School of Computing
National University of Singapore
CS 5240 Theory and Practice of Multimedia
Semester 1, 2010/11

Homework #4
Due: in class on 21 Oct. 2010

To prepare for this homework, please review the material in the following files: `estimation.pdf`, `crlb.pdf`

Q1. Figure 1 shows the joint pdf of two discrete random variables X and Y. Each random variable is drawn from the sample space $\{-1, 0, 1\}$. The joint probabilities are shown as fractions in the figure.

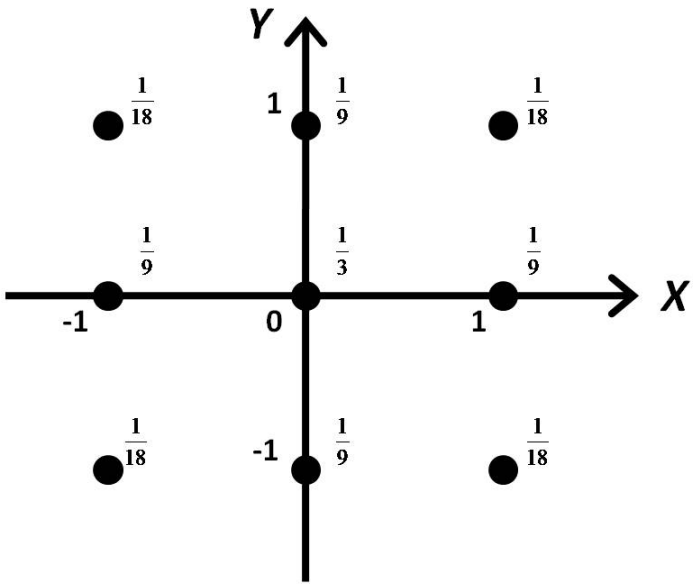

Figure 1: Joint pdf of random variables X, Y.

(a). Find the marginal pdf $P_X(x)$.

(b). Are X and Y (i) orthogonal? (ii) correlated? (iii) statistically independent?

(c). A new random variable Z is defined as $Z = \min(X, Y)$. Find the pdf $P_Z(z)$.

Q2(a). Prove the following theorem.

Theorem: Let X be a random variable with an arbitrary PDF $P_X(x)$. Define a new random variable $Y = F_X(x) = \int_{-\infty}^{x} P_X(w)\,dw$. Then Y follows a Uniform PDF, i.e.

$$P_Y(y) = \begin{cases} 1, & 0 \le y \le 1 \\ 0, & \text{otherwise} \end{cases}$$

Q2(b). Most computer languages provide a `random()` function that generates a (pseudo-) random variable U that is uniformly distributed between 0 and 1. This is not sufficient, however, since you typically want a random variable Z with some other pdf $P_Z(z)$. For instance, you may want a Gaussian pdf. How would you use the theorem in Q2(a) to calculate the desired Z from U?

Q3. Let $\mathcal{X} = \{X_1, X_2, \ldots, X_N\}$ be a set of N observations, where each X_i is an iid random variable drawn from the Uniform pdf: $X_i \sim Uniform(0, \theta)$. It is easy to see that the joint pdf is:

$$P_{\mathcal{X}}(X_1, \ldots, X_N \mid \theta) = \begin{cases} \theta^{-N}, & \text{if all } X_i \in [0, \theta] \\ 0, & \text{otherwise} \end{cases}$$

(a). Sketch the likelihood function, as a function of θ.

(b). Show that the maximum likelihood estimate is $\hat{\theta}_{ML} = \max(X_1, \ldots, X_N)$.

(c). Prove that the CDF of $\hat{\theta}_{ML}$ is: $F_{\hat{\theta}_{ML}}(z) = \left(\frac{z}{\theta}\right)^N$, for $0 \le z \le \theta$

(d). Hence, show that the PDF is $P_{\hat{\theta}_{ML}}(z) = \frac{N z^{N-1}}{\theta^N}$, for $0 \le z \le \theta$

(e). Show that $E[\hat{\theta}_{ML}] = \frac{N}{N+1}\theta$. Is $\hat{\theta}_{ML}$ unbiased?

(f). Determine $Var[\hat{\theta}_{ML}]$. Is $\hat{\theta}_{ML}$ consistent?

Q4. The Land Transport Authority (LTA) is concerned because a particular stretch of the ECP highway is very prone to accidents. The LTA wants to deploy an emergency vehicle to patrol the area, so as to quickly respond to any emergency that may arise.

