Machine Learning in Robotics
Dimensionality Reduction

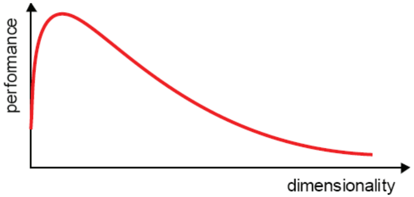
Prof. Dongheui Lee

Institute of Automatic Control Engineering
Technische Universität München

dhlee@tum.de

The curse of dimensionality

- The problems associated with multivariate data analysis as the dimensionality increases
- Toy example : 3-class pattern recognition or an approach which divide the sample space intro equally spaced bins
  - 1D $\rightarrow 3$ bins $\rightarrow 3^2$ examples
  - 2D $\rightarrow 3^2$ bins $\rightarrow 3^3$ examples
  - 3D $\rightarrow 3^3$ bins $\rightarrow 3^4$ examples
- For a given sample size, there is a maximum number of features above which the performance of our classifier will degrade rather than improve



The curse of dimensionality

Implication of the curse of dimensionality

- Exponential growth in the number of examples, required to maintain a given sampling density
- Exponential growth in the complexity of the target function (a density estimate) with increasing dimensionality
- Humans have an extraordinary capacity to discern patterns in 1$\sim$3D. But these capability degrades drastically for higher dimensions

Why dimensionality reduction?

- Learning a target function from data where some features are irrelevant $\rightarrow$ reduce variance. Improve accuracy
- Wish to visualize high dimensional data
- Intrinsic dimensionality of data is smaller than the number of features used to describe it

Dimensionality Reduction

Two approaches

- Feature extraction: creating a subset of new features by combinations of the existing features $m > k$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \to \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = f\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

- Feature selection: choosing a subset of all the features (the ones more informative)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \to \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \vdots \\ x_{i_k} \end{bmatrix}$$

Feature Extraction Problem

- Given a feature space, to find a mapping $y = f(x)$ such that the reduced feature vector y preserves most of the information or structure of original feature x
- The optimal mapping function $f(x)$ will result in no increase in the minimum probability of error
- In general, the mapping function can be a nonlinear function. But, often feature extraction is limited to linear transforms. $\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{x}$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ & & \vdots & \\ u_{k1} & u_{k2} & \cdots & u_{km} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Two Approaches for Feature Extraction

Unsupervised approaches

- Principal Components Analysis
- Singular Value Decomposition
- Independent Components Analysis

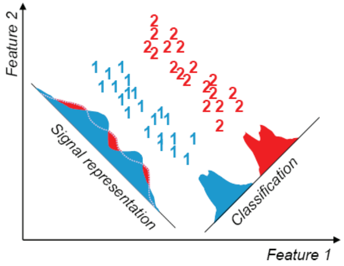
Supervised approaches

- Fisher Linear Discriminant Analysis (LDA)
- Hidden Layers of Neural Networks

Feature Extraction

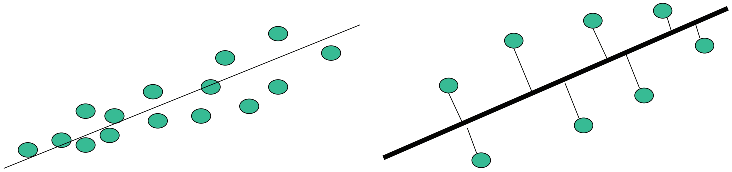
The selection of feature extraction mapping $y = f(x)$ is guided by an objective function to maximize (or minimize)
Depending on the criteria measures by objective function. Two categories

- Signal representation
  - Goal : to represent samples accurately
  - Principal Components Analysis (PCA)
- Classification
  - Goal : to enhance the class-discriminatory information
  - Linear Discriminant Analysis (LDA)



Principal Components Analysis (PCA)

- Given data points in m-dimensional space, project into lower dimensional space while preserving as much information as possible $\Rightarrow$ Choose a line that fits the data so the points are spread out well along the line
- Seeks a projection that best represent the data in a least-square sense $\Rightarrow$ minimize sum of squares of distances to the line
- We wish to explain/summarize the underlying variance-covariance structure of a large set of variables through a few linear combinations of these variables



