Homework 6
unsupervised learning

CMU 10-601: Machine Learning (Spring 2017)
https://piazza.com/cmu/spring2017/10601
OUT: March 22, 2017
DUE: April 03, 2017 11:59 PM
Authors: Megha Arora, Yongjin Cho, Wei Ma

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the collaboration policy on the website for more information: http://www.cs.cmu.edu/~mgormley/courses/10601-s17/about.html
- **Late Submission Policy:** See the late submission policy here: http://www.cs.cmu.edu/~mgormley/courses/10601-s17/about.html
- **Submitting your work:** Each homework will consist of two parts. For this homework, the first part will be submitted via QnA. The second part of the assignment will be a programming question. For this question, you will submit code via Autolab and answer related questions on QnA as detailed below.
  - **QnA:** We will use an online system called QnA for short answer and multiple choice questions. You can log in with your Andrew ID and password using the button labeled "LOGIN WITH CMU ID". (As a reminder, never enter your Andrew password into any website unless you have first checked that the URL starts with "https://" and the domain name ends in ".cmu.edu" – but in this case it's OK since both conditions are met) A link to the Homework 6 QnA section is provided in Problem 1 below. The deadline displayed on QnA may not correspond to the actual deadline for this homework, since we are allowing late submissions (as discussed in the late submission policy on the course site).
  - **Autolab:** You can access the 10601 course on autolab by going to https://autolab.andrew.cmu.edu/ Using All programming assignments will be graded automatically on Autolab using Octave 3.8.2 and Python 2.7. You may develop your code in your favorite IDE, but please make sure that it runs as expected on Octave 3.8.2 or Python 2.7 before submitting. The code which you write will be executed remotely against a suite of tests, and the results are used to automatically assign you a grade. To make sure your code executes correctly on our servers, you should avoid using libraries which are not present in the basic Octave install. For Python users, you are encouraged to use the `numpy` package. The version of `numpy` used on Autolab is 1.7.1. The deadline displayed on Autolab may not correspond to the actual deadline for this homework, since we are allowing late submissions (as discussed in the late submission policy on the course site).

Problem 1: QnA [50 points]

This problem consists of a set of multiple choice and numerical questions posted on the QnA platform at the URL below:

`https://qna.cs.cmu.edu/#/pages/view/123`

Question 24 to 29 on QnA are based on Problem 2 of this homework and should not be attempted until you have read through Problem 2 below.

QnA allows you to submit multiple times for the same question. We will grade the last answer you submit before the homework deadline.

Problem 2: k-means Clustering [50 points]

In this problem you will implement Lloyd's method for the k-means clustering problem and answer several questions about the k-means objective, Lloyd's method, and k-means++.

Recall that given a set $X = \{x_1, \dots, x_n\} \in R^d$ of n points in d-dimensional space, the goal of k-means clustering is to find a set of centers $c_1, \dots, c_k \in R^d$ that minimize the k-means objective:

$$\sum_{j=1}^{n} \min_{i \in \{1,\dots,k\}} \|x_j - c_i\|_2^2 , \tag{1}$$

which measures the sum of squared distances from each point x_j to its nearest center.

In class we discussed that finding the optimal centers for the k-means objective is NP-hard, which means that there is likely no algorithm that can efficiently compute the optimal centers. Instead, we often use Lloyd's method, which is a heuristic algorithm for minimizing the k-means objective that is efficient in practice and often outputs reasonably good clusterings. Lloyd's method maintains a set of centers $c_1, \dots, c_k$ and a partitioning of the data X into k clusters, $C_1, \dots, C_k$. The algorithm alternates between two steps: (i) improving the partitioning $C_1, \dots, C_k$ by reassigning each point to the cluster with the nearest center, and (ii) improving the centers $c_1, \dots, c_k$ by setting c_i to be the mean of those points in the set C_i for $i = 1, \dots, k$. Typically, these two steps are repeated until the clustering converges (i.e, the partitioning $C_1, \dots, C_k$ remains unchanged after an update). Pseudocode is given below:

```
1. Initialize the centers $c_1, \dots, c_k$ and the partition $C_1, \dots, C_k$ arbitrarily.
2. Do the following until the partitioning $C_1, \dots, C_k$ does not change:
    i. For each cluster index $i$, let $C_i = \{x \in X : x$ is closer to $c_i$ than any other center$\}$, breaking ties
       using the indices corresponding to the first occurrence.
    ii. For each cluster index $i$, let $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
```

Programming Instructions

You will implement some functions for training Lloyd's k-means Clustering method for this question. You will submit your code online through the CMU autolab system, which will execute it remotely against a suite of tests. Your grade will be automatically determined from the testing results.

