Principal Component and Correspondence Analyses: Multivariate Approach

Optum Global Analytics

23rd Nov' 2020

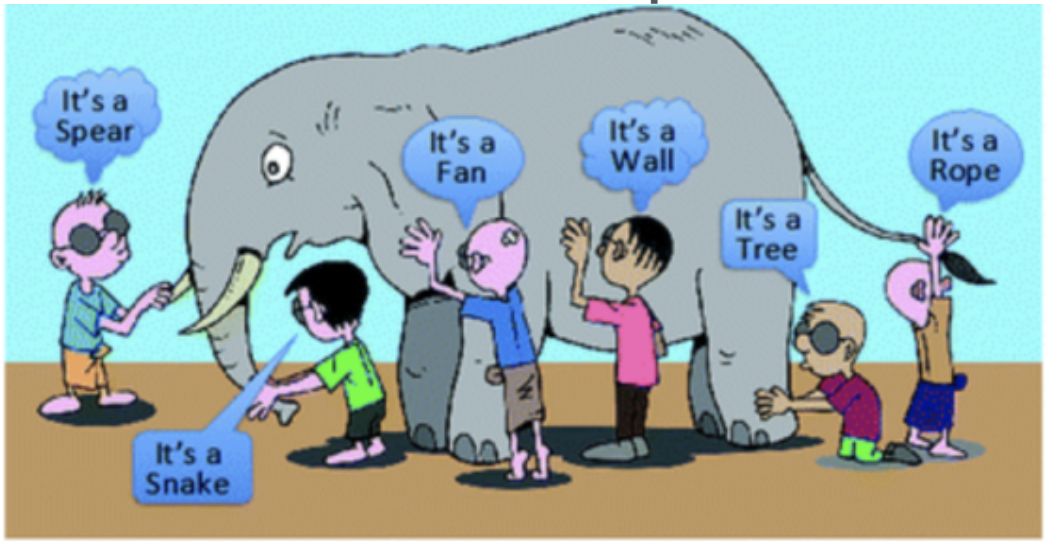
Agenda

- ❑ Intuitive way of thinking Multi Var. Statistics and PCA
- ❑ Concept of Dimensions Reduction
- ❑ Linear projections using PCA and its algebraic derivations
- ❑ PCA as maximizing variance and minimizing reconstruction loss
- ❑ Singular value decomposition (SVD)
- ❑ Non-Linear Projection /Kernelizing PCA
- ❑ Correspondence Analyses
- ❑ Use case Analyses using Python

OPTUM®

Intuition: Multivariate Statistics

The blind men and the elephant

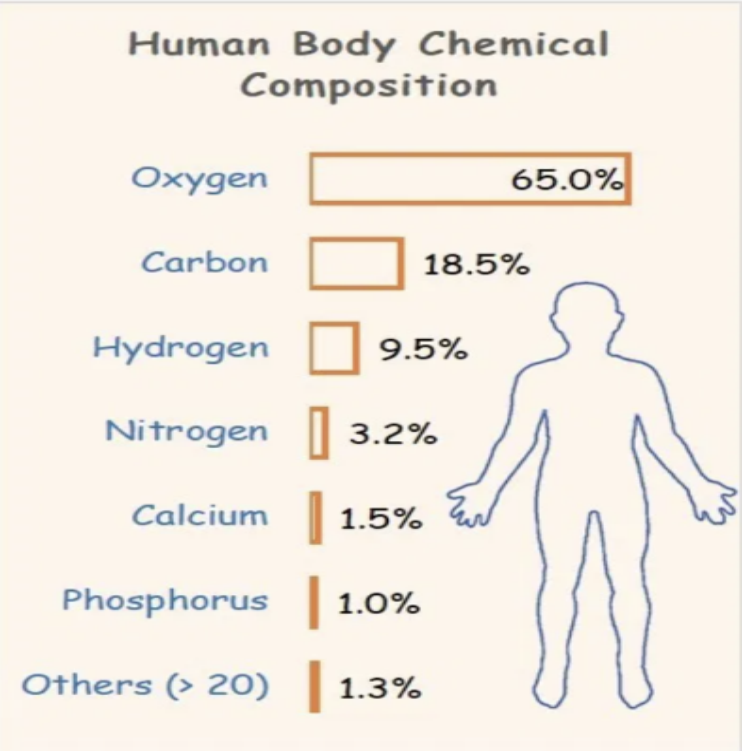


The Blind Men and the Elephant – What is it?

Six blind men who stumble across a strange creature and they try to understand what it is:

The first blind man feels the elephant's body and comes to the conclusion that the creature is in fact a wall, while his friend feels the tusk and declares that the elephant is a spear. Another shakes his head after feeling the trunk of the elephant and claims with some anxiety that it's actually a snake. The fourth blind man, whilst feeling the elephant's leg, states that they are incorrect and that it's indeed a tree. The next man has got hold of the elephant's large ear and announces that it's a fan, whilst the last blind man, who has hold of its tail, declares that his friends are all wrong and that this elephant is in fact a rope.

