Proper Orthogonal Decomposition

Chunyan Li

March 2022

1 Introduction

Outline

Contents

2 Motivation

Motivation PCA is a dimensional reduction method/feature extraction method.

Motivation In case (a), all points lies on a straight line which is a subspace of R^2, then PCA could help us to determine this straight line, then we could represent all these data in $1D$ by using this line as new coordinate system.
In case (b), the points are not exactly lie on the line, but we still could find such a line and use the projection of the points on this line to represent them! Though, we lose some info, but we didn't lose too much as long as the line is well chosen.

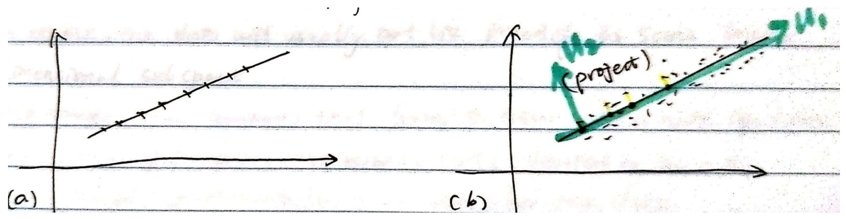

Figure 1: fig1

[1]

. In (b), u_1 is the first principle component, and u_2 is the second one.

- One way to understand PCA is that we are rotating the coordinate system so that we could represent the data in a lower-dimensional subspace without losing too much info.
- Another way to understand PCA is that looking for vector u_1 among all possible vector in this space such that when all points are projected on u_1, the variation of the data set is maximum.
- the goal is to preserve as much of the variance in the original data as possible in the new coordinate system.
- Given data of d variables, the hope is that the data points will lie mainly in a linear subspace of dimension lower that d.

3 Definition and Optimization problem

$\boxed{\text{Definition}}$ For a given set of data vectors $x_i, i = 1,2,...,n.$ the p principal axes are those orthonormal axes onto which the variance retained under projection is maximal.
let $X = [x_1, x_2, ..., x_n]$ where $x_i \in R^d, i = 1,2,...,n$. Then, we could formulate the PCA as the following optimization problem.

$$max_{u_1 \in R^d} Var(u_1^T X) = max_{u_1 \in R^d} u_1^T S u_1 \tag{1}$$

Where S is the $d \times d$ sample covariance matrix of original data X. This is not a well defined problem, since the objective is a quadratic form which has no upper bound. Hence, we need a constrain.
$\boxed{\text{Optimization Problem}}$

- Where the constrain comes from?
  Since we are searching for the direction of PC, but not the length, then, we could add a constrain $||u_1||_2^2 = u_1^T u_1 = 1$.

[1]Ghodsi, *Principal Component Analysis, Lec1 & Lec2*

- Now we get a well-defined optimization problem:

$$max_{u_1} u_1^T S u_1 \tag{2}$$
$$s.t. u_1^T u_1 = 1 \tag{3}$$

$\boxed{\text{Lagrange Multiplier Method}}$ Apply Lagrange Multiplier method, we get:

$$\mathbf{L}(u_1, \lambda_1) = u_1^T S u_1 - \lambda_1 (u_1^T u_1 - 1) \tag{4}$$

where λ_1 is lagrange multiplier or dual variable.
By differentiate this Lagrangian function, we get

$$\frac{\partial \mathbf{L}}{\partial u_1} = 2Su_1 - 2\lambda_1 u_1 = 0. \tag{5}$$
$$\frac{\partial \mathbf{L}}{\partial \lambda_1} = u_1^T u_1 - 1 = 0 \tag{6}$$

4 Structure of PCs

$\boxed{\text{Structure of PC}}$ From (5), we know, $Su_1 = \lambda_1 u_1$, hence, λ_1 and u_1 is eigenpair of S. Then, we could determine the PC by simplifying the objective function using this property.

$$u_1^T S u_1 = \lambda_1 u_1^T u_1 = \lambda_1 \tag{7}$$

covariance matrix S has at most d eigenpairs,

$$\lambda_1 > \lambda_2 > ... > \lambda_d$$
$$u_1 > u_2 > ... > u_d$$

Conclusion: The eigenvector of sample covariance matrix S corresponding to the maximum eigenvalue is the first principal component! Similar results for other PCs.

$\boxed{\text{Centralization and SVD}}$ Centralize X, let $X := X - ones(d,1)M$ where M is a row vector with size 1 by n, with mean of each row of X as entry. By the definition of covariance matrix, we know $S = E[(X-\mu)(X-\mu)^T] = E(XX^T)$. Then, do SVD for the centralized data, we get

$$X_{d\times n} = U \Sigma V^T \tag{8}$$

where $U_{d\times d}$ is the eigenvector matrix of XX^T which is S, $V_{n\times n}$ is the eigenvector matrix of $X^T X$, Σ is diagonal matrix with eigenvalues of $X^T X$ as components. (descending order)
Hence, U is the collection of all principal components.