To reduce the load of Autolab server, you are allowed to submit at most 30 times to Autolab. If you use up all the submissions, we will grade on your last submission regardless of whether it is finished or not. So we suggest you thoroughly test your codes locally and then make the submission.

To get started, you can log into the autolab website (`https://autolab.andrew.cmu.edu`). From there you should see 10-601 in your list of courses. Download the handout for Homework 6 (Options → Download handout) and extract the contents (i.e., by executing `tar xvf hw6handout.tar` at the command line). You will find three folders in the extracted contents. The `data` folder contains the data file for this problem. The `python` folder contains a `kmeans.py` file which contains empty function templates for each of the functions you are asked to implement. Similarly, the `octave` folder contains separate `.m` files for each of the functions that you are asked to implement.

To finish each programming part of this problem, open the `kmeans.py` or the function-specific `.m` template files and complete the function(s) defined inside. When you are ready to submit your solutions, you will create a new tar archive of the files you are submitting. Please create the tar archive exactly as detailed below.

If you are submitting Python code:

```
tar cvf hw6_handin.tar kmeans.py
```

If you are submitting Octave code:

```
tar cvf hw6_handin.tar update_assignments.m update_centers.m lloyd_iteration.m kmeans_obj.m
```

If you are working in Octave and are missing **any** of the function-specific `.m` files in your tar archive, you will receive zero points.

We have provided the test data for this assignment as `csv` files in the `data` folder in your handout. You can load the data using the `numpy.genfromtext` function in Python or the `csvread` and `csv2cell` (`io` package) functions in Octave. We have provided a `hw6_script` file for each language to help get you started. You should build on the `hw6_script` to test out the functions we are asking you to implement and turn in, but you will not submit the script itself. After loading the data, you should have two variables: `X` and `mnist_X`.

- `X` is a $n \times d$ dimensional matrix describing the n data points used for training your k-means clustering method.
- `mnist_X` is from MNIST dataset, it is a $n \times d$ dimensional matrix describing the n figures used in question 9 and 10. The entry `XTrain(i,j)` is the intensity of the pixel j in i^{th} figure.

The grading on autolab may take minutes, especially for the Lloyd's method. So we suggest you thoroughly test your implementation locally before you submit codes, which will also save your time. Also, **you won't see any print outs or errors on Autolab except for what we provide**. This is done with the initial hope that students should find the bug locally instead of on Autolab.

Implementing Lloyd's Method

In the remainder of this problem you will implement and experiment with Lloyd's method and the k-means++ algorithm on the two dimensional dataset shown in Figure 1. First there are several notes for the implementation questions.

Notes:

- *General:*
  - The cluster label in the write up starts from 1 by mathematical conventions, while it can be different in different programming languages.
  - Autolab should take around 1 min to grade your submission. If it is taking longer, there is an error, most likely with convergence.
- *Octave:*
  - Use only native Octave functions. Higher-level functions, such as `pdist2`, are not supported on Autolab.
  - Cluster label starts from 1.

- *Python:*
  - Numpy 2-D array is used to represent matrix X, C and `mnist_X`, and numpy 1-D array is used to represent vector a.
  - Attribute `axis` in `numpy.linalg.norm` is not supported on Autolab.
  - Python object, especially numpy array is passed by reference in functions[1], be very careful when you implement the iterative process.
  - Cluster label starts from 0.

