ICT582 Major Assignment 2020

This assignment aims to provide a Python solution to a specific simulation problem. To solve this problem, you need the knowledge gained from the first Topics of this Unit. You may also need to undertake **independent research** to find out **some Python packages, modules and functions required in your solution**.

This assignment is worth 35% of the total unit assessment.

This is an individual assignment and must be completed by you alone. Any unauthorised collaboration/collusion may be subject to investigation and result in penalties if found to be in breach of University policy.

Learning Outcomes	Assessed in this Assignment
Students should be able to design and write correct and readable small programs practical data processing problems in Python;	☑
Students should be able to read, understand and debug computer programs.	☑
Students should understand some practical limitations on computer programs, including scaling (w.r.t. time and memory) and numeric precision (rounding errors) issues.	☑
Students should be able to solve many kinds of practical problems using programming as a primary tool	☑

Description of the Problem

Automatic Face Recognition System

Face recognition is used as an authentication process in various fields and especially in computer security related activities, such as homeland security, building access security, criminal identification, as well as user identification in small mobile devices. The goal of a face recognition system is to have a negligible misclassification rate. Face recognition also plays a significant role in the research field of biometric. Biometric technology is used for authentication and it may analyse human behaviour.

In this project, you are required to develop an automatic face recognition system. The idea is to develop intelligent software using Python to process images and recognize faces in those images. You will divide your images into training and test data (Image dataset has been provided). In practice, the test data will have different images from the training dataset. For example, if there are 10 images of a subject S1, use 5 images for training and the remaining 5 images for testing. Assuming there is a total of 40 subjects, you will have 40x5 images in the training set and 40x5 images in the test set.

Face Recognition Algorithm

In this project, we will use a simple Linear Regression Classification Algorithm[1] (details below). You are not allowed to use any **machine learning** package/module to complete this assignment. You cannot use any built-in/ready to use linear regression functions. You must implement the following algorithm in Python.

Let there be N number of distinguished classes with p_i number of training images from the ith class, $i = 1, 2, \ldots, N$. Each gray-scale training image is of an order $a \times b$ and is represented as $\overset{(m)}{u_i} \in \mathbb{R}^{a \times b}$, $i = 1, 2, \ldots, N$ and $m = 1, 2, \ldots, p_i$. Each gallery image is downsampled to an order $c \times d$ and transformed to vector through column concatenation such that $\overset{(m)}{u_i} \in \mathbb{R}^{a \times b} \rightarrow \overset{(m)}{\mathbf{w}_i} \in \mathbb{R}^{q \times 1}$, where $q = cd$, $cd < ab$. Each image vector is normalized so that maximum pixel value is 1. Using the concept that patterns from the same class lie on a linear subspace [9], we develop a class-specific model $\mathbf{X}_i$ by stacking the q-dimensional image vectors,

$$\mathbf{X}_i = \left[\overset{(1)}{\mathbf{w}_i} \; \overset{(2)}{\mathbf{w}_i} \; \ldots\ldots \; \overset{(p_i)}{\mathbf{w}_i} \right] \in \mathbb{R}^{q \times p_i}, \quad i = 1, 2, \ldots, N \qquad (1)$$

Each vector $\overset{(m)}{\mathbf{w}_i}$, $m = 1, 2, \ldots, p_i$, spans a subspace of $\mathbb{R}^q$ also called the column space of $\mathbf{X}_i$. Therefore, at the training level, each class i is represented by a vector subspace, $\mathbf{X}_i$, which is also called the *regressor* or *predictor* for class i. Let z be an unlabeled test image and our problem is to classify z as one of the classes $i = 1, 2, \ldots, N$. We transform and normalize the gray-scale image z to an image vector $\mathbf{y} \in \mathbb{R}^{q \times 1}$ as discussed for the gallery. If $\mathbf{y}$ belongs to the ith class, it should be represented as a linear combination of the training images from the same class (lying in the same subspace), i.e.,

$$\mathbf{y} = \mathbf{X}_i \beta_i, \quad i = 1, 2, \ldots, N, \qquad (2)$$

1. Naseem, I., Togneri, R., & Bennamoun, M. (2010). Linear regression for face recognition. *IEEE transactions on pattern analysis and machine intelligence*, *32*(11), 2106-2112.

