The Hong Kong University of Science and Technology
Department of Computer Science and Engineering
CSIT 5410 (Spring 2020)

Assignment 4

Total = 100 marks
Due: 11:55pm, May 22, 2020
Assignments must be submitted via CANVAS
Late Policy: 10% reduction; only one day late is allowed, i.e. 11:55pm, May 23

In this written assignment, you need to give answers to the following three questions.

Written assignment specifics (15%)

1. PCA

(1) There are three 3x1 face images (see below). Each image has three intensity values. You need to compute the scatter matrix. (Round results to 4 decimal places. Show all steps.)

$$\vec{x}_1 = \begin{bmatrix} 2 \\ 1 \\ 5 \end{bmatrix} \qquad \vec{x}_2 = \begin{bmatrix} 3 \\ 2 \\ 7 \end{bmatrix} \qquad \vec{x}_3 = \begin{bmatrix} 2 \\ 6 \\ 6 \end{bmatrix}$$

(2) If the eigenvectors $(\vec{e}_1, \vec{e}_2, \vec{e}_3)$ and eigenvalues $(\lambda_1, \lambda_2, \lambda_3)$ are given as below,

$$\lambda_1 = 0, \vec{e}_1 = \begin{bmatrix} -0.8701 \\ -0.0967 \\ 0.4834 \end{bmatrix} \qquad \lambda_2 = 2.5215, \vec{e}_2 = \begin{bmatrix} -0.4882 \\ 0.0334 \\ -0.8721 \end{bmatrix}$$

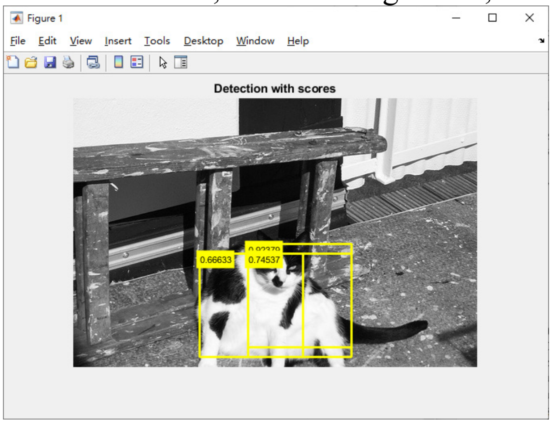
$$\lambda_3 = 14.1452, \vec{e}_3 = \begin{bmatrix} -0.0681 \\ 0.9948 \\ 0.0763 \end{bmatrix}$$

Given a test face image, $\vec{x}_4 = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$, you need to compute the approximation of $\vec{x}_4$ by using the eigenvectors with non-zero eigenvalues. Please identify $\vec{x}_4$'s the closest labeled face among $\vec{x}_1, \vec{x}_2, \vec{x}_3$ based on the nearest neighbor in **the t-dimensional space**. (Round results to 2 decimal places. Show all steps.)

Please save your written parts as PDF file and rename it to "written.pdf".

Programming assignment specifics (85%)

In this section, you are required to implement a simple object detection system with the Adaboost algorithm. Specifically, the object detection system is able to find out all the cats inside an image and label it with a bounding box. The object detection system consists of feature extraction, a set of weak classifiers, Adaboost algorithm, and sliding window.



Prerequisite

Before you start, you need to download the PASCAL VOC 2007 dataset and the toolkit of the dataset at http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html.

The PASCAL VOC 2007:
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCtrainval_06-Nov-2007.tar

The toolkit:
http://host.robots.ox.ac.uk/pascal/VOC/voc2007/VOCdevkit_08-Jun-2007.tar

And download the starting code and the testing images on CANVAS.

