Optimisation in Pattern Recognition:
Principal Component Analysis (PCA)

Tae-Kyun Kim
Senior Lecturer
https://labicvl.github.io/

Backgrounds:
Linear algebra
Optimisation
- Lagrange multipliers
- Gradient method
Matrix and vector derivatives

Further reading:
Chapter 12, C.M.Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

Gradient-based optimisation

– In optimization, **gradient method** is an algorithm to solve problems of the form

$$\min_{\mathbf{x} \in R^n} f(\mathbf{x})$$

with the search directions defined by the gradient of the function at the current point.

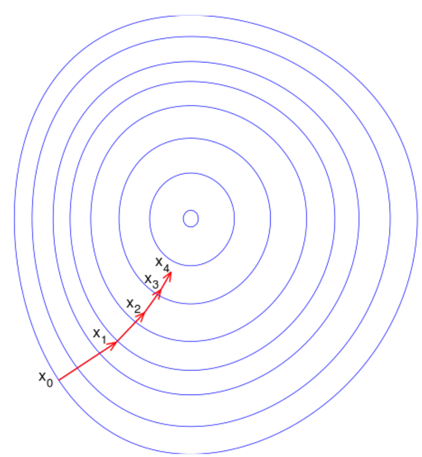
– Examples of gradient method are PCA, LDA, Kernel Machines, Neural Networks.

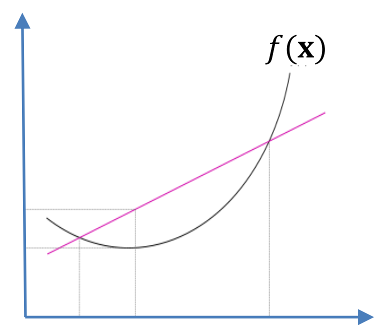
– **Gradient descent** (or ascent) is an iterative optimization algorithm for finding a local minimum (or maximum) of a function, taking steps proportional to the gradient at the current point.

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i \nabla f(\mathbf{x}_i), i \geq 0$$

– When the function f is convex, all local minima are also global minima.

– A function is **convex,** if the line segment between any two points on the graph of the function lies above or on the graph.





Appendix A: Mathematical Foundations, R.Duda, P.Hart, D.Stork, Pattern Classifcation (Second Edition), JOHN WILEY & SONS, Inc. 2001.

https://goo.gl/m6Mnov for questions

Lagrange multipliers for constrained optimisation problems

– The **method of Lagrange multipliers** is a strategy for finding the local maxima/minima of a function subject to equality constraints.

maximize $f(x, y)$

subject to $g(x, y) = 0$, or $g(x, y) = c$

– The **Lagrange function** is

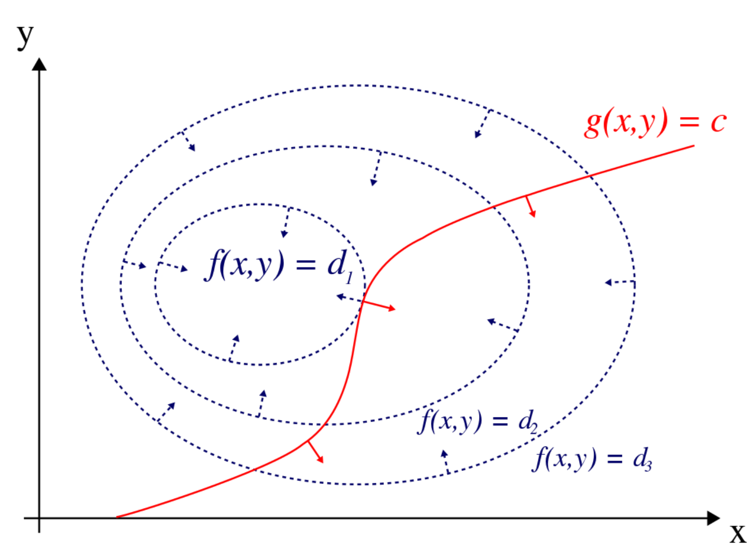
$$\mathcal{L}(x, y, \lambda) = f(x, y) - \lambda \cdot g(x, y)$$

where λ is a constant.