The problem may be modeled as follows: let X_1 denote the position of an accident (when it occurs) on the highway, and let X_2 denote the position of the emergency vehicle at the time of the accident. Let the length of the highway be 1 km. Then we can say that X_1 is uniformly distributed between 0 and 1. That is, $X_1 \sim Uniform(0,1)$. Likewise, $X_2 \sim Uniform(0,1)$ and is independent of X_1. When an accident occurs, the emergency vehicle is immediately notified, and it rushes to the accident site. Assume that the vehicle can U-turn anywhere along the highway.

(a). Let $D = |X_1 - X_2|$. This is the distance the vehicle has to travel to reach the accident site. Prove that the Cumulative Distribution Function (CDF) of D is given by: $F_D(y) = 1 - (1-y)^2, \quad 0 \leq y \leq 1$.

(b). Calculate the numeric values of $E[D]$ and $Var[D]$.

(c). Suppose the emergency vehicle travels at a constant speed S. Then the time taken to reach the accident site is $T = D/S$. Show that the conditional pdf of T, given S, is: $P(t|s) = 2s(1-ts), \quad 0 \leq t \leq 1/s$.

(d). The driver of the vehicle tells you that, in the most recent accident, he took 50 seconds to reach the accident site. How fast was he travelling? Determine the Maximum Likelikhood speed.

For Q1 to Q4, submit your answers as hardcopy in class during lecture. Handwritten answers are fine.

Q5. **Eigenfaces exercise:** Download the file `faces.zip` from the Homework Workbin of IVLE, and extract it to a suitable directory. In this, you will find some face images taken from the CMU PIE database. There are 10 people, with each person captured under 24 different lighting conditions, for a total of 240 images. For convenience, these images have been split into two equal sets in the `test` and `train` directories.

The image files have names `ppppp_xx_yy.bmp`, where `ppppp` denotes the identity of the person; `xx` denotes the head orientation (all are frontal (=27) in this exercise); and `yy` denotes the lighting condition. All images have been cropped and aligned, and their height, width are 160 and 140 pixels, respectively.

The goal in this exercise is to use Principal Components Analysis (PCA) to recognize face images. Read all instructions first before coding. Some planning is required. The main tasks are:

1. Compute the PCA projection matrix $\mathbf{W}$ using the training images.
2. Project all training images using $\mathbf{W}$, and represent each person (class) by the mean projected vector of his training images. Classification will be performed using the nearest mean.
3. Evaluate the performance of the PCA classifier using the test images.

1 Compute the PCA projection matrix

Using the `sys.path.append` command, add the `faces` directory to your Python path, so that the file `EigenFace.py` is visible to Python.

Study the code in `EigenFace.py`. Four functions have been defined for you: `myPCA, ComputeNorm, float2int8, read_train_faces`. Their purpose is described in the code. Use `read_train_faces` to read in all the training images, reshape each image into a vector, and place all the vectors in to a data matrix.

You should now have a 22400×120 matrix called `faces`, whose columns are all the face images from the `train` directory.

Using the `myPCA` function, compute the PCA projection matrix $\mathbf{W}$, the global mean[1] vector $\mathbf{m}$, and the vector of eigenvalues $\mathbf{\Lambda}$. Read the code in `myPCA.m` to understand how this is done. In particular, note the use of the inner product trick (as explained in the lecture notes) to avoid an "Out of memory" error.

Visualize the top 8 eigenfaces (principal components corresponding to the 8 largest eigenvalues) in $\mathbf{W}$ as follows: re-shape each of them back into a 160×140 2D matrix using the Numpy function `reshape`, then use the Pyplot `subplot` command to create a 3×3 array of plots, and display the eigenfaces in these plots. Use the provided `float2int8` function to scale the pixel values to between 0 and 255. In the bottom right plot, display the mean face $\mathbf{m}$. Entitle each plot with "Eigenface 1", "Eigenface 2", ..., "mean", so that it is clear which image is which. **Submit these plots.**

[1] That is, the mean of all images, regardless of their class.

2 Project each training image

Retain only the top K ($K = 30$) eigenfaces, by typing `W = W[:,1:K]`.

