Principal Component Analysis*

Ke Chen

To carry out this assessed coursework, you will need the Python notebook, `pca.ipynb`, which contains code to get you started with each of the assignments, and the data files, `Yale_64x64.npy` for Yale face dataset to be used in the image compression assignments, and a subset of this dataset, `Yale_64x64-part-4-persons.npy`, to be used in the face recognition assignments. In two datasets, data are organised in the column vector notation, i.e., each column corresponds to a data point in the data matrix. These are all available in a zipped file on BlackBoard alongside this document.

As a linear model, principal component analysis (PCA) lays foundation to enable representation learning. As one of the most important dimension reduction techniques, PCA can be applied to high dimensional data for visualisation, data compression and feature extraction. In this coursework, you are asked to implement two versions of PCA algorithm in Python and apply your own PCA implementations to real data sets for visualisation, image compression and feature extraction used in face recognition.

1 PCA Implementation

There are two versions of PCA algorithms, **PCA** and **dual PCA**[1]. The PCA algorithm is applicable to the scenario that the number of instances is larger than that of features in a training dataset, while the dual PCA algorithm is required for efficient computation in the scenario that the number of instances is smaller than that of features in a training dataset. In this part, you are asked to use the appropriate built-in functions in Python to implement two PCA algorithms to produce **all non-zero eigenvalues and their corresponding eigenvectors** for the covariance of a dataset expressed in the column format; i.e., each column refers to a data point. So your implementation must work on this notation and your code must be commented out properly.

Assignment 1 [4 marks] In your answer notebook, implement the PCA algorithm in Python for the requirements stated above. This function signature must be set to `my_pca(X)`, where `X` is an input data matrix in the column format. (**Hint**: in this implementation, you can use the built-in function, `np.linalg.eig`, in the `numpy` library.)

Assignment 2 [4 marks] In your answer notebook, implement the dual PCA algorithm in Python for the requirements stated above. This function signature must be set to `my_dual_pca(X)`, where `X` is an input data matrix in the column format. (**Hint**: in this implementation, you can use the built-in function, `np.linalg.svd`, in the `numpy` library.)

*__Assessed Coursework__: the deadline and requirements can be found at the end of this document.

[1] For the detailed algorithmic description of two PCA algorithms, see "Principal Component Analysis (PCA)" lecture notes.

2 Visualisation

As an off-shelf technique, PCA is often used for visualisation of high-dimensional data. In this part, you are asked to use your own PCA implementation, `my_pca(X)`, to the **IRIS** dataset. The IRIS dataset[2] is a built-in dataset in the `scikit-learn` library where there are 150 4-D data points.

Assignment 3 [4 marks] Apply `my_pca(X)` to the IRIS data set and then project all 150 data points onto a two-dimensional PCA subspace consisting of two principal components for visualisation. Let PC_1, PC_2 and PC_3 denote top three principal components of this data set, respectively. In your answer notebook, (a) plot the 2-D projection results on the PC_1-PC_2, PC_1-PC_3 and PC_2-PC_3 subspaces, respectively. In each plot, you are asked to use the first PC as x-axis and the second PC as y-axis in a specific PC pair, e.g., for the PC_1-PC_2, you should use PC_1 as x-axis and PC_2 as y-axis in the plot, and (b) based on three plots achieved in (a), describe any non-trivial properties you observe for this 4-D data set.

Note: If you fail to complete **Assignment 1**, you can still do the assignment in this part with the built-in PCA function, `sklearn.decomposition.PCA`, in the `scikit-learn` library. In this case, however, up to **3 marks** are given for the assignment in this part.

3 Image Compression

As a dimension reduction technique, PCA can be used as a lossy data compression method. In this part, you are asked to apply your dual PCA implementation, `my_dual_pca(X)`, to the Yale face dataset[3], `Yale_64x64.npy`, for image compression. This facial data set has been split into two subsets: `train` of 150 images for achieving the PCA projection matrix and `test` of 15 images for evaluating the generalisation of your PCA learning.