where $\beta_i \in \mathbb{R}^{p_i \times 1}$ is the vector of parameters. Given that $q \geq p_i$, the system of equations in (2) is well conditioned and β_i can be estimated using least-squares estimation

$$\hat{\beta}_i = \left(\mathbf{X}_i^T \mathbf{X}_i\right)^{-1} \mathbf{X}_i^T \mathbf{y}. \tag{3}$$

The estimated vector of parameters, $\hat{\beta}_i$, along with the predictors $\mathbf{X}_i$ are used to predict the response vector for each class i:

$$\begin{aligned} \hat{\mathbf{y}}_i &= \mathbf{X}_i \hat{\beta}_i, \quad i = 1, 2, \ldots, N \\ \hat{\mathbf{y}}_i &= \mathbf{X}_i \left(\mathbf{X}_i^T \mathbf{X}_i\right)^{-1} \mathbf{X}_i^T \mathbf{y} \\ \hat{\mathbf{y}}_i &= \mathbf{H}_i \mathbf{y}, \end{aligned} \tag{4}$$

where the predicted vector $\hat{\mathbf{y}}_i \in \mathbb{R}^{q \times 1}$ is the projection of $\mathbf{y}$ onto the ith subspace. In other words, $\hat{\mathbf{y}}_i$ is the closest vector, in the ith subspace, to the observation vector $\mathbf{y}$ in the euclidean sense [16]. $\mathbf{H}$ is called a *hat matrix* since it maps $\mathbf{y}$ into $\hat{\mathbf{y}}_i$. We now calculate the distance measure between the predicted response vector $\hat{\mathbf{y}}_i$, $i = 1, 2, \ldots, N$, and the original response vector $\mathbf{y}$,

$$d_i(\mathbf{y}) = \|\mathbf{y} - \hat{\mathbf{y}}_i\|_2, \quad i = 1, 2, \ldots, N \tag{5}$$

and rule in favor of the class with minimum distance, i.e.,

$$\underbrace{\min}_{i} d_i(\mathbf{y}), \quad i = 1, 2, \ldots, N. \tag{6}$$

Implementation Details

There are two phases to develop in this face recognition system (1) the training phase and (2) the testing phase.

Training Phase. In the training phase, you will develop a class-specific model for training images as explained above. Hint: Equation 1 to 3 will be used in this phase. Training phase involves development of matrices as in Eq. 1 to 3.

For a given image dataset, you can use 50% of the images for training. Remember, 50% per subject i.e., if there are 10 images per subject, use 5 images for training. Once you have your training set, down-sample (e.g., 10 x 5) and normalize your training images as stated above.

Testing Phase. Use the remaining 50% images for testing. Down-sample the test images as you did during the training phase. Compute the original response/label (training data/model will be required) and the predicted response (label) for the test image. Compare these responses using the distance measure, as stated above, to predict the class of the test image. Based on the quality and consistency of the prediction, decide which class corresponds best to the test image.

Display your input test image and any random image from the predicted class. Once you have predicted classes for all the test images, report your overall recognition accuracy.

Hint: Eq. 4 to 6 will be used in this phase.

Required Features:

Develop a Python program for face recognition algorithm. The program should have the functionality to trigger training of your linear regression model and load a test image one by one, predict the class of the test face/image and display the results. The recognition should be fully automatic, and you may only specify the directory where the training and test images are present.

All students are required to implement these training and test features. This part will be assessed in terms of completeness in meeting the prescribed feature requirement, correct implementation of the algorithm, quality of your code, the robustness and reliability of your program, and the overall quality of your documentation which includes the adherence to Documentation and Submission Requirements.

Training and Testing using your Data

In this part of assignment, you are required to capture your and your friends' images (10 each). You can create new folders in already provided face dataset to save your images. Train and test your face recognition algorithm using these new images. You are required to upload these new images for the assessment of your algorithm.

Documentation and Submission Requirements

Your submission must be in the form of a ZIP archive file consisting of:

1) one Microsoft Word file named "Assignment1.docx" (details below) containing a critical analysis and documentation of your solution, and challenges faced during this assignment;

2) the Python file for your solution to the problem.

3) your own image dataset.

You must submit the ZIP file using ICT582 Unit LMS on or before the deadline. You can submit up to three files if the file size exceeds the size limit imposed by the LMS.

A Microsoft Word document named "Assignment1.doc" containing the documentation of your solution to the problem. It must contain the following documents and these documents must be layout in the following order:

i. The "ICT582 Major Assignment Check List" is available from the Assignment section. All students must complete this form.

ii. If you have been granted extension, include the email from your Unit Coordinator.

iii. The detailed description of the problem you aim to solve.

iv. **Self-diagnosis, evaluation and declaration.** You must provide a full and detailed declaration of the following: the features that are fully implemented and fully working, the features that are not fully working, and the features that are not implemented. Where possible, you should also identify the possible causes of the problems for those features that are not fully working.

v. A brief description of your solution to the problem. The length of your description should not exceed one page.

vi. Evidence that your solution meets each requirement of the assignment, including each of the requirements you have specified for the self-selected advanced features. You can paste program outputs as evidence.

Please note that although your tutor may test your program to verify the evidence presented in your documentation, it is not the responsibility of your tutor to test your program for the purpose of finding marks for you. It is up to individual student to mount a convincing case that the submitted solution meets all requirements. You will lose a significant number of marks (up to 70% of the assignment) if the evidence you presented is not convincing or not complete, even if your program actually works.

vii. The source code listing - including python file of your application.

The ZIP file name must conform to the following format:

ICT582_MajorAssignment_FirstName_Surname.zip. For example, someone with the name of John Smith should name his assignment ZIP file as ICT582_MajorAssignment_John_Smith.zip

The ZIP file must be submitted using ICT582 Unit LMS. Please be aware that the Unit LMS will not accept your assignment if it is late by more than 5 days.

The above documentation and submission requirements will be strictly enforced. Your assignment will not be marked, or your marks will be significantly reduced, if you fail to adhere to the above requirements.

Distribution of Marks

Item No.	**Feature/Functionality**	**Percentage (%)**
	Python Program (Correctness of algorithm will be tested for item 1, 2 and 3)	
1	Correct training implementation (Eq. 1 to 3)	20
2	Correct testing implementation (Eq. 4 to 6)	25
3	Display of images (predicted and test)	10
4	Quality of your code	10
	Word Document	
5	Description of the problem	5
6	Self-diagnosis, evaluation and declaration	20
7	Description of your solution	10
	Total	100

Deadline and Penalty for Late Submission

The deadline for submitting this assignment is given in the respective Assignment Submission section of the Unit LMS.

Unless you have exceptional circumstances, be advised that late submissions will incur a penalty of 10% of the total marks per day (including weekends and public holidays). Please see the further explanation on how the number of days is calculated in the following paragraph. Work submitted more than **5 days** late will not be accepted.

In calculating the number of days late, a fractional day is rounded up to one whole day. For example, if you submit your assignment one day and three hours (1.125 days) after the deadline, your submission is considered to be late for two whole days under this rule, hence your marks will be deducted by 20%.

Applications for extension of your assignment deadline can only be made via email to the Unit Coordinator or his delegate, normally a week before the specified due date of the assignment. If an extension is granted (also by email), you must attach a copy of the email to your submission (see Documentation and Submission Requirements). Applications for extension by phone or in person do not count even if granted. The above policy will be rigorously enforced.

Errata

This document is subject to change based on errors found. Any errors found and corrected will be posted in the Unit Announcements page of the Unit LMS. If you print out a copy of this page, please follow the news in the Unit Announcements page.