Homework #1

Deep Learning for Computer Vision

Due: 109/10/13 (Tue.) 02:00 AM

Total Score: 120 points

Contact TAs by ntudlcv@gmail.com

- **Academic Honesty**

  Plagiarism/cheating is strictly prohibited and against university policy. If any form of plagiarism is discovered, **ALL** students involved would receive an *F* score for this course (not just for HW).

- **Collaboration Policy**

  Searching for online materials or discussing with fellow classmates are highly encouraged. However, you must provide the code or solution by yourself. Please specify, if any, the references for any parts of your HW solution in your report (e.g., the name and student ID of your collaborators and/or the Internet URL you consult with). If you complete the assignment all by yourself, you must also specify "*no collaborators*".

- **Late HW Submission Policy**

  No late submission is allowed for this homework.

Problem 1: K-Means Clustering (24%)

K-Means Clustering is an unsupervised learning algorithm for data grouping. In image segmentation, it can be applied to partition image pixels into different groups based on the associated pixel values or features. In this problem, you will learn how to segment the provided image by using K-means clustering.

1. (10%) For $K = 2, 4, 8, 16$, and 32, perform K-means clustering on the provided `bird.jpg` by taking the RGB values of each pixel as the feature of interest. Take a 64×64 pixel color image for example, we have a total of $64 \times 64 = 4096$ data points for K-means clustering, and each data point is described as a three dimensional vector (i.e., (R, G, B)). To visualize your image segmentation results, plot the clustering results by replacing all pixels' RGB value in each cluster with the that of the corresponding cluster center.

2. (6%) Repeat 1. but take both RGB values and the location (x and y) as a five dimensional vector as the feature for describing each pixel. Show the segmentation results.

3. (8%) Compare your results obtained in 1. and 2., and briefly explain the differences between the two methods under the same K. If further improved segmentation results would be desirable, please provide possible modification or extension to the above feature definition (and visualize your results).

✓ ***Hint***

- When performing K-means clustering, your pixel values (RGB) should be in the range of $[0, 255]$.
- In 2., x and y denote the horizontal and vertical distances (i.e., number of pixels in each direction from the upper left corner of the image), respectively.

Problem 2: Principal Component Analysis (60%)

Principal component analysis (PCA) is a technique of dimensionality reduction, which linearly maps data onto a lower-dimensional space, so that the variance of the projected data in the associated dimensions would be maximized. In this problem, you will perform PCA on a dataset of face images.

The folder `p2_data` contains face images of 40 different subjects (classes) and 10 grayscale images for each subject, all of size $(56, 46)$ pixels. Note that `i_j.png` is the j-th image of the i-th person, which is denoted as **person**$_i$**image**$_j$ for simplicity.

First, split the dataset into two subsets (i.e., training and testing sets). The first subset contains the first 9 images of each subject, while the second subset contains the remaining images. Thus, a total of $9 \times 40 = 360$ images are in the training set, and $1 \times 40 = 40$ images in the testing set.

In this problem, you will compute the eigenfaces of the training set, and project face images from both the training and testing sets onto the same feature space with reduced dimension.

1. (15%) Perform PCA on the training set. Plot the mean face and the first four eigenfaces.
2. (12%) Take **person**$_2$**image**$_1$, and project it onto the PCA eigenspace you obtained above. Reconstruct this image using the first $n = 3, 50, 170, 240, 345$ eigenfaces. Plot the four reconstructed images.
3. (6%) For each of the four images you obtained in 2., compute the mean squared error (MSE) between the reconstructed image and the original image. Record the corresponding MSE values in your report.
4. (15%) Now, apply the k-nearest neighbors algorithm to classify the testing set images. First, you will need to determine the best k and n values by 3-fold cross-validation. For simplicity, the choices for such hyperparameters are $k = \{1, 3, 5\}$ and $n = \{3, 50, 170\}$. Show the cross-validation results and explain your choice for (k, n).
5. (12%) Use your hyperparameter choice in 4. and report the recognition rate of the testing set.

✓ ***Hint***

- When plotting eigenfaces, be sure to start from the most dominant one to the least dominant one. Note that the calculated eigenvalues may be sorted in either ascending or descending order depending on the programming language/packages you use.
- Display your output faces in grayscale colormap instead of other ones.
- When calculating MSE, your pixel values should be in the range of $[0, 255]$.

Problem 3: Image Filtering (36%)

Image filtering is a basic method to smoothen or sharpen an image. The general operation of image filtering is to compute the function of local neighborhood and output the new value for each pixel of interest. For example, a 3×3 box filter simply replaces the value of each pixel of interest with average values of itself and its eight neighbors. For more details about image filtering, please refer to the slides.

In this problem, you will be implementing **Gaussian filters** and apply them to images for filtering purposes. Below is the definition of 2D kernel of a Gaussian filter:

$$\text{2D kernel :} \qquad G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1. (12%) Implement a discrete 2D Gaussian filter using a 3×3 kernel with $\sigma \approx \frac{1}{2\ln 2}$. Use the provided `lena.png` as input, and plot the output image in your report. Briefly describe the effect of the filter.

2. (16%) Consider the image $I(x,y)$ as a function $I : \mathbb{R}^2 \to \mathbb{R}$. When detecting edges in an image, it is often important to extract information from the derivatives of pixel values. Denote the derivatives as follows:

$$I_x(x,y) = \frac{\partial I}{\partial x} \approx \frac{1}{2}(I(x+1,y) - I(x-1,y))$$
$$I_y(x,y) = \frac{\partial I}{\partial y} \approx \frac{1}{2}(I(x,y+1) - I(x,y-1)).$$

   Implement the 1D convolution kernels $k_x \in \mathbb{R}^{1\times 3}$ and $k_y \in \mathbb{R}^{3\times 1}$ such that

$$I_x = k_x * I$$
$$I_y = k_y * I.$$

   Write down your answers of k_x and k_y. Also, plot the resulting images I_x and I_y using the provided `lena.png` as input.

3. (8%) Define the **gradient magnitude** image I_m as

$$I_m(x,y) = \sqrt{I_x(x,y)^2 + I_y(x,y)^2}.$$

   Use **both** the provided `lena.png` **and** the Gaussian-filtered image you obtained in 1. as input images. Plot the two output gradient magnitude images in your report. Briefly explain the differences in the results.

✓ ***Hint***

- When using `lena.png` for this problem, be sure to load it as a *grayscale* image.

Remarks

- For this homework, we will **not** grade your source code. Thus, you may use **any** programming language you feel comfortable with. You are also allowed to use any related packages, libraries, and functions for your implementation. However, you must provide image outputs with detailed discussions or explanations.

- Please convert your report into a single `.pdf` file (with arbitrary file name) and upload it to CEIBA before the deadline. You do **NOT** need to upload anything other than the pdf report for this homework.