STAT 37710: Homework 1

Caleb Derrickson

March 29, 2024

Collaborators: The TA's of the class, as well as Kevin Hefner, and Alexander Cram.

Contents

Problem 1

As we saw in class, k-means clustering minimizes the average square distance distortion

$$J_{avg^2} = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2, \tag{1}$$

where $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$ and C_j is the set of points belonging to cluster j. Another distortion function that we mentioned is the intra-cluster sum of squared distances,

$$J_{IC} = \sum_{j=1}^{k} \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}')^2.$$

Problem 1, part a

Given that in k-means, $\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}$, show that $J_{IC} = 2J_{avg^2}$.

Solution:

Fix $j \in \{1, 2, ..., k\}$, that is, let us focus on one cluster. The two distortion functions for cluster j would then be

$$J_{avg^2} = \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2, \quad J_{IC} = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}')^2.$$

Starting with the intra-cluster distortion, we will show the equality for C_j.

$$J_{IC} = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} d(\mathbf{x}, \mathbf{x}')^2 \qquad \text{(Given.)}$$

$$= \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} (\mathbf{x} - \mathbf{x}')^\intercal (\mathbf{x} - \mathbf{x}') \qquad \text{(Euclidean distance.)}$$

$$= \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} [\mathbf{x}^\intercal \mathbf{x} - 2\mathbf{x}^\intercal \mathbf{x}' + (\mathbf{x}')^\intercal (\mathbf{x}')] \qquad \text{(Expanding.)}$$

$$= \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x} - \frac{2}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' + \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} (\mathbf{x}')^\intercal (\mathbf{x}') \qquad \text{(Expanding.)}$$

$$= \sum_{\mathbf{x} \in C_j} \mathbf{x}^\intercal \mathbf{x} - \frac{2}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' + \sum_{\mathbf{x}' \in C_j} (\mathbf{x}')^\intercal (\mathbf{x}') \qquad \text{(Independent summations.)}$$

$$= \sum_{\mathbf{x} \in C_j} \mathbf{x}^\intercal \mathbf{x} - \frac{2}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' + \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x} \qquad \text{(Dummy index.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \mathbf{x}^\intercal \mathbf{x} - \frac{2}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' \qquad \text{(Simplifying.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \mathbf{x}^\intercal \mathbf{x} - \frac{4}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' + \frac{2}{|C_j|} \sum_{\mathbf{x} \in C_j} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' \qquad \text{(Adding a zero.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \left[\mathbf{x}^\intercal \mathbf{x} - \frac{2}{|C_j|} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' + \frac{1}{|C_j|} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' \right] \qquad \text{(Grouping Summations.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \left[\mathbf{x}^\intercal \mathbf{x} - \frac{2}{|C_j|} \sum_{\mathbf{x}' \in C_j} \mathbf{x}^\intercal \mathbf{x}' + \frac{1}{|C_j|^2} \sum_{\mathbf{x}' \in C_j} \sum_{\mathbf{x}'' \in C_j} (\mathbf{x}')^\intercal \mathbf{x}'' \right] \qquad \text{(Multiplying by 1.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \left[\mathbf{x}^\intercal \mathbf{x} - 2\mathbf{x}^\intercal \left(\sum_{\mathbf{x}' \in C_j} \frac{1}{|C_j|} \mathbf{x}' \right) + \frac{1}{|C_j|^2} \left(\sum_{\mathbf{x}' \in C_j} \mathbf{x}' \right)^\intercal \left(\sum_{\mathbf{x}'' \in C_j} \mathbf{x}'' \right) \right] \qquad \text{(Linearity of inner product.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \left[\mathbf{x}^\intercal \mathbf{x} - 2\mathbf{x}^\intercal \mathbf{m}_j + \mathbf{m}_j^\intercal \mathbf{m}_j \right] \qquad \text{(Given form of } \mathbf{m}_j\text{.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \left[(\mathbf{x} - \mathbf{m}_j)^\intercal (\mathbf{x} - \mathbf{m}_j) \right] \qquad \text{(Factoring.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} \|\mathbf{x} - \mathbf{m}_j\|^2 \qquad \text{(Euclidean Distance.)}$$

$$= 2 \sum_{\mathbf{x} \in C_j} d(\mathbf{x}, \mathbf{m}_j)^2 \qquad \text{(Given distance.)}$$

$$\implies J_{IC} = 2J_{avg^2} \qquad \text{(Given.)}$$

Note that the above proof has only in mind one cluster. It is easy to see however, that this behavior should hold for all clusters since clusters should not overlap (i.e., no point should be assigned to two clusters). Therefore, the claim has been shown.

Problem 1, part b

Let $\gamma_i \in \{1, 2, ..., k\}$ be the cluster that the i-th data-point is assigned to, and assume that there are n points in total, $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$. Then (1) can be written as

$$J_{avg^2}(\gamma_1, ..., \gamma_n, \mathbf{m}_1, ..., \mathbf{m}_k) = \sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{m}_{\gamma_i})^2. \tag{2}$$

Recall the k-means clustering alternates the following two steps:

1. Update the cluster assignments:

$$\gamma_i \leftarrow \underset{j \in \{1,2,...,k\}}{\operatorname{argmin}} d(\mathbf{x}_i, \mathbf{m}_j), \quad i = 1, 2, ..., n.$$

2. Update the centroids:

$$\mathbf{m}_j \leftarrow \frac{1}{|C_j|} \sum_{i:\gamma_i = j} \mathbf{x}_i, \quad j = 1, 2, ..., k.$$

Show that the first of these steps minimizes (2) as a function of $\gamma_1, ..., \gamma_n$, while holding $\mathbf{m}_1, ..., \mathbf{m}_k$ constant, while the second step minimizes it as a function of $\mathbf{m}_1, ..., \mathbf{m}_k$, while holding $\gamma_1, ..., \gamma_n$ constant. The notation "$i : \gamma_i = j$" should be read as "all i for which $\gamma_i = j$".

