# hw8

March 7, 2024

## 1 Homework

Evelina Teran & Kevin Smith

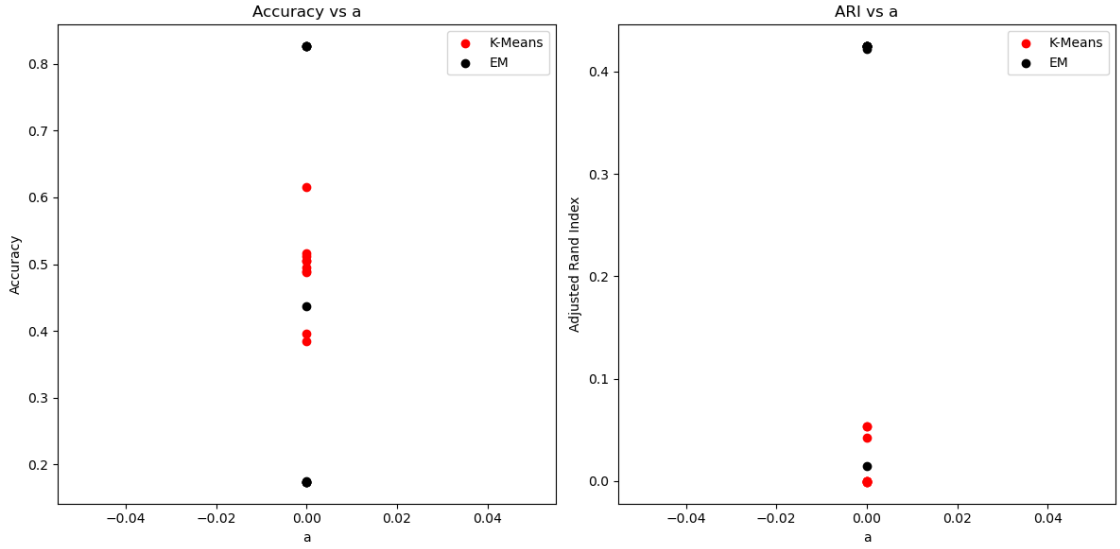
[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score, adjusted_rand_score, cluster
from numpy.linalg import svd
from tabulate import tabulate
```

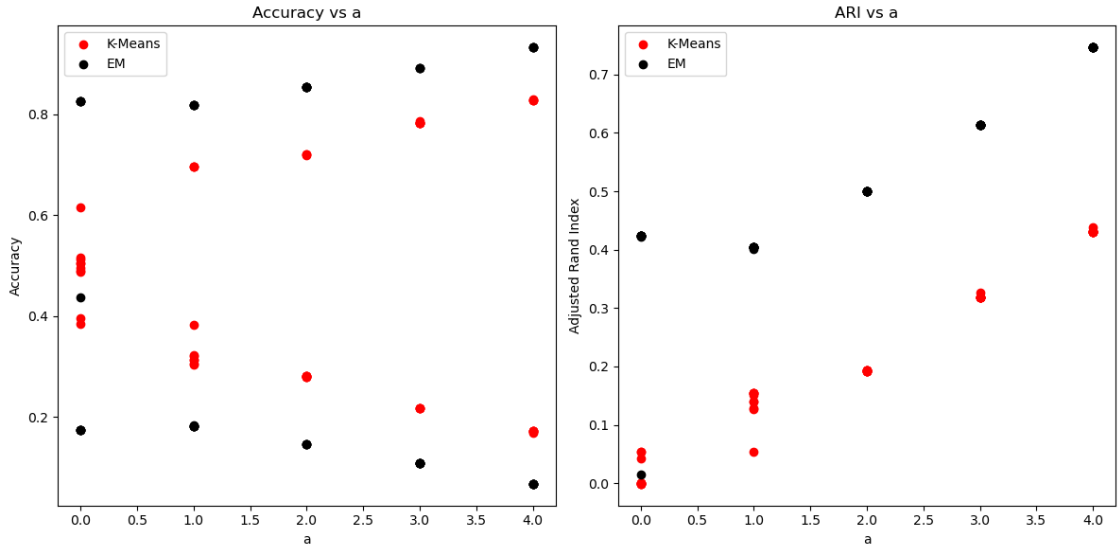
### 1.1 Part A

[ ]:
```
def generate_data(a, sigma=3, n=500):
    Xa = np.random.normal(loc=[a, 0], scale=1, size=(n, 2))
    Xq = np.random.normal(loc=[0, 0], scale=sigma, size=(n, 2))
    X = np.vstack((Xa, Xq))
    y = np.array([1] * n + [0] * n)
    return X, y

