Top highlight

Understanding Principal Component Analysis

Rishav Kumar
Follow
Jan 2, 2018 · 8 min read

The purpose of this post is to give the reader detailed understanding of Principal Component Analysis with the necessary mathematical proofs.

> ***Should be read on big screen. Phone screen doesn't render the png images properly, and images are important. :)***
>
> *I have used text and formulas in images because medium doesn't support latex. So, I had to export latex to png to display it here.*
> *You can skip the proofs if you don't want to indulge in Linear Algebra.*
> *It's my first medium post and I expect the readers to really try to understand what*
> *I'm showing here.*

In real world data analysis tasks we analyze complex data i.e. multi dimensional data. We plot the data and find various patterns in it or use it to train some machine learning models. One way to think about dimensions is that suppose you have an data point **x** , if we consider this data point as a physical object then dimensions are merely a basis of view, like where is the data located when it is observed from horizontal axis or vertical axis.

As the dimensions of data increases, the difficulty to visualize it and perform computations on it also increases. So, how to reduce the dimensions of a data-
* Remove the redundant dimensions
* Only keep the most important dimensions

> ***break1***

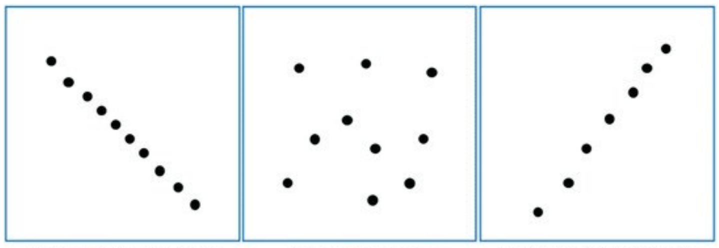
First try to understand some terms -

Variance : It is a measure of the variability or it simply measures how spread the data set is. Mathematically, it is the average squared deviation from the mean score. We use the following formula to compute variance *var(x).*

$$var(x) = \frac{\sum(x_i - \bar{x})^2}{N} \qquad cov(x,y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N}$$

Covariance : It is a measure of the extent to which corresponding elements from two sets of ordered data move in the same direction. Formula is shown

above denoted by *cov(x,y)* as the covariance of *x* and *y*.
Here, *xi* is the value of x in *ith* dimension.
x bar and *y bar* denote the corresponding mean values.
One way to observe the covariance is how interrelated two data sets are.



Positive covariance means X and Y are positively related i.e. as X increases Y also increases. Negative covariance depicts the exact opposite relation. However zero covariance means X and Y are not related.

> ***Continue break1***

Now lets think about the requirement of data analysis.
Since we try to find the patterns among the data sets so we want the data to be spread out across each dimension. Also, we want the dimensions to be independent. Such that if data has high covariance when represented in some *n* number of dimensions then we replace those dimensions with *linear combination* of those n dimensions. Now that data will only be dependent on linear combination of those related n dimensions. *(related = have high covariance)*

So, what does Principal Component Analysis (PCA) do?

> *PCA finds a new set of dimensions (or a set of basis of views) such that all the dimensions are orthogonal (and hence linearly independent) and ranked according to the variance of data along them. It means more important principle*
> *axis occurs first. (more important = more variance/more spread out data)*

How does PCA work -

1. Calculate the covariance matrix *X* of data points.
2. Calculate eigen vectors and corresponding eigen values.
3. Sort the eigen vectors according to their eigen values in decreasing order.
4. Choose first k eigen vectors and that will be the new k dimensions.
5. Transform the original n dimensional data points into k dimensions.

To understand the detail working of PCA , you should have knowledge of eigen vectors and eigen values. Please refer to this visual explanation of eigen vectors and values.

> *Make sure you understand the eigenvectors and eigen values before proceeding*

> *further.*

$$[Covariance\ \ matrix] \cdot [Eigenvector] = [eigenvalue] \cdot [Eigenvector]$$

Assuming we have the knowledge of variance and covariance, Lets look into what a ***Covariance matrix*** is.

$$\begin{bmatrix} V_a & C_{a,b} & C_{a,c} & C_{a,d} & C_{a,e} \\ C_{a,b} & V_b & C_{b,c} & C_{b,d} & C_{b,e} \\ C_{a,c} & C_{b,c} & V_c & C_{c,d} & C_{c,e} \\ C_{a,d} & C_{b,d} & C_{c,d} & V_d & C_{d,e} \\ C_{a,e} & C_{b,e} & C_{c,e} & C_{d,e} & V_e \end{bmatrix}$$

A covariance matrix of some data set in 4 dimensions a,b,c,d.
Va : variance along dimension a
Ca,b : Covariance along dimension a and b

If we have a matrix X of m*n dimension such that it holds n data points of m dimensions, then covariance matrix can be calculated as

$$C_x = \frac{1}{n-1}(X - \bar{X})(X - \bar{X})^T \qquad X^T = Transpose\ of\ X$$

It is important to note that the covariance matrix contains -
* variance of dimensions as the main diagonal elements.
* covariance of dimensions as the off diagonal elements.

