10.09.2024

Statistical Methods in AI (CS7.403)

Lecture-11: PCA – 2, PPCA, FA, Feature Selection

Ravi Kiran (ravi.kiran@iiit.ac.in)

https://ravika.github.io

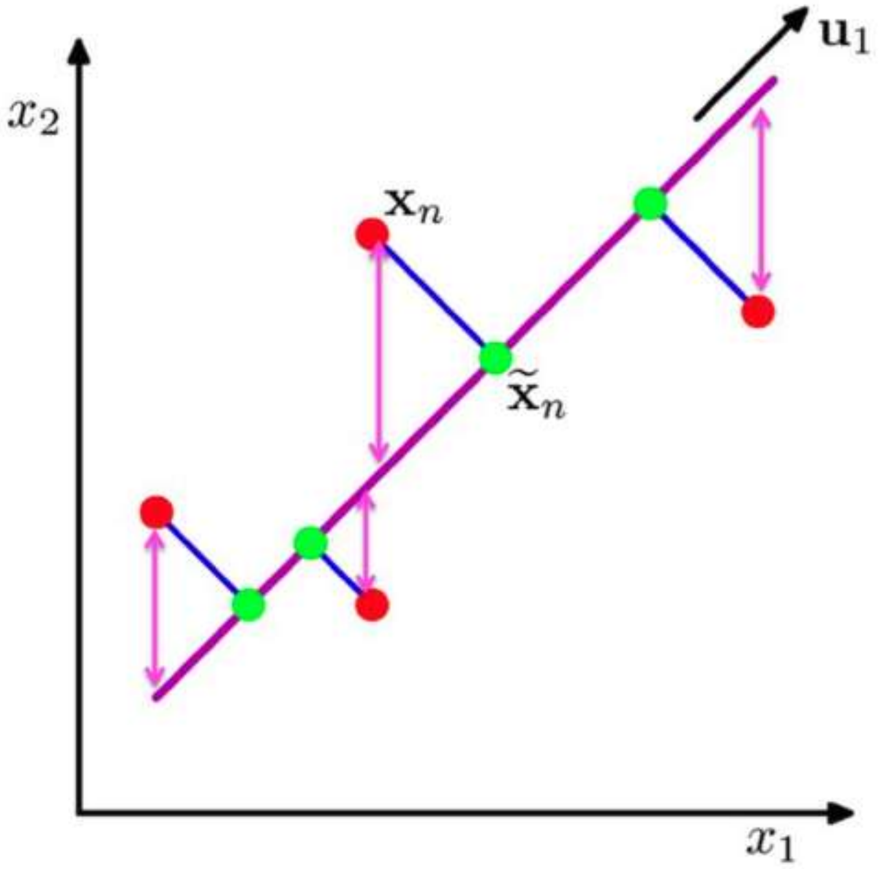
@vikataravi

Center for Visual Information Technology (CVIT)

IIIT Hyderabad

PCA: Why eigenvectors ?

PCA vs linear regression



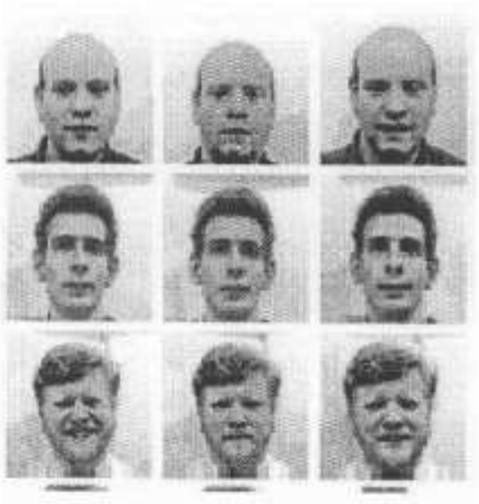
In contrast: in regression we'd minimize square error on *one* dimension (x_2) using a linear combination the *other dimensions*

Application to Faces

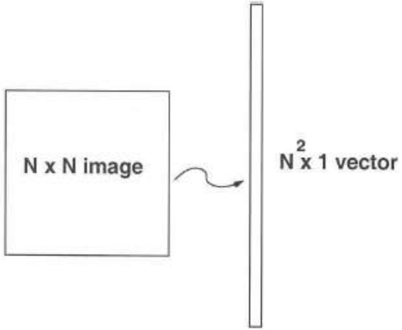
- Computation of low-dimensional basis (i.e.,eigenfaces):

Step 1: obtain face images $I_1, I_2, ..., I_M$ (training faces)

(**very important:** the face images must be *centered* and of the same *size*)



Step 2: represent every image I_i as a vector Γ_i



Application to Faces

- Computation of the eigenfaces – cont.

Step 3: compute the average face vector Ψ:

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

Step 4: subtract the mean face:

$$\Phi_i = \Gamma_i - \Psi$$

Step 5: compute the covariance matrix C:

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = \frac{1}{M} A A^T \quad (N^2 \mathrm{x} N^2 \text{ matrix})$$

where $A = [\Phi_1 \; \Phi_2 \cdots \Phi_M] \quad (N^2 \mathrm{x} M \text{ matrix})$

Application to Faces

- Computation of the eigenfaces – cont.

Step 6: compute the eigenvectors u_i of AA^T ➡ $AA^T u_i = \lambda_i u_i$

The matrix AA^T is very large --> not practical !!

Application to Faces

- Computation of the eigenfaces – cont.

<u>Step 6:</u> compute the eigenvectors u_i of AA^T $\Rightarrow$ $AA^T u_i = \lambda_i u_i$

<u>The matrix AA^T is very large --> not practical !!</u>

<u>Step 6.1:</u> consider the matrix $A^T A$ (M x M matrix)

<u>Step 6.2:</u> compute the eigenvectors v_i of $A^T A$

$$A^T A v_i = \mu_i v_i$$

<u>What is the relationship between u_i and v_i?</u>

Application to Faces

- Computation of the eigenfaces – cont.