def cluster_and_evaluate(X, y, n_clusters=2, n_runs=10):
    accuracies_kmeans = []
    accuracies_em = []
    ari_kmeans = []
    ari_em = []

    for _ in range(n_runs):
        kmeans = KMeans(n_clusters=n_clusters, n_init=1).fit(X)
        em = GaussianMixture(n_components=n_clusters, n_init=1).fit(X)

        labels_kmeans = kmeans.labels_
        labels_em = em.predict(X)

        accuracies_kmeans.append(accuracy_score(y, labels_kmeans))
        accuracies_em.append(accuracy_score(y, labels_em))

        ari_kmeans.append(adjusted_rand_score(y, labels_kmeans))
```

```
        ari_em.append(adjusted_rand_score(y, labels_em))

    return accuracies_kmeans, accuracies_em, ari_kmeans, ari_em

def plot_results(a_values, accuracies_kmeans, accuracies_em, ari_kmeans,␣
 ↪ari_em):
    plt.figure(figsize=(12, 6))

    # Repeat each value of a 10 times to match the number of runs
    if len(accuracies_em) == 5 :
        a_values_repeated = np.repeat(a_values, 10)
    else:
        a_values_repeated = np.repeat(0, 10)

    plt.subplot(1, 2, 1)
    plt.scatter(a_values_repeated, accuracies_kmeans, color='red',␣
 ↪label='K-Means')
    plt.scatter(a_values_repeated, accuracies_em, color='black', label='EM')
    plt.xlabel('a')
    plt.ylabel('Accuracy')
    plt.title('Accuracy vs a')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.scatter(a_values_repeated, ari_kmeans, color='red', label='K-Means')
    plt.scatter(a_values_repeated, ari_em, color='black', label='EM')
    plt.xlabel('a')
    plt.ylabel('Adjusted Rand Index')
    plt.title('ARI vs a')
    plt.legend()

    plt.tight_layout()
    plt.show()


a_values = [0, 1, 2, 3, 4]
accuracies_kmeans = []
accuracies_em = []
ari_kmeans = []
ari_em = []

for a in a_values:
    X, y = generate_data(a)
    acc_kmeans, acc_em, ari_k, ari_e = cluster_and_evaluate(X, y)
    accuracies_kmeans.append(acc_kmeans)
    accuracies_em.append(acc_em)
    ari_kmeans.append(ari_k)
```

```
    ari_em.append(ari_e)
    if a == 0:
        plot_results(a_values, accuracies_kmeans, accuracies_em, ari_kmeans,␣
↪ari_em)