– We solve $\nabla_{x,y,\lambda}\mathcal{L}(x, y, \lambda) = 0$

$\rightarrow$

$$\begin{cases} \nabla_{x,y} f(x, y) = \lambda \nabla_{x,y} g(x, y) \\ \nabla_{\lambda}\mathcal{L}(x, y, \lambda) = 0 \rightarrow g(x, y) = 0 \end{cases}$$



Appendix A: Mathematical Foundations, R.Duda, P.Hart, D.Stork, Pattern Classifcation (Second Edition), JOHN WILEY & SONS, Inc. 2001.

https://goo.gl/m6Mnov for questions

Matrix and vector derivatives

Matrix and vector derivatives are obtained first by element-wise derivatives and then reforming them into matrices and vectors.

$$\frac{\partial \mathbf{x}}{\partial t} = \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \vdots \\ \frac{\partial x_n}{\partial t} \end{bmatrix} \qquad \frac{\partial \mathbf{F}}{\partial t} = \begin{bmatrix} \frac{\partial F_{1,1}}{\partial t} & \cdots & \frac{\partial F_{1,m}}{\partial t} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_{n,1}}{\partial t} & \cdots & \frac{\partial F_{n,m}}{\partial t} \end{bmatrix}$$

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix} \qquad \frac{\partial f}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial X_{1,1}} & \cdots & \frac{\partial f}{\partial X_{n,1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial X_{1,m}} & \cdots & \frac{\partial f}{\partial X_{n,m}} \end{bmatrix}$$

Matrix and vector derivatives

Useful formula for linear and quadratic functions:

$$\frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}^T$$

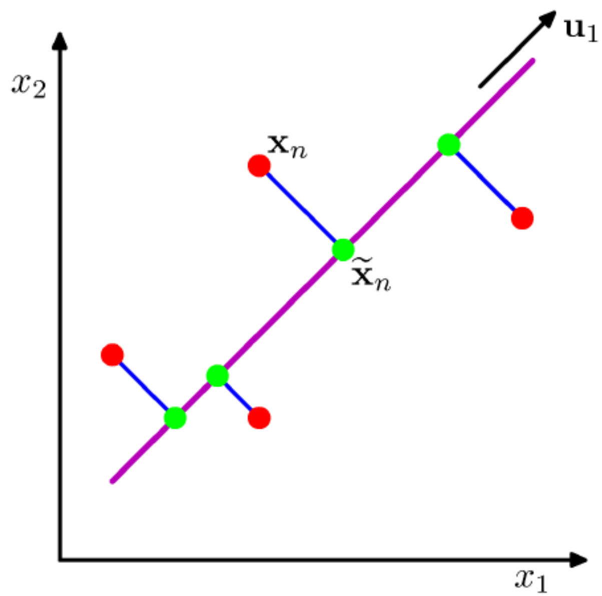
$$\frac{\partial \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^T\mathbf{A}}{\partial \mathbf{x}^T} = \mathbf{A}$$

$$\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = \mathbf{x}^T(\mathbf{A}^T + \mathbf{A}) = \mathbf{x}^T\mathbf{A}^T + \mathbf{x}^T\mathbf{A}$$

Maximum variance formulation of PCA

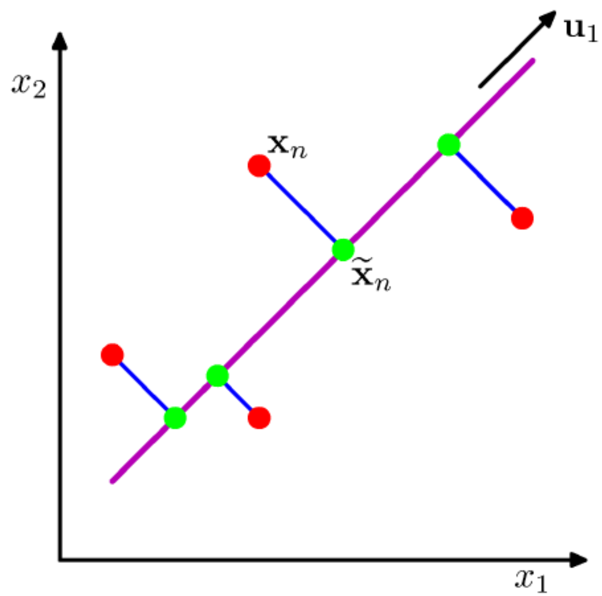
– PCA (also known as Karhunen-Loeve (KL) transform) is a technique for: dimensionality reduction, lossy data compression, feature extraction, and data visualisation.

– PCA is defined as the orthogonal projection of the data onto a lower dimensional linear space such that the variance of the projected data is maximised.