Solution:

For the first step, we are given that the updated γ_i is chosen such that each point is matched with the closest centroid, for all i. Since both the problem of $d(\cdot)$ and $d(\cdot)^2$ are minimized for the same values, we can say that γ_i also minimizes the *squared* distances between all points and the centroids. Thus, for the updated γ, the value of the summation of the squared distances is minimized. Then,

$$\gamma = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{m}_{\gamma_j})^2 = \underset{\gamma}{\operatorname{argmin}}\, J_{avg^2}(\gamma, \vec{\mathbf{m}}).$$

Thus, γ minimizes (2), while holding the centroids constant. We will now investigate the second update schema. This minimization will be proven in a comparatively methodical manner, taking the gradient of J^2_{avg} and setting it to zero. The gradient with respect to $\mathbf{m}$ is then

$$\nabla_{\mathbf{m}} \sum_{i=1}^{n} d(\mathbf{x}_i, \mathbf{m}_{\gamma_j})^2 = -2 \sum_{\mathbf{x}_i \in C_j} (\mathbf{x}_i - \mathbf{m}_{\gamma_j})$$

Setting this equal to zero gives

$$\sum_{\mathbf{x} \in C_j} \mathbf{x} = \sum_{\mathbf{x} \in C_j} \mathbf{m}_{\gamma_j} = |C_j| \mathbf{m}_{\gamma_j} \iff \mathbf{m}_{\gamma_j} = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} \mathbf{x}.$$

Which is what we wanted to find.

Problem 1, part c

Prove that as k-means progresses, the distortion decreases monotonically iteration-by-iteration.

Solution:

It is easy to see that taking the minimum in each parameter separately will allow the function to iteratively proceed to the minimum. Since each subpart of the k-means algorithm minimizes their respective piece, we see that their joined influence produces a lower value than the previous iteration. Therefore, the change in each iteration of k-means will provide us a distortion value bounded above by the previous iteration value, implying the algorithm decreases monotonically.

Problem 1, part d

Give an upper bound on the maximum number of iterations required for full convergence of the algorithm, i.e., the point where neither the centroids, nor the cluster assignments change anymore (note: we do not expect you to prove the bound in Inaba et al., something much looser and simpler will do.

Solution:

Note that the elements of γ are sampled entirely from the values $\{1, 2, ..., k\}$. If no progress have been made in a new iteration, we surely cannot expect the next iteration to improve as well. This will repeat ad infinitum. Thus, our stopping criterion to ensure no excessive looping is when we observe no (meaningful) change in our chosen distortion function. Initially, γ will have a random value, in which each entry is chosen at random. By the previous part, since the algorithm should monotonically decrease our distortion function, no value of γ should repeat. In the worst case, the assignment of γ will never repeat, only stopping once we have exhausted possible unique values of γ. Thus, we will see in the worst case our algorithm will iterate k^n times, as we sample values up to k, n times. Thus, the iteration is bounded by k^n.

Problem 2

Implement the k-means algorithm in a language of your choice (Python is recommended), initializing the cluster centers randomly, as explained in the slides. The algorithm should terminate when the cluster assignments (and hence, the centroids) don't change anymore.

Problem 2, part a

The toy dataset **toydata.txt** contains 500 points in $\mathbb{R}^2$ coming from 3 well separated clusters. Test your code on this data and plot the final result as a 2D plot in which each point's color or symbol indicates its cluster assignment. Note that because of the random initialization, different runs may produce different results, and in some cases the algorithm might not correctly identify the three clusters. Plot the value of the distortion function as a function of iteration number for 20 separate runs of the algorithm on the same plot. Comment on the plot.

Solution:

I have implemented the k-means algorithm in C++ (just for the experience), and plotted in Python. I will provide the two given plots below. The first plot, Figure 1, is a scenario where the algorithm reached a global minima: all clusters are being utilized. I make this distinction here, since in Figure 2, we see two different convergent values - one just below 700 and another below 2000. It seems as though the chance where we utilize only two clusters happens in the case where one of the clusters, which have their initial positions uniformly distributed, is simply to far away to catch any points. This means that the initial distortion value will be noticeably larger, and catch the higher minima. Note that I have plotted the distortion value on a log scale to separate the plot lines for ease of viewing.

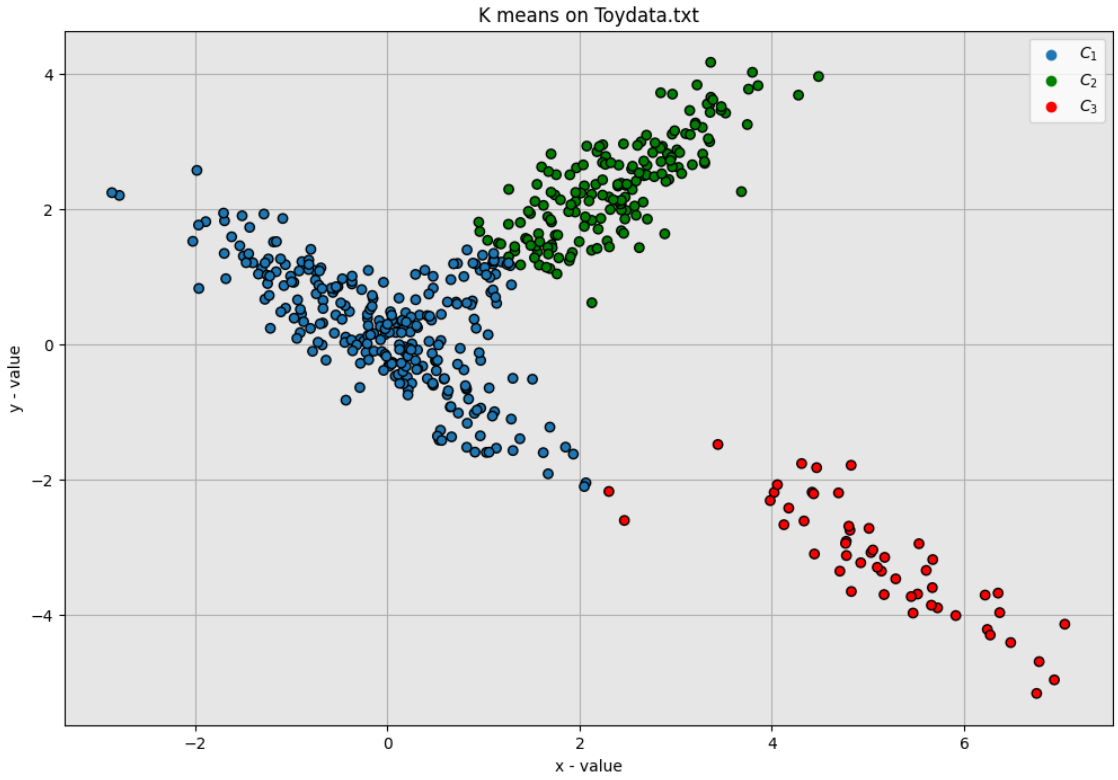


Figure 1: K-means on Toy Data.