Step 6: compute the eigenvectors u_i of AA^T ➡ $AA^T u_i = \lambda_i u_i$

The matrix AA^T is very large --> not practical !!

Step 6.1: consider the matrix $A^T A$ (M x M matrix)

Step 6.2: compute the eigenvectors v_i of $A^T A$

$$A^T A v_i = \mu_i v_i$$

What is the relationship between u_i nd v_i?

$A^T A v_i = \mu_i v_i$ => $AA^T A v_i = \mu_i A v_i$ =>

$u_i = Av_i$ and $\lambda_i = \mu_i$: Av_i

Thus, AA^T and $A^T A$ have the same eigenvalues and their eigenvectors are related as follows: $u_i = Av_i$!!

Application to Faces

- Computation of the eigenfaces – cont.

Note 1: AA^T can have up to N^2 eigenvalues and eigenvectors.

Note 2: $A^T A$ can have up to M eigenvalues and eigenvectors.

Note 3: The M eigenvalues of $A^T A$ (along with their corresponding eigenvectors) correspond to the M *largest* eigenvalues of AA^T (along with their corresponding eigenvectors).

Step 6.3: compute the M best eigenvectors of AA^T: $u_i = Av_i$

(**important:** normalize u_i such that $\|u_i\| = 1$)

Step 7: keep only K eigenvectors (corresponding to the K largest eigenvalues)

Eigendecomposition vs. Singular Value Decomposition

- Eigendecomposition
  - Must be a diagonalizable matrix
  - Must be a square matrix
  - Matrix (n x n size) must have n linearly independent eigenvector
    - e.g. symmetric matrix ..

$$\boxed{A} = \boxed{P}\boxed{\Lambda}\boxed{P^{-1}}$$

Eigendecomposition vs. Singular Value Decomposition

- Eigendecomposition
  - Must be a diagonalizable matrix
  - Must be a square matrix
  - Matrix (n x n size) must have n linearly independent eigenvector
    - e.g. symmetric matrix ..

$$A = P \Lambda P^{-1}$$

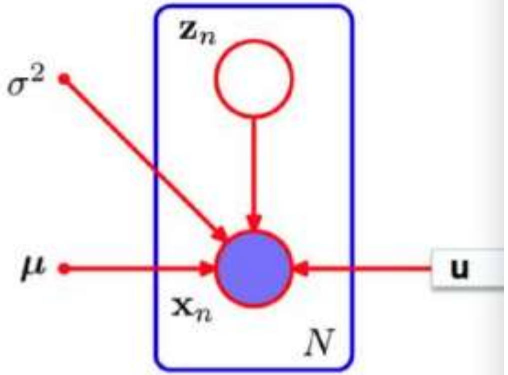
- Singular Value Decomposition
  - Computable for any size (M x n) of matrix

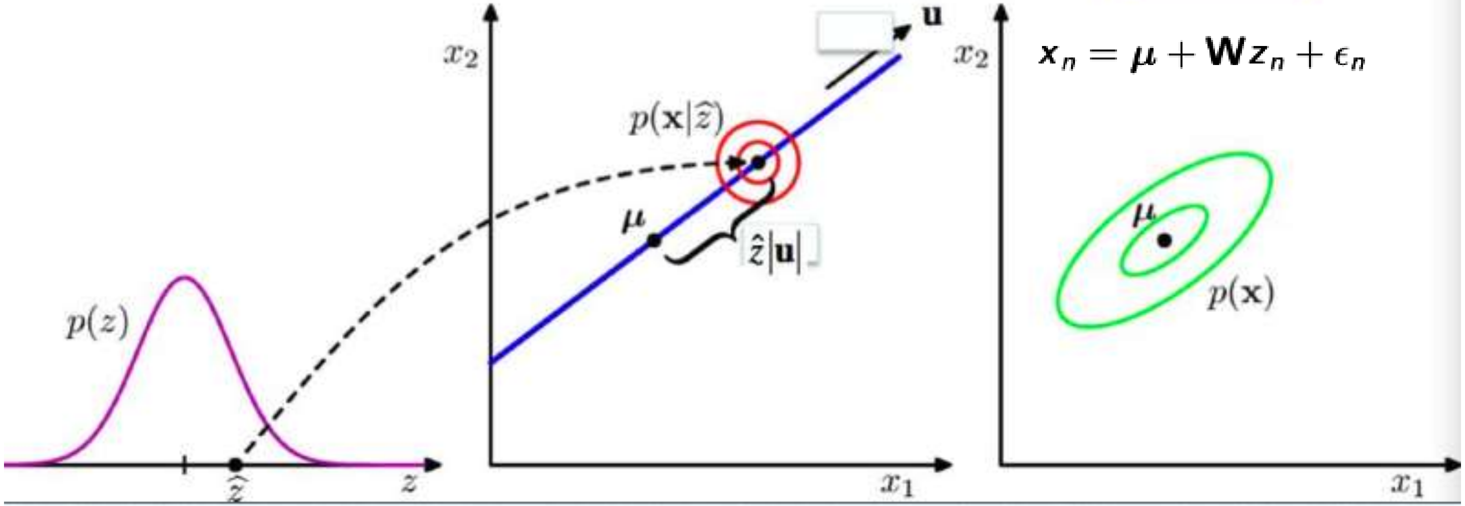
$$A = U \Sigma V^{T}$$

Probabilistic PCA

PCA

- Pick a *continuous* value z, which will be used to combine the "prototypes" **u** in the model
- Pick the the point **x** from a spherical Gaussian centered on ***zu***





Probabilistic PCA

- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$$

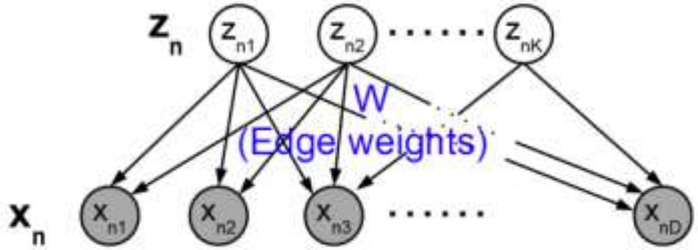
- Note: We'll assume data to be centered, otherwise $\boldsymbol{x}_n = \boldsymbol{\mu} + \mathbf{W}\boldsymbol{z}_n + \epsilon_n$