plot_results(a_values, accuracies_kmeans, accuracies_em, ari_kmeans, ari_em)
```



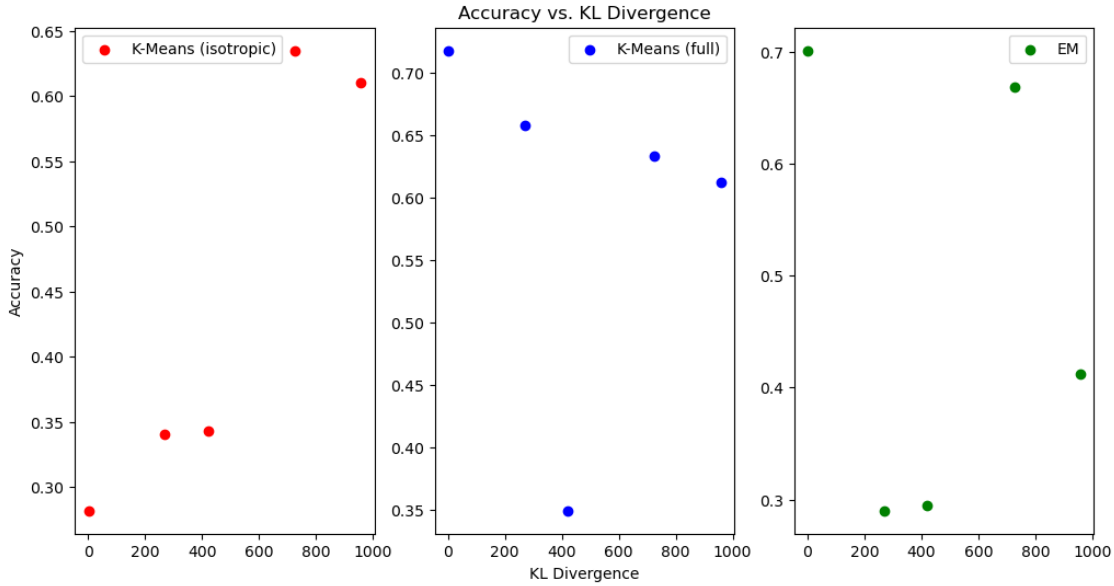
### 1.2 Part B

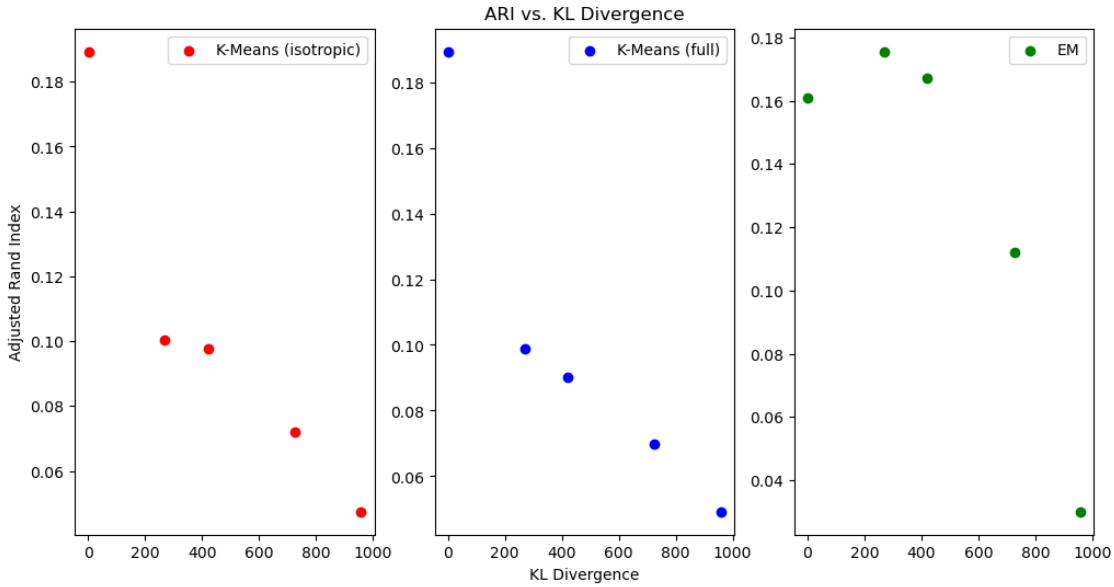
[ ]:
```
def generate_random_rotation_matrix():
    M = np.random.normal(0, 1, (2, 2))
    U, _, _ = svd(M)
    return U

def compute_kl_divergence(cov1, cov2):
    inv_cov2 = np.linalg.inv(cov2)
    return 0.5 * (np.log(np.linalg.det(cov2) / np.linalg.det(cov1))
                  - len(cov1) + np.trace(inv_cov2 @ cov1)
                  + (np.mean(cov2, axis=0) - np.mean(cov1, axis=0)).T @␣
 ↪inv_cov2 @ (np.mean(cov2, axis=0) - np.mean(cov1, axis=0)))

def generate_dataset(U):
    cov = U @ np.diag([100, 1]) @ U.T
    Xq = np.random.multivariate_normal([0, 0], cov, 500)
    Xp = np.random.multivariate_normal([10, 0], cov, 500)
    X = np.vstack((Xq, Xp))
    y = np.array([0] * 500 + [1] * 500)
    return X, y, cov

