University of Illinois Fall 2023

ECE 544: Pattern Recognition
Problem Set 4

Due: Tuesday, October 24, 2023, 11:59 pm

1. **[K-Means]**

   We are given a dataset $\mathcal{D} = \{(x)\}$ of 2d points $x \in \mathbb{R}^2$ which we are interested in partitioning into K clusters, each having a cluster center μ_k ($k \in \{1, \ldots, K\}$) via the k-Means algorithm. This algorithm optimizes the following cost function:

   $$\min_{\mu_k, r} \sum_{x \in \mathcal{D}, k \in \{1,\ldots,K\}} \frac{1}{2} r_{x,k} \|x - \mu_k\|_2^2 \qquad \text{s.t.} \quad \begin{cases} r_{x,k} \in \{0,1\} & \forall x \in \mathcal{D}, k \in \{1, \ldots, K\} \\ \sum_{k \in \{1,\ldots,K\}} r_{x,k} = 1 & \forall x \in \mathcal{D} \end{cases} \tag{1}$$

   (a) What is the domain for μ_k?

   (b) Given fixed cluster centers μ_k $\forall k \in \{1, \ldots, K\}$, what is the optimal $r_{x,k}$ for the program in Eq. (1)? Provide a reason?

   (c) Given fixed $r_{x,k}$ $\forall x \in \mathcal{D}, k \in \{1, \ldots, K\}$, what are the optimal cluster centers μ_k $\forall k \in \{1, \ldots, K\}$ for the program in Eq. (1)? Reason by first computing the derivative w.r.t. μ_k.

   (d) Using Pseudo-code, sketch the algorithm which alternates the aforementioned two steps. Is this algorithm guaranteed to converge? Reason? Is this algorithm guaranteed to find the global optimum? Reason?

   (e) Complete `A7_KMeans.py` by implementing the aforementioned two steps. For the given dataset, after how many updates does the algorithm converge, what cost function value does it converge to and what are the obtained cluster centers?

2. **[Gaussian Mixture Models and K-Means]**

   Consider a Gaussian mixture model with K components ($k \in \{1, \ldots, K\}$), each having mean μ_k, variance σ_k^2, and mixture weight π_k. Further, we are given a dataset $\mathcal{D} = \{x_i\}$, where $x_i \in \mathbb{R}$. We also use z_{ik} to denote the latent variables.

   (a) What is the log-likelihood of the data according to the Gaussian Mixture Model? (use μ_k, σ_k, π_k, K, x_i and $\mathcal{D}$). Don't use any abbreviations.

   (b) Assume $K = 1$, find the maximum likelihood estimate for the parameters $(\mu_1, \sigma_1^2, \pi_1)$.

   (c) What is the probability distribution on the latent variables, *i.e.*, what is the distribution $p(z_{i,1}, z_{i,2}, \ldots, z_{i,K})$ underlying Gaussian mixture models. Also give its name.

   (d) For general K, what is the posterior probability $p(z_{ik} = 1 | x^{(i)})$? To simplify, wherever possible, use $\mathcal{N}(x_i | \mu_k, \sigma_k)$, a Gaussian distribution over $x_i \in \mathbb{R}$ having mean μ_k and variance σ_k^2.

   (e) How are kMeans and Gaussian Mixture Model related? (There are three conditions)

   (f) Show that the objective for kMeans and Gaussian Mixture Model are equivalent under the conditions you provided in the previous part (e).

3. **[K-Means, GMM, and EM]**

In this problem we will use KMeans to cluster data generated from a Gaussian Mixture models (GMM). Start a Python script and load the provided dataset 'dataset_problem3.npy'.

Note: You will create your own Python code from scratch in this problem. Please attach the screenshot of your code along with your answers.

The GMM is five dimensional and each of its components has diagonal ($\sigma^2 I$) covariance matrix. The number of components and centroids are not given.

(a) Take two (out of five) coordinates and plot each point to visualize the dataset in the lower (2D) dimension. Repeat this operation for at least four pairs of coordinates. (Use *subplot* from *Matplotlib* to produce plots in preferably one figure). Do all of these plots appears to contain the same number of clusters?

(b) Use the K-means algorithm with Euclidean distance to cluster the given dataset. Here, You may directly use *Kmeans* from *scikit-learn* to fit the dataset, hence need not implement it from scratch. Use random initialization of centers to start Kmeans. Plot the within-cluster scatter versus the iterations for 10 iterations. A way of achieving this in *scikit-learn* is as follows.

i. Use the *inertia* attribute of *Kmeans* object to access the within-cluster scatter.

ii. Initialize the centroids by setting the *init* argument of *Kmeans* equal to 'random' and run *Kmeans* using the *fit* method for 1 iteration by setting the *max_iter* argument of *Kmeans* equal to 1.

iii. Iteratively (in a loop) use *fit* method to train with *max_iter* argument of *Kmeans* set equal to 1. Supply the centroids obtained from previous iteration by setting the *init* argument of *Kmeans* using the centroid from previous iteration.

(c) Using the EM algorithm with random initialization of parameters, fit the GMM on above dataset. You can use *GaussianMixture* from *scikit-learn* python to fit the GMM and need not implement EM algorithm from scratch. You need to make a reasonable guess about the number of clusters. How many iterations does the algorithm take to converge ?(Use the *n_iter* attribute of *GaussianMixture*). Report the mean values of each of the cluster. Next, initialize the EM with centers obtained using Kmeans. Is there any improvement in number of iterations for convergence? Justify.

4. **[Spectral Clustering]**

In this problem we apply spectral clustering to a problem with non-convex clusters. Start a Python script and load the provided dataset 'dataset_problem4.npy'. The dataset contains 600 training examples in $\mathbb{R}^2$. Obtain a scatter plot to visualize it. You can use *SpectralClustering* from *scikit-learn* to train and fit the data.

Note: You will create your own Python code from scratch in this problem. Please attach the screenshot of your code along with your answers.

(a) Using the K-Means algorithm, cluster the given dataset. Plot the resulting clusters. Comment on your results.

(b) One needs to obtain an affinity (similarity) matrix for the data to fit it using spectral clustering. Using the radial-basis function kernel and tuning the parameters, obtain an affinity matrix and use it to perform spectral clustering. (The Python class *SpectralClustering* contains arguments to do it easily). You can use *n_init* argument of *SpectralClustering* to improve the result.

(c) Set *affinity = 'nearest neighbors'*. This obtains a weighted k-nearest neighbor graph for the dataset and uses its connectivity matrix as the affinity matrix. Using this *affinity*, vary the number of neighbors k and see if the clustering algorithm finds the clusters correctly as one increases k. Compare the clustering performance with part (b).