Brett Castello
Erali Shah
Rohan Oswal

ML - Midterm Project

Implementation of K-Means Algorithm

Our algorithm takes in 4 parameters including the number of clusters, distance metric, maximum iterations and random state.

1. The algorithm initializes cluster by randomly selecting 'k' samples and choosing them as the cluster centroids. A default value of k = 8 is chosen. It then runs through the maximum number of iterations, to identify the labels and simultaneously updates the cluster centroids, until convergence is achieved.
2. To identify the labels, we use the 'nearest()' method which identifies the nearest cluster to a given datapoint and returns the index of that cluster. The default distance metric used to calculate the distance between each datapoint in the sample space and the cluster centroids is the Euclidean distance metric. It iteratively tries to minimize the distances to make sure the datapoints have been well separated to form the most compact clusters possible with least variance among them.
3. The '_distance()' method in our algorithm provides the flexibility to run over several distance metrics and later we can choose the best one for a given dataset using 'best_dist_metric()'.
4. Once it identifies the corresponding cluster of each datapoint(sample), it groups the data points together according to the cluster indices. The cluster centroids are updated by averaging these samples inside each cluster until the cluster centroid stays the same. When it stops changing, we can say that convergence is achieved. After convergence, the algorithm measures an error criterion called inertia and tries to minimize it. Inertia or within-cluster sum of squares criterion measures the sum of square distances of samples to their closest cluster centroids and determines how internally coherent clusters are.
5. The algorithm returns the final labels which were identified in the fit() method, using the 'predict()' method. The transform() method is used to transform the original data points into a cluster-distance space if required. Instead of original value of each datapoint , the space now contains its distance to the cluster centroid.

To automatically determine the optimal number of clusters, we utilized 4 performance measurement metrics namely: Inertia score, Silhouette score, Calinski-Harabaz index & Davies Bouldin score.
Lower the inertia , better the clustering, but it comes at a cost of increasing 'k' which increases the bias.Thus, the optimal number of cluster is determined by trying to find the sweet spot, i.e. a cluster whose inertia score is closer to the mean of all inertia scores for varying ranges of k. Silhouette Score measures the within cluster distances and value closer to 1 indicates a better cluster. A higher Calinski-Harabaz score implies dense and well separated clusters. The Davies-Bouldin score is defined as the average similarity between each cluster and its most

similar one, and a lower value indicates better cluster. Our algorithm finds the cluster which is better in maximum evaluation results of these metrics and returns the optimal number of clusters.
Additionally, we determine the best distance metric among Euclidean, Manhattan, Chebyshev, Cosine, Canberra for any given dataset by evaluating the results on the above 4 metrics.
Using our best distance metric and optimal number of clusters for the provided 13-dimensional dataset, the final result we obtained is:

No. of iterations to convergence: 5
Inertia Score: 48.97029115513914
Silhouette Score: 0.29936674064895064
Calinski-Harabaz Score: 83.3170271984246
Davies Bouldin Score: 1.3095913112417223

Further, we performed evaluation analysis to verify our output for optimal number of clusters using the Elbow Plot and Silhouette Analysis. We usually determine k where the sum of squared distance starts to flatten out and forms an elbow. In our case, k = 3 proves to be a good choice. In Silhouette analysis, value closer to 1 is the best choice and n_clusters = 3 has the best average silhouette score among all values of k.

Model Output:

	k	max_iter	random_s	Inertia	Silhouette	Calinski-H	Davies-Bo	Time(ms)
Custom K-Means	2	200	2	64.53767	0.298722	84.7085	1.354512	128
	3	200	2	48.97029	0.299367	83.31703	1.309591	45.1
	4	200	2	46.68785	0.28866	60.76266	1.638517	58.1
	5	200	2	42.9871	0.250145	52.93419	1.642835	190
	6	200	2	41.56136	0.185977	44.72697	1.761339	194

We improved the time performance using Early Stopping Technique. The algorithm stops iterating once the labels assigned to each data point no longer change. Thus, it does not iterate through the maximum number of iterations and thereby improves the time performance considerably. Our major emphasis was to improve the performance of our algorithm which we achieved using several metrics and obtaining the optimal cluster number and best distance metric.