After that, create an empty folder "{{your 8-digits student ID}}_assignment4", unzip the above files into the folder and put the provided csit5410_test.txt into the folder VOC2007\ImageSets\Main. Your workspace for assignment 4 should contain:

"{{your 8-digits student ID}}_assignment4"
- -test_images % Images for testing the object detection system
- -VOC2007 % VOC 2007 dataset with annotations
- -VOCcode % Some helper functions for VOC 2007
- -csit5410_assignment4.m % Main function of this assignment 4, Programming Task 3
- -feature_extract.m % Programming Task 1
- -train_weak_classifier.m % Programming Task 2
- -viewanno.m % Example of reading an image with its annotation from VOC 2007
- -and some other *.m files % Not necessary for this assignment

Programming task 1 - Feature extraction (20%)

In this task, you are required to complete the function in `feature_extract.m`. The function `feature_extract`, with format `fea= feature_extract(Im)`, computes the feature vector `fea` from the input image `Im`. The feature type must be either the Harr-like feature (Rectangle features) or the Local binary patterns (LBP) described in the lecture. You are allowed to modify the arguments list of the function. You must complete this function with mathematic operators and matrix operators. Third-party libraries or any related build-in functions are not allowed.

If you are not able to finish this function, you may want to use "`extractLBPFeatures`" from Matlab. In that case, no marks will be given for this task.

Programming task 2 - Weak classifiers (20%)

In this task, you are required to complete the function in `train_weak_classifier.m`. The function `train_weak_classifier.m` with format `model=train_weak_classifier(feature_type, ...)`, returns a trained weak classifier `model`, trained with feature specified by the input string `feature_type`. In this assignment, you are required to create **at least five unique weak classifiers.** Each weak classifier should be different in feature type/feature-length/classifier. For example, Harr-like features computed on different set of spatial positions can be regarded as different features.

2.1 Training data

For each weak classifier, it must be trained with images specified in VOC2007\ImageSets\Main\cat_train.txt. The example of visualizing the training data can be found in `viewanno.m`, e.g.: Run the command `viewanno('Main/cat_train')`. Note that it is not necessary to utilize all the images specified in cat_train.txt.

2.2 Pre-processing (optional)

As the images may vary in light condition or size, pre-processing may be necessary for object detection. A suggested pre-processing pipeline is shown below:

1. Convert the RGB image into grayscale using rgb2gray function
2. Histogram equalization
3. Resize the bounding box images into fixed size `h x w`, e.g.: `128 x 128`.

2.3 Training a classifier

Given a pre-processed image cropped by the annotation (bounding box), the purpose of the classification is to classify the image into two categories, "cat" or "non-cat". In order to achieve that, you need to train your own classifier. First, you are required to compute the feature vectors from a set of pre-processed images using the function in `feature_extract.m` (Programming task 1). After that, fed the computed feature vector and the ground truth ("cat" or "non-cat") into a custom classifier, e.g.: Support-vector machine (SVM) or other classifier. For the classifier, you are allowed to use any built-in function related to it. The example of using SVM for binary classification is available in https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html.

2.4 Model size limitation

To facilitate the grading process, you are required to save your trained classifier into "{{model name}}_model.mat" using the command "save(filename,variables)". The model/variable size for each weak classifier **must not exceed 15MB**. If your size of the model is exceeding the size limit, you may need to reduce the length/dimension of the computed feature vector.

Programming task 3 - Adaboost algorithm and sliding window (30%)

In this task, you are required to complete the script in `csit5410_assignment4.m`. Specifically, this script first loads all the trained weak classifiers and classifies the unseen validation data specified in cat_val.txt. Then, an Adaboost algorithm is applied to all the weak classifiers to compute a strong classifier. Finally, the strong classifier classifies the image patches, extracted by a sliding window algorithm, to locate the object "cat" within the images (provided in the "test_images" folder).