Also, covariance matrix is symmetric. (observe from the image above)
As, we discussed earlier we want the data to be spread out i.e. it should have high variance along dimensions. Also we want to remove correlated dimensions i.e. covariance among the dimensions should be zero (they should be linearly independent). Therefore, our covariance matrix should have -
* large numbers as the main diagonal elements.
* zero values as the off diagonal elements.
We call it a *diagonal matrix*.

> *A cat, just to get your attention. :)*

So, we have to transform the original data points such that their covariance is a diagonal matrix. The process of transforming a matrix to diagonal matrix is called *diagonalization*.

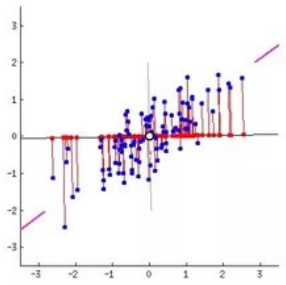
> ***Always normalize your data before doing PCA because if we use data(features here) of different scales, we get misleading components. We can also simply use correlation matrix instead of using covariance matrix if features are of different scales. For the simplicity of the post,***

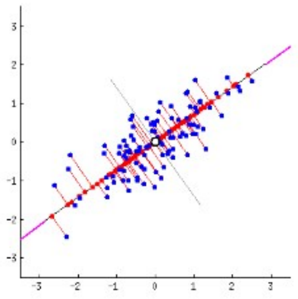
> ***I'm assuming we have already normalized data.***

This defines the goal of PCA -

1. Find linearly independent dimensions (or basis of views) which can losslessly represent the data points.
2. Those newly found dimensions should allow us to predict/reconstruct the original dimensions. The reconstruction/projection error should be minimized.

Lets try to understand what I mean by projection error. Suppose we have to transform a 2 dimensional representation of data points to a one dimensional representation. So we will basically try to find a straight line and project data points on them. (A straight line is one dimensional). There are many possibilities to select the straight line. Lets see two such possibilities -





Say magenta line will be our new dimension.
If you see the red lines (connecting the projection of blue points on magenta line) i.e. the perpendicular distance of each data point from the straight line is the projection error. Sum of the error of all data points will be the total projection error.
Our new data points will be the projections (red points) of those original blue data points. As we can see we have transformed 2 dimensional data points to one dimensional data points by projection them on 1 dimensional space i.e. a straight line. That magenta straight line is called *principal axis.* Since we are projecting to a single dimension, we have only one principal axis.

Clearly, Second choice of straight line is better because -
* The projection error is less than that in the first case.
* Newly projected red points are more widely spread out than the first case. i.e. more variance.

The above mentioned two points are related i.e. if we minimize the reconstruction error, the variance will increase.
How ?
Proof : (Optional) https://stats.stackexchange.com/questions/32174/pca-objective-function-what-is-the-connection-between-maximizing-variance-and-m/136072#136072

Steps we have performed so far -
* We have calculated the covariance matrix of original data set matrix **X.**

Now we want to transform the original data points such that the covariance matrix of transformed data points is a diagonal matrix. How to do that ?

$$
\begin{aligned}
&C_x = covariance\ matrix\ of\ original\ data\ set\ X \\
&C_y = covariance\ matrix\ of\ transformed\ data\ set\ Y \\
&such\ that, \\
&Y = PX \\
&For\ simplicity,\ we\ discard\ the\ mean\ term\ and\ assume\ the \\
&data\ to\ be\ centered.\ i.e.\ X = (X - \bar{X}) \\
&So,\ C_x = \tfrac{1}{n}XX^T \\
&C_y = \tfrac{1}{n}YY^T \\
&= \tfrac{1}{n}(PX)(PX)^T \\
&= \tfrac{1}{n}PXX^TP^T \\
&= P(\tfrac{1}{n}XX^T)P^T \\
&= PC_xP^T
\end{aligned}
$$

Here's the trick- If we find the matrix of eigen vectors of Cx and use that as P (P is used for transforming X to Y, see the image above) , then Cy (covariance of transformed points) will be a diagonal matrix. Hence Y will be the set of new/transformed data points.
Now, if we want to transform points to k dimensions then we will select first k eigen vectors of the matrix Cx (sorted decreasingly according to eigen values) and form a matrix with them and use them as P.