Maximum variance formulation of PCA

- Given a data set $\{\mathbf{x}_n\}$, n = 1,...,N and $\mathbf{x}_n \in R^D$, our goal is to project the data onto a space of dimension $M \ll D$ while maximising the projected data variance.
- For simplicity, M = 1. The direction of this space is defined by a vector $\mathbf{u}_1 \in R^D$ s.t. $\mathbf{u}_1^\top \mathbf{u}_1 = 1$.
- Each data point $\mathbf{x}_n$ is then projected onto a scalar value $\mathbf{u}_1^\top \mathbf{x}_n$.



Maximum variance formulation of PCA

– The mean is $\mathbf{u}_1^T \overline{\mathbf{x}}$,

where

$$\overline{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n.$$

– The variance is given by

$$\frac{1}{N} \sum_{n=1}^{N} \{\mathbf{u}_1^T \mathbf{x}_n - \mathbf{u}_1^T \overline{\mathbf{x}}\}^2 = \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1$$

where **S** is the data covariance matrix defined as

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T.$$

Maximum variance formulation of PCA

– We maximise the projected variance

$$J = \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$$

with respect to $\mathbf{u}_1$ with the normalisation condition $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$.

– The Lagrange multiplier formulation is

$$L = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1).$$

– By setting the derivative with respect to $\mathbf{u}_1$ to zero, we obtain

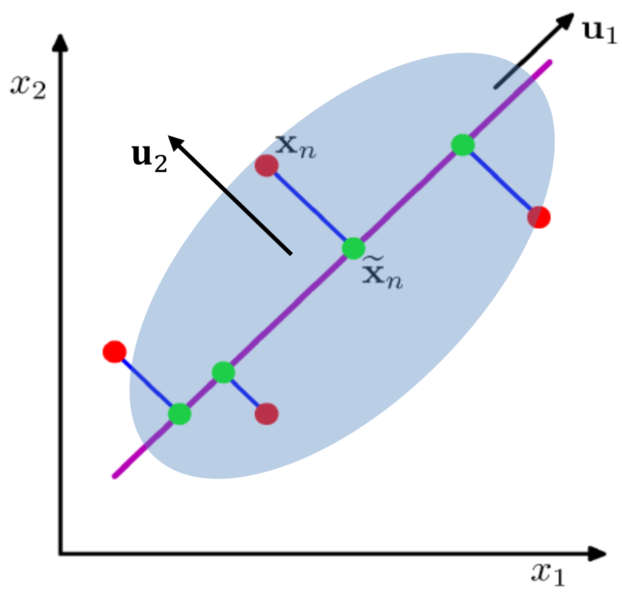
$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$$

$\longrightarrow$ $\mathbf{u}_1$ is an eigenvector of $\mathbf{S}$.

– By multiplying $\mathbf{u}_1^T$ to both sides and using the condition $\mathbf{u}_1^{\mathrm{T}}\mathbf{u}_1 = 1$, the variance is obtained by $\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \lambda_1.$

Maximum variance formulation of PCA

- We obtain the maximum variance, when $\mathbf{u}_1$ is the eigenvector with the largest eigenvalue λ_1.
- The eigenvector is also called the *principal component*.
- For the general case of an *M* dimensional subspace, we obtain the *M* eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M$ of the data covariance matrix $\mathbf{S}$ corresponding to the *M* largest eigenvalues $\lambda_1, \lambda_2 \dots, \lambda_M$.



$$\mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}.$$

$$\delta_{i,j} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

Minimum error formulation of PCA

- Alternative (equivalent) formulation of PCA is to minimise the reconstruction error.
- We minimise the distortion measure (or reconstruction error)

$$J = \frac{1}{N} \sum_{n=1}^{N} ||\mathbf{x}_n - \widetilde{\mathbf{x}_n}||^2.$$

where $\widetilde{\mathbf{x}}_n$ is the reconstruction of n-th data point $\mathbf{x}_n \in R^D$.

- The solution is to choose the eigenvectors of the covariance matrix with *M* largest eigenvalues:

$$\mathbf{S}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

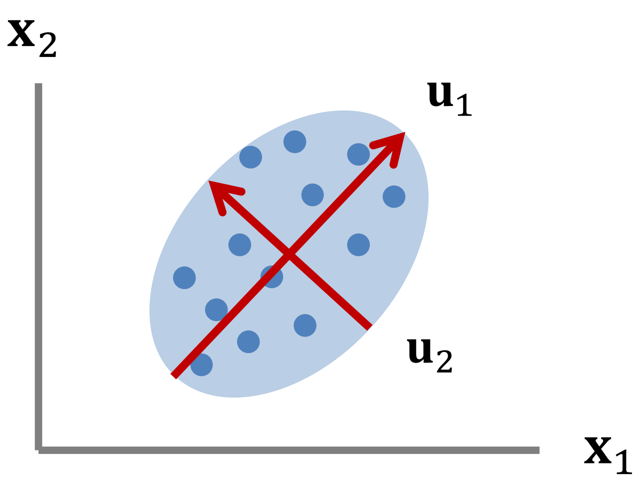
where $i = 1, \ldots, M$.