5 Encoder and Decoder

$\boxed{\text{encoder and decoder/reconstruction}}$ let $X_{d\times n} = [x_1, ..., x_n]$
We could project the first sample $x_1 \in R^d$ into a point $y_1 \in R$ in $u_1 \in R^d$ axes by the following projection.

$$y_1 = u_1^T x_1 \tag{9}$$

One can also project it back by

$$\hat{x}_1 = u_1 y \tag{10}$$

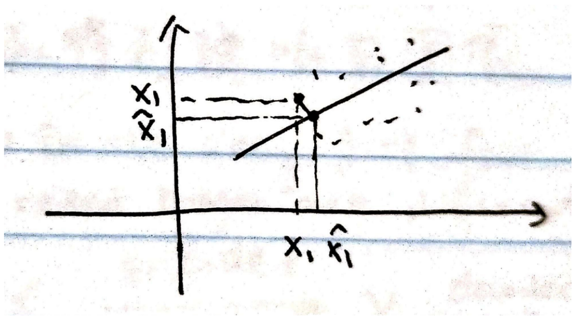

Figure 2: fig2

$\boxed{\text{encoder and decoder}}$ ${u_i}^T X$ is the projection of the data on the direction of the $i-th$ PC. Let $U_{d\times p} = [u_1, ..., u_p]$, where $p \leqslant d$, we could treat PCA as a linear autoencoder:

$$X_{d\times n} \xrightarrow[U_{d\times p}^T]{encoder} Y_{p\times n} \xrightarrow[U_{d\times p}]{decoder} \hat{X}_{d\times n} \tag{11}$$

or for single sample x:

$$x_{d\times 1} \xrightarrow[U_{d\times p}^T]{encoder} y_{p\times 1} \xrightarrow[U_{d\times p}]{decoder} \hat{x}_{d\times 1} \tag{12}$$

Remark:
The variance of the original data is the same as the projected data on all PCs. This conclusion comes from the fact:

$$\sum_{i=1}^{d} Var(u_i^T X) = \sum_{i=1}^{d} \lambda_i = Tr(S) = Var(X) \tag{13}$$

$\boxed{\text{Algorithm 1}}$

- Centralized original data X to get new X.
- Calculate XX^T and let U = eigenvectors of XX^T corresponding to the top p eigenvalues.
- Encode the training data: $Y = U^T X$ where $Y \in \mathbf{R}^{p\times n}$ is a matrix of encodings of the original data.
- Reconstruct training data: $\hat{X} = UY = UU^T X$.
- Encode the test sample: $y = U^T x$ where y is a $p$dimensional encoding of x.
- Reconstruct test sample: $\hat{x} = Uy = UU^T x$.

6 Dual PCA

Dual PCA We learned the PCA for the case $d < n$, now, we learn Dual PCA for $d > n$.
Motivation: In reality, there is a situation that $d > n$, for example genes and patients. Then XX^T is $d \times d$ a larger matrix, which is computational cost to do SVD, but, X^TX is $n \times n$ a smaller matrix, hence, in Dual PCA, we would like to decompose X^TX and represent U in terms of X and the SVD results(Σ, V) of X^TX.
Let $X = U\Sigma V^T$, then $XV = U\Sigma$, then

$$U = XV\Sigma^{-1} \tag{14}$$

where Σ, V comes from the svd of X^TX.

Dual PCA Algorithm 2

- Centralized original data X to get new X.
- Calculate X^TX and do SVD for it to get Σ and V.
- Encode the training data: $Y = U^TX = \Sigma V^T$ where V is the matrix of eigenvectors of X^TX corresponding to the top p eigenvalues. $Y \in \mathbf{R}^{p\times n}$ is a matrix of encodings of the original data.
- Reconstruct training data: $\hat{X} = UY = XV\Sigma^{-1}Y = XV\Sigma^{-1}\Sigma V^T = XVV^T$.
- Encode the test sample: $y = U^Tx = \Sigma^{-1}V^TX^Tx$ where y is a $p$dimensional encoding of x.
- Reconstruct test sample: $\hat{x} = Uy = UU^Tx = XV\Sigma^{-1}\Sigma^{-1}V^TX^Tx = XV\Sigma^{-2}V^Tx$.

