z5286124 Jinghan Wang

(a).

- First Iteration.

$$\mu_1^{(0)} = \begin{bmatrix} 5 \\ 2 \end{bmatrix} \quad \mu_2^{(0)} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} \rightarrow \mu_1^{(0)} \quad \begin{bmatrix} 4 \\ 4 \end{bmatrix} \rightarrow \mu_2^{(0)} \quad \begin{bmatrix} 6 \\ 5 \end{bmatrix} \rightarrow \mu_1^{(0)}$$

$$\begin{bmatrix} 5 \\ 6 \end{bmatrix} \rightarrow \mu_2^{(0)} \quad \begin{bmatrix} 2 \\ 3 \end{bmatrix} \rightarrow \mu_2^{(0)}$$

As $\mu_k^{(t)} = \frac{1}{|C_k^{(t)}|} \sum_{x_i \in C_k^{(t)}} x_i$

$$\mu_1^{(1)} = \left\{ \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \begin{bmatrix} 6 \\ 5 \end{bmatrix} \right\} = \begin{bmatrix} 5.50 \\ 3.00 \end{bmatrix} \quad \mu_2^{(1)} = \left\{ \begin{bmatrix} 4 \\ 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \right\} = \begin{bmatrix} 3.67 \\ 3.67 \end{bmatrix}$$

- Second Iteration

$$\begin{bmatrix} 5 \\ 3 \end{bmatrix} \rightarrow \mu_1^{(1)} \quad \begin{bmatrix} 4 \\ 4 \end{bmatrix} \rightarrow \mu_2^{(1)} \quad \begin{bmatrix} 6 \\ 5 \end{bmatrix} \rightarrow \mu_1^{(1)}$$

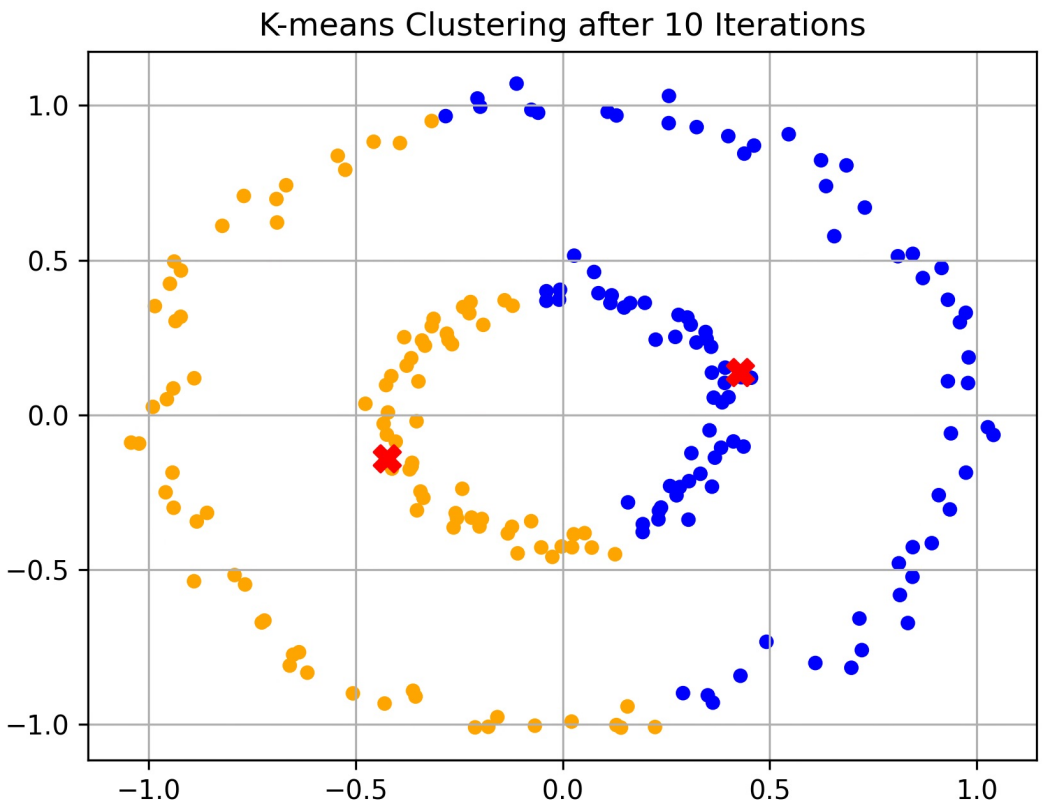
$$\begin{bmatrix} 5 \\ 6 \end{bmatrix} \rightarrow \mu_1^{(1)} \quad \begin{bmatrix} 2 \\ 3 \end{bmatrix} \rightarrow \mu_2^{(1)}$$

$$\mu_1^{(2)} = \left\{ \begin{bmatrix} 5 \\ 3 \end{bmatrix}, \begin{bmatrix} 6 \\ 5 \end{bmatrix} \begin{bmatrix} 5 \\ 4 \end{bmatrix} \right\} = \begin{bmatrix} 5.33 \\ 5.33 \end{bmatrix} \quad \mu_2^{(2)} = \left\{ \begin{bmatrix} 4 \\ 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} \right\} = \begin{bmatrix} 3 \\ 3.5 \end{bmatrix}$$

(b) -

Clustering a dataset with 10,000 features using K-means may encounter the problem of "dimensionality catastrophe", which makes the Euclidean distance less discriminative in high-dimensional spaces. In addition, K-means is very sensitive to initialization and feature scaling. Before applying K-means, consider using dimensionality reduction techniques such as PCA and make sure that the features have been scaled appropriately.