- The distortion measure (or reconstruction error) becomes

$$J = \sum_{i=M+1}^{D} \lambda_i.$$

(Recap) PCA

- Principal components are the vectors in the direction of the maximum variance of the projection data.
- For given 2D data points, u1 and u2 are found as PCs.



- For dimension reduction,
  - Each 2D data point is transformed to a single variable z1 representing the projection of the data point onto the eigenvector u1.
  - The data points projected onto u1 has the max variance.
- PCA infers the inherent structure of high dimensional data.
- The intrinsic dimensionality of data is much smaller.

(Recap) PCA

PCA (also known as Karhunen-Loeve transform) is a useful technique for:

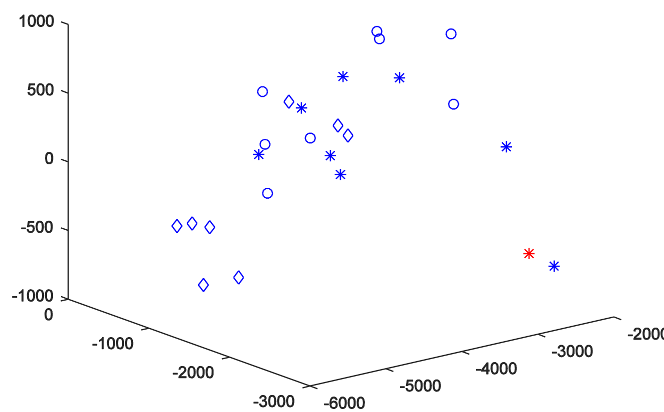
- feature extraction,
- lossy data compression,

- dimensionality reduction,
- and data visualisation.

$$\widetilde{\mathbf{x}}_n = \overline{\mathbf{x}} + \sum_{i=1}^{M} a_{ni}\mathbf{u}_i$$

Original face

M=50



Low-dimensional computation of Eigenspace, when D>>N

Given a data set $\{\mathbf{x}_n\}$, n = 1,...,N and $\mathbf{x}_n \in R^D$, our goal is to project the data onto a space of dimension $M << D$.

– We compute the eigenvectors $\mathbf{u}_i$ of the matrix AA^T (for simplicity, instead of $S=(1/N)AA^T$).
– The matrix AA^T (DxD matrix) is typically very large (not practical).

We consider the matrix A^TA (NxN matrix) instead.

– Compute the eigenvectors $\mathbf{v}_i$ of A^TA:

$$A^TA\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

– What is the relationship between $\mathbf{u}_i$ and $\mathbf{v}_i$?

$$A^TA\mathbf{v}_i = \lambda_i\mathbf{v}_i \rightarrow AA^TA\mathbf{v}_i = \lambda_i A\mathbf{v}_i \rightarrow SA\mathbf{v}_i = \lambda_i A\mathbf{v}_i$$
$$\rightarrow S\mathbf{u}_i = \lambda_i\mathbf{u}_i, \textit{ where } \mathbf{u}_i=A\mathbf{v}_i$$

Thus, AA^T and A^TA have the same eigenvalues and their eigenvectors are related s.t. $\mathbf{u}_i=A\mathbf{v}_i$

Low-dimensional computation of Eigenspace, when D>>N

- Note:

  1: AA^T can have up to *D* eigenvalues and eigenvectors.

  2: A^TA can have up to *N (or N-1)* eigenvalues and eigenvectors.

