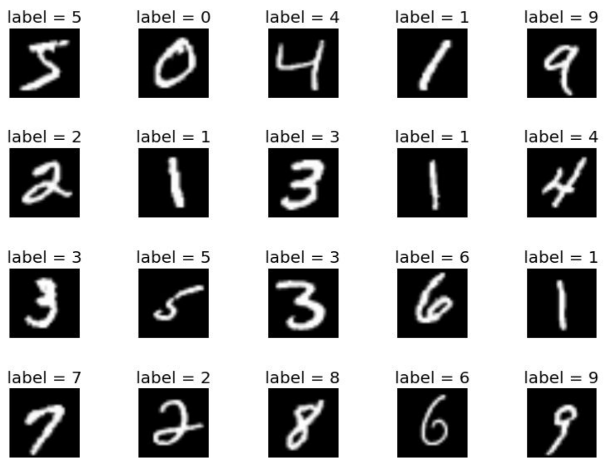
Project 1: Clustering

Release date: 10th of Mar, 2019
Due date: 24th of Mar, 2019

Description

In this project, you will implement the clustering algorithm k-means on handwritten digits dataset called MNIST digit dataset. Sample images is shown below.
The MNIST data consists of 20,000 examples of 28 × 28 images of digits (i.e., numbers from 0-9).



Dataset:

- **digits-embedding.csv**: contains a precomputed 2 features for each image (first column: image id, second column: class label, third and fourth columns: image features).

Design Tasks

You are required to implement k-means clustering algorithm on the MNIST dataset. You are free to design your project the way you see fit in terms of functions, parameters…etc. However, your project should meet certain design constraints. You may ***NOT*** use ready implementation for kmeans.

You must implement a Class called **MyKmeans**. ***All*** your needed functions and attributes should be included in this class, such that a user using your class should be able to use it as follows:

```
km = MyKmeans() #creating an object

parsedData = km.readData('digits-embedding.csv') #reading, parsing the data.
clusters = km.cluster(parsedData, iterCount=50, k=5, centroids=[]) #perform clustering with initially random centroids
SC = km.calculateSC(clusters) #calculating the SC coeff. for the clustering performed

parsedData = km.readData('digits- embedding.csv')
clusters = km.cluster(parsedData, iterCount=10, k=3, centroids=[10,20,30]) #perform clustering using the provided initial centroids
SC = km.calculateSC(clusters) #calculating the SC coeff. for the clustering performed
```

As shown in the previous example, your class should implement those functions:

1. ***readData (filename)***

   Which takes a file name (digits-embedding.csv) as an input then parse it.

   Inputs:
   - filename (string)

   Returns:
   - parsedData. Parsing the dataset in any format you see fit.

2. ***cluster (parsedData, iterCount, k, centroids)***

   Inputs:
   - **parsedData:** The data parsed from the readData function.
   - **iterCount** (int): the number of iterations for the algorithm.
   - **k** (int): the number of clusters.
   - **centroids** (list<int>) (optional): if passed, it's an array of image ids that should be used as the initial cluster heads (centroids).
     If this parameter is not passed, then centroids are selected randomly. (*set the random seed of the program to be 1111*)

   Returns:
   - The list of clusters (list<list<int>>). Each cluster is a list containing all the image ids belonging to it. Clusters order in the list should match the order of their corresponding input centroids. E.g. if the passed centroids=[5,10,15], then the clusters to be returned would be a list whose 1st element is the list of image ids corresponding to the first centroid (5), the second element corresponds to the second centroid (10) … etc.

3. ***calculateSC (clusters)***

   This function takes the clusters formed and calculate the Silhouette Coefficient (SC) for it.

Clustering Tasks

Consider two versions of the data for each of the tasks below:

(i) Use only the subset of the data (digits-embedding.csv) consisting of the digits 2, 4, 6 and 7.

(ii) Use only the subset of the data (digits-embedding.csv) consisting of the digits 6 and 7.

For each dataset configuration (i, ii), it's required

1. Visualize the images using their 2D features from (**digits-embedding.csv**), coloring the points to show their corresponding class labels. I.e. all images of the digit 1 colored in black, images of digit 2 colored in blue … and so on.

2. Cluster the data with different values of K ∈ [2,4,8,16]. For each K repeat the experiment for 5 different times each with random centroids and calculate the average Silhouette Coefficient (SC) of each K after the 5 trials.

3. Construct a plot showing the average Silhouette Coefficient (SC) on y-axis as a function of K on the x-axis.

4. Choose which k you think is best number of clusters for this dataset version.

You are required to prepare a small document to report the previous plots, visualization and results.

Turning in Your Work

- Submit your code and report on Blackboard.
  If your email is abc@purdue.edu, then submit abc_project.py and abc_report.pdf

Rubric

Project 1	
Task 1: Reading Data	**5%**
Task 2: Implementation of k-means	**30%**
Task 3: Calculation of Silhouette Coefficient (SC)	**25%**
Task 4: Successfully running experiments	**20%**
Task 5: Choosing the right k	**5%**
Task 6: Report	**15%**
TOTAL	**100%**