Final Project: Sparse Representation Based Face Recognition

This project will introduce a widely used face recognition method -- **sparse representation based face recognition** [1].

◆ **Project Requirements:**

1. This is an individual project. No teamwork required.
2. No plagiarism. Stanford MOSS will be used for plagiarism detection.
3. Basic MATLAB skills are required. You can either refer to the documents, which is highly recommended, or come to tutorial sessions for help.
4. Report (written in **English** and **LATEX** Only) covers the following topics:
   a) The algorithm. Don’t copy and paste words from this description. Use your own words. Otherwise, you may fail to pass the plagiarism test.
   b) Related linear algebra knowledge for each section.
   c) How to run your code.
   d) How you implemented.
   e) Possible improvements you can make.

   ****Don’t** put your code into the report. Pack it into zip file.
5. Test your program on The Extended Yale Face Database B. Use the cropped images.
6. Name your files in the following manner:
   a) In total, you should hand in at most 2 MATLAB code files.
      i. **SRBFR.m** contains all the steps required. It should have the following input-output interface: **[ACCURACY] = SRBFR(numTrainSamples, filePath)**
      ii. If you write your own PCA function, it should have interface **[COEFF, SCORE, latent] = PCA(X)**. Name the file as **PCA.m**

   b) Name your report as **SRBFR_Report_StudentID_yourNameOnEmail.pdf** (e.g., SRBFR_Report_12345678_zhangqq.pdf).
   c) Pack all the above files into a zip file: **SRBFR_StudentID_yourNameOnEmail.zip** (e.g. SRBFR_12345678_zhangqq.zip).
7. Please email the zip file to **si131_2016@163.com** with subject **FinalProject_StudentID_name** (e.g., FinalProject_12345678_ZhangQiqi), before **24:00 Dec 28, 2016.**

◆ **Problem description:**

In this project, we want you to go through a simple face recognition algorithm, using the knowledge of linear algebra. Here is what you need to do:

1. From the database, randomly read in a same number of images for each person as training samples; create a face matrix using the training samples.
2. Use **Principal Component Analysis (PCA)** to reduce the dimension of the face matrix. Obtain the Eigenface matrix. Here are two choices for you (the second one comes with bonus credits):
   a) You can use the built-in functions **[COEFF, SCORE, latent] = princomp(X)** to do this. You may read the documentation for further details.
   b) Write your own PCA function with the same input-output interface of the built-in function. For further information, please read the background knowledge part.
3. Use the provided feature_sign.m to solve the following problem:

$$\min_{\boldsymbol{x}}: \frac{1}{2}|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}|_2^2 + \lambda\,|\boldsymbol{x}|_1$$

   i.e. to minimize the square error $|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}|_2^2$ under the l_1-norm constraint of the coefficient vector **x**. Note that ***y*** is a test face image, ***A*** is the Eigenface matrix obtained in Step 2, λ is a Lagrange multiplier to be optimized, and ***x*** is a sparse coefficient vector. The l_1-norm ensures that the coefficient vector **x** is sparse. **Why should x be sparse?**
4. Evaluate your program. Check your results with the ground truth. Give the accuracy. Optimize your program to run fast. Give numbers of pairs containing the ground truth and the estimated face, show them in report.

◆ **Background knowledge:**

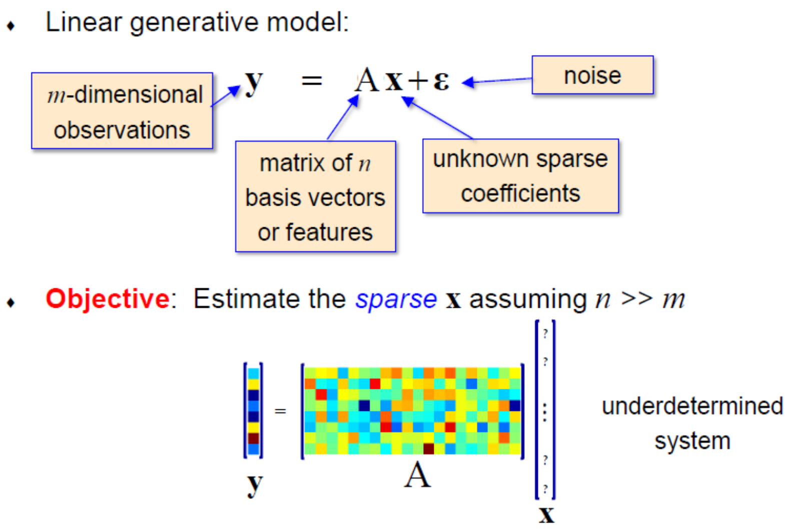
The method we use is a supervised machine learning method. “Learning” is a fancy word for extracting information from the inputs. Here, the machine needs to “learn” a function f(x) which can categorize the test image by trained classifier. “Supervised” means that when you “train” your classifier. Every time you input a pair of an image and its label for training. Whether labeled or not distinguishes the supervised and unsupervised learning mechanism.