Space Representation

- High-dimensional space representation $\boldsymbol{x}$
- Low-dimensional space representation $\boldsymbol{y}$

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}$$

- PCA allows us to compute a linear transformation that maps data from a high dimensional space to a lower dimensional sub-space

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ & & \vdots & \\ u_{k1} & u_{k2} & \cdots & u_{km} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

Maximum variance formulation

- Goal: to project the data onto a space having dimensionality $k < m$ while maximizing the variance of the projected data
- Begin with $k = 1$
- m-dimensional unit vector $\boldsymbol{u}_1 = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \end{bmatrix}^T$, $\boldsymbol{u}_1^T \boldsymbol{u}_1 = 1$
- Each data point $\boldsymbol{x}^{(i)}$ is then projected onto a scalar value $y^{(i)} = \boldsymbol{u}_1^T \boldsymbol{x}^{(i)}$
- Mean, covariance of the original data
  $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}^{(i)}$, $\boldsymbol{S} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^T$
- mean of the projected data is $\boldsymbol{u}_1^T \boldsymbol{\mu}$
- variance of the projected data $\frac{1}{n} \sum_{i=1}^{n} \{\boldsymbol{u}_1^T \boldsymbol{x}^{(i)} - \boldsymbol{u}_1^T \boldsymbol{\mu}\}^2 = \boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1$
- Maximize the projected variance $\boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1$ wrt $\boldsymbol{u}_1$

$$\frac{\partial}{\partial \boldsymbol{u}_1} \left(\boldsymbol{u}_1^T \boldsymbol{S} \boldsymbol{u}_1 + \lambda_1 (1 - \boldsymbol{u}_1^T \boldsymbol{u}_1) \right) = 0, \quad \boldsymbol{S} \boldsymbol{u}_1 = \lambda_1 \boldsymbol{u}_1$$

- Variance will be a maximum value when $\boldsymbol{u}_1$ is equal to the eigenvector having the largest eigenvalue λ_1
- When k increases, choose $\boldsymbol{u}_1 \ldots \boldsymbol{u}_k$ as corresponding eigenvectors to k largest eigenvalues $\lambda_1, \lambda_2 \ldots \lambda_k$

PCA in General

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \to \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ & & \vdots & \\ u_{k1} & u_{k2} & \cdots & u_{km} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

- $\{\boldsymbol{u}_j\}$ are orthogonal basis vectors.
- $\boldsymbol{u}_1 = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \end{bmatrix}^T$ is 1st eigenvector of covariance matrix, and known as the first principal component. This axis explains as much as possible of original variance in data set
- $\boldsymbol{u}_2 = \begin{bmatrix} u_{21} & u_{22} & \cdots & u_{2m} \end{bmatrix}^T$ is 2nd eigenvector of covariance matrix, and the 2nd principal component
- $\boldsymbol{u}_k = \begin{bmatrix} u_{k1} & u_{k2} & \cdots & u_{km} \end{bmatrix}^T$ is kth eigenvector of covariance matrix, and the kth principal component

PCA algorithm

1. Compute the mean and covariance matrix

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}^{(i)}, \quad \boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^T$$

2. Find eigenvectors and eigenvalues of $\boldsymbol{S}$
3. Choose the k largest eigenvalues $\lambda_1, \lambda_2 \ldots \lambda_k$
4. Choose the corresponding eigenvectors $\boldsymbol{u}_1 \ldots \boldsymbol{u}_k$ and make transformation matrix $\boldsymbol{W} = [\boldsymbol{u}_1 \boldsymbol{u}_2 \ldots]$
5. Project original data $\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}$

PCA algorithm: Alternate formulation

Often the data is made zero mean as a preprocess step:

1. Compute the mean
$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)}$$
2. Replace each $\boldsymbol{x}^{(i)}$ by $\boldsymbol{x}^{(i)} - \boldsymbol{\mu}$ (preprocessing step).
3. Calculate the covariance matrix
$$\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{x}^{(i)})(\boldsymbol{x}^{(i)})^T$$
4. Find eigenvectors and eigenvalues of $\boldsymbol{S}$
5. Choose the k largest eigenvalues $\lambda_1, \lambda_2 \ldots \lambda_k$
6. Choose the corresponding eigenvectors $\boldsymbol{u}_1 \ldots \boldsymbol{u}_k$ and make transformation matrix $\boldsymbol{W} = [\boldsymbol{u}_1 \boldsymbol{u}_2 \ldots]$
7. Project the zero mean data $\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}$
8. For a new $\boldsymbol{x}'$, PCA projection will be $\boldsymbol{y}' = \boldsymbol{W}^T(\boldsymbol{x}' - \boldsymbol{\mu})$

Principal Components Analysis

- Principal Components Analysis is the oldest technique in multivariate analysis
- PCA is also known as the Karhunen-Loève transform
- PCA was first introduced by Pearson in 1901, and generalized by Loève in 1963
- The main limitation of PCA is that it does not consider class separability since it does not take into account the class label of the feature vector
  - PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance
  - There is no guarantee that the directions of maximum variance will contain good features for discrimination

PCA - simple example

Compute the principal components for the following two-dimensional dataset
$X = (x_1, x_2) = (1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8)$

PCA - simple example

Compute the principal components for the following two-dimensional dataset
$\boldsymbol{X} = (x_1, x_2) = (1,2), (3,3), (3,5), (5,4), (5,6), (6,5), (8,7), (9,8)$
Solution

- The (biased) covariance estimate of the data is

$$\boldsymbol{S} = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$$

- The eigenvalues are the zeros of the characteristic equation

$$\boldsymbol{S}\boldsymbol{u} = \lambda \boldsymbol{u} \Rightarrow |\boldsymbol{S} - \lambda \boldsymbol{I}| = 0 \Rightarrow \lambda_1 = 9.34, \lambda_2 = 0.41$$

- The eigenvectors are the solutions of the system

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix} = \begin{bmatrix} \lambda_1 u_{11} \\ \lambda_1 u_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$$

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix} = \begin{bmatrix} \lambda_2 u_{21} \\ \lambda_2 u_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix} = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}$$

Linear Discriminant Analysis

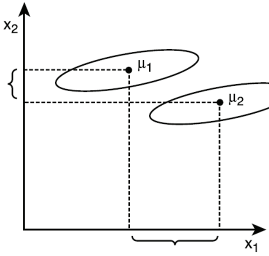
- Multiple classes and PCA
  - Suppose there are C classes in the training data
  - PCA is based on the sample covariance which characterizes the scatter of the entire data set, irrespective of class-membership
  - The projection axes chosen by PCA might not provide good discrimination power
- What is the goal of LDA?
  - Perform dimensionality reduction while preserving as much of the class discriminatory information as possible
  - Seeks to find directions along which the classes are best separated
  - Takes into consideration the scatter within-classes but also the scatter between-classes
  - More capable of distinguishing image variation due to identity from variation due to other sources such as illumination and expression

Linear Discriminant Analysis

In order to find a good projection vector, we need to define a measure of separation between the projected classes

- Projection $\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{x}$
- The mean of original and projected dataset
  $\boldsymbol{\mu}_j = \frac{1}{n_j}\sum_{\boldsymbol{x}\in X_j}\boldsymbol{x}$, $\widetilde{\mu}_j = \frac{1}{n_j}\sum_{y\in Y_j} y = \frac{1}{n_j}\sum_{\boldsymbol{x}\in X_j}\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{w}^T\boldsymbol{\mu}_j$
- The case of two classes, the distance between the projected means
  $|\widetilde{\mu}_1 - \widetilde{\mu}_2| = |\boldsymbol{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)|$

$\Longrightarrow$ Not so good measure, since it does not take into account the standard deviation within the class.