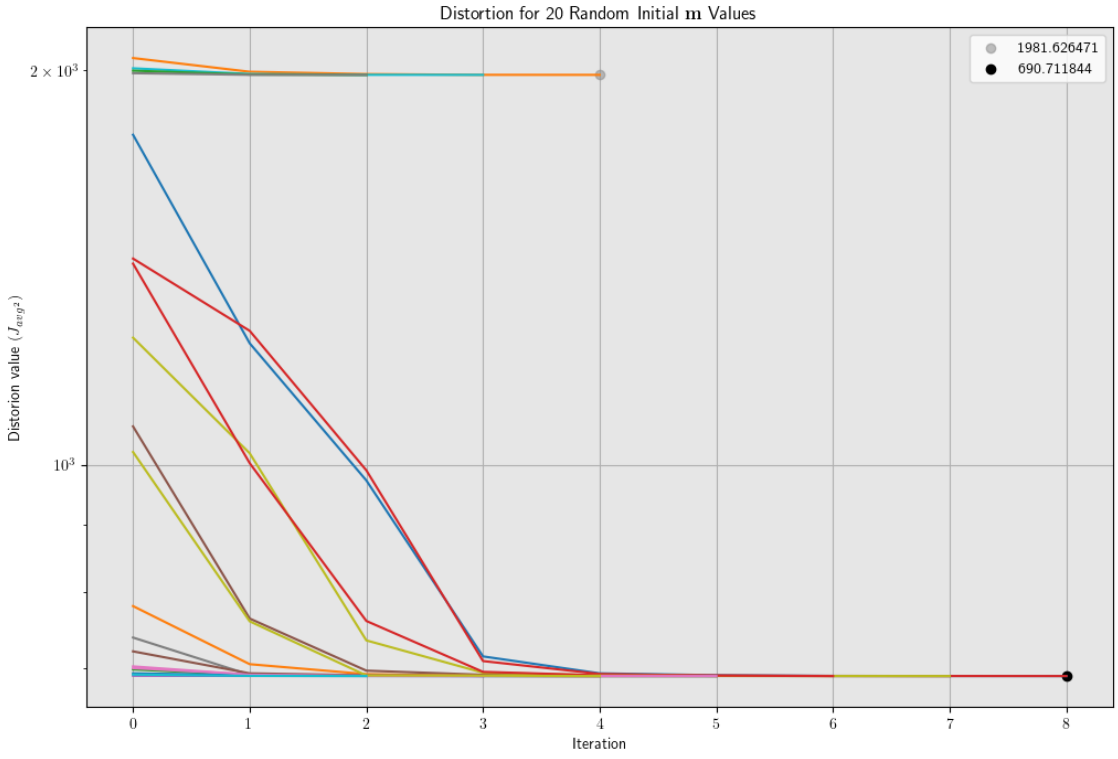


Figure 2: 20 Different values and their converging minima.

Problem 2, part b

Now implement the k–means++ algorithm discussed in class and repeat part (a) using its result as initialization (except for the 2D plot). Comment on the convergence behavior of k–means++ vs. the original algorithm.

Solution:

I have (hopefully) implemented the k-means++ algorithm correctly, and present the resulting distortion plot in Figure 3, we can see that the convergence for k-means++ is less (6 compared to 8), with one outlier condition, which ran for noticeably longer. All points converged to the global minimum, and the iterations needed to get to the minimum is 2 to 6 (with the outlier removed).