Intuition: Principal Component Analyses(PCA)

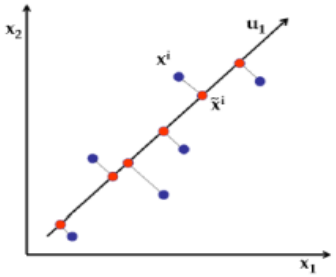


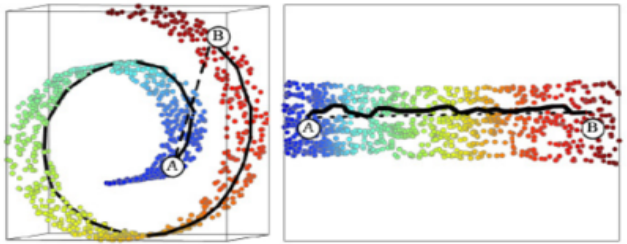
What happens when people die? Where do they go? All of us have pondered this at some point or other. There are several religious and a few non-religious theories to explain events after death. A theory which makes sense to me is: after death, we go back to our fundamental elements. Here, I am not talking about metaphysical or divine elements but chemistry. There are close to 120 known elements including lithium, oxygen, uranium, hydrogen, argon, and carbon. Despite the choice of roughly 120 elements, the human body does not have all of them in the same proportion. Incidentally, six elements, as shown in the chart, constitute close to 99% of the human body. Think of these elements as the principal components of the human body. The remaining ~1% of the human body has a little over 20 other elements. This means, around three-fourth of the elements in the periodic table are completely absent in the human body.

We noticed that about 1% of the human body has formed with ~ 20 elements. **The idea with PCA is that if you remove these 20 elements you will lose just 1% of the essence of the human body. This will also mean that your information load will decline by ~77% (20/26).** These ideas will form the basis of our understanding of principal component analysis

Dimensionality Reduction

- Usually considered an unsupervised learning method
- Used for learning the low-dimensional structures in the data



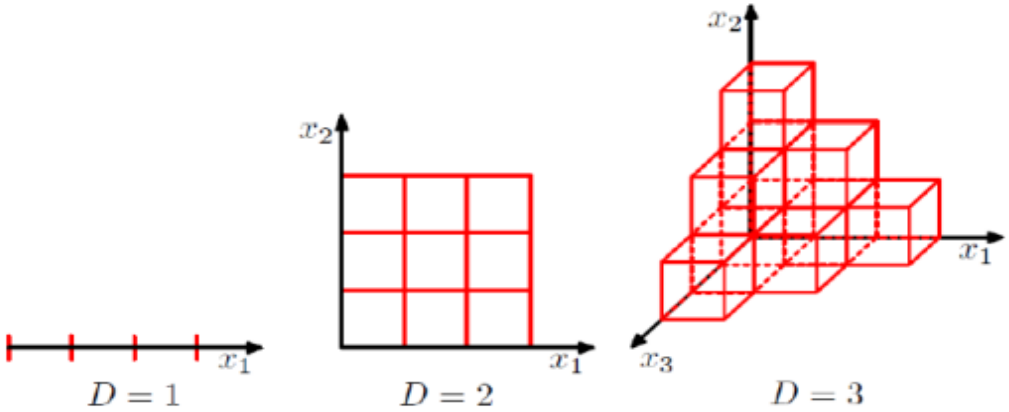


- Also useful for "feature learning" or "representation learning" (learning a better, often smaller-dimensional, representation of the data), e.g.,
  - Documents using using topic vectors instead of bag-of-words vectors
  - Images using their constituent parts (faces - eigenfaces)
- Can be used for speeding up learning algorithms
- Can be used for data compression

OPTUM®

Curse of Dimensionality

- Exponentially large # of examples required to "fill up" high-dim spaces

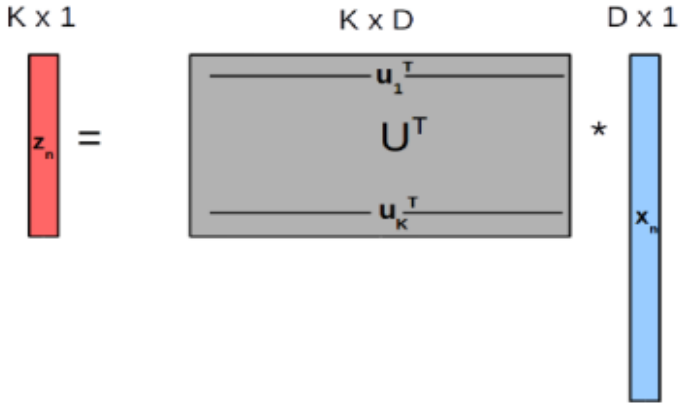


- Fewer dimensions ⇒ Less chances of overfitting ⇒ Better generalization
- Dimensionality reduction is a way to beat the curse of dimensionality