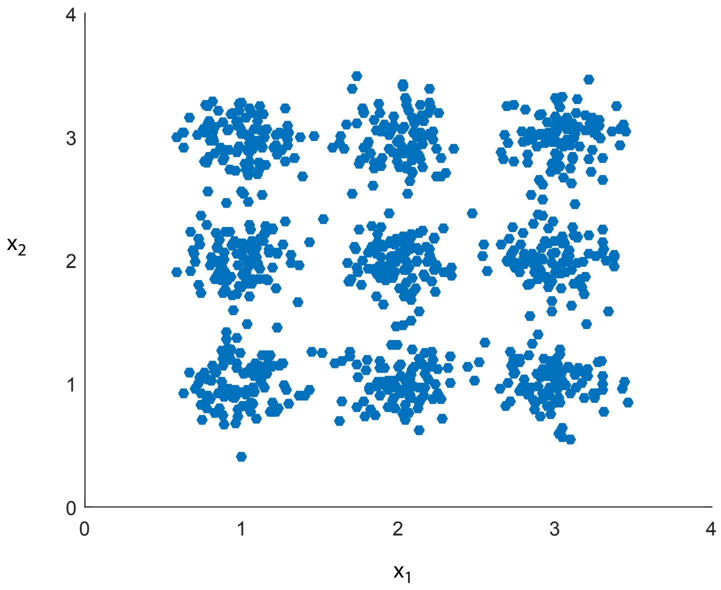

Figure 1: The 2D dataset X used for problem 1.

1. [7 pts] Complete the function `[a] = update_assignments(X, C)`. The input `X` is the $n \times d$ data matrix, `C` is the $k \times d$ matrix of current centers. Your function should output the $n \times 1$ vector of cluster assignments `a`. That is, `C(i,:)` is the center for cluster i, and `a(j)` denotes the cluster label $[1, \ldots, k]$ (label in octave $[1, \ldots, k]$, label in Python $[0, \ldots, k-1]$) for the j^{th} data point. so that each point is assigned to the cluster with the center that has the smallest Euclidean distance with the point.

2. [7 pts] Complete the function `[C] = update_centers(X, C, a)`. The input arguments are as in part 1. Your function should output a $k \times d$ matrix `C` whose i^{th} row is the optimal cluster center for those points in cluster i.

3. [9 pts] Complete the function `[C, a] = lloyd_iteration(X, C)`. This function takes a data matrix `X`, initial centers `C` and runs Lloyd's method until convergence. Specifically, alternate between updating the assignments and updating the centers until the the assignments stop changing. Your function should output the final $k \times d$ matrix `C` of centers and final the $n \times 1$ vector `a` of assignments.

4. [7 pts] Complete the function `[obj] = kmeans_obj(X, C, a)`. This function takes the $n \times d$ data matrix `X`, a $k \times d$ matrix `C` of centers, and a $n \times 1$ vector `a` of cluster assignments. Your function should output the value of the k-means objective of the provided clustering.

[1] http://stackoverflow.com/questions/17886890/python-numpy-and-memory-efficiency-pass-by-reference-vs-value

Experiments

We provide you with a function `[best_C, best_a, best_obj] = kmeans_cluster(X, k, init, num_restarts)` that takes an $n \times d$ data matrix `X`, the number `k` of clusters, a string `init` which must be `'random'` or `'kmeans++'` or `'fixed'`, and a number of restarts `num_restarts`. This function runs Lloyd's method `num_restarts` times and outputs the best clustering it finds. You may use this function when answering the following questions. When `init` is set to `'random'` then the centers are initialized uniformly at random, when `init` is set to `'kmeans++'` the centers are initialized using the D^2 weighting of k-means++, and when `init` is set to `'fixed'` the centers are initialized using the first k data points. Setting `init` to be `'fixed'` is only used for grading purpose.

Note the function is not available until you implement functions `lloyd_iteration` and `kmeans_obj` correctly. You are not supposed to randomize the dataset `X` and `mnist_X` for grading purpose.

The effect of k on Lloyd's method

In questions 5 and 6 of this problem you will investigate the effect of the number of clusters k on Lloyd's method and a simple heuristic for choosing a good value for k. You can load the data for this problem by running the given script in Octave or Python. This will introduce a 900×2 matrix `X`, where each row corresponds to a single point in the dataset shown in Figure 1.