7 Kernel PCA

Kernel PCA

- What is kernel?
  Any positive definite function could be a kernel function.
- Why it helps?/Why we need it?
  Usually, higher dimensional data is easier to explore. Changing the nonlinear data into a linear case(quasi-linear case) by kernel. And this is usually done by going to higher dimensional space.

x^Ty is a measurement of the similarity between x and y in Euclidean metric, then $\phi(x)^T\phi(y)$ is another measurement of similarity between $\phi(x)$ and $\phi(y)$ which is based on other metric.

what is kernel? let $x = [x1, x2]'$, $\phi(x) = [x_1^2, x_2^2, \sqrt{2}x_1x_2]'$, $y = [y_1, y_2]'$, $\phi(y) = [y_1^2, y_2^2, \sqrt{y_1y_2}]'$.
Then, $\phi(x)^T \cdot \phi(y) = x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 = (x_1y_1 + x_2y_2)^2$, One could verify easily that there is a function $k(x,y)$ such that $k(x,y) = \phi(x)^T\phi(y)$.
$k(x,y) = (x_1y_1 + x_2y_2)^2$
Examples of kernel functions:

- linear kernel: $k(x,y) = \langle x,y \rangle$
- polynomial kernel: $k(x,y) = (1 + \langle x,y \rangle)^p$
- Gaussian kernel: $k(x,y) = e^{-\dfrac{||x-y||^2}{2\sigma^2}}$

Kernel PCA Algorithm 3

- Centralized original data X to get new X.
- Calculate $\phi(X)^T\phi(X)$ and do SVD for it to get Σ and V.
- Encode the training data: $Y = U^T\phi(X) = \Sigma V^T$ where V is the matrix of eigenvectors of X^TX corresponding to the top p eigenvalues. $Y \in \mathbf{R}^{p\times n}$ is a matrix of encodings of the original data.
- Reconstruct training data: We can't!
  since $\hat{X} = UY = XV\Sigma^{-1}Y = XV\Sigma^{-1}\Sigma V^T = \phi(X)VV^T$ where $\phi(X)$ is unknown.
- Encode the test sample: $y = U^T\phi(x) = \Sigma^{-1}V^T\phi(X)^T\phi(x)$ where y is a $p$dimensional encoding of x.
- Reconstruct test sample: We can't!
  $\hat{x} = Uy = UU^T\phi(x) = \phi(X)V\Sigma^{-1}\Sigma^{-1}V^T\phi(X)^T\phi(x) = \phi(X)V\Sigma^{-2}V^T\phi(x)$. We don't know $\phi(X)$.

Scree plot and biplot The residual error from only using first p PCs is given by the sum of discarded eigenvalues:

$$Error = \sum_{j=p+1}^{d} \lambda_j \tag{15}$$

Hence, one can plot the retained eigenvalues in decreasing order,known as scree plot, to find the relationship between the reconstruction error and number of PCs.

To better understand the meaning of the principle components, we can project unit vectors corresponding to each of the feature dimensions, $e_1 = (1,...,0) \in \mathbf{R}^d, e_2 = (0,1,...,0)$, etc. into the low dimensional space. Usually, project the unit vector of features to the first 2 PCs. This is known as a biplot. Note: One can check that the projection of $e_1 = [u_{11}, u_{21}, ..., u_{p1}]$.

8 Categorical PCA

categorical PCA

9 Proper Orthogonal Decomposition (POD)

$\boxed{\text{Proper Orthogonal Decomposition (POD)}}$ POD is a method of reduced order modeling.

$$u_t(t,x) = \mathscr{L}u(t,x) + \mathscr{N}u(t,x) \tag{16}$$

For such a PDE, after spatial discretization (for example, finite difference discretization to represent an element u in infinite dimensional by a finite dimensional vector $\mathbf{u}$), we obtain a collection of ODE (a large ODE system)

$$u(t,x) \to u(t,x_j) = u_j(t), \quad j = 1,...,n \text{ (spatial discretization)} \tag{17}$$

$$\frac{d\mathbf{u}(t)}{dt} = L\mathbf{u}(t) + N\mathbf{u}(t) \tag{18}$$

Where $\mathbf{u} = [u_1(t),...,u_n(t)]$, L and N are the discrete version of $\mathscr{L}$ and $\mathscr{N}$ respectively, and n could be very large if the mesh size of spatial domain is quite small or the dimension of the spatial domain is large (For example, 1D case with 100 grids, then, $\mathbf{u} \in R^{10^2}$, 2D case with 100 grids, then, $\mathbf{u} \in R^{10^4}$, and 3D case with 100 grids, then, $\mathbf{u} \in R^{10^6}$). Solving this kind of large dynamical system will be very time consuming or even out of our computational capacity or memory capacity. Hence, we need to develop the reduced order model to approximate this kind of dynamic system accurately and could be computed very efficiently. POD is such a method to embed the high dimensional dynamic into a lower dimensional space by using a proper basis.

Now, for solving eq.18, we first do temporal discretization with mesh Δt, then RK method or other numerical scheme for ODE could be applied to obtain snapshot $u_i = [u_{i1},...,u_{in}]^T$ at time t_i. Let $\mathbf{X} = [\mathbf{u}_1,\mathbf{u}_2,...,\mathbf{u}_m]$ be the centered snapshots matrix (minus the mean of each column/take average over space) of dynamics at m different time.