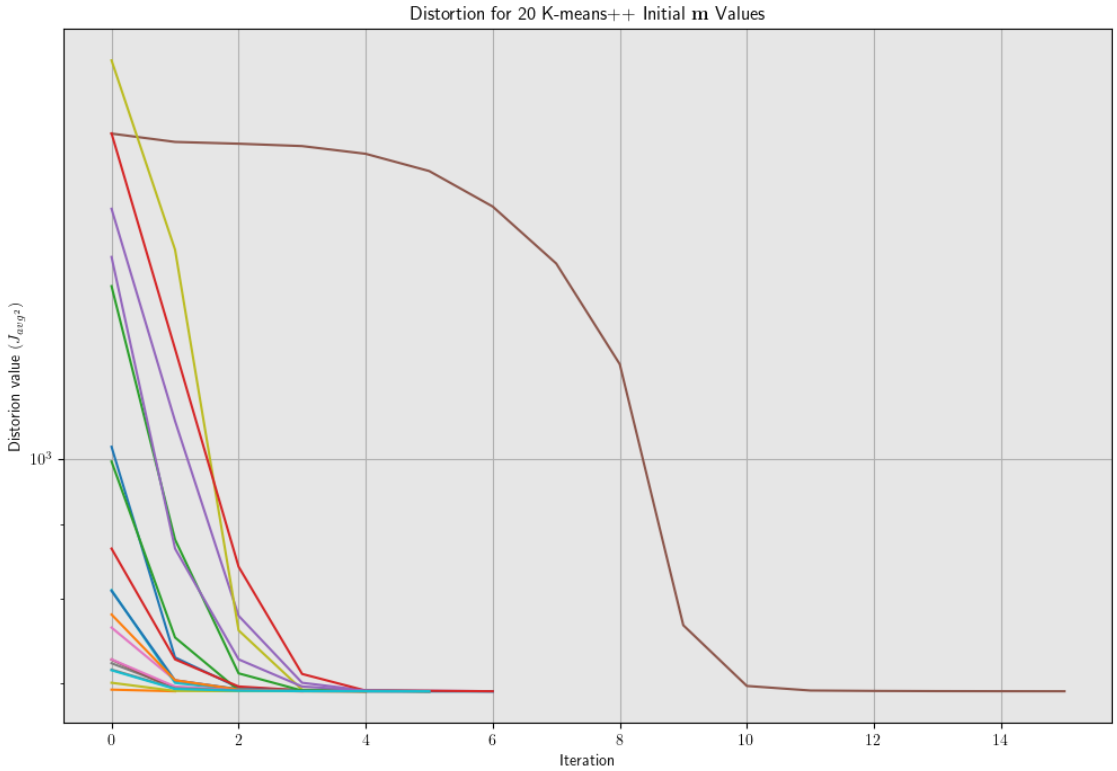


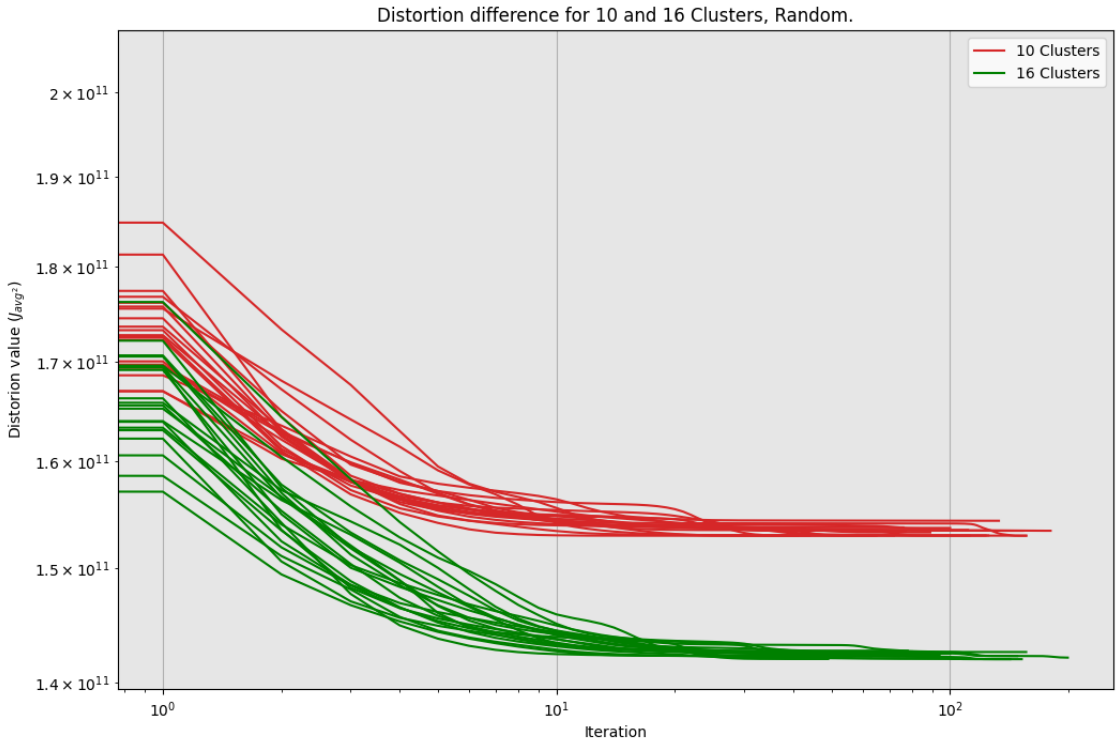
Figure 3: k-means initialized by k-means++.