Probabilistic PCA

- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$$

- Note: We'll assume data to be centered, otherwise $\boldsymbol{x}_n = \boldsymbol{\mu} + \mathbf{W}\boldsymbol{z}_n + \epsilon_n$
- Think of it as low dimensional $\boldsymbol{z}_n \in \mathbb{R}^K$ "generating" a higher-dimensional $\boldsymbol{x}_n \in \mathbb{R}^D$ via a mapping matrix $\mathbf{W} \in \mathbb{R}^{D \times K}$, plus some noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_D)$

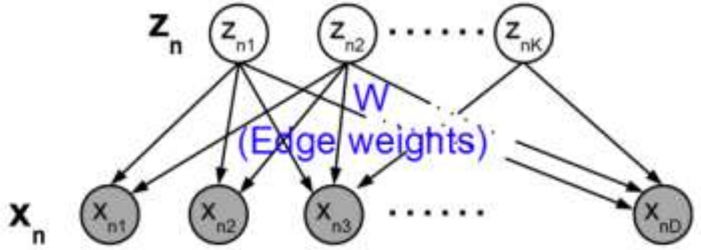


Probabilistic PCA

- Assume the following generative model for each observation $\boldsymbol{x}_n$

$$\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$$

- Note: We'll assume data to be centered, otherwise $\boldsymbol{x}_n = \boldsymbol{\mu} + \mathbf{W}\boldsymbol{z}_n + \epsilon_n$
- Think of it as low dimensional $\boldsymbol{z}_n \in \mathbb{R}^K$ "generating" a higher-dimensional $\boldsymbol{x}_n \in \mathbb{R}^D$ via a mapping matrix $\mathbf{W} \in \mathbb{R}^{D \times K}$, plus some noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_D)$



- Intuitively, this generative model is "inverse" of what the traditional PCA does. Here we assume a latent low-dim $\boldsymbol{z}_n$ that "generates" the high-dim $\boldsymbol{x}_n$ via the mapping $\mathbf{W}$ (plus adding some noise)
- A directed graphical model linking $\boldsymbol{z}_n$ and $\boldsymbol{x}_n$ via "edge weights" $\mathbf{W}$

Probabilistic PCA

- Can also write $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$ as each example $\boldsymbol{x}_n$ being a linear comb. of columns of $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$, plus some example-specific random noise ϵ_n

$$\boldsymbol{x}_n = \sum_{k=1}^{K} \boldsymbol{w}_k z_{nk} + \epsilon_n$$

- The K columns of $\mathbf{W}$ (each $\mathbb{R}^D$) are like "prototype vectors" shared by all examples. Each $\boldsymbol{x}_n$ is a linear combination of these vectors (the combination coefficients are given by $\boldsymbol{z}_n \in \mathbb{R}^K$ which is basically the low-dim rep. of $\boldsymbol{x}_n$).

Probabilistic PCA

- Can also write $\boldsymbol{x}_n = \mathbf{W}\boldsymbol{z}_n + \epsilon_n$ as each example $\boldsymbol{x}_n$ being a linear comb. of columns of $\mathbf{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K]$, plus some $\boxed{\text{example-specific random noise } \epsilon_n}$

$$\boldsymbol{x}_n = \sum_{k=1}^{K} \boldsymbol{w}_k z_{nk} + \epsilon_n$$

- The K columns of $\mathbf{W}$ (each $\mathbb{R}^D$) are like "prototype vectors" shared by all examples. Each $\boldsymbol{x}_n$ is a linear combination of these vectors (the combination coefficients are given by $\boldsymbol{z}_n \in \mathbb{R}^K$ which is basically the low-dim rep. of $\boldsymbol{x}_n$).
- Some examples:
  - In case of images, columns of $\mathbf{W}$ would correspond to "basis images"
  - In case of text documents, columns of $\mathbf{W}$ (with non-negativity imposed on it) would correspond to "topics" in the corpus

Probabilistic PCA - EM

- Since noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ is Gaussian, the conditional distrib. of $\boldsymbol{x}_n$

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

- Given a set of observations $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the goal is to learn $\mathbf{W}$ and the low-dim. representation of data, i.e., $\mathbf{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$

Probabilistic PCA - EM

- Since noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ is Gaussian, the conditional distrib. of $\boldsymbol{x}_n$

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

- Given a set of observations $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, the goal is to learn $\mathbf{W}$ and the low-dim. representation of data, i.e., $\mathbf{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$

- Assume a Gaussian prior on the low-dimensional latent representation, i.e.,

$$p(\boldsymbol{z}_n) = \mathcal{N}(0, \mathbf{I}_K)$$

- Observed data: $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, latent variable: $\mathbf{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$
- Parameters: $\mathbf{W}, \sigma^2$
- The complete data log -likelihood

$$\begin{aligned}\log p(\mathbf{X}, \mathbf{Z} | \mathbf{W}, \sigma^2) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, \boldsymbol{z}_n | \mathbf{W}, \sigma^2) &= \log \prod_{n=1}^{N} p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) p(\boldsymbol{z}_n) \\ &= \sum_{n=1}^{N} \{\log p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) + \log p(\boldsymbol{z}_n)\}\end{aligned}$$

Benefits of Probabilistic PCA

- Can handle missing data (can treat it as latent variable in E step)
- Doesn't require computing the $D \times D$ cov. matrix of data and doing expensive eigen-decomposition. When K is small (i.e., we only want few eigen vectors), this is especially nice because only inverting $K \times K$ is required

