REPORT

HIERARCHICAL CLUSTERING FOR SEED CATEGORIZATION

NAME - KUNAL MEHTA

Steps performed

1) Training dataset is grouped into n clusters using hierarchical agglomerative clustering.

2) New features based on cluster representatives are created and added to the existing features in the training dataset. cluster representatives are found and based on these representatives, new features one for each cluster representative are added to the training dataset. Distances to each cluster representative are computed and encoded as new features

3) Each training example will have extra feature values that is calculated based on the similarity between data point and cluster using cluster representative.

4) Each unlabelled test data point is transformed into the same feature space as the training data set prior to being classified. This is done using the previously obtained cluster representatives.

5) The transformed training set is passed to the classifier for classification on unlabeled data points.

6) Performance of the classifier is measured using K-Fold cross validation

7) The classifier undergoes training using the transformed training data set and hence classification is performed using optimized parameters found from k-Fold cross validation.

Build of the code

1) Built agglomerative clustering model to divide the training data set into n clusters
2) Built KNN model for classification of labels for the unlabelled data set. For which, a classifier was trained using transformed training data set.
3) Built K-Fold validation technique in which the data set was divided into k-subsets and the holdout method was repeated k-times where each of the k subsets are used as test set and other k-1 subsets are used for the training purpose.
4) Built a function to determine good number of n clusters for every linkage criterion
5) Code segment to predict the labels on unlabelled data points

Experimentation

1) In accordance to the different linkage criterions, training dataset was divided into n different clusters
2) With respect to each linkage criterion, multiple clustering was performed.
3) For each cluster, KNN classifier was evaluated using kfold cross validation technique.

Generating new features and adding those to the existing data:

```
clustering = AgglomerativeClustering(n_clusters=n, linkage=linkage) #Instatiation of agglomerative clustering class
pred_clusters = clustering.fit_predict(train_set.values) #training data divided into n clusters

train_set['labels'] = pred_clusters #cluster ID's are assigned as lables to training data

for i in range(n): #Loop iterates the number of times as the number of clusters

    indexes = np.where(train_set['labels'] == i) #indexes are fetched where training data has labels pertaining to cluster ID
    df_i = train_set.iloc[indexes[0], :-1].reset_index(drop = True) #creating a dataframe using the fetched indexes
    centroid = list(df_i.mean()) #Cluster representative is determined as centroid for the training set
    dataset = train_set.iloc[:,:-1]
    column_name = "cluster" + str(i)

    train_set[column_name] = [np.power(np.sum(np.square(row-centroid)), 0.5) for row in dataset.values] #Features are added t
    train_set[column_name] = train_set[column_name] - train_set[column_name].min() #Scaled column using the equation for min
    train_set[column_name] = train_set[column_name]/train_set[column_name].max()

    test_set[column_name] = [np.power(np.sum(np.square(row-centroid)), 0.5) for row in test_set.values] #unlabeled test data
    test_set[column_name] = test_set[column_name] - test_set[column_name].min()
    test_set[column_name] = test_set[column_name]/test_set[column_name].max()

train_set['labels'] = train_set.pop('labels')

return train_set, test_set
```

Samples of transformed train and test data after adding features based on the cluster representatives i.e., centroid.

Training data with additional features based on cluster representatives

```
train_set.head(3)
```

	A	P	C	LK	WK	A_Coef	LKG	cluster0	cluster1	cluster2	cluster3	cluster4	labels
0	13.78	14.06	0.8759	5.479	3.156	3.136	4.872	0.123033	0.282826	0.341461	0.645702	0.721988	0
1	12.37	13.47	0.8567	5.204	2.960	3.919	5.001	0.084579	0.115066	0.482256	0.761588	0.791066	1
2	15.38	14.77	0.8857	5.662	3.419	1.999	5.222	0.317004	0.480020	0.049031	0.403345	0.545278	2

Test data points are transformed into the same feature space as the training data set, using cluster representatives obtained from training data

```
test_set.head(3)
```

	A	P	C	LK	WK	A_Coef	LKG	cluster0	cluster1	cluster2	cluster3	cluster4
0	18.83	16.29	0.8917	6.037	3.786	2.553	5.879	0.684813	0.769886	0.426564	0.007005	0.169549
1	19.46	16.50	0.8985	6.113	3.892	4.308	6.009	0.771502	0.799991	0.587154	0.051190	0.020106
2	18.94	16.32	0.8942	6.144	3.825	2.908	5.949	0.698965	0.771939	0.449342	0.000000	0.133884

Evaluating KNN through Kfold Cross Validation technique-

```
n_clusters = [3,4,5,6,7]
knn = [3,4,5,6]
linkages = ['complete', 'single', 'average']
Accuracy_linkage = []


for linkage in linkages:

    print(f"\033[1m-------------------------For linkage = {linkage}-------------------------\033[0m\n")
    Accuracy = []

    for n in n_clusters:

        train_set, test_set = dataset(df, n, linkage)
        print(f"      \033[1mFor clusters n = {n}\033[0m")
        acc_clusters = []

        for k in knn:

            result = kfoldCV(train_set, 10, k)
            acc = sum(result)/len(result)
            print(f"       Accuracy using {k} knn: {acc}")
            acc_clusters.append(acc)

        Accuracy.append(sum(acc_clusters)/len(acc_clusters))

        print(f"\n       \033[1mAccuracy for {n} clusters : {sum(acc_clusters)/len(acc_clusters)} \033[0m")
        print("\n")
    print(f"             \033[1mGood number of clusters would be: {n_clusters[np.argmax(Accuracy)]}\033[0m\n\n")

    Accuracy_linkage.append(max(Accuracy))

print(f"                         \033[1mAccuracy using {linkage} : {max(Accuracy_linkage)}\033[0m")
```

Results Procedure

1) After experimenting with different values of n clusters, linkage, nearest neighbours "***result = KfoldCV(train_set, 10, k)***"

This code segment provides evaluation of KNN classifier for a particular number of k nearest neighbours through list of accuracies for f number of folds.

2) Accuracy for KNN is calculated as mean of all the accuracies of these folds found above in the list.

3) These steps are repeated for different values of k nearest neighbours.

4) Average accuracy of particular number of clusters is calculated as the mean of accuracies of different k nearest neighbours.

5) These steps are repeated for different number of clusters

6) Best number of clusters is determined based on the highest accuracy for the corresponding linkage.

Results

Clusters	Accuracy		
	Complete	**Single**	**Average**
3	0.993	0.98	0.991
4	0.96	0.986	0.955
5	0.97	0.95	0.970
6	0.95	0.956	0.951
7	0.93	0.946	0.983
Average Accuracy	0.960	0.963	0.992

- The table above is the sample output for one of the multiple iterations performed to evaluate the model.
- From the results averaged over these iterations, best parameters to be used for the KNN Classifier are
    - Linkage -> average
    - No. of clusters -> 3
- Evaluation of KNN through Kfold cross validation on original data vs transformed data set using cluster representatives.

K	Accuracy	
	Original dataset (Original labels = 3)	**Transformed dataset (For clusters n=3)**
3	0.886	0.99
4	0.89	0.991
5	0.9	0.993
6	0.913	0.98

- After comparing the above results over multiple iterations, it can be concluded that adding new features to the feature space using cluster representatives improves performance of the model.