Fisher Linear Discriminant Analysis

- Proposed by Ronald Aylmer Fisher
- To maximize a function that represents the difference between means normalized by a measure of the within-class scatter
  $J(\boldsymbol{w}) = \frac{|\widetilde{\mu}_1 - \widetilde{\mu}_2|^2}{\widetilde{S}_1 + \widetilde{S}_2}$
- Define the scatter matrix for each class
  - Original space $\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in X_i} \boldsymbol{x}$, $\boldsymbol{S}_i = \sum_{\boldsymbol{x} \in X_i} (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^T$
  - projected space
    $\widetilde{\mu}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in X_i} \boldsymbol{w}^T \boldsymbol{x}$, $\widetilde{S}_i = \sum_{\boldsymbol{x} \in X_i} \boldsymbol{w}^T (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \boldsymbol{w} = \boldsymbol{w}^T \boldsymbol{S}_i \boldsymbol{w}$
- Within-class scatter matrix $\boldsymbol{S}_W = \boldsymbol{S}_1 + \boldsymbol{S}_2$, $\widetilde{S}_1 + \widetilde{S}_2 = \boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}$
- Between-class scatter matrix
  $\boldsymbol{S}_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T$, $(\widetilde{\mu}_1 - \widetilde{\mu}_2)^2 = \boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}$
- The criterion function $J(\boldsymbol{w}) = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$

Linear Discriminant Analysis

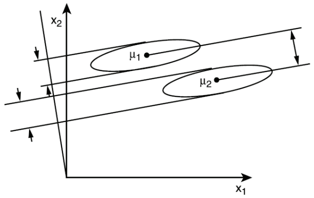
Maximize $J(\boldsymbol{w})$

$$\frac{\partial}{\partial \boldsymbol{w}}[J(\boldsymbol{w})] = \frac{\partial}{\partial \boldsymbol{w}}[\frac{\boldsymbol{w}^T \boldsymbol{S_B w}}{\boldsymbol{w}^T \boldsymbol{S_W w}}] = 0$$

$$\vdots$$

$$\boldsymbol{S}_{\boldsymbol{W}}^{-1} \boldsymbol{S_B w} = J\boldsymbol{w}$$

$$\boldsymbol{w}^\star = arg \max_{\boldsymbol{w}} \left[\frac{\boldsymbol{w}^T \boldsymbol{S_B w}}{\boldsymbol{w}^T \boldsymbol{S_W w}} \right] = \boldsymbol{S}_{\boldsymbol{W}}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

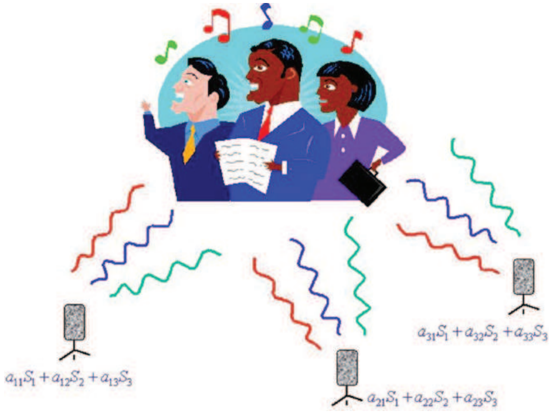


Properties of LDA

- LDA is a parametric method since it assumes unimodal Gaussian likelihoods
- If the distributions are significantly non-Gaussian, the LDA projections will not be able to preserve any complex structure of the data that may be needed for classification

Independent component analysis

Source separation : Cocktail Party Problem

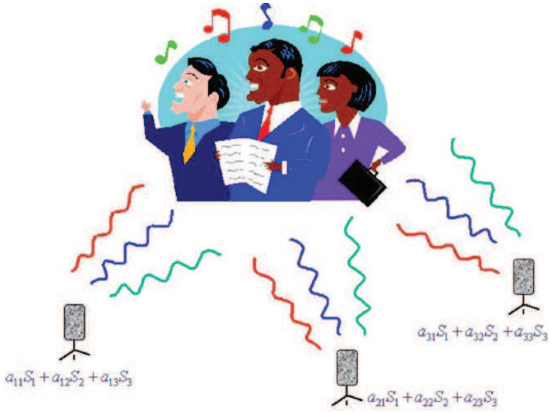


$a_{11}S_1 + a_{12}S_2 + a_{13}S_3$
$a_{21}S_1 + a_{22}S_2 + a_{23}S_3$
$a_{31}S_1 + a_{32}S_2 + a_{33}S_3$

- separate the mixed signal into sources.