OPTUM

Linear Dimensionality Reduction

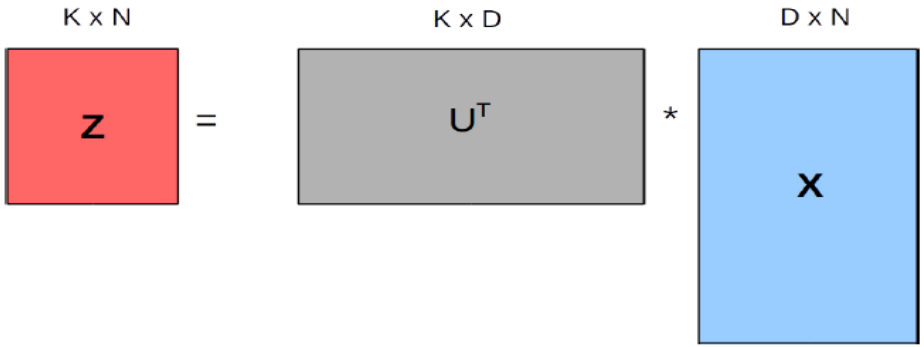
- A projection matrix $\mathbf{U} = [\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \ldots \ \boldsymbol{u}_K]$ of size $D \times K$ defines K linear projection directions, each $\boldsymbol{u}_k \in \mathbb{R}^D$, for the D dim. data (assume $K < D$)
- Can use $\mathbf{U}$ to transform $\boldsymbol{x}_n \in \mathbb{R}^D$ into $\boldsymbol{z}_n \in \mathbb{R}^K$ as shown below



- Note that $\boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n = [\boldsymbol{u}_1^\top \boldsymbol{x}_n \ \boldsymbol{u}_2^\top \boldsymbol{x}_n \ \ldots \ \boldsymbol{u}_K^\top \boldsymbol{x}_n]$ is a K-dim projection of $\boldsymbol{x}_n$
  - $\boldsymbol{z}_n \in \mathbb{R}^K$ is also called low-dimensional "embedding" of $\boldsymbol{x}_n \in \mathbb{R}^D$

Linear Dimensionality Reduction

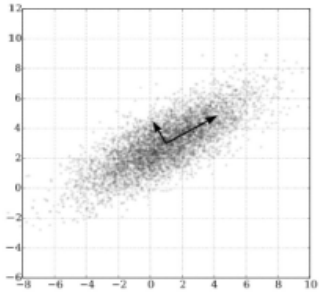
- $\mathbf{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \ldots\ \boldsymbol{x}_N]$ is $D \times N$ matrix denoting all the N data points
- $\mathbf{Z} = [\boldsymbol{z}_1\ \boldsymbol{z}_2\ \ldots\ \boldsymbol{z}_N]$ is $K \times N$ matrix denoting **embeddings** of data points
- With this notation, the figure on previous slide can be re-drawn as below



- How do we learn the "best" projection matrix **U**?
- What criteria should we optimize for when learning **U**?
- Principal Component Analysis (PCA) is an algorithm for doing this

Principal Component Analysis (PCA)

- A classic linear dim. reduction method (Pearson, 1901; Hotelling, 1930)
- Can be seen as
  - Learning projection directions that capture maximum variance in data
  - Learning projection directions that result in smallest reconstruction error
- Can also be seen as changing the basis in which the data is represented (and transforming the features such that new features become decorrelated)

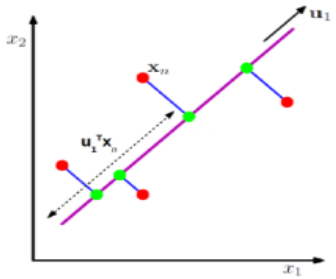


- Also related to other classic methods, e.g., Factor Analysis (Spearman, 1904)

PCA as Maximizing Variance

Variance Captured by Projections

- Consider projecting $\boldsymbol{x}_n \in \mathbb{R}^D$ on a one-dim subspace defined by $\boldsymbol{u}_1 \in \mathbb{R}^D$
- Projection/embedding of $\boldsymbol{x}_n$ along a one-dim subspace $\boldsymbol{u}_1$ $= \boldsymbol{u}_1^\top \boldsymbol{x}_n$ (location of the green point along the purple line representing $\boldsymbol{u}_1$)



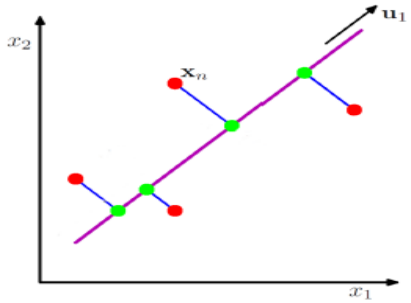
- Mean of projections of all the data: $\frac{1}{N}\sum_{n=1}^N \boldsymbol{u}_1^\top \boldsymbol{x}_n = \boldsymbol{u}_1^\top (\frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n) = \boldsymbol{u}_1^\top \boldsymbol{\mu}$
- Variance of the projected data ("spread" of the green points)

$$\frac{1}{N}\sum_{n=1}^N \left(\boldsymbol{u}_1^\top \boldsymbol{x}_n - \boldsymbol{u}_1^\top \boldsymbol{\mu}\right)^2 = \frac{1}{N}\sum_{n=1}^N \{\boldsymbol{u}_1^\top (\boldsymbol{x}_n - \boldsymbol{\mu})\}^2 = \boldsymbol{u}_1^\top \boldsymbol{S} \boldsymbol{u}_1$$