For each vector $\mathbf{x}_i$ in `faces`, project it onto "PCA space" using: $\mathbf{y}_i = \mathbf{W}^\top(\mathbf{x}_i - \mathbf{m})$. Group these $\mathbf{y}_i$ vectors according to their classes,[2] and for each class j, compute the mean vector $\mathbf{z}_j$ from all the $\mathbf{y}_i$ belonging to that class. Each person j is now represented by his PCA mean vector $\mathbf{z}_j$. Store these vectors as columns in a Numpy matrix `Z`.

For convenience, it is best to store the columns of `Z` so that `Z(:,1)` corresponds to person `04010`, `Z(:,2)` corresponds to person `04011`, etc. Check that your `Z` matrix has size $K \times 10$.

3 Evaluate the performance

Your PCA classifier is now ready. To classify a new face image $\mathbf{x}$ (re-shaped into a vector), first project it onto PCA space using $\mathbf{y} = \mathbf{W}^\top(\mathbf{x} - \mathbf{m})$. Then search for the mean vector in the matrix `Z` that is closest to $\mathbf{y}$, using the Euclidean distance metric. The index of the nearest mean vector in `Z` reveals the identity of $\mathbf{x}$. For example, if the nearest mean vector is column 2, then the identity is `04011`.

You can now evaluate the performance of your classifier. Using the images in the `test` directory, classify each of them with your classifier as explained above.

The *Confusion Matrix* is a useful way to evaluate the performance a classifier. Each element c_{ij} of an $M \times M$ Confusion Matrix $\mathbf{C}$ is the number of times an image from person i is classified as person j by the classifier. Thus the perfect classifier should produce a diagonal Confusion Matrix. Any non-zero off-diagonal element in $\mathbf{C}$ represents an error. The overall accuracy of the classifier can be calculated as $trace(\mathbf{C})/(\sum_{i,j} c_{i,j})$, i.e. the trace divided by the sum of all elements.

Calculate the 10×10 Confusion Matrix as follows. First, initialize all entries in the Confusion Matrix to 0. Then, for each test image, let *output_id* be the identity that your classifier outputs, and let *actual_id* be the actual identity of the test image (which you can determine from its filename). Add 1 to the entry in the *actual_id*-th row and *output_id*-th column of the Confusion Matrix. **Answer the following questions:**

(a). Submit your Confusion Matrix. What is the accuracy A of your classifier?

(b). Of all the off-diagonal entries in the Confusion Matrix, which has the largest value?

(c). In (b) above, suppose the largest value occurs in row R and column C, display one face image of person R and one of person C. Submit these faces. Do they look alike?

[2] If you used the `read_train_faces` function to read the training images, the filenames should already be sorted and grouped by classes. Thus the first 12 vectors in `faces` will belong to class 1, the next 12 to class 2, etc.

(d). Is there an entire column (or row) in the Confusion Matrix that is non-zero? If so, display that face, and suggest an explanation why this face is problematic.

(e). Let $K = 5, 6, 8, 10, 15, 20, 25$. Re-train your classifier and re-calculate its accuracy A for each K. Plot accuracy A versus K, for $K = 5, 6, 8, 10, 15, 20, 25, 30$. Submit this plot. What do you observe?

What to submit for Q5

Write a MS Word or PDF file containing your answers to all the questions for Q5, and submit it to the HW5_submission folder in IVLE Workbin. For the plot of eigenfaces, and the plot of accuracy versus K, you may include the plots as figures in your MS Word or PDF file, or else submit them as JPEG images. If you are submitting multiple files, please put all your files into a single ZIP file and submit only 1 file. Remember to write your name and matric. number in your report. Please do *not* submit your Python code.

Submit your answers to Q5 via the Workbin. Submit your answers for Q1 to Q4 as hardcopy (handwritten is fine) in class during the lecture.

Useful functions

Some useful functions are listed below. Please refer to the online documentation for more details: `www.scipy.org/Tentative_NumPy_Tutorial`,
`www.pythonware.com/library/pil/handbook/index.htm`,
`http://matplotlib.sourceforge.net/users/index.html`

`Numpy.dot` multiplies two arrays.

`Numpy.transpose` transposes an array.

`Numpy.tile` repeats a column multiple times.

`Numpy.astype` changes the data type of the array.

`Numpy.asarray` converts a PIL Image to a Numpy array.

`Image.fromarray` converts an array to a PIL Image.

`Numpy.zeros` creates an array of 0's.

`Numpy.trace, Numpy.sum, Numpy.sqrt` compute the trace, sum and square-root.

— END —