Koushik Sahu
118CS0597
Soft Computing Laboratory – I
10th January 2022

Code:

```
import numpy as np
from tqdm import tqdm
import seaborn as sn
import matplotlib.pyplot as plt
import pandas as pd


def get_distance(x1, x2):
    sum = 0
    for i in range(len(x1)):
        sum += (x1[i] - x2[i]) ** 2
    return np.sqrt(sum)


def kmeans(X, k, max_iters):
    print('Applying K-MEANS:\n')
    centroids = X[np.random.choice(range(len(X)), k, replace=False)]
    converged = False
    current_iter = 0

    while (not converged) and (current_iter < max_iters):
        cluster_list = [[] for _ in range(len(centroids))]

        for x in tqdm(X):
            distances_list = []
            for c in centroids:
                distances_list.append(get_distance(c, x))
            cluster_list[int(np.argmin(distances_list))].append(x)

        cluster_list = list((filter(None, cluster_list)))
        prev_centroids = centroids.copy()
        centroids = []

        for j in tqdm(range(len(cluster_list))):
            centroids.append(np.mean(cluster_list[j], axis=0))

        pattern = np.abs(np.sum(prev_centroids) - np.sum(centroids))
        print('K-MEANS pattern count: ', int(pattern))
```

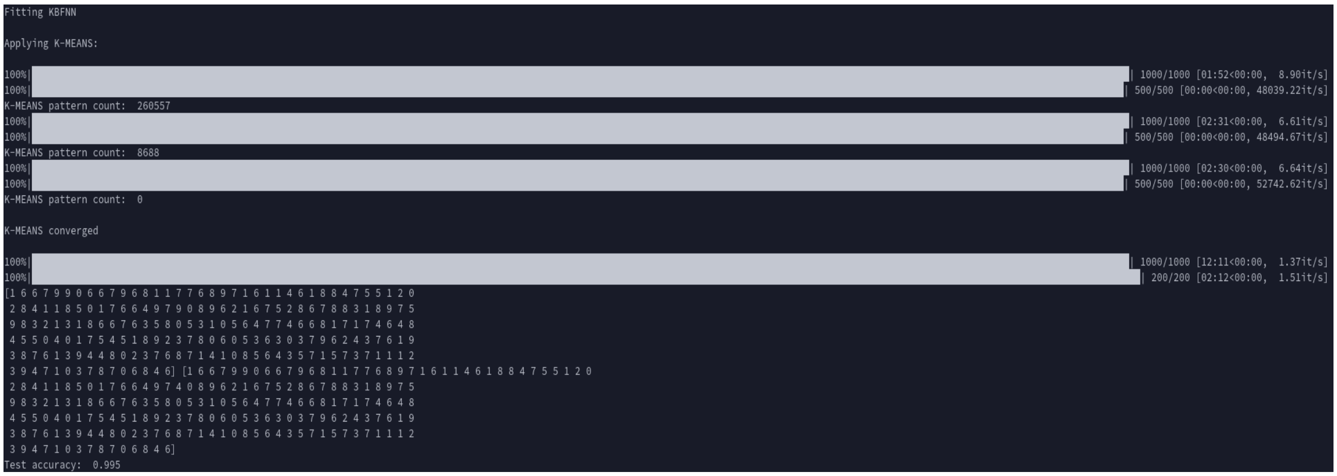
```
        converged = (pattern == 0)
        current_iter += 1

    print('\nK-MEANS converged\n')
    return np.array(centroids), [np.std(x) for x in cluster_list]


class RBF:
    def __init__(self, X, y, tX, ty, num_of_classes,
            k, std_from_clusters=True):
        self.X = X
        self.y = y

        self.tX = tX
        self.ty = ty

        self.number_of_classes = num_of_classes
        self.k = k
        self.std_from_clusters = std_from_clusters

    def convert_to_one_hot(self, x, num_of_classes):
        arr = np.zeros((len(x), num_of_classes))
        for i in range(len(x)):
            c = int(x[i])
            arr[i][c] = 1
        return arr

    def rbf(self, x, c, s):
        distance = get_distance(x, c)
        return 1 / np.exp(-distance / s ** 2)

    def rbf_list(self, X, centroids, std_list):
        RBF_list = []
        for x in tqdm(X):
            RBF_list.append([self.rbf(x, c, s) for (c, s) in zip(centroids, std_list)])
        return np.array(RBF_list)

    def plot_confusion_matrix(self):
        mat = [[0 for _ in range(10)] for _ in range(10)]
        print(self.ty, self.pred_ty)
        for idx, val in enumerate(self.ty):
            mat[int(val)][int(self.pred_ty[idx])] += 1
        plt.figure(figsize = (10,7))
        sn.heatmap(mat, annot=True)
```

```
        plt.show()

    def fit(self):
        print('Fitting KBFNN\n')
        self.centroids, self.std_list = kmeans(self.X, self.k, max_iters=1000)

        if not self.std_from_clusters:
            dMax = np.max([get_distance(c1, c2) for c1 in self.centroids for c2 in self.centroids])
            self.std_list = np.repeat(dMax / np.sqrt(2 * self.k), self.k)

        RBF_X = self.rbf_list(self.X, self.centroids, self.std_list)
        self.w = np.linalg.pinv(RBF_X.T @ RBF_X) @ RBF_X.T @ self.convert_to_one_hot(self.y,
self.number_of_classes)
        RBF_list_tst = self.rbf_list(self.tX, self.centroids, self.std_list)
        self.pred_ty = RBF_list_tst @ self.w
        self.pred_ty = np.array([np.argmax(x) for x in self.pred_ty])
        diff = self.pred_ty - self.ty

        self.plot_confusion_matrix()

        print('Test accuracy: ', len(np.where(diff == 0)[0]) / len(diff))


if __name__ == '__main__':
    df = pd.read_csv('mnist_train.csv')
    df = df.sample(frac=1)

    train_y = df.iloc[0:1000, 0].to_numpy()
    train_x = df.iloc[0:1000, 1:].to_numpy()

    test_y = df.iloc[0:200, 0].to_numpy()
    test_x = df.iloc[0:200, 1:].to_numpy()

    RBF_CLASSIFIER = RBF(train_x, train_y, test_x, test_y, num_of_classes=10,
            k=500, std_from_clusters=False)

    RBF_CLASSIFIER.fit()
```