- **S** is the $D \times D$ **data covariance matrix**: $\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^N (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^\top$. If data already centered $(\boldsymbol{\mu} = 0)$ then $\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n \boldsymbol{x}_n^\top = \frac{1}{N}\mathbf{X}^\top \mathbf{X}$

OPTUM

Direction of Maximum Variance



- We want $\boldsymbol{u}_1$ s.t. the variance of the projected data is maximized

$$\arg\max_{\boldsymbol{u}_1} \ \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1$$

- To prevent trivial solution (max var. $=$ infinite), assume $||\boldsymbol{u}_1|| = 1 = \boldsymbol{u}_1^\top \boldsymbol{u}_1$
- We will find $\boldsymbol{u}_1$ by solving the following constrained opt. problem

$$\boxed{\arg\max_{\boldsymbol{u}_1} \ \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)}$$

where λ_1 is a Lagrange multiplier

OPTUM

Direction of Maximum Variance

- The objective function: $\arg\max_{\boldsymbol{u}_1} \ \boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 + \lambda_1(1 - \boldsymbol{u}_1^\top \boldsymbol{u}_1)$
- Taking the derivative w.r.t. $\boldsymbol{u}_1$ and setting to zero gives

$$\mathbf{S}\boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$$

- Thus $\boldsymbol{u}_1$ is an eigenvector of $\mathbf{S}$ (with corresponding eigenvalue λ_1)
- But which of $\mathbf{S}$'s (D possible) eigenvectors it is?
- Note that since $\boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1$, the variance of projected data is

$$\boldsymbol{u}_1^\top \mathbf{S} \boldsymbol{u}_1 = \lambda_1$$

- Var. is maximized when $\boldsymbol{u}_1$ is the (top) eigenvector with largest eigenvalue
- The top eigenvector $\boldsymbol{u}_1$ is also known as the first Principal Component (PC)
- Other directions can also be found likewise (with each being orthogonal to all previous ones) using the eigendecomposition of $\mathbf{S}$ (this is PCA)

Steps in Principal Component Analysis

1. Standardize the d-dimensional dataset.
2. Construct the covariance matrix.
3. Decompose the covariance matrix into its eigenvectors and eigenvalues.
4. Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
5. Select k eigenvectors which correspond to the k largest eigenvalues, where k is the dimensionality of the new feature subspace ($k \leq d$).
6. Construct a projection matrix $\mathbf{W}$ from the "top" k eigenvectors.
7. Transform the d-dimensional input dataset $\mathbf{X}$ using the projection matrix $\mathbf{W}$ to obtain the new k-dimensional feature subspace.

OPTUM®

PCA as Minimizing the Reconstruction Error

Data as Combination of Basis Vectors

- Assume *complete* orthonormal basis vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_D$, each $\boldsymbol{u}_d \in \mathbb{R}^D$
- We can represent each data point $\boldsymbol{x}_n \in \mathbb{R}^D$ **exactly** using this new basis

$$\boldsymbol{x}_n = \sum_{k=1}^{D} z_{nk} \boldsymbol{u}_k$$

$$\begin{bmatrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nD} \end{bmatrix} = \begin{bmatrix} | & | & & | \\ u_1 & u_2 & \cdots & u_D \\ | & | & & | \end{bmatrix} * \begin{bmatrix} z_{n1} \\ z_{n2} \\ \vdots \\ z_{nD} \end{bmatrix}$$

- Denoting $\boldsymbol{z}_n = [z_{n1} \; z_{n2} \ldots z_{nD}]^\top$, $\mathbf{U} = [\boldsymbol{u}_1 \; \boldsymbol{u}_2 \; \ldots \boldsymbol{u}_D]$, and using $\mathbf{U}^\top \mathbf{U} = \mathbf{I}_D$

$$\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n \quad \text{and} \quad \boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n$$

- Also note that each component of vector $\boldsymbol{z}_n$ is $z_{nk} = \boldsymbol{u}_k^\top \boldsymbol{x}_n$

OPTUM

Reconstruction of Data from Projections

- Reconstruction of $\boldsymbol{x}_n$ from $\boldsymbol{z}_n$ will be exact if we use all the D basis vectors
- Will be approximate if we only use $K < D$ basis vectors: $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$
- Let's use $K = 1$ basis vector. Then the one-dim embedding of $\boldsymbol{x}_n$ is

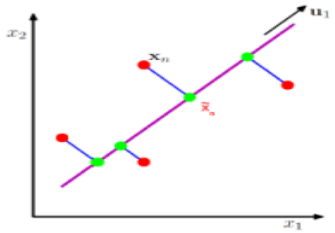
$$\boldsymbol{z}_n = \boldsymbol{u}_1^\top \boldsymbol{x}_n \qquad \text{(note: this will just be a scalar)}$$

- We can now try "reconstructing" $\boldsymbol{x}_n$ from its embedding $\boldsymbol{z}_n$ as follows

$$\tilde{\boldsymbol{x}}_n = \boldsymbol{u}_1 \boldsymbol{z}_n = \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{x}_n$$

- Total error or "loss" in reconstructing all the data points

$$= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \tilde{\boldsymbol{x}}_n||^2 = \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1 \boldsymbol{u}_1^\top \boldsymbol{x}_n||^2$$