5. [3 pts] Compute the k-means objective value obtained by running `kmeans_cluster(X, k,'fixed', 10)` for each value of `k` in $\{1, \ldots, 10\}$. What is the minimum objective value on the dataset `X` among all the experimented k's. Please submit your answer to question 24 on QnA.
6. [3 pts] Plot the objective values computed by running `kmeans_cluster(X, k,'random', 10)` against the number of cluster k in $\{1, \ldots, 20\}$. One heuristic for choosing the value of k is to pick the value at the "elbow" or bend of the plot, since increasing k beyond this point results in relatively little gain. Based on your plot and the data scatter plot in Figure 1, what value of k that makes the most sense to you for practical use? Please submit your answer to question 25 on QnA. You do not need to submit the plot.

The effect of initialization on Lloyd's method

In questions 7 and 8 you will investigate the effect of the center initialization on Lloyd's method. In particular, you will compare random initialization with the k-means++ initialization. Note that the provided `kmeans_cluster` function has both kinds of initialization already implemented. In this problem, you just need to run the code and answer questions.

You can load the data for this problem by running the given script in Octave or Python. This will introduce a 900×2 matrix `X`, where each row corresponds to a single point in the dataset shown in Figure 1.

Note since the initialization is random, your answers will vary each time you perform this experiment, while our grading script will consider the randomness.

7. [4 pts] For the value of `init` being `'random'`, report the mean of the k-means objective value returned by `kmeans_cluster(X, 9, init, 1)` over 1000 runs. Note that `num_restarts` $= 1$. Please submit your answer to question 26 on QnA.
8. [4 pts] For the value of `init` being `'kmeans++'`, report the mean of the k-means objective value returned by `kmeans_cluster(X, 9, init, 1)` over 1000 runs. Note that `num_restarts` $= 1$. Please submit your answer to question 27 on QnA.

Combining PCA with k-means Clustering

In question 9 and 10 you will work with the real world data. You are given part of the MNIST dataset[2], the dataset contains handwritten digits from 0 to 9. Each row of the data matrix represent a 32×32 figure, the values of features is the intensity of pixels.

You can load the data for this problem by running the given script in Octave or Python. This will introduce a 2876×784 matrix `mnist_X`, where each row corresponds to a figure in the MNIST dataset.

Since the dimension of each figure is relatively high, you will first use PCA to project the data to low dimensional space, then conduct the k-means clustering method on the low dimensional data.

The experiments can be done by either Octave or Python.

- *Octave:* For Octave users, we provide you with a function `X_reduced = pca_fit_transform(X, n_components)`. The function takes the original dataset $X \in R^{n \times d}$ and number of components `n_components` as inputs, outputs the reduced dataset `X_reduced` $\in R^{n \times \texttt{n\_components}}$. You are required to read and understand the implementation of function `pca_fit_transform`, there will be question related to the implementation.
- *Python:* For Python users, library scikit-learn provides you with the `fit_transform` function in `sklearn.decomposition.PCA`. The function is able to reduce the dimension of the original dataset and output the reduced dataset `X_reduced` $\in R^{n \times \texttt{n\_components}}$, where `n_components` is the number of components in PCA. You are required to read and understand the document of PCA in scikit-learn[3], there will be question related to the document.

9. [3 pts] In this question, you will first compute the reduced dataset `mnist_X_reduced` from the original dataset `mnist_X`, where the number of components is 5. Then you will run the k-means clustering method on the reduced dataset. Report the objective value of the k-means returned by `kmeans_cluster(mnist_X_reduced, 3, 'fixed', 1)`, where `mnist_X_reduced` is a $n \times$ `n_components` matrix produced by PCA. You can use either decimal representation (e.g. 11111111111.111) or scentific representation (e.g. $1111111111.1111e+001$). Please submit your answer to question 28 on QnA.

10. [3 pts] After conducting k-means clustering on the reduced dataset, you get the center of each cluster in `C` for the reduced dataset, how do you find the center of each cluster for the original dataset? This question should be answered by short sentences. You can either describe the process by plain text or paste the codes you wrote. Please submit your answer to question 29 on QnA.

[2] http://yann.lecun.com/exdb/mnist/

[3] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html