Independent component analysis

Source separation : Cocktail Party Problem


$a_{11}S_1 + a_{12}S_2 + a_{13}S_3$
$a_{21}S_1 + a_{22}S_2 + a_{23}S_3$
$a_{31}S_1 + a_{32}S_2 + a_{33}S_3$

- separate the mixed signal into sources.
- Assumption: different sources are **independent**

Demo: `http://research.ics.aalto.fi/ica/cocktail/cocktail_en.cgi`

Independent component analysis

$$x = As \text{ which implies } s = Wx \text{ where } W = A^{-1}$$

- x : observations
- s : the independent components
- A : the **mixing matrix**
- W : the **unmixing matrix.**
- Goal: to recover the sources $s^{(i)}$ that had generated our data $x^{(i)} = As^{(i)}$

Independent component analysis

- the distribution of each source s_i : p_s. By the condition of independence,

$$p(s) = \prod_{i=1}^{m} p_s(s_i) \tag{1}$$

If $x = As$, the density of x is given as $p_x(x) = p_s(Wx) \cdot |W|$.

$$p(x) = \prod_{i=1}^{m} p_s(w_i^T x) \cdot |W| \tag{2}$$

- A choice of cdf can be a sigmoid function $g(s) = \frac{1}{1+exp(-s)}$. $p_s(s) = g'(s)$
- Learn W that maximizes the log-likelihood function for a given dataset $\{x^{(i)}\}$

$$l(W) = \sum_{i=1}^{n} \Big(\sum_{j=1}^{m} \log g'(w_j^T x^{(i)}) + \log |W| \Big) \tag{3}$$

For a training sample $x^{(i)}$, the gradient ascent rule:

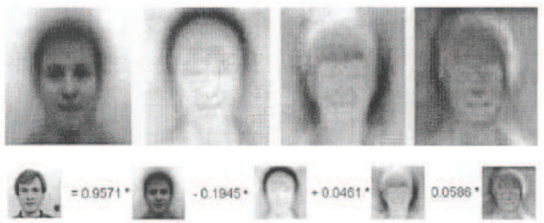
$$W := W + \alpha \left(\begin{bmatrix} 1 - 2g(w_1^T x^{(i)}) \\ 1 - 2g(w_2^T x^{(i)}) \\ \vdots \\ 1 - 2g(w_m^T x^{(i)}) \end{bmatrix} x^{(i)} + (W^T)^{-1} \right) \tag{4}$$

PCA application: Face recognition

Eigenfaces for face detection/recognition

- Template matching problem
- Difficult to perform recognition in a high-dimensional space
- Improvements can be achieved by mapping data into a lower dimensionality space

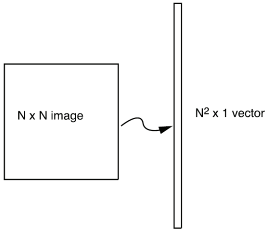
Data is projected into a lower dimensional space using PCA.



M. Turk, A. Pentland, *Eigenfaces for Recognition*. Journal of Cognitive Neuroscience, 3(1), pp. 71-86, 1991.

PCA application: Face recognition

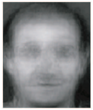
- Step 1. Obtain face images (training data set)
- Step 2. Preprocess the face images (centered and same size) and represent each image as a vector $I^{(i)} \rightarrow x^{(i)}, \ \forall i = 1, \ldots, n$



PCA application: Face recognition

- Step 3. Compute the average face

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)}$$



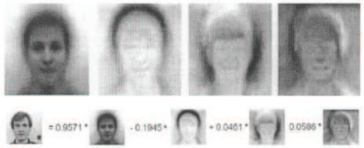
- Step 4. Compute the covariance matrix

$$\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{x}^{(i)} - \boldsymbol{\mu})(\boldsymbol{x}^{(i)} - \boldsymbol{\mu})^T$$

