Tsinghua-Berkeley Shenzhen Institute
Learning From Data
Fall 2020

Programming Assignment 3

Issued: Sunday 22nd November, 2020 **Due:** Sunday 6th December, 2020

POLICIES

- **Acknowledgments:** We expect you to make an honest effort to solve the problems individually. As we sometimes reuse problem set questions from previous years, covered by papers and web pages, we expect the students **NOT** to copy, refer to, or look at the solutions in preparing their answers (relating to an unauthorized material is considered a violation of the honor principle). Similarly, we expect you not to google directly for answers (though you are free to google for knowledge about the topic). If you do happen to use other material, it must be acknowledged in `README.md`, with a citation on the submitted solution.
- **Required homework submission format:** You should submit the assignment by the invitation link `https://classroom.github.com/a/_2SPOPUG`. This link will create a private GitHub repository from the code template. Upload your modification to the master branch and view the auto-grading results afterwards. The teaching assistant will grade your assignment mainly based on the rightness of your programming implementation. Optionally you can write your ideas in `README.md` or in code comments.
- **Collaborators:** If you collaborated with others, in `README.md`, list their names and GitHub ID and for which question(s).

3.1. (*kmeans*) Recall that kmeans is solving the following optimization problem:

$$\min_{C,\mu} \sum_{j=1}^{k} \sum_{x \in C_j} ||x - \mu_j||^2$$

Exact solution of the above problem is NP-hard. We proceed by an iterative scheme instead. The iterative algorithm first initializing the cluster centroids $\mu_1, \ldots, \mu_k$. We can choose k different points from $x_1, \ldots, x_m$ as the random initialization. Then the algorithm iterates between updating the centroids (E step) and assigning labels (M step).

(a) (2 points) Please use such algorithm to implement kmeans by completing the code in **kmeans.py**. Your implementation should follow Algorithm 1.

Algorithm 1 K-Means Clustering

```
Input: data points $x^{(1)}, \dots, x^{(m)}$ and cluster size $k$
Output: clustering label vector $y$
1: Initialize cluster centroids $\mu_1, \dots, \mu_k \in \mathbb{R}^n$ randomly
2: while not convergent do
3:     for $i = 1, \dots, n$ do
4:         $y_i = \arg\min_j ||x^{(i)} - \mu_j||^2$
5:     for $j = 1, \dots, k$ do
6:         $\mu_j = \frac{\sum_{i=1}^m \mathbf{1}\{y_i=j\} x^{(i)}}{\sum_{i=1}^m \mathbf{1}\{y_i=j\}}$
```

(b) (2 points) kmeans can be regarded as a special Gaussian mixture model with known uniform variance. If the dataset does not follow this assumption, the clustering result may behave poorly and contrary to the expectation. In this question, you are required to apply kmeans clustering to an artificial dataset with two eclipse contour and present the clustering result in the form of figure. Besides, you should write down some analysis in `README.md` to explain why the unexpected clustering result happens. For the experiment code, please see **kmeans-experiment.py** for detail of artificial data generation and result visualization.

3.2. (*spectral clustering with rbf kernel*) In this question, we consider unnormalized spectral clustering, which deals with the unnormalized Laplacian matrix of a graph, $L = D - W$. The weighted matrix W is constructed using rbf-kernel,

$$W_{ij} = \exp(-\gamma ||x^{(i)} - x^{(j)}||^2) \tag{1}$$

while the diagonal matrix D is obtained by summing each row of W. After making dimension reduction from L by choosing its first k eigenvectors V (corresponding to k smallest eigenvalues), we make clustering in the reduced feature space by k-means, which you have already implemented in the previous question.

(a) (4 points) Please implement spectral clustering by completing the code in **spectral_clustering.py**. Your implementation should follow Algorithm 2.

Algorithm 2 Spectral Clustering

```
Input: data points $x^{(1)}, \dots, x^{(n)}$ and cluster size $k$
Output: clustering label vector $y$
1: Build the similarity matrix $W$ by (1)
2: Construct unnormalized Laplacian matrix $L$
3: Compute first $k$ eigenvectors $V = [v_1, \dots, v_k]$ of $L$
4: Define $u_i \in \mathbb{R}^k$ as the $i$-th row of $V$, cluster $u_1, \dots, u_n$ into $k$ clusters using k-means
   and obtain the cluster label $y_1, \dots, y_n$
```

(b) (2 points) The performance of spectral clustering is influenced by the scaling parameter γ. In this question, we use grid search to get optimal γ. We assume the ground truth label is known in advance and use adjusted rank index to evaluate the clustering performance for different γ. adjusted rank index is a similarity metric of two label vectors, which gives the highest score 1 if the two underlining clusters are exactly the same. Please use this method to tune the parameter γ for the given three circle dataset. You should report your optimal γ and plot the clustering result for this γ. Besides, write down your analysis in `README.md` why γ in the range you choose can produce the desirable result. For the experiment code, please see **spectral-experiment.py** for detail of artificial data generation and result visualization.

3.3. (bonus, 2.5 points) (*spectral clustering with normalized Laplacian*) In this question, you are required to implement spectral clustering with normalized Laplacian. Following the convention of `scipy`, given the weight matrix W, the normalized Laplacian is defined mathematically as:

$$\begin{aligned} W' &= W - \operatorname{diag}\{W\} \\ D &= \operatorname{diag}\{d_1, \ldots, d_n\} \text{ where } d_i = \sum_{j=1}^{n} W'_{ij} \\ L &= I - D^{-1}W' \end{aligned} \tag{2}$$

Please extend your existing implementation in the class function `_get_embedding` of **spectral_clustering.py** to support normalized Laplacian. Your implementation should follow Algorithm 3.

Algorithm 3 Normalized Spectral Clustering

```
Input: data points $x^{(1)}, \ldots, x^{(n)}$ and cluster size $k$
Output: clustering label vector $y$
1: Build the similarity matrix $W$ by (1)
2: Construct normalized Laplacian $L$ from (2)
3: Compute first $k$ eigenvectors $V = [v_1, \ldots, v_k]$ of $L$
4: Define $u_i \in \mathbb{R}^k$ as the $i$-th row of $V$, cluster $u_1, \ldots, u_n$ into $k$ clusters using k-means
   and obtain the cluster label $y_1, \ldots, y_n$
```

Notice:

i) Use matrix operations other than loops for efficiency. If the running time of Auto-Grading steps exceeds 5 minutes, you will get point deductions.

ii) For algorithm implementation questions, You can **only** use `numpy`. Using any API of `scipy` and `sklearn` will lead to point deductions. However, for analysis question you are allowed to use any other third-party packages.

iii) For analysis problem, please write down your answers in `README.md`. You can follow `report-template.md` and click **Preview changes** on GitHub webpage to view the rendered contents after modification. In rare cases when you could not get familiar with the specific syntax of markdown, you could upload a `.docx` or `pdf` file to the root directory of your GitHub repository and declare where your answers are written in `README.md`.