def run_clustering(X, y):
    kmeans_iso = KMeans(n_clusters=2, n_init=10).fit(X)
    em_iso = GaussianMixture(n_components=2, covariance_type='spherical').fit(X)
    em_full = GaussianMixture(n_components=2, covariance_type='full').fit(X)

    acc_kmeans_iso = accuracy_score(y, kmeans_iso.labels_)
    acc_em_iso = accuracy_score(y, em_iso.predict(X))
    acc_em_full = accuracy_score(y, em_full.predict(X))

    ari_kmeans_iso = adjusted_rand_score(y, kmeans_iso.labels_)
    ari_em_iso = adjusted_rand_score(y, em_iso.predict(X))
    ari_em_full = adjusted_rand_score(y, em_full.predict(X))

    return acc_kmeans_iso, acc_em_iso, acc_em_full, ari_kmeans_iso, ari_em_iso,␣
 ↪ari_em_full


n_runs = 10
results = []
temp_results = []

for _ in range(n_runs):
    U = generate_random_rotation_matrix()
    X, y, cov = generate_dataset(U)
```

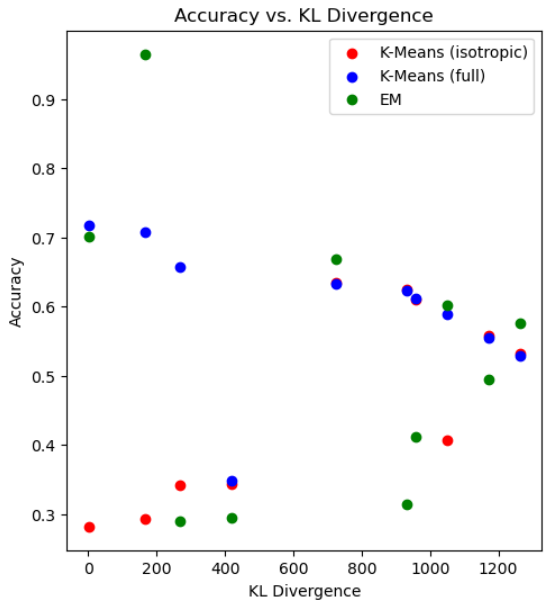
```
    acc_kmeans_iso, acc_kmeans_full, acc_em, ari_kmeans_iso, ari_kmeans_full,␣
 ↪ari_em = run_clustering(X, y)
    kl_divergence = compute_kl_divergence(np.diag([100, 1]), cov)
    results.append([acc_kmeans_iso, acc_kmeans_full, acc_em, ari_kmeans_iso,␣
 ↪ari_kmeans_full, ari_em, kl_divergence])
    if _ <= 4:
        temp_results.append([acc_kmeans_iso, acc_kmeans_full, acc_em,␣
 ↪ari_kmeans_iso, ari_kmeans_full, ari_em, kl_divergence])

# Convert results to a numpy array for easier slicing
results = np.array(results)
temp_results = np.array(temp_results)

# Plotting accuracy vs. KL divergence
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.scatter(temp_results[:, 6], temp_results[:, 0], color='red', label='K-Means␣
 ↪(isotropic)')
plt.ylabel('Accuracy')
plt.legend()
plt.subplot(1, 3, 2)
plt.scatter(temp_results[:, 6],temp_results[:, 1], color='blue', label='K-Means␣
 ↪(full)')
plt.xlabel('KL Divergence')
plt.title('Accuracy vs. KL Divergence')
plt.legend()
plt.subplot(1, 3, 3)
plt.scatter(temp_results[:, 6], temp_results[:, 2], color='green', label='EM')
plt.legend()
plt.show()
plt.close()