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

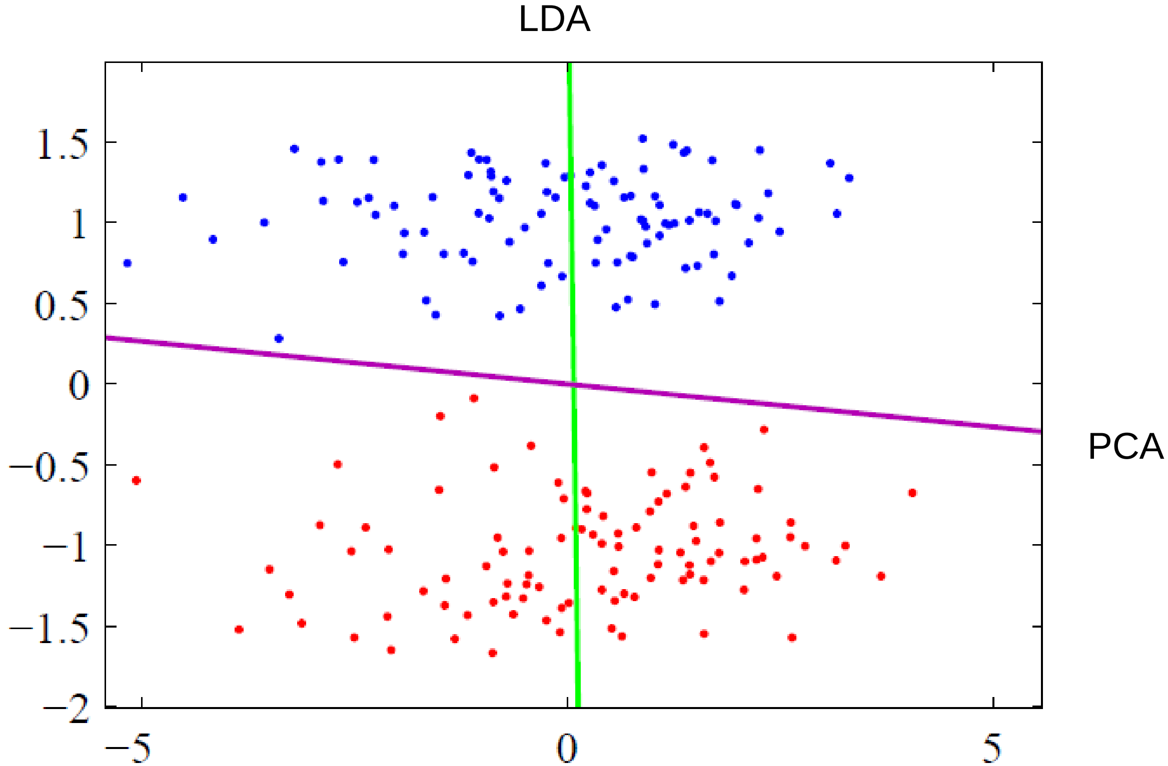
Compute the *M* best eigenvectors of AA^T : $\mathbf{u}_i = A\mathbf{v}_i$

(**important:** normalize $\mathbf{u}_i$ such that $||\mathbf{u}_i|| = 1$)

Limitations of PCA

Unsupervised learning

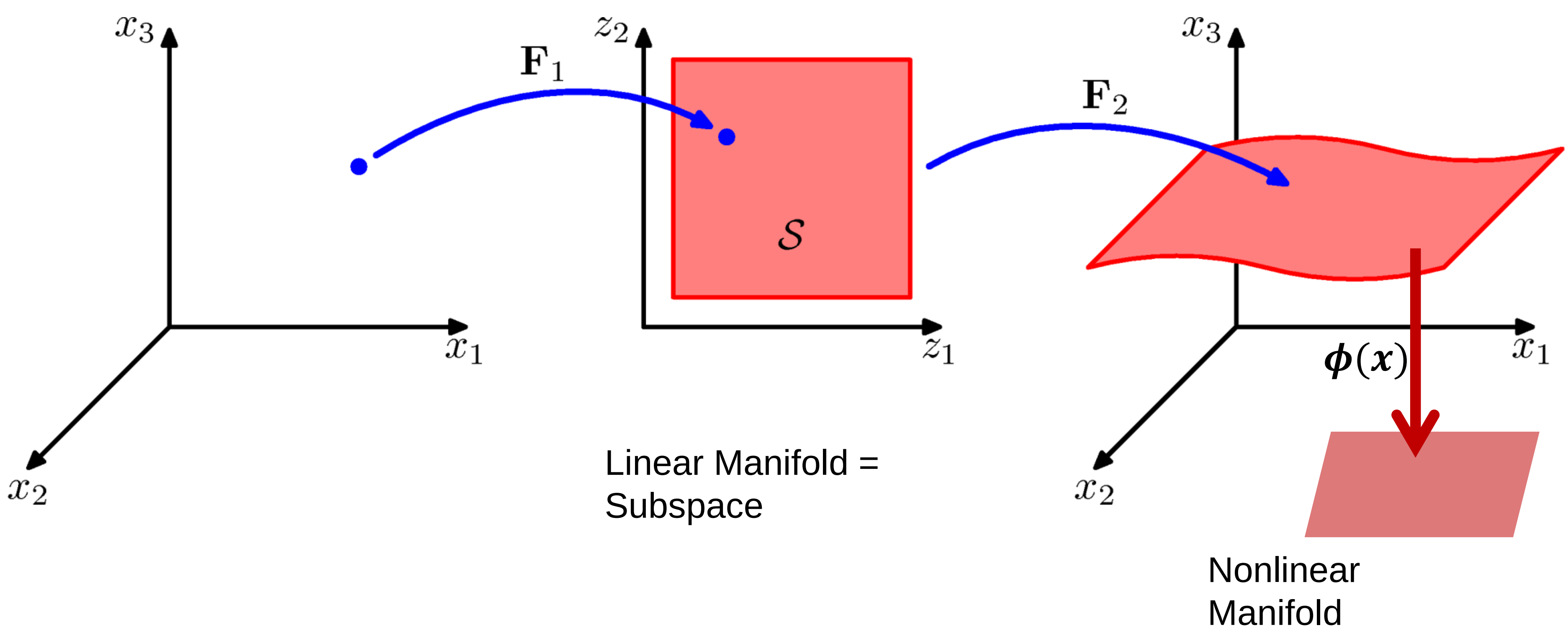
- PCA finds the direction for maximum variance of data (unsupervised), while LDA (Linear Discriminant Analysis) finds the direction that optimally separates data of different classes (discriminative or supervised).