Koushik Sahu **118CS0597**

Output:

Terminal Screenshot:



```
Fitting KBFNN

Applying K-MEANS:

100%|                                                  | 1000/1000 [01:52<00:00,  8.90it/s]
100%|                                                  | 500/500 [00:00<00:00, 48039.22it/s]
K-MEANS pattern count:  260557
100%|                                                  | 1000/1000 [02:31<00:00,  6.61it/s]
100%|                                                  | 500/500 [00:00<00:00, 48494.67it/s]
K-MEANS pattern count:  8688
100%|                                                  | 1000/1000 [02:30<00:00,  6.64it/s]
100%|                                                  | 500/500 [00:00<00:00, 52742.62it/s]
K-MEANS pattern count:  0

K-MEANS converged

100%|                                                  | 1000/1000 [12:11<00:00,  1.37it/s]
100%|                                                  | 200/200 [02:12<00:00,  1.51it/s]
[1 6 6 7 9 9 0 6 6 7 9 6 8 1 1 7 7 6 8 9 7 1 6 1 1 4 6 1 8 8 4 7 5 5 1 2 0
 2 8 4 1 1 8 5 0 1 7 6 6 4 9 7 9 0 8 9 6 2 1 6 7 5 2 8 6 7 8 8 3 1 8 9 7 5
 9 8 3 2 1 3 1 8 6 6 7 6 3 5 8 0 5 3 1 0 5 6 4 7 7 4 6 6 8 1 7 1 7 4 6 4 8
 4 5 5 0 4 0 1 7 5 4 5 1 8 9 2 3 7 8 0 6 0 5 3 6 3 0 3 7 9 6 2 4 3 7 6 1 9
 3 8 7 6 1 3 9 4 4 8 0 2 3 7 6 8 7 1 4 1 0 8 5 6 4 3 5 7 1 5 7 3 7 1 1 1 2
 3 9 4 7 1 0 3 7 8 7 0 6 8 4 6] [1 6 6 7 9 9 0 6 6 7 9 6 8 1 1 7 7 6 8 9 7 1 6 1 1 4 6 1 8 8 4 7 5 5 1 2 0
 2 8 4 1 1 8 5 0 1 7 6 6 4 9 7 4 0 8 9 6 2 1 6 7 5 2 8 6 7 8 8 3 1 8 9 7 5
 9 8 3 2 1 3 1 8 6 6 7 6 3 5 8 0 5 3 1 0 5 6 4 7 7 4 6 6 8 1 7 1 7 4 6 4 8
 4 5 5 0 4 0 1 7 5 4 5 1 8 9 2 3 7 8 0 6 0 5 3 6 3 0 3 7 9 6 2 4 3 7 6 1 9
 3 8 7 6 1 3 9 4 4 8 0 2 3 7 6 8 7 1 4 1 0 8 5 6 4 3 5 7 1 5 7 3 7 1 1 1 2
 3 9 4 7 1 0 3 7 8 7 0 6 8 4 6]
Test accuracy:  0.995
```

Accuracy: 0.995 (99.5%)

Confusion matrix: