K-Means is a greedy algorithm for data clustering.

Input: items $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n \in \mathcal{X}$

Desired clusters $K$

Output: $\tilde{\sigma} \in [K]^n$

With a cost fn. representing the avg distance between the datapoints and their centers $(\tilde{c} = (c_1, \ldots, c_K) \in \mathcal{X})$

$$\mathcal{L}(\tilde{\sigma}, \tilde{c}) = \frac{1}{n} \sum_{i=1}^{n} l(\tilde{x}_i, \tilde{c}_{\sigma(i)})$$

$l : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a measure of distance.

We compute the centers of current clusters (update $\tilde{c}$) and then update the labels assigning data to the closest center (update $\tilde{\sigma}$).

$\tilde{\sigma}^0$ is randomly initialized

$\tilde{\sigma}^{\text{new}} = \tilde{\sigma}^0$

Do:

$\tilde{\sigma} = \tilde{\sigma}^{\text{new}}$

for $j \in \{1, \ldots, K\}$

compute center $c_j$

$$c_j = \arg\min_{c_j} \sum_{i \in \{1, \ldots, n\}; \tilde{\sigma}_i = j} l(\tilde{x}_i, c_j)$$

for $i \in \{1, \ldots, n\}$

compute new label $\tilde{\sigma}_i^{\text{new}}$

$$\tilde{\sigma}_i^{\text{new}} = \arg\min_{\tilde{\sigma}_i \in [K]} l(\tilde{x}_i, \tilde{c}_{\tilde{\sigma}_i})$$

while $\mathcal{L}(\tilde{\sigma}^{\text{new}}) \leq \mathcal{L}(\tilde{\sigma}) - \epsilon$

return $\tilde{\sigma}^{\text{new}}$
We are considering the case where
\[ x_i \in X \in \mathbb{R}^d \]
\[ \ell(\hat{x}, \hat{y}) = \| \hat{x} - \hat{y} \|_2^2 \]
so the center of the cluster is the empirical mean
and is updated by for \( j = 1, \ldots, k \)
\[ c_j = \frac{1}{n_j} \sum_{i \in \{i : \sigma_i = j\}} \hat{x}_i \]
where \( n_j = |\{i : \sigma_i = j\}| \) = cardinality of \( \sigma \)

We can also update the computation of the new label
given the function \( \ell \)
\( \hat{\sigma} \) is updated by for \( i = 1, \ldots, n_f \)
\[ \sigma_i = \arg\min_{\sigma \in \{1, \ldots, k\}} \ell(\hat{x}_i, \hat{c}_\sigma) \]
\[ \sigma_i = \arg\min_{\sigma \in \{1, \ldots, k\}} \| \hat{x}_i - \hat{c}_\sigma \|_2^2 \]
\[ \sigma_i = \arg\min_{\sigma \in \{1, \ldots, k\}} \| x_i \|_2^2 - 2 \langle \hat{x}_i, \hat{c}_\sigma \rangle + \| \hat{c}_\sigma \|_2^2 \]

since \( \| x_i \|_2^2 \) doesn't change
\[ \min \left( \| x_i \|_2^2 - 2 \langle \hat{x}_i, \hat{c}_\sigma \rangle + \| \hat{c}_\sigma \|_2^2 \right) = \min \left( -2 \langle \hat{x}_i, \hat{c}_\sigma \rangle + \| \hat{c}_\sigma \|_2^2 \right) \]

We will call this \( \ell \_\text{simplified} \)
The three clusters can only be identified as separate, they can’t be identified to have a specific label. Thus, they are identified up to a permutation of the labels. If error is calculated with the wrong permutation of labels, it would be artificially inflated. Therefore, the most accurate error is the minimum error across all possible permutations of labels. The error is calculated for each possible permutation, and the minimum of these errors is the true error. This is done with the helper function KMeansError.
As $s$ increases, the error (fraction misclassified) decreases. This is logical, because $s$ increasing further separates the clusters.

With large $s$, there are one or two trials where there are high errors, which increases the mean error for that value of $s$ (here at $s = 6, 7, 8, 9.5$). In these trials, two of the three centroids combine into each other. This might just be due to that particular initialization leading to a saddle point when optimizing.
\( \bar{X} \in \mathbb{R}^{n \times d} \) rows are \( \bar{x}_i^T, \ldots, \bar{x}_n^T \)

\[
\mathcal{L}(\bar{C}, \tilde{C}) = \frac{1}{n} \sum_{i=1}^{n} L(\bar{x}_i, \tilde{C}_{\sigma(i)}) \quad L(\bar{x}_i, \tilde{y}) = \|\bar{x} - \tilde{y}\|_2^2
\]

\[
\mathcal{L}(\bar{C}, \tilde{C}) = \frac{1}{n} \sum_{i=1}^{n} \|\bar{x}_i - \tilde{C}_{\sigma(i)}\|_2^2
\]

We know that for a given \( \sigma \), the minimizing centroids satisfy

\[
C_j = \frac{1}{n_j} \sum_{i \epsilon i\sigma, j} \bar{x}_i \quad \sigma^{-1}(a) = \{ i \epsilon \mathbb{N} : \sigma_i = a \}
\]

\[
C_a = \frac{1}{n_a} \sum_{i \epsilon \sigma^{-1}(a)} \bar{x}_i \quad n_a = |\sigma^{-1}(a)|
\]

rewrite sum as mutl by vector

\[
C_a = \frac{1}{n_a} \bar{X}^T 1_{\sigma^{-1}(a)}
\]

\[
\mathcal{L}(\bar{C}, \tilde{C}) = \frac{1}{n} \sum_{a=1}^{k} \sum_{i \epsilon \sigma^{-1}(a)} \|\bar{x}_i - \frac{1}{n_a} \bar{X}^T 1_{\sigma^{-1}(a)}\|_2^2
\]

\[
= \frac{1}{n} \sum_{a=1}^{k} \left( \sum_{i \epsilon \sigma^{-1}(a)} \|\bar{x}_i\|_2^2 - \frac{1}{n_a} \|\bar{X}^T 1_{\sigma^{-1}(a)}\|_2^2 \right)
\]

\[
= \frac{1}{n} \sum_{a=1}^{k} \sum_{i \epsilon \sigma^{-1}(a)} \|\bar{x}_i\|_2^2 - \frac{1}{n} \sum_{a=1}^{k} \frac{1}{n_a} \|\bar{X}^T 1_{\sigma^{-1}(a)}\|_2^2 \geq \|\bar{X}\|_F^2 = \sum \sum \|\bar{x}_i\|
\]

\[
\sqrt{\mathcal{L}(\bar{C}, \tilde{C})} = \frac{1}{n} \|\bar{X}\|_F^2 - \frac{1}{n} \sum_{a=1}^{k} \frac{1}{n_a} \|\bar{X}^T 1_{\sigma^{-1}(a)}\|_2^2
\]
\[ \chi(\sigma) = \frac{1}{n} \| X \|_F^2 - \frac{1}{n} \sum_{a=1}^{k} \frac{1}{\sqrt{n_a}} \| X^T 1_{\sigma - 1(a)} \|_2^2 \]

\[ = \frac{1}{n} \| X \|_F^2 - \frac{1}{n} \sum_{a=1}^{k} \| X^T \frac{1}{\sqrt{n_a}} 1_{\sigma - 1(a)} \|_2^2 \]

\[ = \frac{1}{n} \| X \|_F^2 - \frac{1}{n} \| X^TM \|_F^2 \]

where \( M = \left[ \frac{1}{\sqrt{n_1}} 1_{\sigma - 1(1)}, \ldots, \frac{1}{\sqrt{n_k}} 1_{\sigma - 1(k)} \right] \in \mathbb{R}^{n \times k} \)

Since \( \{ 1_{\sigma - 1(a) : a \in [k]} \} \) is an orthogonal set

\[ \left\{ \frac{1}{\sqrt{n_a}} 1_{\sigma - 1(a)} : a \in [k] \right\} \text{ is an orthonormal set} \]

\( M \) is the projection matrix for this orthonormal set

We know from SVD projection property that for \( k \) factors

\[ \| X^TM \|_F^2 \leq \sum_{j=1}^{k} \sigma_j(\bar{X})^2 \]

so

\[ \frac{1}{n} \| X^TM \|_F^2 \geq \frac{1}{n} \sum_{j=1}^{k} \sigma_j(\bar{X})^2 \]

\[ \chi(\sigma) = \frac{1}{n} \| X \|_F^2 - \frac{1}{n} \| X^TM \|_F^2 \geq \frac{1}{n} \| X \|_F^2 - \frac{1}{n} \sum_{j=1}^{k} \sigma_j(\bar{X})^2 \]

\[ \geq \frac{1}{n} \sum_{j=k+1}^{nvd} \sigma_j(\bar{X})^2 \]

\[ \geq \frac{1}{n} \sum_{j=k+1}^{nvd} \sigma_j(\bar{X})^2 \]

\[ \geq \frac{1}{n} \sum_{j=k+1}^{nvd} \sigma_j(\bar{X})^2 \]

combine sums
The lower bound calculated from SVD is around 7 consistently. The mean calculated loss is around 10. The mean loss is also affected by the few trials where two centroids combine (here at s= 6, 7, 8, 9.5), leading to much higher losses.
PART C
run on the provided data from seeds_dataset.txt
without pre-processing

<table>
<thead>
<tr>
<th>Loss</th>
<th>Mean Error (fractional)</th>
<th># Misclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.80372377</td>
<td>0.10952381</td>
<td>23</td>
</tr>
<tr>
<td>2.79675529</td>
<td>0.1047619</td>
<td>22</td>
</tr>
<tr>
<td>2.79675529</td>
<td>0.1047619</td>
<td>22</td>
</tr>
<tr>
<td>2.79675529</td>
<td>0.1047619</td>
<td>22</td>
</tr>
<tr>
<td>2.79675529</td>
<td>0.1047619</td>
<td>22</td>
</tr>
<tr>
<td>2.80372377</td>
<td>0.10952381</td>
<td>23</td>
</tr>
<tr>
<td>2.80372377</td>
<td>0.10952381</td>
<td>23</td>
</tr>
<tr>
<td>2.80372377</td>
<td>0.10952381</td>
<td>23</td>
</tr>
<tr>
<td>2.79675529</td>
<td>0.1047619</td>
<td>22</td>
</tr>
<tr>
<td>2.80372377</td>
<td>0.10952381</td>
<td>23</td>
</tr>
<tr>
<td>2.80372377</td>
<td>0.10952381</td>
<td>23</td>
</tr>
</tbody>
</table>

We normalized the data so each column has 0 mean and std 1. This makes it so no one geometrical kernel property dominates, since they likely have different units.

<table>
<thead>
<tr>
<th>Loss</th>
<th>Mean Error (fractional)</th>
<th># Misclassified</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.05299324</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.0527833</td>
<td>0.06666667</td>
<td>14</td>
</tr>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
<tr>
<td>2.05075702</td>
<td>0.08095238</td>
<td>17</td>
</tr>
</tbody>
</table>

The lower bound of Loss determined by the SVD (as in Part B) = 0.0932225284757, which is much lower than the actual loss we observed. This difference is approximately the same as in part A (loss – svd lower bound ~ = 2), but here it is proportionally much larger since the loss itself is smaller.
import numpy as np
import itertools
import matplotlib.pyplot as plt

# KMeans
# INPUTS:
# X_bar - items to cluster of size n by d
# k - number of desired clusters
# sigma_0 - initialized labeling
# epsilon - convergence criterion
# OUTPUTS:
# sigma_new - the new labeling of clusters
# L_new - the loss for this labeling
# C - the centroids of the new clusters
def KMeans(X_bar, k, sigma_0, epsilon):
    n, d = X_bar.shape
    sigma_new = sigma_0
    C = np.zeros((k, d))  # initialize C to zeros
    L_new = -1  # initialize L_new to something impossible
    while True:
        # update variables
        sigma = sigma_new
        L = L_new
        # compute the new centroids
        for j in range(k):
            ind_j = (sigma == j)
            if any(ind_j):
                C[j, :] = np.mean(X_bar[ind_j, :], 0)
        # assign new labels
        norm_C_matrix = np.dot(np.ones((n, 1)), np.reshape(np.power(np.linalg.norm(C, axis=1), 2), (1, k)) + np.linalg.norm(X_bar, 2))
        l_simplified_matrix = -2 * np.dot(X_bar, np.transpose(C)) + norm_C_matrix
        sigma_new = np.argmin(l_simplified_matrix, axis=1)
        # compute new loss
        sum_norm_x = np.linalg.norm(X_bar)**2
        L_new = (sum([l_simplified_matrix[i, sigma_new[i]] for i in range(n)]) + sum_norm_x) / n
        # check break condition
        if (L_new > (L - epsilon)) and (L != -1):
            break
    return sigma_new, L_new, C

# KMeansError
# INPUTS:
# k - number of clusters
# sigma_true - true labeling
# sigma - provided labeling
# OUTPUT:
# clustering_error - error in clustering
def KMeansError(k, sigma_true, sigma):
    min_error = 100  # initialize with huge value
    for curr_permutation in itertools.permutations(range(k)):
        sigma_permuted = [curr_permutation[i] for i in sigma]
        curr_error = np.mean([sigma_true[i] != sigma_permuted[i] for i in range(len(sigma_true))])
        if curr_error < min_error:
            min_error = curr_error
            clustering_error = min_error
    return clustering_error

# PART B
# LowerBound
# INPUTS:
# X_bar - items being clustered of size n by d
# k - number of clusters
# OUTPUT:
# L_lower - SVD lower bound
def LowerBound(X_bar, k):
    n, d = X_bar.shape
    u, s, vh = np.linalg.svd(X_bar)
    L_lower = np.sum(np.power(s[k:], 2)) / n
    return L_lower
# PART A
\( n = 10000 \)
\( d = 10 \)
\( k = 3 \)

\[ s_{values} = \text{np.linspace}(0.5,10,20) \]
\[ n_{trials} = 10 \]
\[ \text{num}_s_{values} = \text{len}(s_{values}) \]
\[ \text{Loss} = \text{np.zeros}((\text{num}_s_{values},n_{trials})) \]
\[ \text{Error} = \text{np.zeros}((\text{num}_s_{values},n_{trials})) \]
\[ \text{Loss}_{lower} = \text{np.zeros}((\text{num}_s_{values},n_{trials})) \]

for \( i \) in range(\( \text{num}_s_{values} \)):
    \[ s = s_{values}[i] \]
    for \( j \) in range(\( n_{trials} \)):
        # make \( X_{\text{bar}} \)
        \[ X_{\text{bar}} = \text{np.random.randn}(n,d) \]
        \[ \text{sigma}_{true} = \text{np.random.randint}(k,size=n) \] # assign to one of \( k \) clusters
        for \( p \) in range(n):
            \[ X_{\text{bar}}[p,\text{sigma}_{true}[p]] += s \]
        # run KMeans
        \[ \text{sigma}_0 = \text{np.random.randint}(k,size=n) \]
        \[ \text{epsilon} = 1e-6 \]
        \[ \text{sigma}, L, C = \text{KMeans}(X_{\text{bar}},k,\text{sigma}_0,\text{epsilon}) \]
        \[ \text{Loss}[i,j] = L \]
        \[ \text{Error}[i,j] = \text{KMeansError}(k,\text{sigma}_{true},\text{sigma}) \]

# PART B
\[ \text{mean}_{\text{Error}} = \text{np.mean}(	ext{Error},1) \]

\text{print}('s values:')
\text{print}(s_{values})
\text{print}('mean error:')
\text{print}(\text{mean}_{\text{Error}})

\text{plt.plot}(s_{values},\text{mean}_{\text{Error}},'o-')
\text{plt.xlabel}('s')
\text{plt.ylabel}('fraction missclassified')
\text{plt.title}('Mean Fraction Missclassified')
\text{plt.savefig}('FractionMissclassified.jpg')
\text{plt.close}()

# PART B
\[ \text{mean}_{\text{Loss}} = \text{np.mean}(	ext{Loss},1) \]
\[ \text{mean}_{\text{Loss}_{lower}} = \text{np.mean}(	ext{Loss}_{lower},1) \]
\text{print}('mean loss:')
\text{print}(\text{mean}_{\text{Loss}})
\text{print}('mean loss lower bound:')
\text{print}(\text{mean}_{\text{Loss}_{lower}})

\text{plt.plot}(s_{values},\text{mean}_{\text{Loss}},'o-',\text{label}='KMeans loss')
\text{plt.plot}(s_{values},\text{mean}_{\text{Loss}_{lower}},'o-',\text{label}='loss lower bound')
\text{plt.xlabel}('s')
\text{plt.ylabel}('Loss')
\text{plt.title}('Comparing Loss with Lower Bound')
\text{plt.savefig}('CompareWithLower.jpg')
\text{plt.close}()}
# PART C

data = np.genfromtxt('seeds_dataset.txt', delimiter='t')
n,d = data.shape
d = d - 1
X_bar = data[:, :d]
sigma_true = data[:, d] - 1

n_trials = 10
k = 3
epsilon = 1e-6
Loss = np.zeros(n_trials)
Error = np.zeros(n_trials)
for j in range(n_trials):
sigma_0 = np.random.randint(k, size=n)
sigma, L, C = KMeans(X_bar, k, sigma_0, epsilon)
Loss[j] = L
Error[j] = KMeansError(k, sigma_true, sigma)

Loss_lower_seeds = LowerBound(X_bar, k)

print('loss: %s' % Loss)
print('error frac: %s' % Error)
print('num wrong: %s' % Error*n)
print(Loss_lower_seeds)

# pre-process the data by having each column mean 0 and std 1
X_bar_proc = np.zeros((n, d))
for i in range(d):
    X_bar_proc[:, i] = (X_bar[:, i] - np.mean(X_bar[:, i])) / np.std(X_bar[:, i])

Loss = np.zeros(n_trials)
Error = np.zeros(n_trials)
for j in range(n_trials):
sigma_0 = np.random.randint(k, size=n)
sigma, L, C = KMeans(X_bar_proc, k, sigma_0, epsilon)
Loss[j] = L
Error[j] = KMeansError(k, sigma_true, sigma)

Loss_lower_seeds = LowerBound(X_bar_proc, k)

print('loss: %s' % Loss)
print('error frac: %s' % Error)
print('num wrong: %s' % Error*n)
print(Loss_lower_seeds)