# CS589: Machine Learning - Fall 2017

## Homework 5: Unsupervised learning

Assigned: November $20^{th}$ Due: December $11^{th}$

**Getting Started:** In this assignment, you will perform unsupervised learning to compress images from two datasets. **Please install Python 3.6 via Anaconda on your personal machine**. Download the homework file HW05.zip via Moodle. Unzipping this folder will create the directory structure shown below,

```
HW05
--- HW05.pdf
--- Data
    |--Faces
    |--Scene
--- Submission
    |--Code
    |--Figures
```

The data files are in 'Data' directory respectively. You will write your code under the Submission/Code directory. Make sure to put the deliverables (explained below) into the respective directories.

**Deliverables:** This assignment has two types of deliverables:

- **Report:** The solution report will give your answers to the homework questions (listed below). Try to keep the maximum length of the report to 5 pages in 11 point font, including all figures and tables. Reports longer than five pages will only be graded up until the first five pages. If you have answered extra credit questions then you can exceed the five page limit but make sure put only your solutions to extra credit questions on pages six and after. You can use any software to create your report, but your report must be submitted in PDF format.
- **Code:** The second deliverable is the code that you wrote to answer the questions, which will involve performing unsupervised learning. Your code must be in Python 3.6 (no iPython notebooks or other formats). You may create any additional source files to structure your code. However, you should aim to write your code so that it is possible to reproduce all of your experimental results exactly by running *python run_me.py* file from the Submissions/Code directory.

**Submitting Deliverables:** When you complete the assignment, you will upload your report and your code using the Gradescope.com service. Place your final code in Submission/Code. If you generated any figures place them under Submission/Figures. Finally, create a zip file of your submission directory, Submission.zip (NO rar, tar or other formats). Upload this single zip file on Gradescope as your solution to the 'HW05-Unsupervised-Programming' assignment. Gradescope will run checks to determine if your submission contains the required files in the correct locations. Finally, upload your pdf report to the 'HW05-Unsupervised-Report' assignment. When you upload your report please make sure to select the correct pages for each

question respectively. Failure to select the correct pages will result in point deductions. The submission time for your assignment is considered to be the later of the submission timestamps of your code and report submissions.

**Academic Honesty Statement:** Copying solutions from external sources (books, internet, etc.) or other students is considered cheating. Sharing your solutions with other students is also considered cheating. Posting your code to public repositories like GitHub, stackoverflow is also considered cheating. Any detected cheating will result in a grade of -100% on the assignment for all students involved, and potentially a grade of F in the course.

**Task:**

Unsupervised learning: Contrary to supervised learning (classification, regression), unsupervised learning algorithms learn patterns from unlabeled examples. There are several popular algorithms in unsupervised learning such as principal component analysis (PCA), k-means, independent component analysis (ICA) and density estimation. In this project you will use PCA and k-means to compress images.

Algorithms and datasets:

**PCA:** This algorithm finds directions that maximize the variance in the data (or minimize the information loss) when the data from the original dimensions is *projected* onto these directions. You are given a dataset of 100 gray scale $50 \times 50$ images. Sample images are shown in Figure 1. You will use PCA to compress these images.

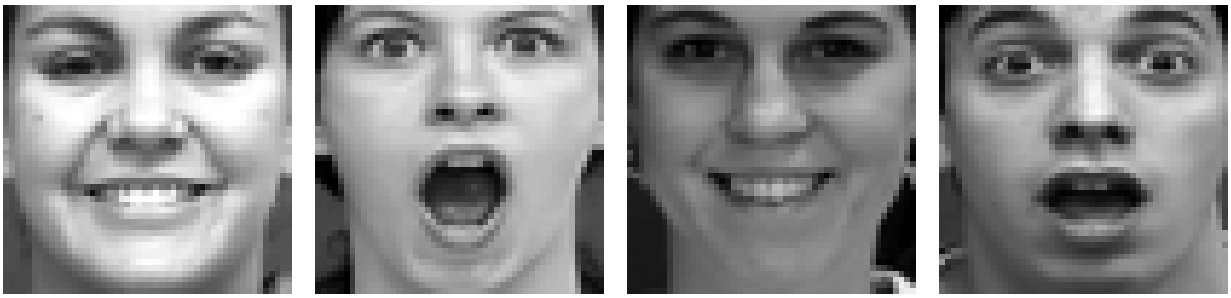

Figure 1: Sample faces

**k-means:** Is a clustering algorithm that finds centroids of clusters and assigns each sample to one and only one of these clusters according to some criteria. You will use k-means to compress an image of times square as shown in Figure 2.


Figure 2: Image to compress using k-means

**Questions:**

## 1. (*55 points*) PCA:

**(10) a.** Suppose that you are given a dataset with $N$ samples, each of dimension $p$. Show that the direction that maximizes the variance (minimizes reconstruction error) of the data is generated by the eigenvector associated to the largest eigenvalue of the estimated covariance matrix of the data.

*Hint:* There are many ways of doing this. One is to solve

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} ||x_i - \hat{x_i}||^2 \text{ subject to } ||\mathbf{w}|| = 1$$

with $x_i$ being the i-th sample placed as a column vector, and $\hat{x_i} = (x_i^T \cdot \mathbf{w})\mathbf{w}$ being the projection of the i-th sample onto the direction $\mathbf{w}$. Recall that the estimated covariance matrix of the data is $C_x = \frac{1}{N} \sum_{i=1}^{N} x_i \, x_i^T$.

**(5) b.** Show that the subspace of dimension two that maximizes the variance (minimizes the reconstruction error) of the data is generated by the two eigenvectors associated to the largest two eigenvalues of the estimated covariance matrix of the data.