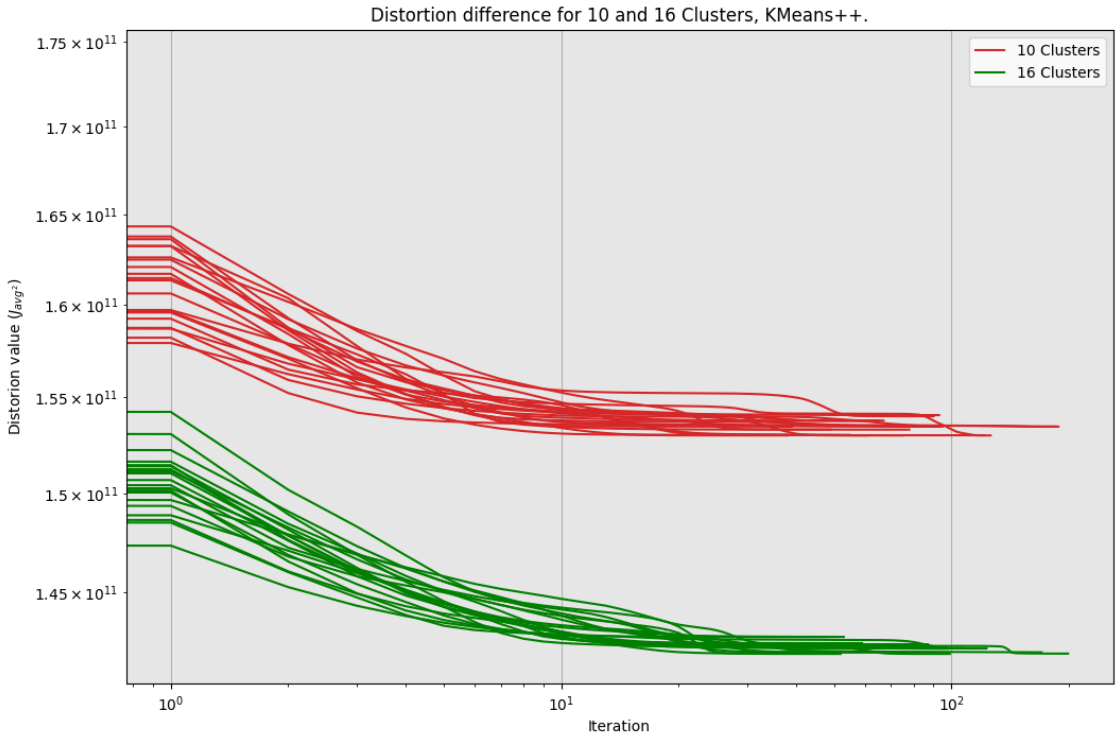
Problem 2, part c

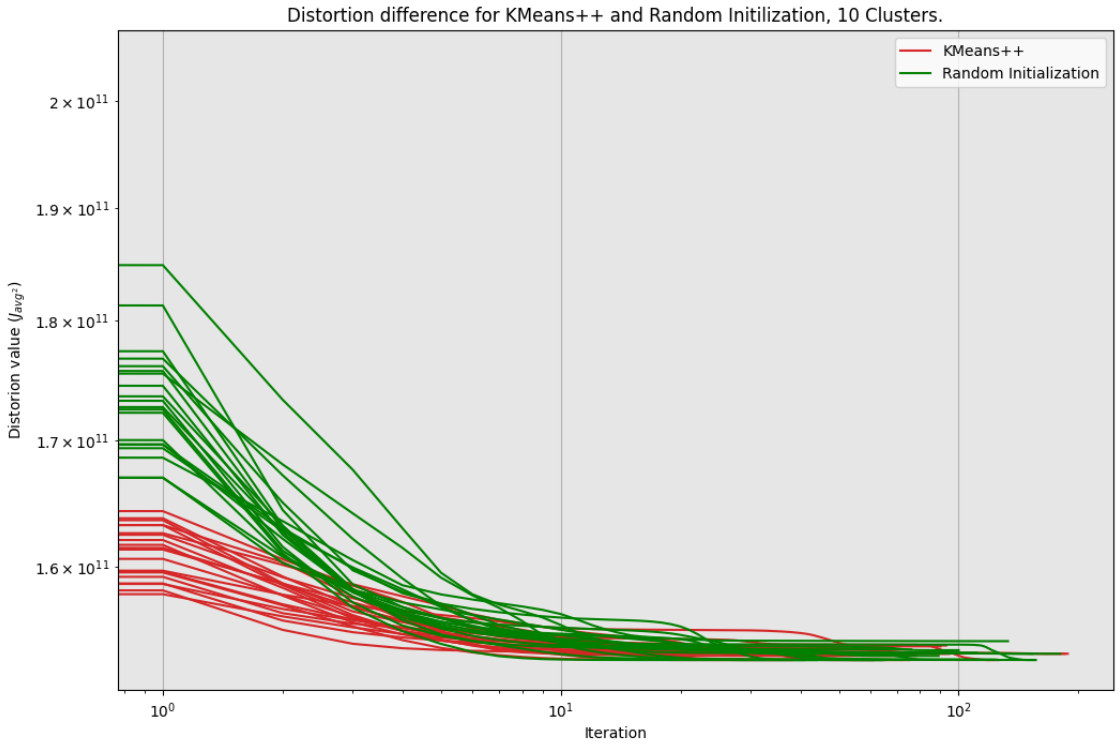
Test both the original k-means and k-means++ algorithms on the MNIST dataset. Each image in MNIST is 28x28 pixels, represented as a 784-dimensional vector. Again, comment on the convergence behavior of k-means++ vs. the original algorithm. Comment on how the results differ when you set the number of clusters to 10 and 16.

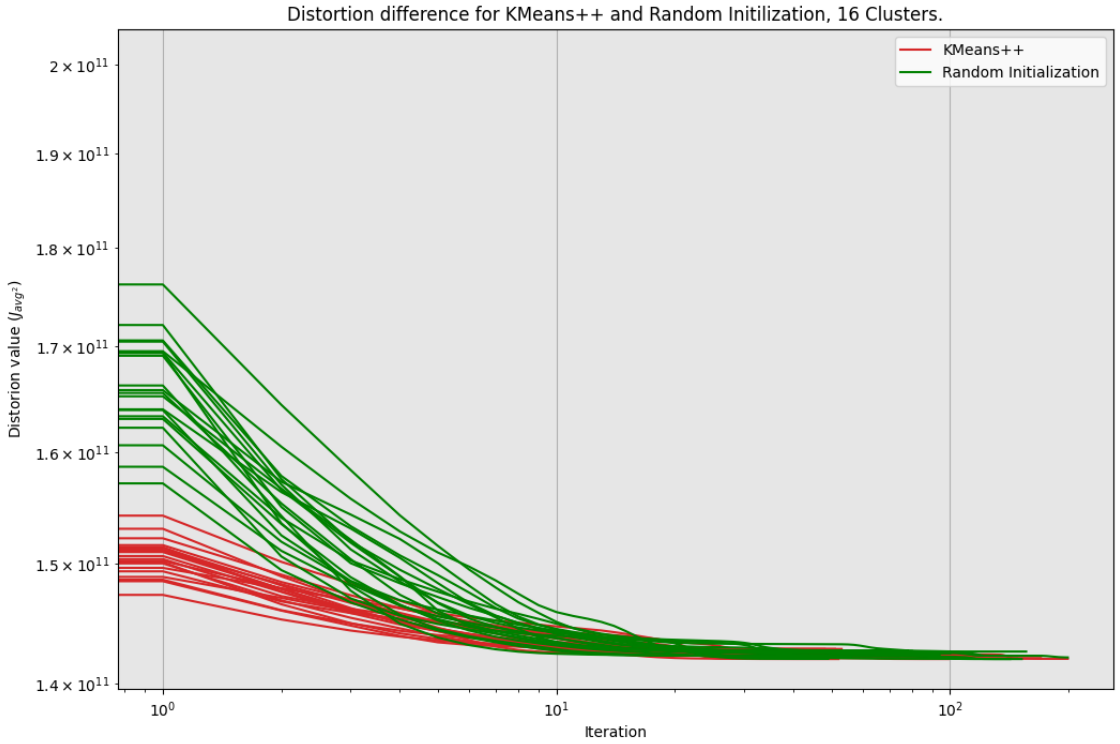
Solution:

The comparison plots are given below, with respect to all notable combinations. We see that k-means++ converges to the same minimum as k-means, but does it in consistently less iterations and lesser initial distortion value. For the difference in k-means++ and random initialization, we see an average iteration length of 86 and 100, respectively.