Benefits of Probabilistic PCA

- Can handle missing data (can treat it as latent variable in E step)
- Doesn't require computing the $D \times D$ cov. matrix of data and doing expensive eigen-decomposition. When K is small (i.e., we only want few eigen vectors), this is especially nice because only inverting $K \times K$ is required
- Easy to "plug-in" PPCA as part of more complex problems, e.g., mixtures of PPCA models for doing nonlinear dimensionality reduction, or subspace clustering (i.e., clustering when data in each cluster lives on a lower dimensional subspace).
- Possible to give it a fully Bayesian treatment (which has many benefits such as inferring K)

Factor Analysis

- Similar to PPCA except that the Gaussian conditional distribution $p(\boldsymbol{x}_n | \boldsymbol{z}_n)$ has diagonal instead of spherical covariance

$$\boldsymbol{x}_n \sim \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \boldsymbol{\Psi})$$

where $\boldsymbol{\Psi}$ is a diagonal matrix

Probabilistic PCA - EM

- Since noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ is Gaussian, the conditional distrib. of $\boldsymbol{x}_n$

$$p(\boldsymbol{x}_n | \boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2 \mathbf{I}_D)$$

Factor Analysis

- Similar to PPCA except that the Gaussian conditional distribution $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ has diagonal instead of spherical covariance

$$\boldsymbol{x}_n \sim \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \boldsymbol{\Psi})$$

where $\boldsymbol{\Psi}$ is a diagonal matrix

- In Factor Analysis, the projection matrix $\mathbf{W}$ is also called the Factor Loading Matrix and $\boldsymbol{z}_n$ is called the factor scores for example n

---

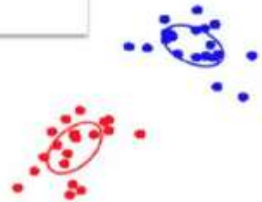
Probabilistic PCA - EM

- Since noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$ is Gaussian, the conditional distrib. of $\boldsymbol{x}_n$

$$p(\boldsymbol{x}_n|\boldsymbol{z}_n, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$$

PCA vs mixtures of Gaussians

Mixture of Gaussians



For each point:

- Pick the index of the (latent) Gaussian *Z=k*
- Pick the the point **x** from that the k-th Gaussian, **x** ~ $N(\mu_k,\Sigma_k)$

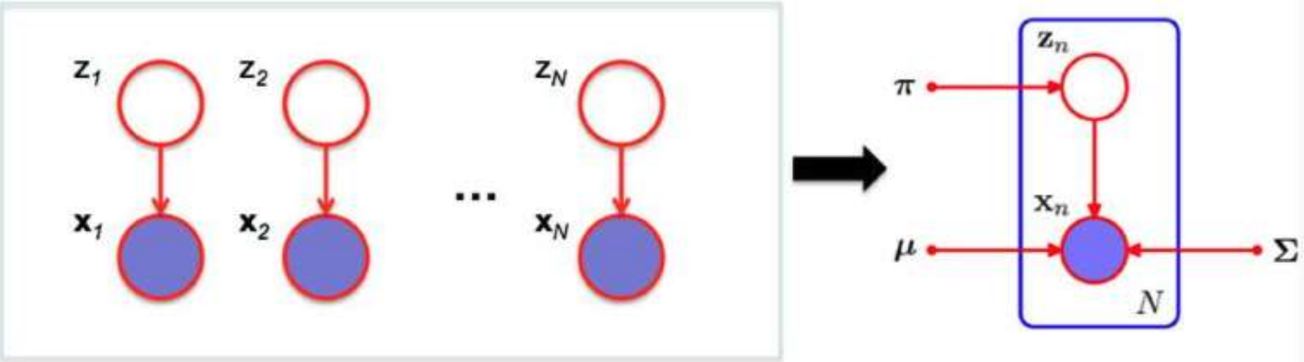
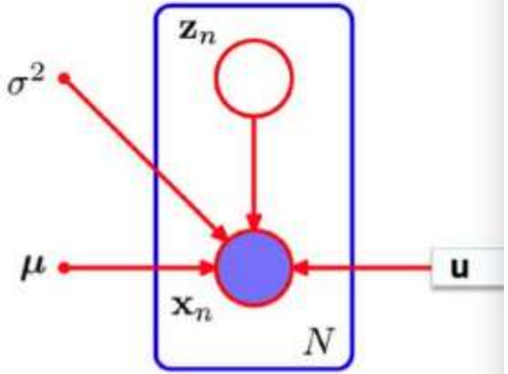


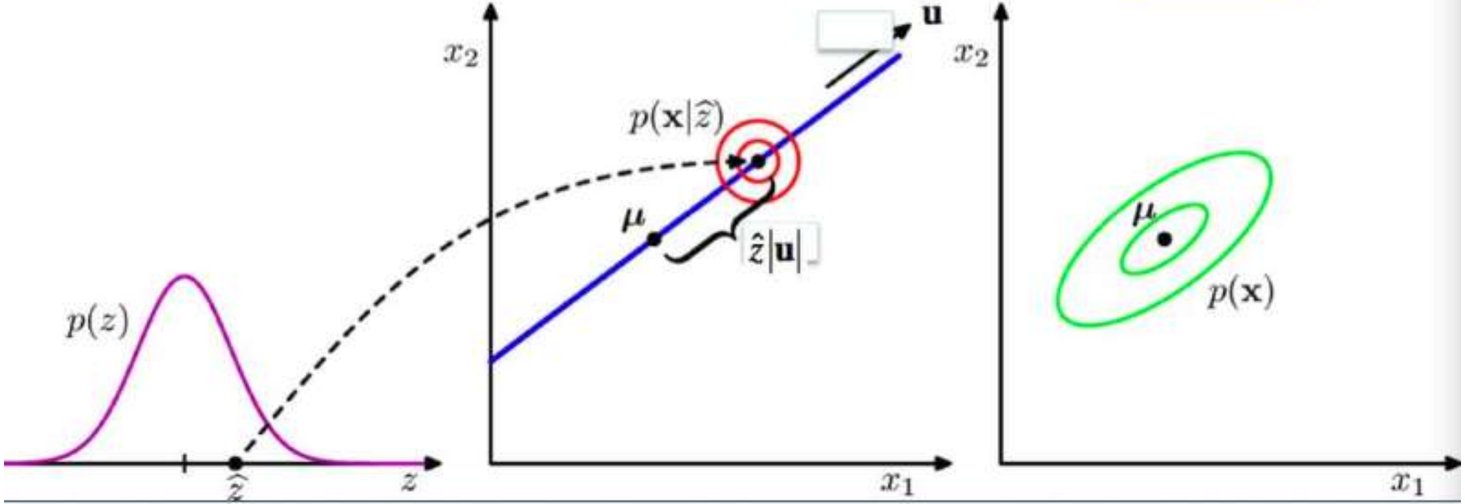
Plate notation

PCA vs mixtures of Gaussians

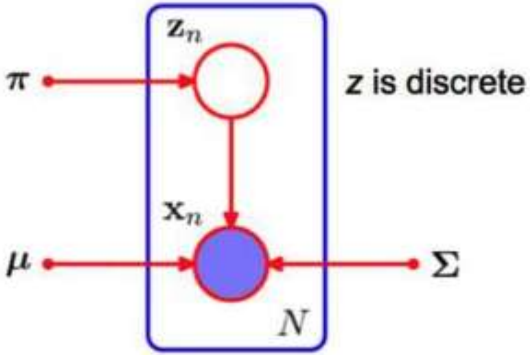
PCA

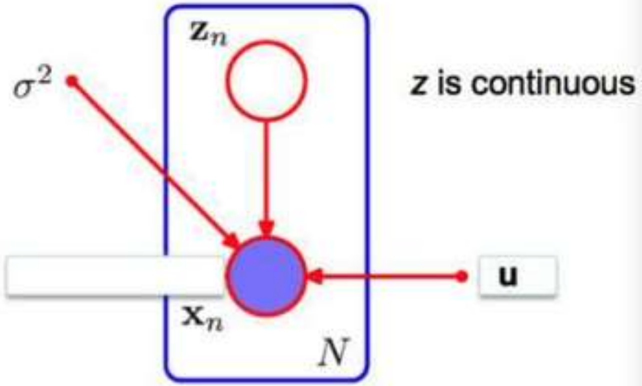
- Pick a *continuous* value z, which will be used to combine the "prototypes" **u** in the model
- Pick the the point **x** from a spherical Gaussian centered on ***zu***



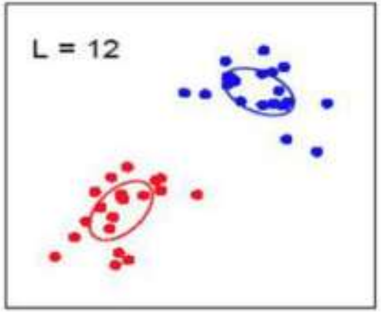


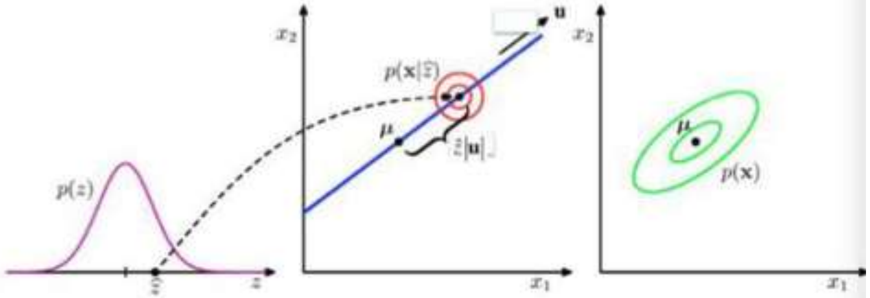
PCA vs mixtures of Gaussians



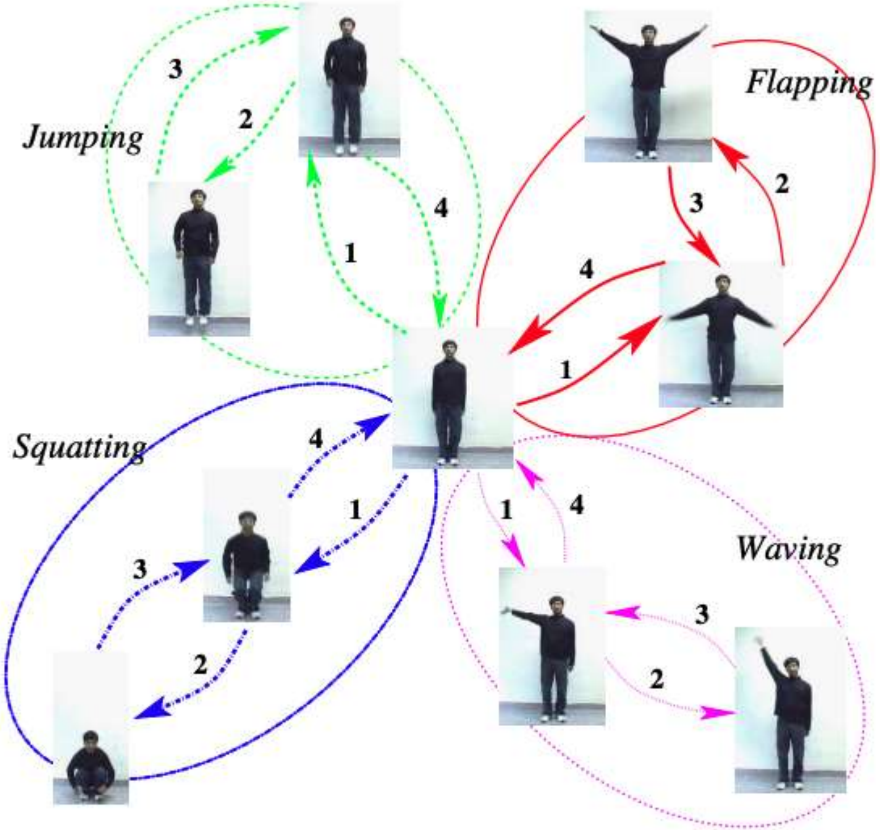


Comment: we can preprocess the data so that the mean is **0** to simplify the model





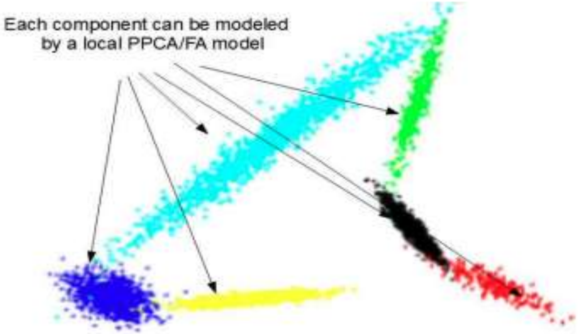
Recognizing Human Activities from Constituent Actions



https://cvit.iiit.ac.in/images/ConferencePapers/2005/ravi05humanact.pdf

Mixture of PPCAs/Mixture of Factor Analyzers

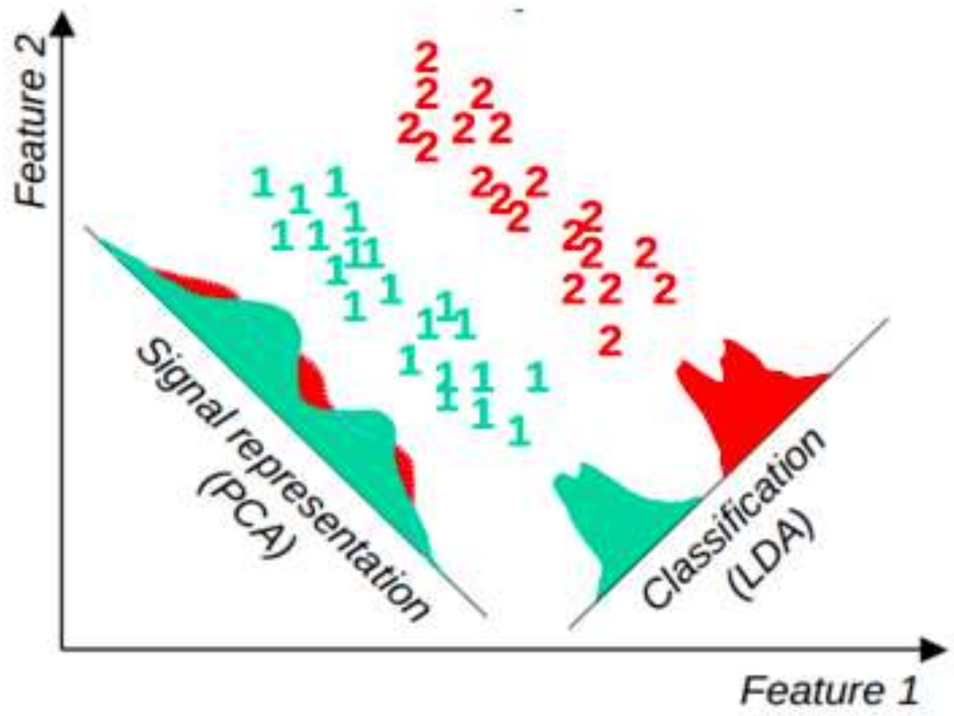
- PPCA and FA learn a linear projection of the data (i.e., are linear dimensionality reduction methods)



- Similar to mixture of Gaussians, except that now each Gaussian is replaced by a PPCA or FA model

https://www.cse.iitk.ac.in/users/piyush/courses/pml_winter16/slides_lec10.pdf

Linear Discriminant Analysis (LDA)



Source: https://www.mygreatlearning.com/blog/linear-discriminant-analysis-or-lda/

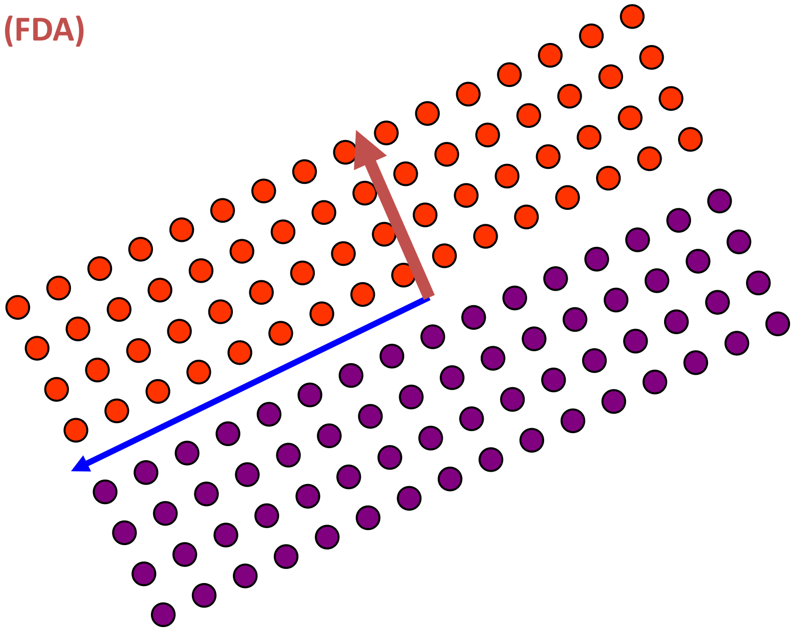
Linear Discriminant Analysis (LDA)

- Also known as “Fisher Discriminant”
- Does dimensionality Reduction
  - Also use the label “y”
  - Or Supervised Dimensionality Reduction
- PCA, LDA are both linear -- there are also **nonlinear** Dimensionality Reduction schemes (kernel PCA, t-SNE, autoencoders)

Beyond PCA

Are the maximal variance dimensions the relevant dimensions for preservation?

