Fast Prototyping Exercise 1

Exercises 5, 6, 7
CSC872
Pattern Analysis and Machine Intelligence

https://bidal.sfsu.edu/~kazokada/csc872/DATA/FaceRecognition_Data.zip

Fast Prototyping Exercise

- **Fast Prototyping**
  - **Learn how to do a quick proof of concept by building a "prototype" (from papers you read, no public codes)**
  - **Correctness matters (no sloppy algorithm!)**
  - **Speed matters (no beautification!)**
  - **No perfect SE necessary**
  - **No copying of codes online (but use base Matlab functions).**
  - **When Done: Parameterization/Visualization/Experimentation**
    - Find out what are **free parameters** in your algorithm whose value must be hand-picked by you
    - Learn how to view internal variable's current values
    - Learn how to visualize your prototype's results in plots/images etc
    - Tweak the parameter values and study your prototype's behavior **quantitatively** to understand the how algorithm works
- **Group Work**
  - **You are encouraged to freely exchange ideas and codes in class**
  - **Contributions to others are as valuable as making your own work**

Fast Prototyping Exercise

- **Please upload your matlab codes thru Canvas Discussion for my grading and your playing!**
  - **Submit Codes: Fast-Prototyping: Exercise #1: Face Recognition by Eigenface/PCA**
  - **First two sessions: Due on midnight of the day (just what you did during the sessions)**
  - **Third last session: Due on midnight next day (complete version with some doc/screen shots of running the code)**
- **Your grade on FP exercises will be partly based on these submitted codes and what I observe during the in-class sessions.**
- **If received helps from others and/or used codes from others, please credit the person who helped you.**

Platforms

- MATLAB
  - MathWorks: http://www.mathworks.com/
  - http://en.wikipedia.org/wiki/MATLAB
- MATLAB @ SFSU
  - https://at.sfsu.edu/at-mathworks-matlab
- Various tutorials available online
  - https://matlabacademy.mathworks.com/?s_tid=acb_tut

Public Libraries

- OpenCV (Computer Vision)
  - http://www.intel.com/technology/computing/opencv/overview.htm
- ITK (Medical Imaging)
  - http://www.itk.org/
- WEKA (Machine Learning)
  - http://www.cs.waikato.ac.nz/~ml/weka/index.html

Face Recognition by Eigenface

- Let's create a face recognition system using one of the most basic algorithm called "**Eigenface**".
  - You have not studied this in the lecture yet but
  - You read a paper on this (Turk & Pentland)
- You will need to implement 3 components
  1) **Image I/O + visualization**
  2) **PCA for learning/training**
  3) **Recognition by nearest neighbor classification**

Paper 1

- **M. Turk, A. Pentland,**
- ***Eigenfaces for Recognition*, Journal of Cognitive Neuroscience, 3(1): 71-86 (1991)**
- http://portal.acm.org/citation.cfm?id=1326887.1326894&coll=&dl=
- http://en.wikipedia.org/wiki/Eigenface

Data

- I provide a set of facial images
- https://bidal.sfsu.edu/~kazokada/csc872/DATA/FaceRecognition_Data.zip
- Images are organized in 3 folders
- **ALL** = FA+FB (for **Training**) ← PCA
- **FA**: 12 32x32 8bit facial images (for **Known faces DB**)
- **FB**: 23 facial images (for **Test Set**)

Principal Component Analysis

- Conceptual Steps
  1) Collect M Training Images (must be aligned, Nx by Ny matrix)
  2) Vectorize the Images: $X = \{x_1, .., x_M\}$ Each of M images is a column vector with N coefficients where $N = Nx$ times Ny
  3) Compute mean image: $\mu = mean(X)$; a vector of N coeffs
  4) Construct Covariance Matrix: $C = (X - \mu^T)(X - \mu^T)^T$ N by N mat
  5) Solve Eigenvalue Problem: $Cv_i = \lambda_i v_i$
  6) Sort resulting eigen vectors in decreasing order of corresponding eigen values.
  7) Select the top K Eigenvectors $W = \{v_1, .., v_K\}$, resulting in a face model $\{\mu, W\}$

Nearest Neighbor Recognition

- Learning & Database Construction
  1) Do PCA, yielding a face model $\{\mu, W\}$
  2) Construct DB of known faces with codes $y_j = W^T(x_j - \mu^T)$ for all known faces $\{x_j\}$ in FA
- Face Recognition by NN Classification
  1) Test face z is also projected to the model $W^T(z - \mu^T) = y_z$
  2) Nearest neighbor classification of y_z with $\{y_i\}$ by picking the index "i" that best match to y_z according to Euclidean distance

TRAINING | TESTING

TRAINING

$X \leftarrow$ {all images in ALL}

① DO PCA on X

↓ Eigen Face space ↓ mean face

$W = \{ [\,] [\,] \cdots [\,] \}$ (ev_1 ev_2 … ev_N), $\mu = [\,]$

② Choose K = 10/20

→ W_K {[ ] … [ ]} (ev_1 … ev_K)

TESTING

Ⓐ Prepare Know Face DB. M=12

For all images in FA: $\{y_1, \dots, y_M\}$

Compute $c_i' = W_K^T (y_i - \mu)$ (K coefficients)

Resulting in $\{c_1, \dots, c_M\}$

Ⓑ Testing! (FR!)

Given a test image $z \in$ FB.

Compute $d_z = W_K^T (z - \mu)$

Compute distances from d_z to $\{c_i\}$

Find index $i^* \in 1, \dots M$ with min dist.

Useful MATLAB Codes

For PCA doc ← cov

- Set X as a matrix with each row is a vectorized face
- m = mean(X): sample mean of X, pay attention to dim.
- M = repmat(μ',1,N); create a matrix by repeating a column matrix μ' N times (M will be length of μ x N)
- S = cov(X): covariance matrix (mean removed)
- [V D] = eig(S): eigen value decomposition of a matrix S
  - Each column of V is an eigen vector.
  - D is a diagonal matrix of eigen values.
  - Columns of V and D are corresponding to each other
- d = diag(D); vectorize the diagonal component of a matrix
- Use for-loop to get cumulative distribution of eigen values then divide it by the total variance (sum(diag(D)))
- Plot(cumulative distribution of eigen values)