Direction with Best Reconstruction

- We want to find $\boldsymbol{u}_1$ that minimizes the reconstruction error

$$
\begin{aligned}
L(\boldsymbol{u}_1) &= \sum_{n=1}^{N} ||\boldsymbol{x}_n - \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n||^2 \\
&= \sum_{n=1}^{N} \{\boldsymbol{x}_n^\top \boldsymbol{x}_n + (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n)^\top (\boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n) - 2\boldsymbol{x}_n^\top \boldsymbol{u}_1\boldsymbol{u}_1^\top \boldsymbol{x}_n\} \\
&= \sum_{n=1}^{N} -\boldsymbol{u}_1^\top \boldsymbol{x}_n\boldsymbol{x}_n^\top \boldsymbol{u}_1 \qquad \text{(using } \boldsymbol{u}_1^\top \boldsymbol{u}_1 = 1 \text{ and ignoring constants w.r.t. } \boldsymbol{u}_1\text{)}
\end{aligned}
$$

- Thus the problem is equivalent to the following maximization

$$
\underset{\boldsymbol{u}_1 : ||\boldsymbol{u}_1||^2 = 1}{\arg\max}\ \boldsymbol{u}_1^\top \left(\frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n\boldsymbol{x}_n^\top \right) \boldsymbol{u}_1 = \underset{\boldsymbol{u}_1 : ||\boldsymbol{u}_1||^2 = 1}{\arg\max}\ \boldsymbol{u}_1^\top \mathbf{S}\boldsymbol{u}_1
$$

  where $\mathbf{S}$ is the covariance matrix of the data (data assumed centered)

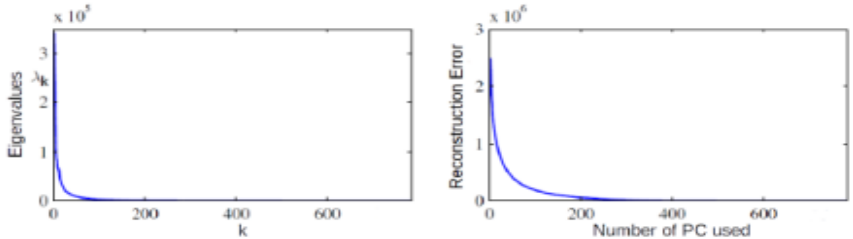
- It's the same objective that we had when we maximized the variance

How many Principal Components to Use?

- Eigenvalue λ_k measures the variance captured by the corresponding PC $\boldsymbol{u}_k$
- The "left-over" variance will therefore be

$$\sum_{k=K+1}^{D} \lambda_k$$

- Can choose K by looking at what fraction of variance is captured by the first K PCs
- Another direct way is to look at the spectrum of the eigenvalues plot, or the plot of reconstruction error vs number of PC



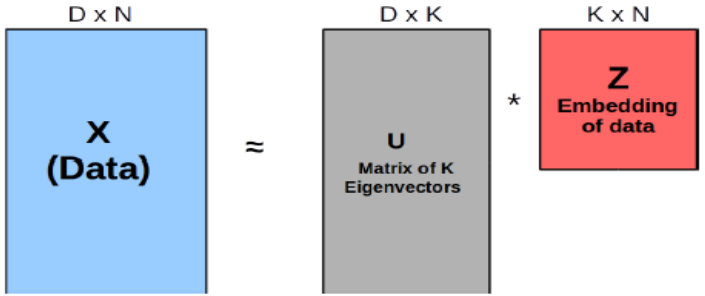
- Can also use other criteria such as AIC/BIC (or more advanced probabilistic approaches to PCA using nonparametric Bayesian methods)

PCA as Matrix Factorization

- Note that PCA represents each $\boldsymbol{x}_n$ as $\boldsymbol{x}_n = \mathbf{U}\boldsymbol{z}_n$
- When using only $K < D$ components, $\boldsymbol{x}_n \approx \mathbf{U}\boldsymbol{z}_n$
- For all the N data points, we can write the same as

$$\mathbf{X} \approx \mathbf{UZ}$$

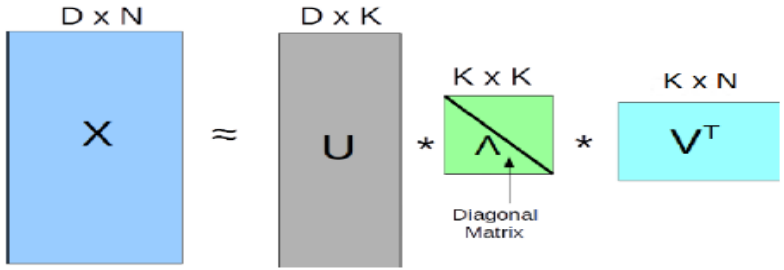
where $\mathbf{X}$ is $D \times N$, $\mathbf{U}$ is $D \times K$ and $\mathbf{Z}$ is $K \times N$



- The above approx. is equivalent to a low-rank matrix factorization of $\mathbf{X}$
  - Also closely related to Singular Value Decomposition (SVD); see next slide