This project is about how a computer “see” the world. For computer, an image is simply a set of numbers, a kind of 2-dimensional signal. As human beings, we treat an image as a whole, rather than a set of numbers. And the more we see, the better our knowledge of the given object will be. Analogously, a computer will be supplied with a set of face photos for a certain person as the training samples, so as to learn the face of the person. Other pictures of the same person will be used as the test data.

Solving the recognition problem translates to solving the following linear system:

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{e} = \boldsymbol{y}$$

$\boldsymbol{A}$ is the matrix consisting of features from the training samples, and $\boldsymbol{x}$ is the coefficients vector, $\boldsymbol{y}$ is your test samples, and $\boldsymbol{e}$ is the error. We want to solve for a coefficient vector that is as sparse as possible, while minimizing the error $\boldsymbol{e}$. A “sparse” vector means many zeros in the vector. This picture should give you intuition:

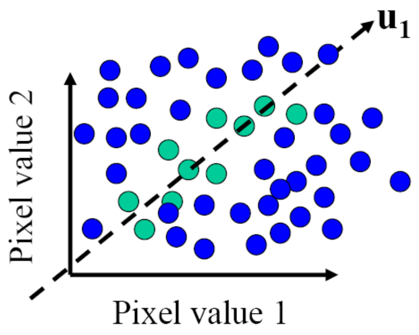


Linear generative model:
m-dimensional observations
$\mathbf{y} = A\mathbf{x} + \varepsilon$
noise
matrix of n basis vectors or features
unknown sparse coefficients
Objective: Estimate the *sparse* $\mathbf{x}$ assuming $n >> m$
y
=
A
x
underdetermined system

■ Principal Component Analysis (PCA)

We use **the space of all face images** to illustrate why we need this:

- When viewed as vectors of pixel values, face images are extremely high-dimensional
    - 100x100 image = 10,000 dimensions
    - Slow and lots of storage
- But very few 10,000-dimensional vectors are **valid** face images. Lots of redundancy
- We want to effectively model the subspace of face images
- **Eigenface** idea: construct a low-dimensional linear subspace that best **explains the variation** in the set of face images
- A useful link: https://en.wikipedia.org/wiki/Principal_component_analysis



● A face image
● A (non-face) image

Here comes how you do this:

- Given: N data points $\mathbf{x}_1, \dots, \mathbf{x}_N$ in R^d
- We want to find a new set of features that are linear combinations of original ones:

$$U(\mathbf{x}_i) = \mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})$$

  ($\boldsymbol{\mu}$: mean of data points)

- Choose unit vector **u** in R^d that captures the most data variance
- Direction that maximizes the variance of the projected data:

  Maximize (subject to $\|\boldsymbol{u}\|=1$)

$$Var(\boldsymbol{u}) = \frac{1}{N}\sum_{i=1}^{N-1} \underbrace{\mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})}_{\text{Projection of data point}} \left(\mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})\right)^T$$

$$= \boldsymbol{u}^T \underbrace{\left[\frac{1}{N}\sum_{i=1}^{N-1}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T\right]}_{\text{Covariance matrix of data}} \boldsymbol{u}$$

$$= \boldsymbol{u}^T \Sigma \boldsymbol{u}$$

The direction that maximizes the variance is the eigenvector associated with the largest eigenvalue of Σ. How to solve this?

■ **Eigenfaces (PCA on face images)** [2]

1. Compute the principal components ("eigenfaces") of the covariance matrix

$$\boldsymbol{X} = [(\boldsymbol{x}_1 - \boldsymbol{\mu}) \ (\boldsymbol{x}_2 - \boldsymbol{\mu}) \ \dots \ (\boldsymbol{x}_n - \boldsymbol{\mu})]$$

$$[\boldsymbol{U}, \boldsymbol{\lambda}] = \text{eig}(\boldsymbol{X}^T\boldsymbol{X})$$

$$\boldsymbol{V} = \boldsymbol{X}\boldsymbol{U}$$

2. Keep K eigenvectors with largest eigenvalues. Usually we want this dimension reduction preserves **95%** of the original information.

$$\boldsymbol{V} = \boldsymbol{V}(:,\ \text{largest\_eig})$$

3. Represent all face images in the dataset as linear combinations of eigenfaces
   - Perform nearest neighbor on these coefficients

$$\boldsymbol{X_{pca}} = \boldsymbol{V}\left(:,\ \textbf{largest}_{\textbf{eig}}\right)^{\boldsymbol{T}} \boldsymbol{X}$$

If you use the built-in function **princomp, please carefully read the MATLAB document for its details.

Some instructions you might need to use (you want to read the MATLAB documents for details and examples):

imread: read the image file by its name string

reshape: reshape the input matrix

A’: taking transpose of input matrix A

A\B: for solving linear equation A*x = B, x = A\B.

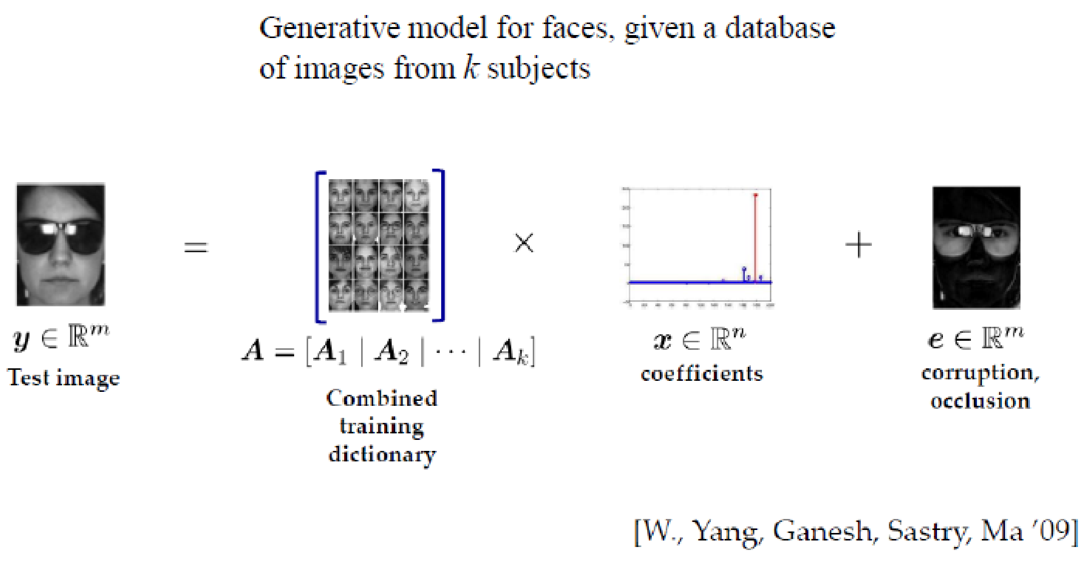
inv: taking inverse of input matrix.

.*, ./: dot product and dot division, entry-wise operations.

Option:

■ **Sparse representation based face recognition:**

This picture will give you some intuitions.


Generative model for faces, given a database of images from k subjects
$y \in \mathbb{R}^m$
Test image
=
$A = [A_1 \mid A_2 \mid \cdots \mid A_k]$
Combined training dictionary
×
$x \in \mathbb{R}^n$
coefficients
+
$e \in \mathbb{R}^m$
corruption, occlusion
[W., Yang, Ganesh, Sastry, Ma '09]

◆ **Reference paper:**

1. Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., & Ma, Y. (2009). Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, *31*(2), 210-227.

2. Turk, M. A., & Pentland, A. P. (1991, June). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*(pp. 586-591). IEEE.