# Plotting ARI vs. KL divergence
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.scatter(temp_results[:, 6], temp_results[:, 3], color='red', label='K-Means␣
 ↪(isotropic)')
plt.ylabel('Adjusted Rand Index')
plt.legend()
plt.subplot(1, 3, 2)
plt.scatter(temp_results[:, 6], temp_results[:, 4], color='blue',␣
 ↪label='K-Means (full)')
plt.xlabel('KL Divergence')
plt.title('ARI vs. KL Divergence')
plt.legend()
```

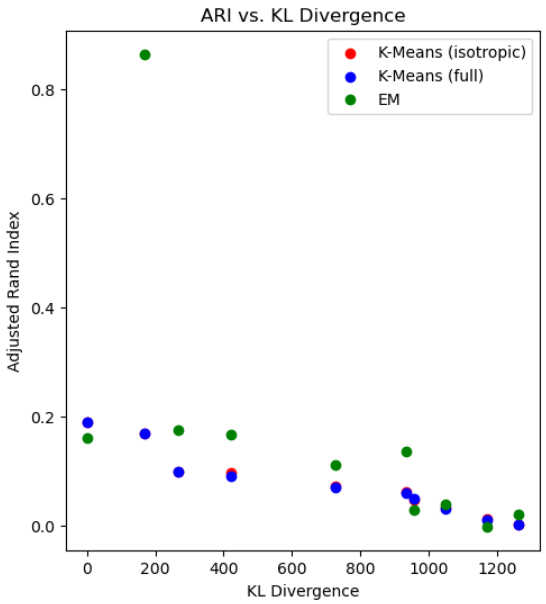
```
plt.subplot(1, 3, 3)
plt.scatter(temp_results[:, 6], temp_results[:, 5], color='green', label='EM')
plt.legend()
plt.show()
plt.close()

# Plotting accuracy vs. KL divergence
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(results[:, 6], results[:, 0], color='red', label='K-Means␣
 ↪(isotropic)')
plt.ylabel('Accuracy')
plt.title('Accuracy vs. KL Divergence')
plt.scatter(results[:, 6], results[:, 1], color='blue', label='K-Means (full)')
plt.xlabel('KL Divergence')
plt.title('Accuracy vs. KL Divergence')
plt.scatter(results[:, 6], results[:, 2], color='green', label='EM')
plt.title('Accuracy vs. KL Divergence')
plt.legend()


# Plotting ARI vs. KL divergence
plt.subplot(1, 2, 2)
plt.scatter(results[:, 6], results[:, 3], color='red', label='K-Means␣
 ↪(isotropic)')
plt.xlabel('KL Divergence')
plt.ylabel('Adjusted Rand Index')
plt.title('ARI vs. KL Divergence')
plt.scatter(results[:, 6], results[:, 4], color='blue', label='K-Means (full)')
plt.xlabel('KL Divergence')
plt.ylabel('Adjusted Rand Index')
plt.title('ARI vs. KL Divergence')
plt.scatter(results[:, 6], results[:, 5], color='green', label='EM')
plt.title('ARI vs. KL Divergence')
plt.legend()
plt.show()
plt.close()


# Printing the results table
headers = ["KM(iso)", "KM(full)", "EM", "ARI(iso)", "ARI(full)", "ARI(EM)", "KL␣
 ↪Div"]

# Format the results to display with three decimal points
formatted_results = [[item if isinstance(item, str) else f"{item:.3f}" for item␣
 ↪in row] for row in results]
```

```
# Print the table
print("Results Table:")
print(tabulate(formatted_results, headers=headers, tablefmt="pretty"))
```






Results Table:

| KM(iso) | KM(full) | EM | ARI(iso) | ARI(full) | ARI(EM) | KL Div |
|---|---|---|---|---|---|---|
| 0.282 | 0.718 | 0.701 | 0.189 | 0.189 | 0.161 | 0.813 |
| 0.635 | 0.633 | 0.668 | 0.072 | 0.070 | 0.112 | 726.207 |
| 0.343 | 0.349 | 0.295 | 0.098 | 0.090 | 0.167 | 420.782 |
| 0.610 | 0.612 | 0.412 | 0.047 | 0.049 | 0.030 | 957.557 |
| 0.341 | 0.658 | 0.290 | 0.100 | 0.099 | 0.176 | 268.627 |
| 0.558 | 0.554 | 0.494 | 0.012 | 0.011 | -0.001 | 1171.205 |
| 0.532 | 0.528 | 0.576 | 0.003 | 0.002 | 0.022 | 1263.264 |
| 0.407 | 0.589 | 0.602 | 0.034 | 0.031 | 0.041 | 1050.019 |
| 0.625 | 0.623 | 0.314 | 0.062 | 0.060 | 0.138 | 932.563 |
| 0.293 | 0.707 | 0.965 | 0.171 | 0.171 | 0.865 | 167.399 |