Distortion difference for 10 and 16 Clusters, Random.
10 Clusters
16 Clusters
Distorion value (J_{avg^2})
2×10^{11}
1.9×10^{11}
1.8×10^{11}
1.7×10^{11}
1.6×10^{11}
1.5×10^{11}
1.4×10^{11}
10^0
10^1
10^2
Iteration


Distortion difference for 10 and 16 Clusters, KMeans++.
10 Clusters
16 Clusters
Distorion value (J_{avg^2})
1.75×10^{11}
1.7×10^{11}
1.65×10^{11}
1.6×10^{11}
1.55×10^{11}
1.5×10^{11}
1.45×10^{11}
10^0
10^1
10^2
Iteration


Distortion difference for KMeans++ and Random Initilization, 10 Clusters.
KMeans++
Random Initialization
2×10^{11}
1.9×10^{11}
1.8×10^{11}
1.7×10^{11}
1.6×10^{11}
Distorion value (J_{avg^2})
10^0
10^1
10^2
Iteration


Distortion difference for KMeans++ and Random Initilization, 16 Clusters.
KMeans++
Random Initialization
2×10^{11}
1.9×10^{11}
1.8×10^{11}
1.7×10^{11}
1.6×10^{11}
1.5×10^{11}
1.4×10^{11}
Distorion value (J_{avg^2})
10^0
10^1
10^2
Iteration

Problem 3

Recall the "Mixture of k Gaussians" model used in clustering

$$p(\mathbf{x}, z) = \pi_z \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z), \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^d$, $z \in \{1, 2, ..., k\}$ is its cluster assignment, and $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$ is the d-dimensional normal density

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) = (2\pi)^{-d/2} |\boldsymbol{\Sigma}_z|^{-1/2} \exp\left(-(\mathbf{x} - \boldsymbol{\mu}_z)^\intercal \boldsymbol{\Sigma}_z^{-1} (\mathbf{x} - \boldsymbol{\mu}_z)/2\right).$$

The parameters of this restricted model are $\theta = (\pi_1, ..., \pi_k, \boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k)$.

Problem 3, part a

Let $\{(\mathbf{x}_1, z_1), (\mathbf{x}_2, z_2), ..., (\mathbf{x}_n, z_n)\}$ be an n-point sample from this model. Write down the corresponding log-likelihood $\ell(\theta)$. As usual, you may ignore constant terms in the log-likelihood.

Solution:

I am relatively new to statistics, so forgive me if my explanations are poorly written. The likelihood function seeks to describe a given set of parameters θ are more likely to be the true set of parameters, assuming we have fixed random variables $\mathbf{x}$. Since the given parameters are discrete and constitute all probable cases for the model, we will sum over them for each random instance encountered. Furthermore, since each random instance is assumed independent, we have the likelihood function to be

$$L(\theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \pi_i \mathcal{N}(\mathbf{x}_j; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j).$$

This results in a log-likelihood function of

$$\ell(\theta) = \sum_{i=1}^{n} \ln \left(\sum_{j=1}^{k} \pi_j \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right)$$

by properties of the natural log. I am refraining from expanding out the $\mathcal{N}$ term, just to preserve ease of readability.

Problem 3, part b

Let $p_{i,j}$ be the probability $p(z_i = j|\mathbf{x}_i)$ of the i-th data point coming from the j-th cluster (given that its position is $\mathbf{x}_i$). Derive an expression for this probability.

Solution:

Seeing as though this is an occurrence of one event happening from a set of k events ($\mathbf{x}_i$ being in the j-th cluster), we would imagine that the probability of this event occurring would result in the proportional behavior of the values of j in the overall parameter θ. In other words, we would set

$$p_{i,j} = \frac{p(\mathbf{x}_i, j)}{\sum_{l=1}^{k} p(\mathbf{x}_i, l)} = \frac{\pi_j \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{l=1}^{k} \pi_l \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}.$$

Problem 3, part c

Derive the expected log-likelihood $\overline{\ell}_{\theta_{old}}(\theta)$ in terms of these $p_{i,j}$ conditional probabilities. Here the expectation is taken over the values of the hidden variables $(z_1, ..., z_n)$, and the subscript θ_{old} signifies that the $p'_{i,j}s$ are computed with respect to the old values of the parameters, whereas $\overline{\ell}$ itself is a function of the new values of the parameters.

Solution:

The expected value of the log-likelihood is given as the general expected value formula,

$$E(\theta, \theta_{old}) = \sum_{l=1}^{k} p_{i,l} \ln\left(p(\mathbf{x}, z)\right),$$

where the term in the natural log is the probability function of our new parameters, and $p_{i,l}$ is our old probabilities. This new probability represents the probability over all input data, $\mathbf{x}$, thus, we can break them apart by data entry, to write

$$E(\theta, \theta_{old}) = \sum_{l=1}^{k} p_{i,l} \sum_{n=1}^{N} \ln\left(\pi_n \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\right).$$

Note that I have pulled out the product from the natural log and turned it into a sum. Since this new sum is independent of the old, we can pull it to the front as follows:

$$E(\theta, \theta_{old}) = \sum_{n=1}^{N} \sum_{l=1}^{k} p_{i,l} \ln\left(\pi_n \mathcal{N}(\mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\right).$$

Next, we notice the old conditional probability term is the expected value of the second term, implying that we can rewrite the expected value as

$$E(\theta, \theta_{old}) = \sum_{n=1}^{N} \sum_{l=1}^{k} p_{l,l} \ln\left(\pi_{\mathcal{N}}(\mathbf{x}_n; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)\right).$$

Plugging in what we have, we see that the expectation value is of the form

$$E(\theta, \theta_{old}) = \sum_{n=1}^{N} \sum_{l=1}^{k} p_{n,l} \left[\ln(\pi_l) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\det(\boldsymbol{\Sigma}_l)|) - \frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_l)^\intercal \boldsymbol{\Sigma}_l^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_l)\right]$$

Problem 3, part d

The expectation maximization algorithm updates the parameters of the mixture by maximizing $\bar{\ell}_{\theta_{old}}(\theta)$. Let us start with the "cluster priors" $\pi_1, ..., \pi_k$. Since $(\pi_1, ..., \pi_k)$ is a discrete distribution, we must have that $\sum_{j=1}^{k} \pi_j = 1$. Using this constraint, derive the update rule for the cluster priors.

Solution:

Taking the constraint we have on the cluster priors, we can substitute in a single term for the whole sum. In particular, when replacing the last cluster prior, we have that

$$\pi_k = 1 - \sum_{j=1}^{k-1} \pi_j$$

This will be used in differentiation. When finding the particular pi_j which maximizes $\bar{\ell}_{\theta_{old}}(\theta)$, we can employ simple calculus: taking the derivative with respect to π_j and setting equal to zero. We see that, when differentiating, we should keep in mind the one instance of $p_{i,j}$ which is not a constant with respect to differentiation, in particular, π_j. Therefore, the sum over clusters can be dropped, since the only one which is not a constant is the j-cluster. We then see that, when differentiating,

$$\frac{dE}{d\pi_j} = \sum_{i=1}^{N} \left[p_{i,j} \frac{1}{\pi_j} - p_{k,j} \frac{1}{1 - \sum_{l=1}^{k-1} \pi_l} \right] = 0.$$

After rearranging, we have that

$$\frac{1}{\pi_j} \sum_i p_{i,j} = \frac{1}{1 - \sum_{j=1}^{k-1} \pi_j} \sum_i p_{k,j}.$$

Finally, after simplification, we have

$$\pi_j = \frac{1}{N} \sum_{i=1}^{N} p_{i,j}.$$

Problem 3, part e

Similarly derive the update rule for the $\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k$ location parameters and $\boldsymbol{\Sigma}_1, ..., \boldsymbol{\Sigma}_k$ covariance matrices.

Solution:

The update rules for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are done similarly. We will show the update scheme for the $\boldsymbol{\mu}$'s first, then the $\boldsymbol{\Sigma}$'s after.

$$\nabla_{\boldsymbol{\mu}_j} = E(\theta, \theta_{old}) \qquad \text{(Given.)}$$

$$= -\frac{1}{2}\sum_{i=1}^{N} p_{i,j} \nabla_{\boldsymbol{\mu}_j} (\mathbf{x}_i - \boldsymbol{\mu}_j) \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j) \qquad \text{(Simplifying.)}$$

$$= -\frac{1}{2}\sum_{i=1}^{N} p_{i,j} (2\boldsymbol{\Sigma}_j^{-1}\mathbf{x}_i + 2\boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j) = 0 \qquad \text{(Simplifying and setting to zero.)}$$

$$\implies \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j \sum_{i=1}^{N} p_{i,j} = \sum_{i=1}^{N} p_{i,j}\boldsymbol{\Sigma}_j^{-1}\mathbf{x}_i \qquad \text{(Rearranging.)}$$

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} p_{i,j}\mathbf{x}_i}{\sum_{i=1}^{N} p_{i,j}} \qquad \text{(Simplifying.)}$$

$$\nabla_{\boldsymbol{\Sigma}_j^{-1}} E(\theta, \theta_{old}) = \sum_{i=1}^{N} p_{i,j} \nabla \left[-\frac{1}{2}\ln\left(|\det \boldsymbol{\Sigma}_j|\right) - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j) \right] \qquad \text{(Applying Gradient.)}$$

$$= \sum_{i=1}^{N} p_{i,j} \nabla \left[\frac{1}{2}\ln\left(|\det \boldsymbol{\Sigma}_j^{-1}|\right) - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j) \right] \qquad \text{(inverse properties.)}$$

$$= \sum_{i=1}^{N} p_{i,j} \left[((\boldsymbol{\Sigma}_j^{-1})^\intercal)^{-1} - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j) \right] \qquad (\bigstar)$$

$$= \sum_{i=1}^{N} p_{i,j} \left[((\boldsymbol{\Sigma}_j^{-1})^\intercal)^{-1} - (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal \right] \qquad (\bigstar\bigstar)$$

$$= \sum_{i=1}^{N} p_{i,j} \left[\boldsymbol{\Sigma}_j - (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal \right] \qquad (\boldsymbol{\Sigma}_j \text{ is symmetric.})$$

$$\implies \boldsymbol{\Sigma}_j \sum_{i=1}^{N} p_{i,j} = \sum_{i=1}^{N} p_{i,j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal \qquad \text{(Setting to zero.)}$$

$$\boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{N} p_{i,j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^\intercal}{\sum_{i=1}^{N} p_{i,j}}$$

The lines labelled ($\bigstar$) and ($\bigstar\bigstar$) come from The Matrix Cookbook, https://www.math.uwaterloo.ca/ hwolkowi/matrixcookbook.pdf, equations 57 and 72, respectively.

Problem 3, part f

Compare these update rules to the k-means update rules derived in Question 1.

Solution:

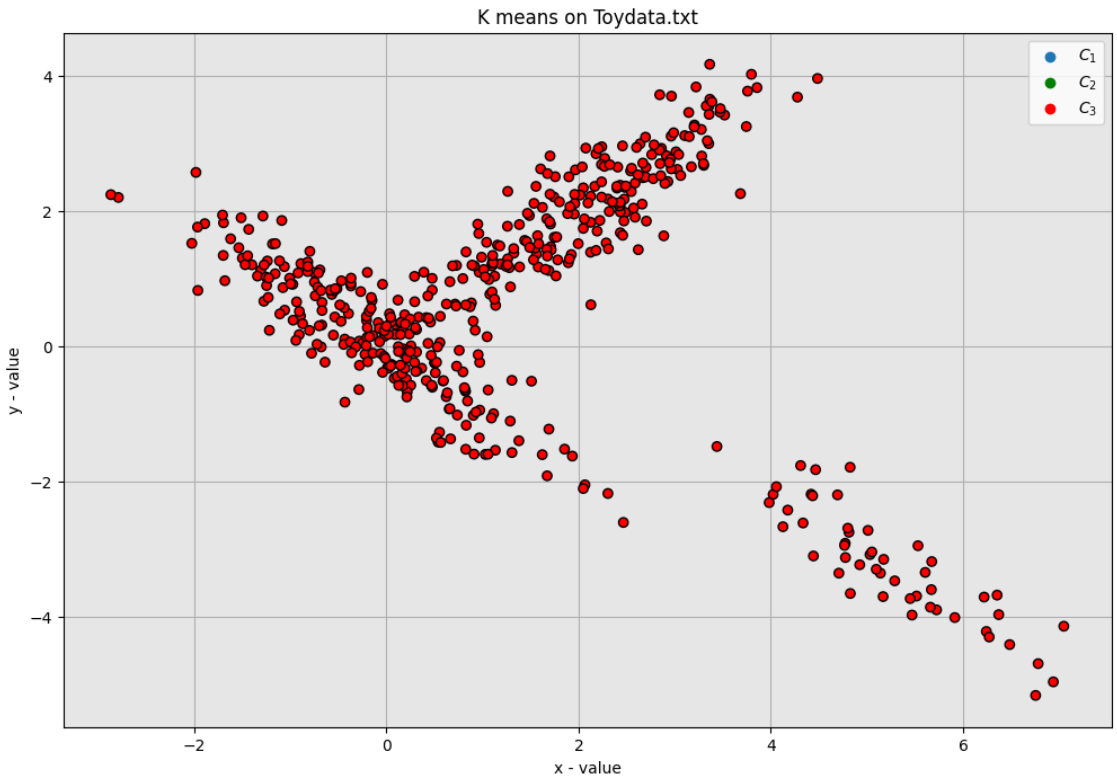
The only update scheme we found in Problem 1 was the centroid update within a given cluster. This formula resembles the cluster priors, with a notable difference that we are summing over all data points, rather than the cluster paradigm seen in k-means.

