0 are weak influences.