PCA and SVD

- A rank-K SVD approximates a data matrix $\mathbf{X}$ as follows: $\mathbf{X} \approx \mathbf{U}\Lambda\mathbf{V}^\top$



- $\mathbf{U}$ is $D \times K$ matrix with top K left singular vectors of $\mathbf{X}$
- Λ is a $K \times K$ diagonal matrix (with top K singular values)
- $\mathbf{V}$ is $N \times K$ matrix with top K right singular vectors of $\mathbf{X}$
- Rank-K SVD is based on minimizing the reconstruction error

$$||\mathbf{X} - \mathbf{U}\Lambda\mathbf{V}^\top||$$

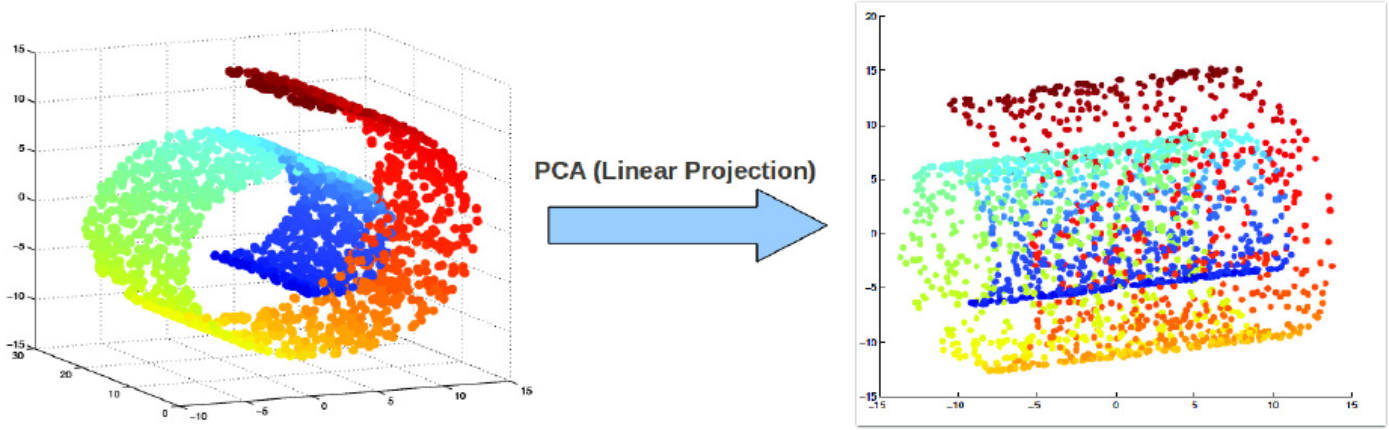
- PCA is equivalent to the best rank-K SVD after centering the data

PCA: Limitations and Extensions

- A linear projection method
  - Won't work well if data can't be approximated by a linear subspace
  - But PCA can be kernelized easily (Kernel PCA)

Beyond Linear Projections..

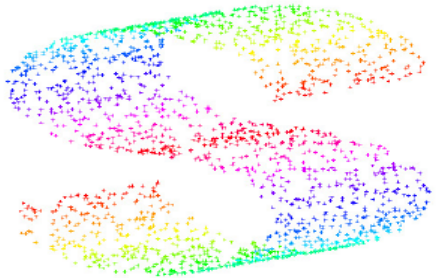
Consider the swiss-roll dataset (points lying close to a manifold):

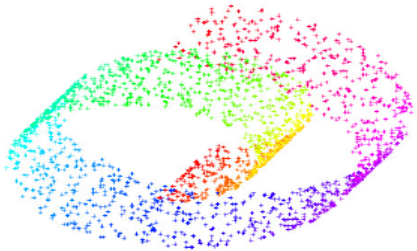


Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities

Nonlinear Dimensionality Reduction

- Given: Low-dim. surface embedded **nonlinearly** in high-dim. space
  - Such a structure is called a **Manifold**



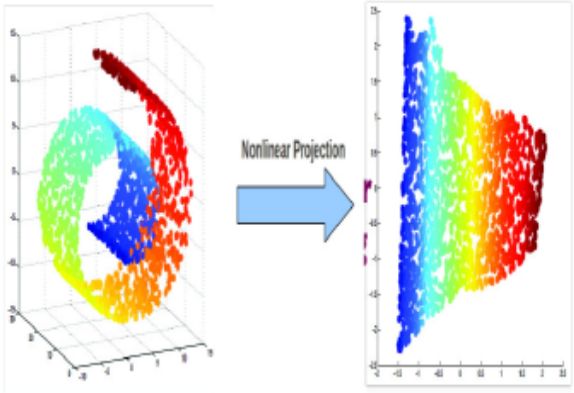


- Goal: Recover the low-dimensional surface

Nonlinear Dimensionality Reduction

Usually two ways of doing this nonlinear low-dim projection:

- **Nonlinearize a linear dimensionality reduction method.**
- ✓ Kernel PCA (nonlinear PCA)
- **Using manifold based methods that intrinsically preserve nonlinear geometry**
- ✓ Locally Linear Embedding (LLE)
- ✓ Isomap
- ✓ Maximum Variance Unfolding
- ✓ Laplacian Eigenmaps
- ✓ And others (tSNE [t-distributed stochastic neighbour embedding], Hessian LLE, etc.)