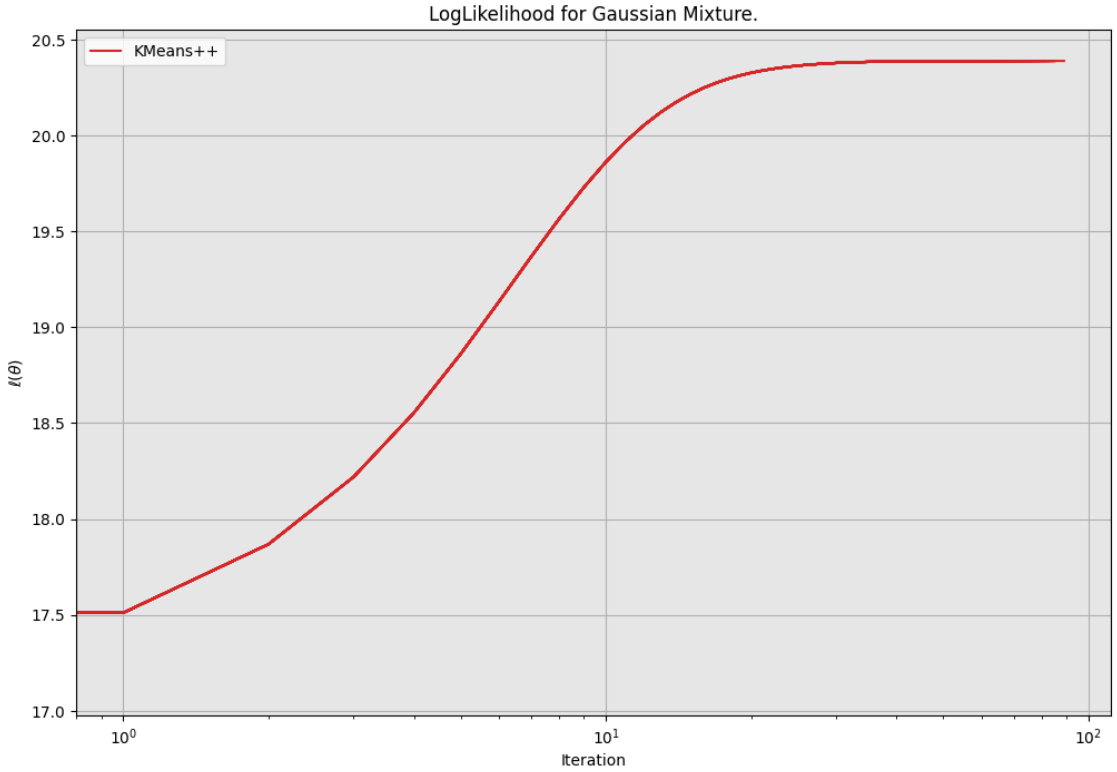
Problem 3, part g

Apply the Mixture of Gaussians clustering algorithm to the toy data in Question 2. Plot the final result as a 2D plot in which each point's color or symbol indicates its cluster assignment. Comment on the plot and the convergence speed compared to that of k-means.

Solution:

I have spent a significant amount of time typing this from scratch in C++ and debugging the whole thing. My best results are shown below. I am not aware of the exact value we should be seeing for the log likelihood, but the results incorrectly cluster all points into the third cluster. This could result from my incorrect initialization of the covariance matrices, since my initial values are just random points within our space. I will include all my code in submission, just for confirmation.


K means on Toydata.txt
C_1
C_2
C_3
y - value
x - value
4
2
0
−2
−4
−2
0
2
4
6


LogLikelihood for Gaussian Mixture.
KMeans++
$\ell(\theta)$
Iteration
20.5
20.0
19.5
19.0
18.5
18.0
17.5
17.0
10^0
10^1
10^2

Problem 4

You likely found that on the “toydata” dataset, most of the time even vanilla k–means clustering produces acceptable solutions. Create a dataset of your own for which (on average over many runs) k–means++ improves the performance of k–means by at least a factor of 10 in terms of the distortion function value of the final clustering. Submit the code that you used to generate the dataset.

Solution:

A "nice" example of a dataset which would generate this disparity is for there to be two clusters of data points, far from each other, where we initialize $k = 2$. This can be done for arbitrary number of clusters, we just require there to be an equal number of clusters of data sufficiently far from each other. In this example, k-means++ will always initialize (or be heavily weighted toward) choosing the initial cluster centroids within different clusters of data, whereas random initialization will have a chance of selecting points within the same cluster as initial steps. Since the two clusters are sufficiently far apart, a cluster centroid will never average out into the other cluster, which would significantly decrease the distortion function. Therefore, we can see that for sufficient distance, we will attain an average distortion function ratio of 10. An example of this is given for the following points in 2D space:

```
-10.0, 1.0
-10.0, -1.0
10.0, 1.0
10.0, -1.0
```

Note the two clusters appearing at -10 and 10 on the x axis. More data points can be included, but this is a concise example to illustrate the disparity.