PCA vs LDA

Linear model

- PCA is a linear projection method.
- When data lies in a nonlinear manifold, PCA is extended to Kernel PCA by the kernel trick.

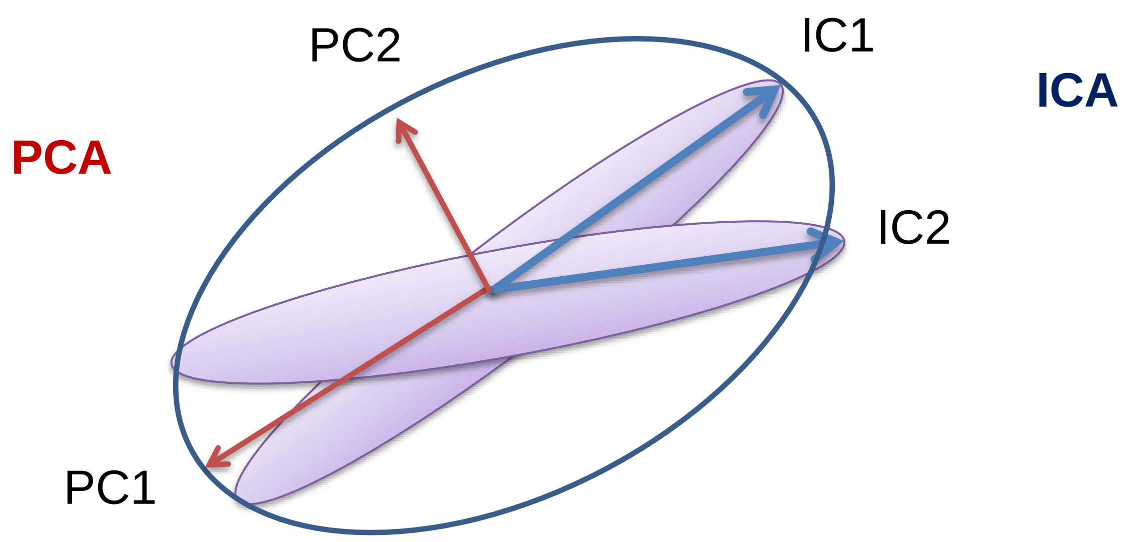


PCA vs Kernel PCA

https://goo.gl/m6Mnov for questions

Gaussian assumption

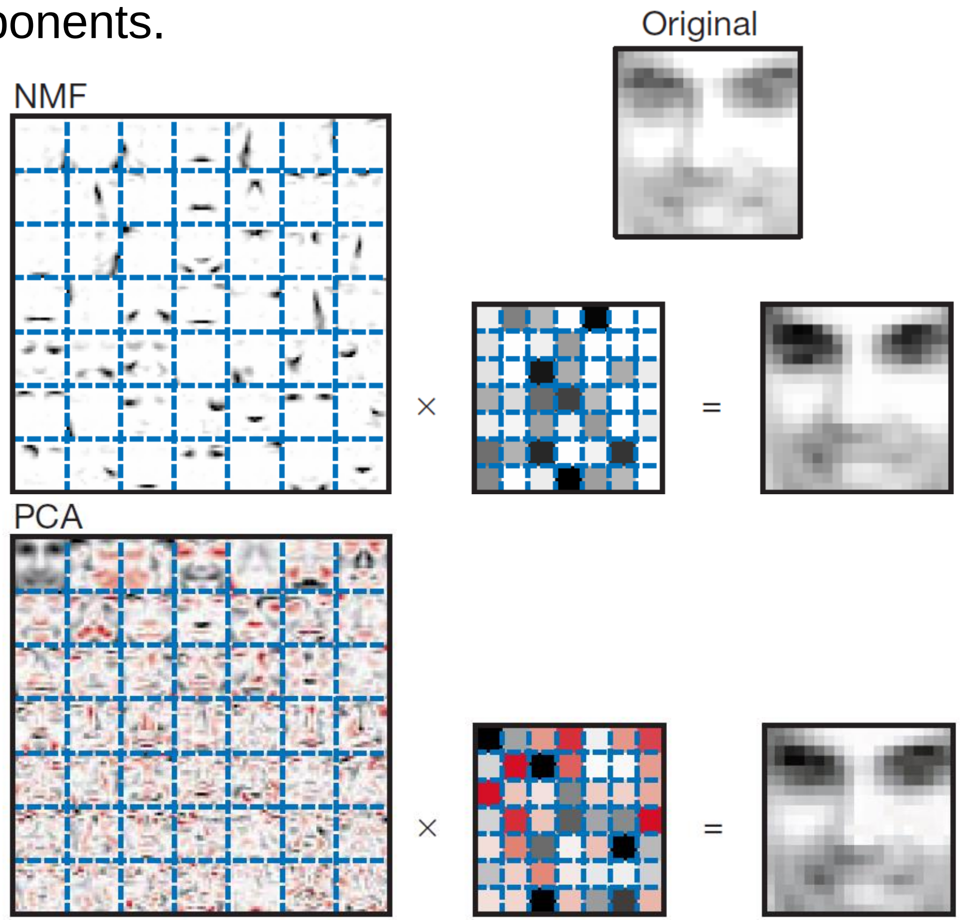
– PCA (Principal Component Analysis) models data as Gaussian distributions (2^{nd} order statistics), whereas