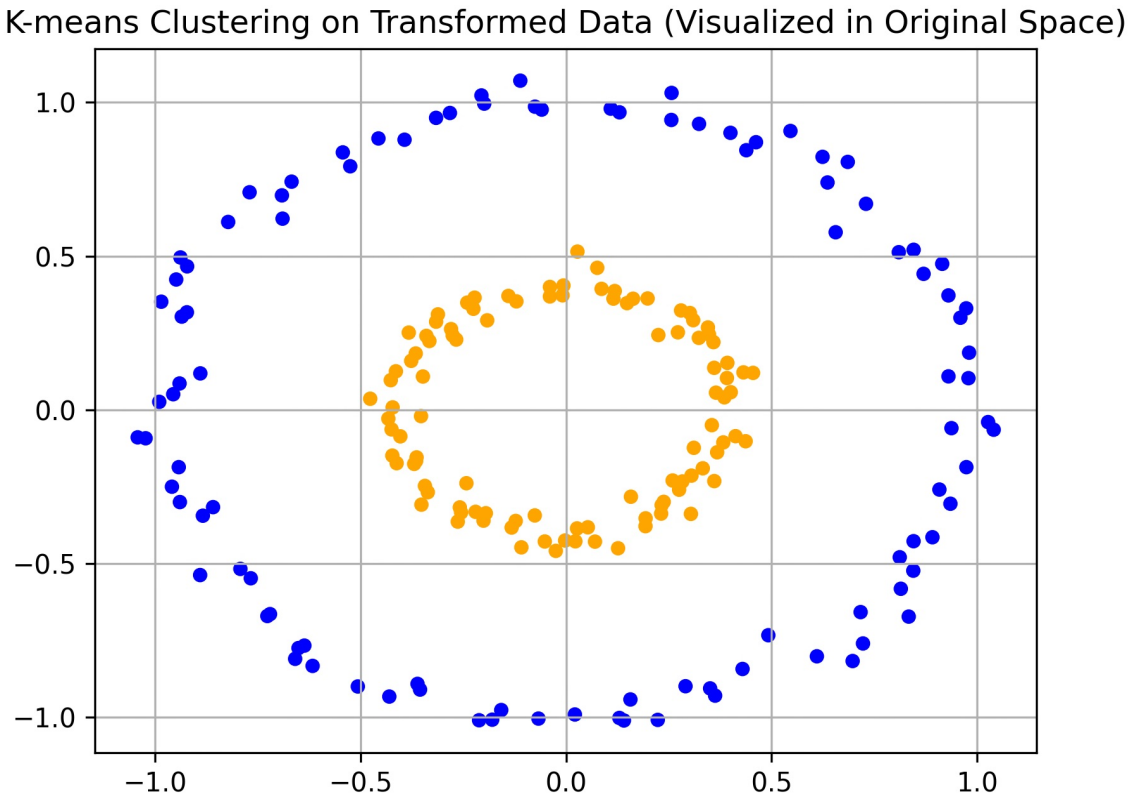
(c).



No, K-means doesn't perform well on this data. The data consists of two concentric circles, but K-means tries to find spherical clusters, leading to incorrect clustering. Alternative algorithms that can handle non-convex clusters would be more suitable for this dataset.

```python
def kmeans(X, initial_centers, num_iterations=10):
    global labels
    centers = initial_centers
    for _ in range(num_iterations):
        distances = np.linalg.norm(X[:, np.newaxis] - centers, axis=2)
        labels = np.argmin(distances, axis=1)

        new_centers = np.array([X[labels == i].mean(axis=0) for i in range(len(centers))])
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return labels, centers


def q2_c():
    X, y = datasets.make_circles(n_samples=200, factor=0.4, noise=0.04, random_state=13)

    initial_centers = np.array([[0, 0], [1, 0]])
    labels, final_centers = kmeans(X, initial_centers, num_iterations=10)

    plt.scatter(X[:, 0], X[:, 1], s=20, color=np.array(['orange', 'blue'])[labels])
    plt.scatter(final_centers[:, 0], final_centers[:, 1], s=100, c='red', marker='X')
    plt.title("K-means Clustering after 10 Iterations")
    plt.grid(True)
    plt.savefig( *args: "q2_c.png", dpi=300)
    plt.show()
```

(d).



By performing feature transformation on the data, K-means successfully distinguishes two concentric circles. This shows that the performance of K-means on certain complex data structures can be improved by suitable feature transformation.

```python
def phi_transform(X):
    x1_sq = X[:, 0] ** 2
    x2_sq = X[:, 1] ** 2
    x1_x2 = np.sqrt(2) * X[:, 0] * X[:, 1]
    return np.vstack((x1_sq, x2_sq, x1_x2)).T


def q2_d():
    X, y = datasets.make_circles(n_samples=200, factor=0.4, noise=0.04, random_state=13)
    X_transformed = phi_transform(X)
    initial_centers_transformed = np.array([[0, 0, 0], [1, 1, 0]])
    labels_transformed, final_centers_transformed = kmeans(X_transformed, initial_centers_transformed,
                                                           num_iterations=10)
    plt.scatter(X[:, 0], X[:, 1], s=20, color=np.array(['orange', 'blue'])[labels_transformed])
    plt.title("K-means Clustering on Transformed Data (Visualized in Original Space)")
    plt.grid(True)
    plt.savefig( *args: "q2_d.png", dpi=300)
    plt.show()
```