3.1 Weak classifiers

The script `csit5410_assignment4.m` first load all the pre-trained weak classifiers, e.g.: using the function "`load(filename)`". Then, these weak classifiers are used to classify all the annotated image patches specified in the given csit5410_test.txt. You are required to report the classification result for all the weak classifiers in the followings format:

```
>> Correctness (Weak Classifier 1): {{number of correct prediction}}/{{number of image patches}}
>> Correctness (Weak Classifier 2): {{number of correct prediction}}/{{number of image patches}}
...
```

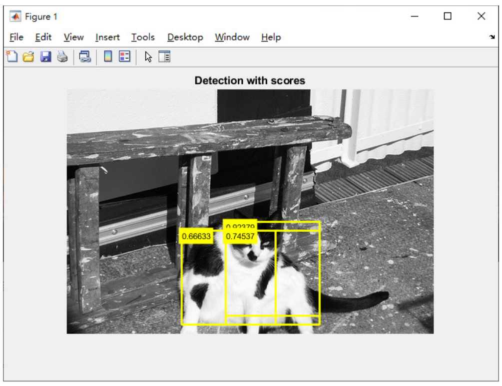
3.2 Adaboost Algorithm

After section 3.1, you are required to implement the Adaboost algorithm described in the lecture (recognizing-faces.pdf). This Adaboost algorithm takes all the weak classifiers and its prediction results on images in “VOC2007\ImageSets\Main\cat_val.txt” as input and outputs a final strong classifier, which formed by at least 5 weighted weak classifiers. Similarity, you are required to report and print out the classification result for the strong classifier in the followings format:

```
>> Correctness (Strong Classifier): {{number of correct prediction}}/{{number of image patches}}
```

3.3 Sliding window

You are required to implement the sliding window algorithm to extract fixed-size, e.g.: 128x128, image patches from images in folder "test_images". The concept of the sliding window is described in https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/. Noted that the stride for the sliding window is flexible. You are allowed to fine-tune the stride size to speed up your object detection algorithm. For each image patches extracted by the sliding window, a strong classifier (computed in section 3.2) is applied to classify whether the image patch contains a "cat" object or not. Finally, you are required to display (e.g. imshow()) the detection results overlay with three top-3 scores/confidences bounding boxes for each image in the folder "test_images" as followings:



Note that we will test your algorithm with a set of new testing images. The detection accuracy for the testing images is accounted for 10% scores in this assignment.

3.4 Running time

The script `csit5410_assignment4.m` must be completed **within 10 minutes**. Running time for `csit5410_assignment4.m` exceed the time limit will subject to marks deduction. Note that you must make sure all your codes can be run without a runtime error. Otherwise, **no marks will be given.**

Programming task 4 – Report (15%)

You are required to submit a report describing the details and results of your object detection system in PDF format and rename it as "report.pdf". Your report should cover the followings:

- A brief explanation of your weak classifiers
- A brief explanation of your preprocessing chain (if any)
- The selected weak classifiers and its weight after the Adaboost algorithm
- The classification accuracy of each weak classifier and strong classifier on images specified in csit5410_test.txt
- The detection results of the given images in the "test_images" folder, a maximum of 3 bounding boxes per image.

Assignment Submission and Marking

1. Your submitted programs must **include all the material excluding the VOC 2007 dataset (folder "VOC2007").**

2. You must include a README.txt file indicating the programming software (Octave or MATLAB) that you are using for this assignment.

3. Runtime requirement: your program must be able to finish all the above tasks within 10 minutes.

4. You must compress all your files with the following filename format: [your 8-digit student ID]_assign4.rar (or zip), e.g. 09654321_assign4.rar, into one file.

5. If your assignment compressed file has been submitted multiple times before the due date (including late submission date), the new version will replace the old version in marking.

6. Note that we take plagiarism seriously. You are allowed to discuss or share your idea to your classmate, but **you are not allowed to share your code/pseudocode** of your assignment. Please also follow the referencing skills at https://libguides.ust.hk/referencing/plagiarism to avoid plagiarism.

7. Marks would be deducted if there are any violations of the above requirements.

~~ End of Assignment 4 ~~