ICA (Independent Component Analysis) captures higher-order statistics.



PCA vs ICA

Holistic bases

- PCA bases are holistic (cf. part-based) and less intuitive.
- NMF (Non-negative Matrix Factorisation) yields bases, which capture local facial components.



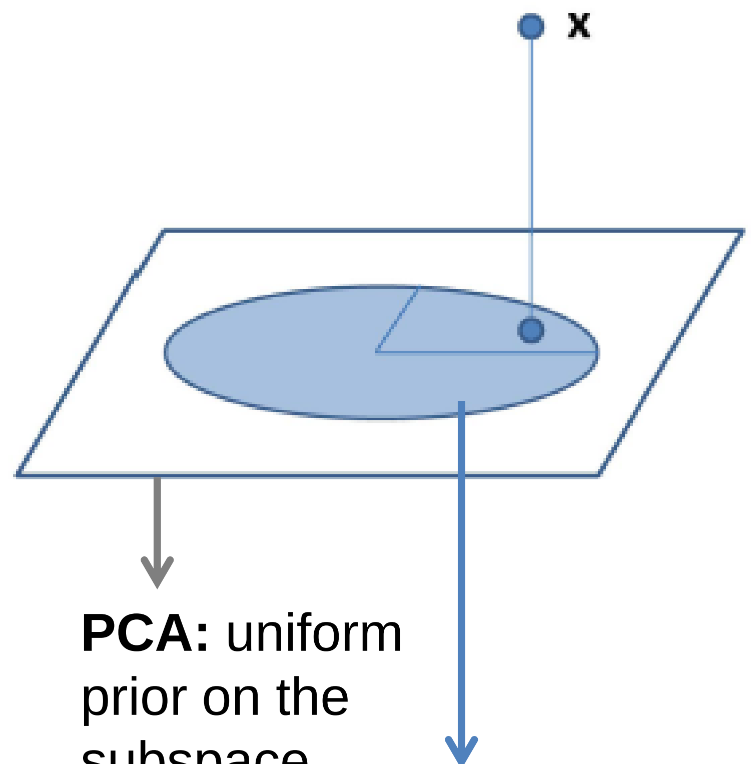
D.Lee and S.Seung (1999). "Learning the parts of objects by non-negative matrix factorization". *Nature* 401 (6755): 788–791.