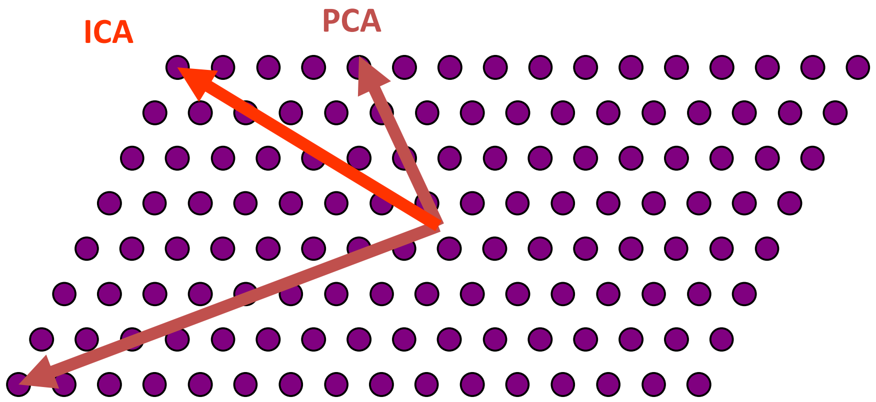
- Neighborhood Component Analysis (NCA)
- Fisher Discriminant analysis (FDA)



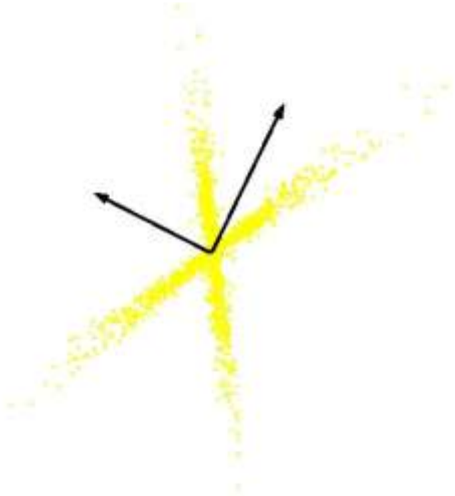
Beyond PCA

Should the goal be finding independent rather than pair-wise uncorrelated dimensions

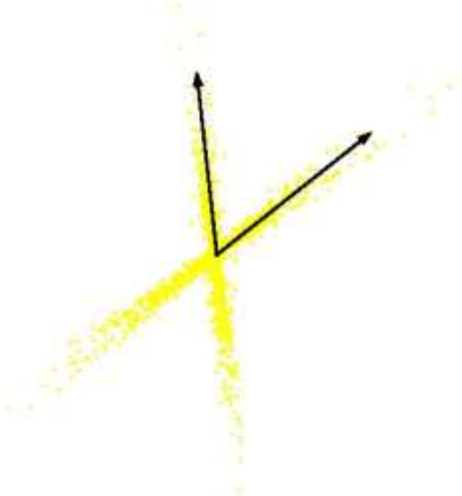
- Independent Component Analysis (ICA)