$\boxed{\text{POD basis}}$ To find a lower dimensional representation of $\mathbf{u}$, one can apply PCA. By using the idea of POD (PCA), one can find the left singular matrix Φ_n of X through SVD or find the eigenmatrix Φ_n of covariance matrix XX^T to get the proper orthogonal basis Φ_r (first r columns of Φ_n) to represent $\mathbf{u}$ in a lower dimension. Note that when X is low-and-fat $(n \ll m)$, the POD basis can be found by using the SVD algorithm. Conversely, if x is tall-and-skinny $(n \gg m)$, the method of snapshots can be applied. When parallel computing is implemented in large-scale computing problems, one can use an approximate partitioned method of snapshots to further reduce the computational complexity and the communication volume for generating the POD basis in (Z. Wang, McBee, and Iliescu, "Approximate partitioned method of snapshots for POD").

$\boxed{\text{Galerkin Projection}}$

$$\mathbf{u}(t) \approx \sum_{i=1}^{r} \phi_i a_i(t) = \Phi_r \mathbf{a}(t) \tag{19}$$

$$\Phi_r^T \Phi_r = I_{r\times r} \tag{20}$$

where $\mathbf{u} \in R^n$, $\Phi_r = [\phi_1,...,\phi_r] \in R^{n\times r}$ is the collection of r POD basis (axis) and $\mathbf{a}(t) = [a_1(t),...,a_r(t)] \in R^r$ is the corresponding coordinate coefficients. eq.20 shows the Orthogonality of the linear transform $\Phi_r \in R^{n\times r}$ which could be a long matrix (so usually, it not an orthogonal matrix which is a square matrix).

$\boxed{\text{Low-Rank Dynamics}}$ In matrix form, we could plug eq.19 into eq.18, and multiply Φ_r^T, we get the Galerkin projected Low-Rank Dynamics

$$\frac{d\mathbf{a}(t)}{dt} = \Phi_r^T L \Phi_r \mathbf{a}(t) + \Phi_r^T N(\Phi_r \mathbf{a}(t)) \tag{21}$$

where Φ_r is determined by POD and the above Low-Rank dynamics is obtained through projecting the original dynamics into lower dimensional space spanned by Φ_r using Galerkin projection.

As we can see, by applying POD and Galerkin projection, the original high dimensional dynamics system 18 of $\mathbf{u}$ with dimension n becomes a lower dimensional dynamics system of $\mathbf{a}$ in new coordinate system with dimension $r \ll n$.

Let $\mathbf{X} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_m]$ be the centered snapshots of m time. For computational efficiency, there are two ways to compute POD basis:

1. standard POD: $n \ll m$ compute eigen-decomposition of XX^T
2. snapshot POD: $n \gg m$ compute eigen-decomposition of $X^T X$ [1]

Note: One need to check the relation between Snapshot POD and dual PCA.

10 Applications

$\boxed{\text{Applications}}$ Deal with images with noise. **Question:** What is the feature in this problem? One could treat PCA as a approximation of matrix by rank one matrix! And each rank one matrix represents a feature!

References

Ghodsi, Ali. *Principal Component Analysis, Lec1 & Lec2*. Youtube. 2017. URL: `https://www.youtube.com/watch?v=L-pQtGm3VS8`.

Ghojogh, Benyamin and Mark Crowley. “Unsupervised and supervised principal component analysis: Tutorial”. In: *arXiv preprint arXiv:1906.03148* (2019).

Gong, Yuezheng, Qi Wang, and Zhu Wang. “Structure-preserving Galerkin POD reduced-order modeling of Hamiltonian systems”. In: *Computer Methods in Applied Mechanics and Engineering* 315 (2017), pp. 780–798.

Lee, Kyunghoon. *Investigation of probabilistic principal component analysis compared to proper orthogonal decomposition methods for basis extraction and missing data estimation*. Georgia Institute of Technology, 2010.

Shlens, Jonathon. “A tutorial on principal component analysis”. In: *arXiv preprint arXiv:1404.1100* (2014).

[1] Sirovich, “Turbulence and the dynamics of coherent structures. I. Coherent structures”

Sirovich, Lawrence. “Turbulence and the dynamics of coherent structures. I. Coherent structures”. In: *Quarterly of applied mathematics* 45.3 (1987), pp. 561–571.

Wang, Zhu, Brian McBee, and Traian Iliescu. “Approximate partitioned method of snapshots for POD”. In: *Journal of Computational and Applied Mathematics* 307 (2016), pp. 374–384.

Weiss, Julien. “A tutorial on the proper orthogonal decomposition”. In: *AIAA Aviation 2019 Forum*. 2019, p. 3333.