PCA vs NMF

Uniform prior on the subspace

- A subspace is spanned by the orthonormal bases i.e. eigenvectors computed from the covariance matrix.
- It interprets each observation with the uniform prior on the subspace.
- PPCA (Probabilistic PCA): It estimates the probability of generating each observation with Gaussian distribution,

$$p(\mathbf{x}|\mu, \Sigma)$$



PCA: uniform prior on the subspace

PPCA: Gaussian distribution on the subspace

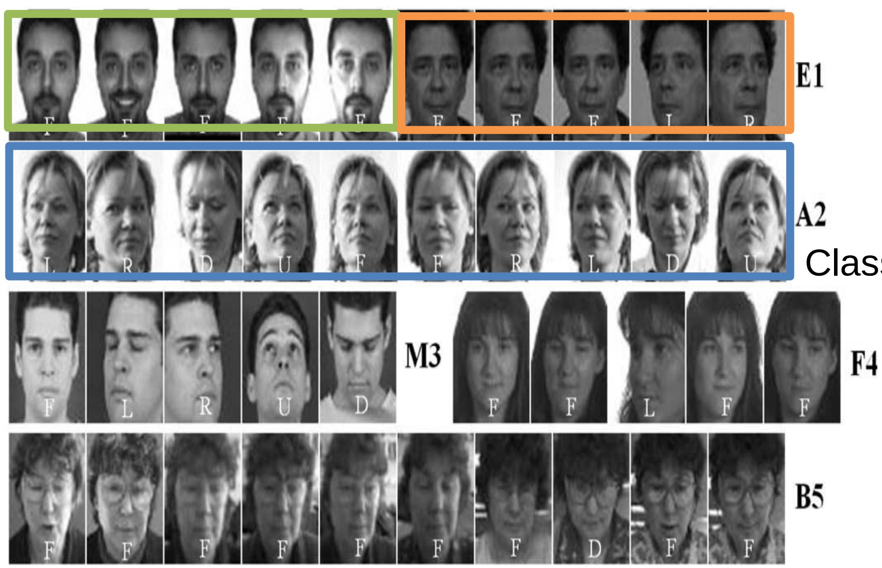
PCA vs PPCA

Face Recognition vs Object Categorisation

- Both are as multi-class classification problems.
- Classes are different object categories in object categorisation, while classes are different person identities in face recognition.

Face dataset

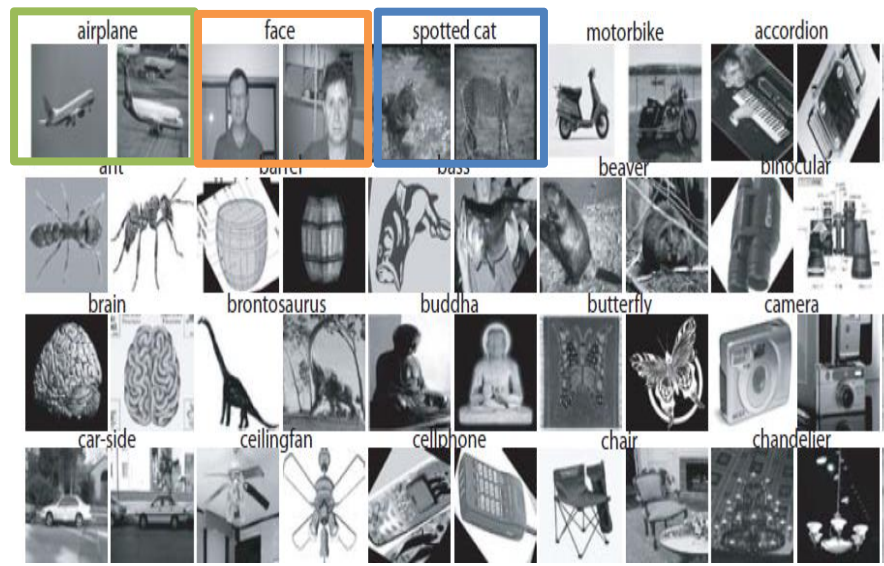
Class 1 Class 2



Class 3

Object category dataset

Class 1 Class 2 Class 3 …



https://goo.gl/m6Mnov for questions

Face Recognition vs Object Categorisation

– Intraclass and Interclass variations in object categorisation are wider, compared to face recognition.

– We extract representations/features that minimise intraclass variations and maximise interclass variations for a classification problem.

– Bag of Words (BoW) is one of dominating-arts for feature extraction for generic object categorisation, while subspace/manifolds are standard techniques for face image analysis.

– Using more advanced classifiers (Support Vector Machine/Randomised Forests/Convolutional Neural Network, cf. NN (Nearest Neighbour) classifier) often improves recognition performance.