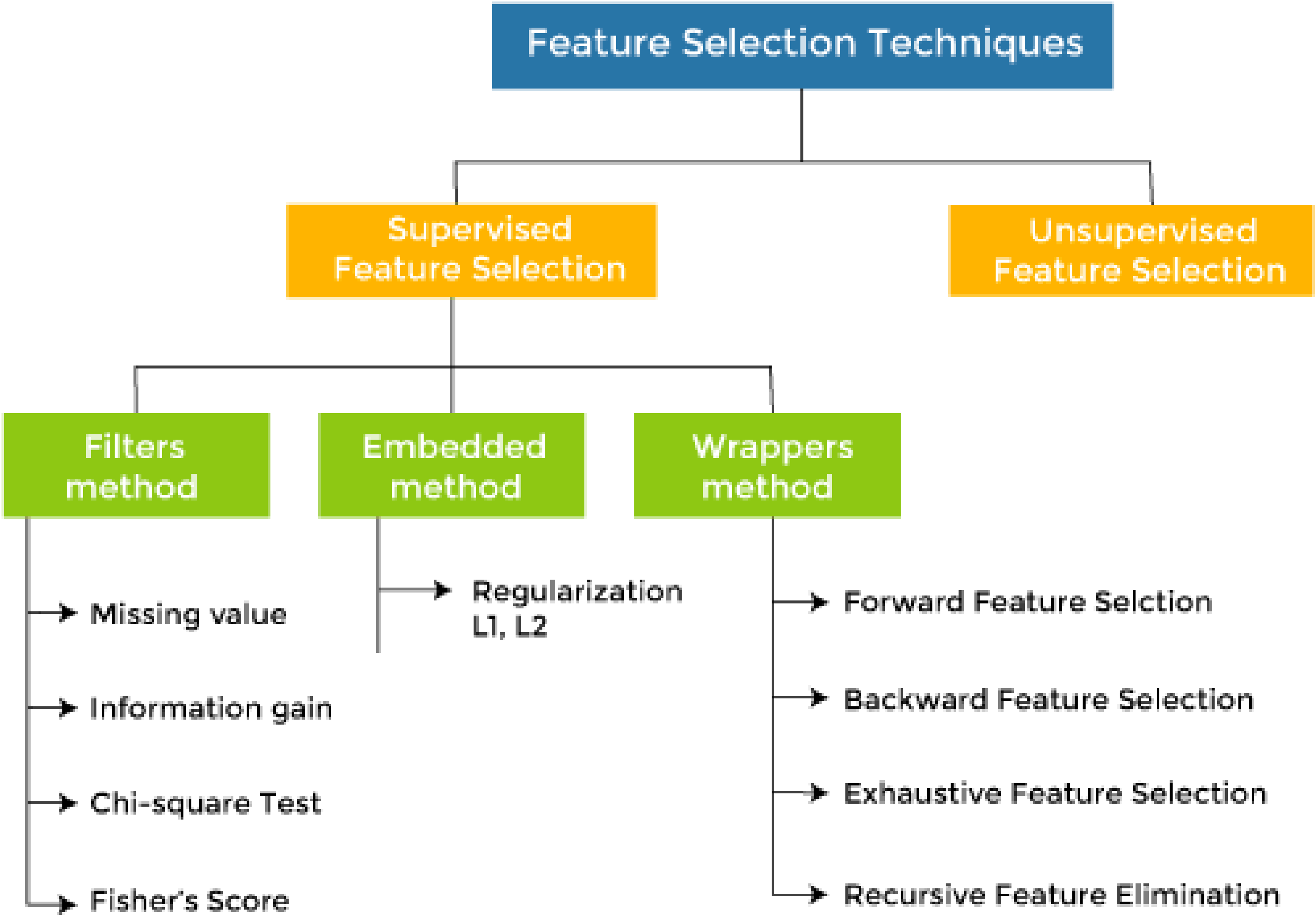
PCA vs ICA

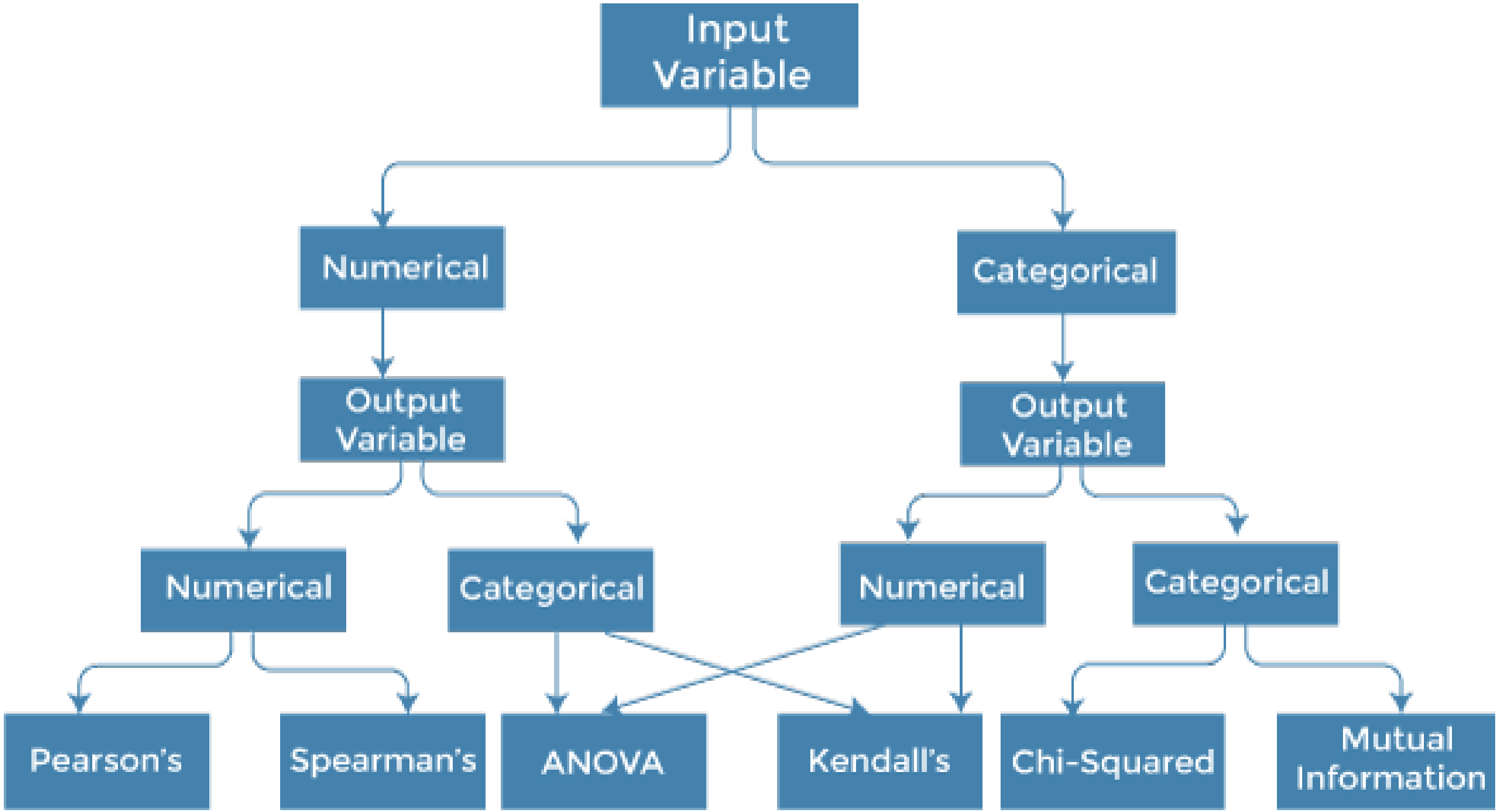


PCA
(orthogonal coordinate)



ICA
(non-orthogonal coordinate)


Feature Selection Techniques
Supervised Feature Selection
Unsupervised Feature Selection
Filters method
Embedded method
Wrappers method
Missing value
Information gain
Chi-square Test
Fisher's Score
Regularization L1, L2
Forward Feature Selction
Backward Feature Selection
Exhaustive Feature Selection
Recursive Feature Elimination


Input Variable
Numerical
Categorical
Output Variable
Output Variable
Numerical
Categorical
Numerical
Categorical
Pearson's
Spearman's
ANOVA
Kendall's
Chi-Squared
Mutual Information

References

- Bishop PRML, 12.1, 12.2, 12.4
- https://jeremy9959.net/Math-3094-UConn/published_notes/notes/PCA.pdf
- https://www.cse.iitk.ac.in/users/piyush/courses/pml_winter16/slides_lec10.pdf