OPTUM

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \mathbf{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

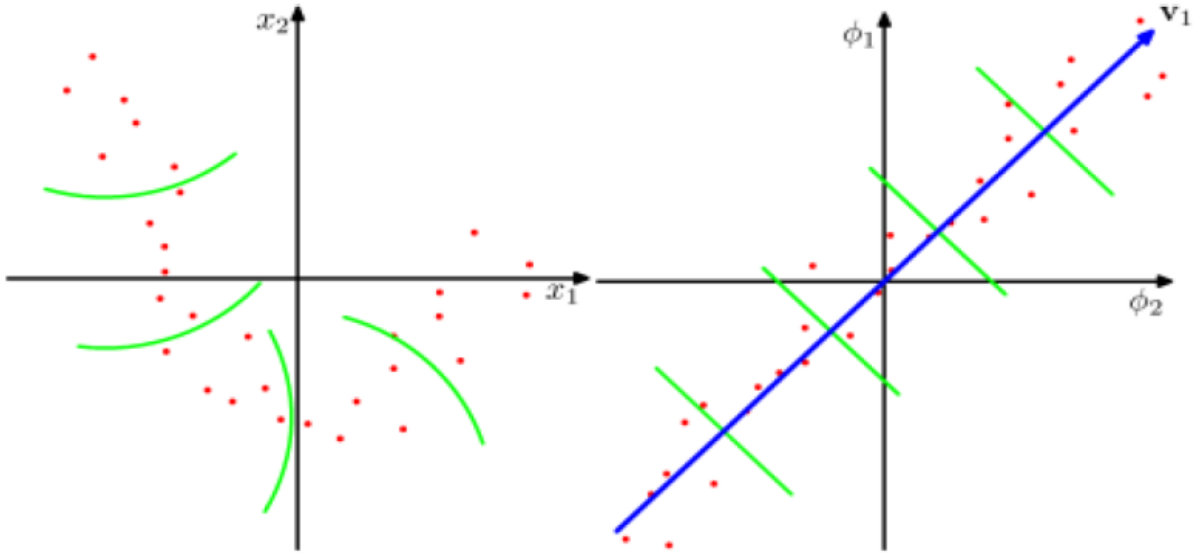
- PCA computes eigenvectors $\boldsymbol{u}_i$ which satisfy $\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \; \forall i = 1, \ldots, D$
- Let's assume a kernel k with associated M dimensional nonlinear map ϕ
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\boldsymbol{x}_n) = \mathbf{0}$)

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n) \phi(\boldsymbol{x}_n)^\top$$

- Kernel PCA: Compute eigenvectors $\boldsymbol{v}_i$ satisfying: $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \; \forall i = 1, \ldots, M$
- We would like to do this without having to compute $\mathbf{C}$ or $\phi(\boldsymbol{x}_n)$'s

** A "**kernel**" is usually used to refer to the **kernel** trick, a method of using a linear classifier to solve a non-linear problem.

Kernel PCA



Right figure: After mapping the data via ϕ, data is now close to a linear subspace

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional
- Plugging in the expression for $\mathbf{C}$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

- Denoting $\alpha_{in} = \frac{1}{\lambda_i N}\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i$, $\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$ (also recall Rep. Thm.)
- Thus we can get $\boldsymbol{v}_i$ by finding $\boldsymbol{\alpha}_i = [\alpha_{i1} \ \ldots \ \alpha_{iN}]$
- Plugging this back in the eigenvector equation $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \sum_{m=1}^{N}\alpha_{im}\phi(\boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$$

- Pre-multiplying both sides by $\phi(\boldsymbol{x}_\ell)^\top$ and re-arranging

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_\ell)^\top\phi(\boldsymbol{x}_n)\sum_{m=1}^{N}\alpha_{im}\phi(\boldsymbol{x}_n)^\top\phi(\boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_\ell)^\top\phi(\boldsymbol{x}_n)$$

OPTUM

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)\sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\boldsymbol{\alpha}_i$ as the $N \times 1$ vector with elements α_{in}
- Using $\boldsymbol{K}$ and $\boldsymbol{\alpha}_i$, the eigenvector equation becomes:

$$\boldsymbol{K}^2\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{K}\boldsymbol{\alpha}_i \quad \Rightarrow \quad \boxed{\boldsymbol{K}\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{\alpha}_i}$$

- Thus $\boldsymbol{\alpha}_i$ is an eigenvector of the $N \times N$ kernel matrix $\boldsymbol{K}$
- Note: Since $\boldsymbol{v}_i^\top \boldsymbol{v}_i = 1$ and $\boldsymbol{v}_i = \sum_{n=1}^{N} \alpha_{in}\phi(\boldsymbol{x}_n)$, we have $\boldsymbol{\alpha}_i^\top \boldsymbol{K}\boldsymbol{\alpha}_i = 1$, which means $\boldsymbol{\alpha}_i^\top \lambda_i N \boldsymbol{\alpha}_i = 1$ and $\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i = 1/(\lambda_i N)$. Thus the original solution with $\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i = 1$ (eigenvec with norm 1) needs to be re-scaled as $\tilde{\alpha}_{in} = \alpha_{in}/\sqrt{\lambda_i N}$