- Step 5. Compute eigenvalues of the covariance matrix
- Step 6. Choose the k largest eigenvalues and their corresponding eigenvectors

PCA application: Face recognition

- Now, we can represent each face as a linear combination of the best k eigenvectors
- Eigenfaces: eigenvectors of face dataset



PCA application: Face recognition

- Step 7. For face recognition of a new image x, normalize the input image. Project on the eigenspace.

$$y_i = u_i^T \boldsymbol{x} \quad \forall i = 1, \ldots, k \quad \boldsymbol{y} = [y_1 \ldots y_k]^T$$

- Step 8. Compare the projected input image $\boldsymbol{y}$ with face classes $\boldsymbol{y}_j$. Find the class which results in smallest (Euclidean / Mahalanobis) distance. If the minimum Euclidean distance is smaller than a threshold, the image is recognized as the face class.
  If $min \parallel \boldsymbol{y} - \boldsymbol{y}_j \parallel < \tau$, $\boldsymbol{y}$ belongs to the j-th face class.

PCA application: Object Grasping

- The PCA analysis of grasping data reveals that 87% of objects can be grasped by the first three synergies (eigen-postures) of the hand, which lead to the design and control of an underactuated gripper.

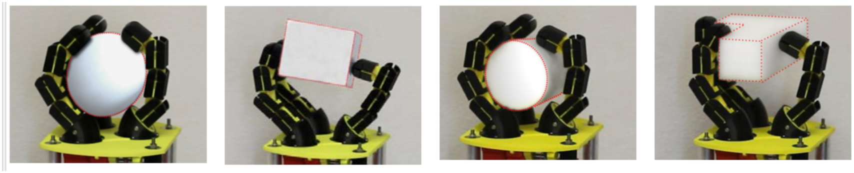


Figure: Hand prototype during the execution of some simple grasps. Only one synergy is used to grasp four test objects. Despite this, depending by the actuation variable, grasped object shape, contact point positions and internal forces, different grasp configurations can be achieved.

Catalano, Manuel G., et al. *Adaptive synergies for the design and control of the Pisa/IIT SoftHand.* The International Journal of Robotics Research 33.5 (2014): 768-782.

PCA application: Object Grasping

- Their approach enables to adopt a model of the hand with a number of independent actuators that is smaller than the number of joints.
- Shape adaptation in underactuated hands using differential transmission e.g. with gears, closed-chain mechanisms or pulleys etc.
- The shape adaptation of the gripper comes from the kinematic model which has non-uniqueness of the shape attained by an underactuated gripper

PCA application: Object Grasping

A total of 107 objects of different shapes were successfully grasped with the 19-joints hand by using only a single actuator.

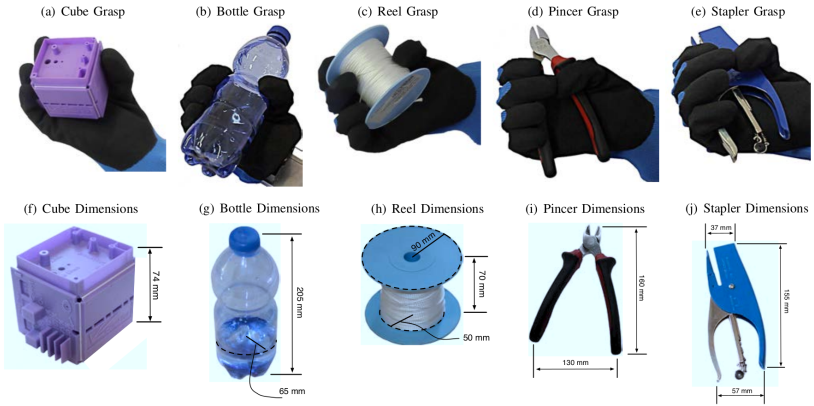


Figure: Some experimental grasps performed with the Pisa/IIT hand, with the object placed in the hand by a human operator.

LDA Application: Emotion recognition

- Biased linear discriminant analysis (BLDA) method that impose large penalties on interclass samples with small differences and small penalties on those samples with large differences

$$S_B = g(i,j)(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

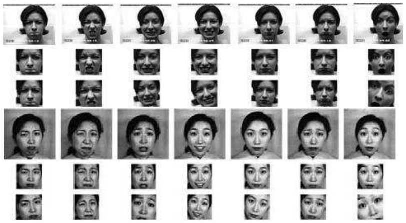


Figure: Examples of the original, well-aligned, and misaligned images of one subject from the (upper half) Cohn-Kanade and (lower half) JAFFE database. From left to right are the facial images with anger, disgust, fear, happy, neutral, sad, and surprise expressions, respectively.

Haibin Yan, Marcelo H. Ang Jr, and Aun Neow Poo, *Weighted biased linear discriminant analysis for misalignment-robust facial expression recognition*. ICRA 2011.

LDA Application

- LDA is a supervised subspace learning approach which searches for a set of most discriminative projections to maximize the ratio of between-class variance to within-class variance simultaneously
- When the class information is available, LDA usually outperforms PCA for classification tasks

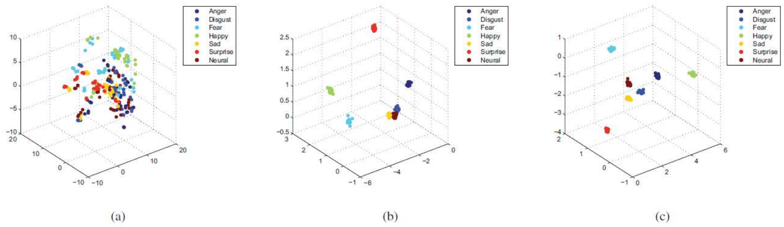


Figure: The projections of the first three components of the original data on the (a) PCA (b) LDA and (c) BLDA feature spaces respectively.

LDA Application

- LDA is a supervised subspace learning approach which searches for a set of most discriminative projections to maximize the ratio of between-class variance to within-class variance simultaneously
- When the class information is available, LDA usually outperforms PCA for classification tasks

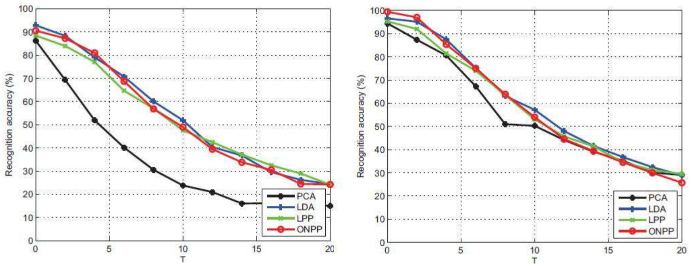


Figure: Recognition accuracy versus different amounts of spatial misalignments. (a) Results obtained on the Cohn-Kanade database. (b) Results obtained on the JAFFE database.

Separating reflections using Independent Components Analysis

A mixed image of a painting and an object is represented as $y_1 = aP + bR$, by photographing through a linear polarizer, the relative strength of the reflections can be adjusted.

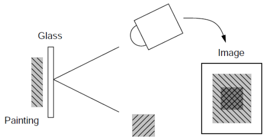


Figure: A photograph of a painting behind glass contains a superposition of the light reflected by the painting and the light reflected directly off the glass.

Farid, Hany, and Edward H. Adelson, *Separating reflections and lighting using independent components analysis*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol.1, 1999.

Separating reflections using Independent Components Analysis

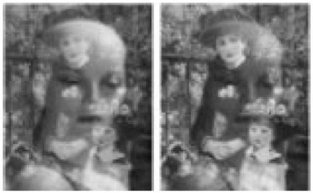


Figure: Mixture of two images as input for ICA.

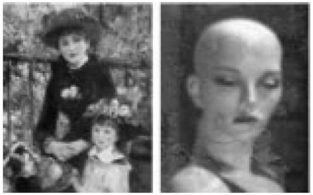


Figure: Output produced by ICA.

Announcements

- Further Reading
  - Duda: Chap 3.7, 3.8.1, 3.8.2, 10.13
  - Bishop: Chap. 12.1-12.3, 12.4.1
  - PCA: Tutorial by J. Schlens