Assignment 4 [3 mark] In your answer notebook, (a) describe how you can decide a number of principal components to enable you to establish a satisfactory image compression system for this data set; (b) using the method given in (a), report the number of of principal components (eigenfaces), k, based on the experimental evidence; and (c) display the k eigenface images by using the provided `print_image` function. In your answer notebook, arrange k eigenface images according to their corresponding eigenvalues from large to small in a grid structure where each row contains 8 images (or fewer in the last row if k is not divisible by 8).

Assignment 5 [4 marks] For 15 images in the test subset, you are asked to report your image compression results with the eigenfaces achieved in **Assignment 4**. In your answer notebook, (a) list all the low-dimensional representations of all the test images in a table in $k \times 15$ format; (b) display all the reconstructed test images by using the provided `print_image` function and arranging 15 test images in a 3×5 grid; (c) describe a proper evaluation criterion for measuring the loss between an original raw and its corresponding reconstructed images; (d) program the evaluation criterion

[2] The detailed information on this dataset can be found on https://archive.ics.uci.edu/ml/datasets/iris.

[3] The detailed information on this dataset can be found on http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html.

given in (c) and report the loss calculated with your code for each of 15 test images.

Note: If you fail to complete **Assignment 2**, you can still do two assignments in this part with the built-in PCA function, `sklearn.decomposition.PCA`, in the `scikit-learn` library. In this case, however, up to **5 marks** are given for all the assignments this part.

4 Face Recognition

For face recognition, PCA is a well-known feature extraction method, leading to so-called "eigenface" features. The features can be incorporated into a standard machine learning pipeline for face recognition. In this part, you are asked to establish two face recognition systems: one by means of the "eigenfaces" achieved in **Assignment 4** and the other directly trained on the raw images. To do this experiment, you are given a facial data set, `Yale_64x64-part-4-persons.npy`, which contains 11 images/person for 2 different people chosen from the Yale face dataset used in **Part 3**. This two-people data set has been split into two subsets: `train` of 8 images/person for training and `test` of 3 images for evaluating the performance of your face recognition systems. In your face recognition system, you must use non-linear support vector machine (SVM) with RBF kernel as your classifier. In this experiment, you need to use knowledge of the SVM learned from COMP61011 and the SVM built-in functions in the `scikit-learn` library (Note: all the functions in the `scikit-learn` library work on data organised in the row format; i.e., each row in data matrix corresponds to a data point, which is different from your PCA implementations).

Assignment 6 [6 marks] In your answer notebook, (a) describe how you establish two face recognition systems (with and without the use of PCA), respectively, with mark-up comments, code and experimental evidence, which should include the details in both the training and the test phases; and (b) for each of two face recognition systems, report its recognition accuracy and the confusion matrix on test subset.

Requirement: Before starting working on this assessed coursework, you need to

1. download all the files required by this coursework from Blackboard as specified at the beginning of this document;
2. unzip the file then you should be see a Jupyter notebook file named `pca.ipynb` and one sub-directory named `Data` (you must keep this directory/sub-directory structure and its name unchanged when you work on this coursework);
3. rename `pca.ipynb` in the directory as `yourfullname_pca.ipynb`. For instance, if your name is "John Smith", your filename should be `john_smith_pca.ipynb`. This file will be your answer notebook to be submitted for marking, so you must include everything required by the coursework in this Jupyter notebook.

Deliverable: Only your answer notebook, `yourfullname_pca.ipynb`, which should include all your code, output, answers and your interpretation/justification. In this Jupyter notebook, all assignments have been separated with the clear delimiters. You must put your stuff regarding an

assignment in those cells related to this assignment and, if necessary, create new cells within the delimiters of this assignment.

Your answer notebook, `yourfullname_pca.ipynb`**, must be submitted via the Blackboard.**

Marking: Marking will be on the basis of (1) correctness of results and quality of comments on your code; (2) rigorous experimentation; (3) how informative and clear your results and answers are presented in your answer book; and (4) your knowledge exhibited, interpretation and justification.

Late Submission Policy: The default departmental late submission policy is applied to this coursework.

Deadline: 23:30, 22nd December 2020 (Tuesday)