Kernel PCA

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m) \\
&= \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)^\top \phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_\ell) \\
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_n, \boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell)
\end{aligned}$$

- In matrix notation, the centered $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{1}_N\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1}_N + \boldsymbol{1}_N\boldsymbol{K}\boldsymbol{1}_N$
- $\boldsymbol{1}_N$ is the $N \times N$ matrix with every element $= 1/N$
- Eigen-decomposition is then done for the centered kernel matrix $\tilde{\boldsymbol{K}}$

OPTUM

Kernel PCA: Computing the Embeddings

- Suppose $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L\}$ are the top L eigenvectors of kernel matrix $\tilde{\boldsymbol{K}}$
- The L-dimensional KPCA projection $\boldsymbol{z}_m = [z_{m1}, \ldots, z_{mL}]$ of a point $\boldsymbol{x}_m$:

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell \qquad \forall \ell = 1, \ldots, L$$

- Using the definition of $\boldsymbol{v}_\ell$, i.e., $\boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell n} \phi(\boldsymbol{x}_n)$, we have

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- Note: Cost of computing the embeddings scales in N
- Note: For linear kernel, KPCA reduces to PCA (but more efficient if $N < D$)

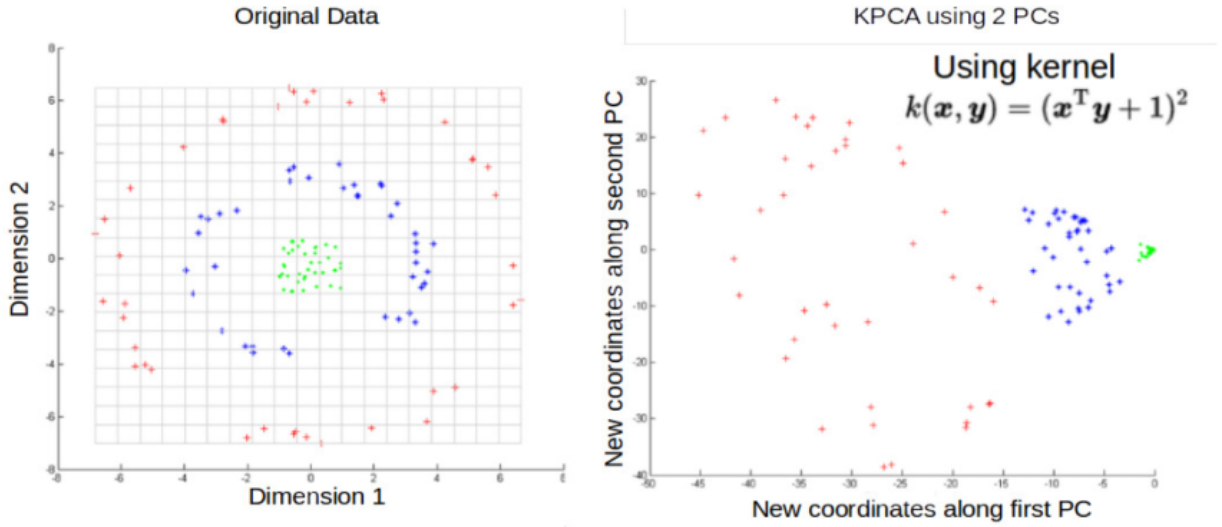
Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$
- Center $\mathbf{K}$ as follows $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix of all $1/N$
- Do eigen-decomposition of $\tilde{\mathbf{K}}$ and find top L eigenvecs $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_L$ with eigenvals $\lambda_1, \lambda_2, \ldots, \lambda_L$
- Re-scale each eigenvector as $\tilde{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i / \sqrt{\lambda_i N}$, $\forall i = 1, \ldots, L$
- Finally, compute embedding $\boldsymbol{z}_m \in \mathbb{R}^L$ of any point $\boldsymbol{x}_m$ as

$$z_{m\ell} = \sum_{n=1}^{N} \tilde{\alpha}_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- Note: For compactness, the $L \times N$ matrix of all L eigenvectors (each is N dimensional) can be written as $\tilde{\boldsymbol{\alpha}} = [\tilde{\boldsymbol{\alpha}}_1 \ \tilde{\boldsymbol{\alpha}}_2 \ \ldots \ \tilde{\boldsymbol{\alpha}}_L]^\top$. Thus we can also write embedding $\boldsymbol{z}_m \in \mathbb{R}^L$ as $\boldsymbol{z}_m = \tilde{\boldsymbol{\alpha}}\boldsymbol{k}_m$ where $\boldsymbol{k}_m$ is $N \times 1$ vector of kernelized similarities of $\boldsymbol{x}_m$ with all the N training data points

Kernel PCA: An Example



Note that even if we throw away the 2nd PC, we get a good 1-D embedding

Thank you

Optum Global Analytics

Swarupananda Adhikari

swarupananda_adhikari@optum.com