> *So, if we have m dimensional original n data points then*
> *X : m*n*
> *P : k*m*
> *Y = PX : (k*m)(m*n) = (k*n)*
> *Hence, our new transformed matrix has n data points having k dimensions.*

But why does this trick work ?
Proof:

First lets look at some theorems -

- Theorem-1 :
  The inverse of an orthogonal matrix is its transpose, why?

$$
\begin{aligned}
&Let\ A\ be\ an\ m * n\ orthogonal \\
&matrix\ where\ a_i\ is \\
&the\ ith\ column\ vector. \\
&The\ ijth\ element\ of\ A^TA\ is \\
\\
&(A^TA)_{ij} = a_i^Ta_j = \\
&\begin{cases} 1\ \ if\ i = j \\ 0\ \ otherwise \end{cases} \\
&\text{Therefore, because } A^TA = I,
\end{aligned}
$$

$it\ follows\ that\ A^{-1} = A^T.$

- Theorem-2 :

$$\begin{array}{l} Let\ A\ be\ a\ real\ symmetric\ matrix\ and\ \lambda_1, \lambda_2, ..., \lambda_k\ be\ \ distinct\ eigenvalues\ of\ A. \\ Let\ u_i \in R^n\ be\ nonzero\ such\ that,\ 1 \leq i \leq k. \\ Then\ \{u_1, u_2, .., u_k\}\ forms\ an\ orthonormal\ set. \end{array}$$

Proof :

$$\begin{array}{l} For\ i \neq j,\ 1 \leq i, j \leq k,\ since\ A^T = A,\ we\ have \\ \lambda_i \langle u_i, u_j \rangle\ =\ \langle \lambda_i u_i, u_j \rangle \\ = \langle Au_i, u_j \rangle\ =\ \langle u_i, A^T u_j \rangle\ =\ \langle u_i, Au_j \rangle \\ = \lambda_j \langle u_i, u_j \rangle \\ Since,\ i \neq j\ we\ have\ \lambda_i \neq \lambda_j\ and\ hence\ \langle u_i, u_j \rangle\ =\ 0 \end{array}$$

- Theorem-3 :

$$\begin{array}{l} Let\ A\ be\ n * n\ real\ symmetric\ matrix\ such\ that \\ all\ its\ eigenvales\ are\ distinct.\ Then,\ there \\ exists\ an\ orthogonal\ matrix\ P\ such\ that, \\ P^{-1}AP = D \\ where\ D\ is\ a\ diagonal\ matrix\ with \\ diagonal\ entries\ being\ the\ eigenvalues\ of\ A. \end{array}$$

Proof :

$$\begin{array}{l} Let\ A\ has\ eigenvalues\ \lambda_1, \lambda_2, ..., \lambda_n\ with\ u_i \in R^n \\ such\ that\ |u_i| = 1\ and\ Au_i\ =\ \lambda_i u_i, 1 \leq i \leq n. \\ By\ corollary,\ the\ matrix \\ P = [u_1, u_2, .., u_n] \\ is\ invertible\ and\ P^{-1}AP = D,\ is\ diagonal\ with\ diagonal\ entries. \\ Further,\ by\ theorem - 2,\ (u_1, u_2, .., u_n)\ is\ an\ orthogonal\ set.\ Hence\ P \\ is\ in\ fact\ an\ orthogonal\ matrix. \end{array}$$

Having these theorems, we can say that

> A symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors. Orthonormal vectors are just normalized orthogonal vectors. (what normalization is? google ;))

$$\begin{array}{l} C_y = PC_xP^T \\ = P(E^TDE)P^T \\ = P(P^TDP)P^T \\ = (PP^T)D(PP^T) \\ = (PP^{-1})D(PP^{-1}) \\ C_Y = D \end{array}$$

It is evident that the choice of P diagonalizes Cy. This was the goal for PCA. We can summarize the results of PCA in the matrices P and Cy.

- The principal components of X are the eigenvectors of Cx.
- The i th diagonal value of Cy is the variance of X along pi

Conclusion -

$$[new\ data]_{k*n} = [top\ k\ eigenvectors]_{k*m}[original\ data]_{m*n}$$

In the next post we will be implementing PCA in python and using it for color data augmentation.

> ***Note:*** *PCA is an analysis approach. You can do PCA using SVD, or you can do PCA doing the eigen-decomposition (like we did here), or you can do PCA using many other methods. SVD is just another numerical method. So, don't confuse the terms PCA and SVD. However, there are some performance factors of sometimes choosing SVD over eigen-decomposition or the other way around(not that you should care much about). We will explore SVD in our upcoming posts.*

Let's discuss in comments if you find anything wrong in the post or if you have anything to add. and give me claps :P
Thanks.

Data Science Machine Learning Deep Learning Statistics Linear Algebra

WRITTEN BY
Rishav Kumar
Follow
I like to write about machine learning.. Contact me - code.rishus23@gmail.com

Understanding the concept of Hierarchical clustering Technique

Chaitanya Reddy in Towards Data Science

Dec 10, 2018 · 7 min read

271