**(8) c.** Let $x$ be a sample and let $x_i$ represent the i-th component of $x$ (as in question a, $x$ has dimension $p$). Suppose that for every sample in the dataset $x_p = \sum_{i=1}^{p-1} \alpha_i \, x_i$. What is the minimum number of directions (eigenvectors of the estimated covariance matrix) needed to store the data perfectly (ie. no loss of information)? Explain in at most 4 sentences.

*Hint:* Thinking about the specific case when $p = 2$ or $p = 3$ might help.

**(22) d.** You will now perform PCA on the faces dataset. You are given a set of 100 images ($50 \times 50$ pixels in gray scale) containing faces. Each data example is a matrix $x_i \in \mathbb{R}^{50 \times 50}$, and can be

stored as a vector $x_i \in \mathbb{R}^{2500}$. The data is organized into a design matrix $\mathbf{X}$ with $N$ rows and $p$ columns, where $N = 100$ and $p = 2500$. You can compute the covariance matrix of the data as $C_x = \frac{1}{N}\sum_{i=1}^{N} x_i\, x_i^T \in \mathbb{R}^{2500\times 2500}$. Use this covariance matrix to compute the eigenvalues and eigenvectors which we denote as $\mathbf{v} \in \mathbb{R}^{2500}$ and $\mathbf{w} \in \mathbb{R}^{2500\times 2500}$ respectively. Your task is to compress the entire dataset using $k$ eigenvectors corresponding to $k$ largest eigenvalues. For this question you have to implement PCA using only a package that computes covariance, eigenvalues and eigenvectors of a matrix (ie. NO package that implements PCA directly can be used).

1. In the first part you will look at reconstructing a compressed image with different values of $k$. For this part you will need to visually inspect *face_1.png* only. The images are numbered from zero and hence face 1 will be the second row in the design matrix, $\mathbf{X}$. Our approach is to perform PCA on the entire dataset and then visually inspect face 1. You will experiment with different values of $k$ in the range $\{3, 5, 10, 30, 50, 100, 150, 300\}$. The reconstructed dataset can be represented as $\mathbf{X}_{recon} = \mathbf{X}_{projected}{\mathbf{w_k}}^T$, where $\mathbf{X}_{projected} = \mathbf{X}\mathbf{w_k}$. Here $\mathbf{w_k}$ denotes a matrix with $k$ eigenvectors corresponding to $k$ largest eigenvalues. Followng this make a $3 \times 3$ grid plot of the original face 1 along with eight reconstructed faces corresponding to eight different values of $k$. Make sure to label the images.
2. Report the average squared reconstruction error as $\sqrt{\text{mean}(\mathbf{X} - \mathbf{X_{recon}})^2}$, for the **whole dataset**. For each value of k report the reconstruction error using the table below.
3. Another way to assess image compression is to look at the compression rate. The compression rate is the memory required to store the compressed images ($\mathbf{X}_{projected}$) divided by the memory required to store the original images ($\mathbf{X}$). In this case, the former includes the space required to store the $k$ principal components (one $\mathbf{w_k}$ for the entire dataset). For each value of $k$ report the compression rate using the table below. In python, you can use numpy's nbytes to get the number of bytes consumed by an array, vector, etc.

| k | Reconstruction error | Compression rate |
|---|---|---|
| 3 | | |
| 5 | | |
| 10 | | |
| 30 | | |
| 50 | | |
| 100 | | |
| 150 | | |
| 300 | | |

Table 1: Reconstruction error and compression rate for faces dataset

Write in atmost four sentences on using PCA for image compression using qualitative plots ($3 \times 3$ grid) that you made above as well as quantitative metrics like reconstruction error and compression rate.

## 2. (*45 points*) K-means:

**(5) a.** K-means is a simple unsupervised learning algorithm that splits the data into clusters. There are

different ways to determine the "optimal" number of clusters; the elbow rule being a very simple one. Explain it in at most 4 sentences.

**(5) b.** Another issue with k-means is that the random initialization of the centroids can sometimes lead to "poor" clusters. A possible solution to this problem is presented in the algorithm called k-means++. Briefly explain the idea behind this algorithm.

**(35) c.** You are given an RGB image *times_square.jpg* as a $400 \times 400 \times 3$ matrix. Each pixel can be seen as a sample of dimension 3 (3 integers between 0 and 255, one for each component of RGB). For this question you will treat each pixel as a data sample. You are encouraged to use *sklearn's* implementation of *kmeans* for this question.


Figure 3: Example of reconstructed image using 5 clusters

1. Apply k-means using $k$ clusters in the range $\{2, 5, 10, 25, 50, 75, 100, 200\}$ (note that in this case each cluster will represent an RGB color triplet). Replace each pixel in the original image with the centroid of the cluster assigned to that pixel. Followng this make a $3 \times 3$ grid plot of the original times square image along with eight reconstructed images corresponding to eight different values of $k$. Make sure to label the images. An example of the original image and reconstructed image is shown in Figure 3. Helper code is given in 'run_me.py' file to convert the $400 \times 400 \times 3$ matrix to $160,000 \times 3$ matrix and vice versa.
2. For each value of $k$ report using a table like above the recontruction error
3. For each value of $k$ report using a table like above the compression rate. Note that in this case each pixel of the original image uses 24 bits, each centroid is represented by 3 *floats* (each one uses 32 bits), and an integer from 1 to $k$ needs $ceiling(\log_2 k)$ bits (for each pixel in the image you store the index of the centroid assigned).
4. Make a plot of the sum of squared errors of each pixel to its respective cluster centroids for diferent values of $k$ (elbow plot). If the range is too big then make the plot in $\log$ space but make sure to indicate that in your report.

Write in atmost five sentences on using k-means for image compression using qualitative plots ($3 \times 3$

grid; elbow plot) that you made above as well as quantitative metrics like reconstruction error and compression rate.