PRACTICAL MACHINE LEARNING

COMPARISON OF UNSUPERVISED LEARNING TECHNIQUES

January 25, 2023

Rares Patrascu

Contents

Chapter 1

Introduction

1.1 Short project description

Unsupervised techniques offer an efficient way to categorize a set of data and eliminate the need for human labor in the manual labeling. The project approach entails optimizing two unsupervised algorithms on a multi-class image classification task and comparing the best results to those of a supervised method.

1.2 Used clustering algorithms

The chosen algorithms fall into the following two types of clustering algorithms.

- Centroid-based: they use distance metrics to assign each point to the nearest cluster;

- Density-based: clusters are defined as dense sections of space divided by low density zones;

K-means is a centroid-based technique that seeks to divide n data into k categories by allocating a specific data point to a category by minimizing the squared Euclidian distance from the cluster centroid to the data point. The second used algorithm is a density-based clustering technique called DBSCAN.

Chapter 2

Dataset

For this research I have utilized a public image classification kaggle dataset named "Architectural Heritage Elements". The dataset consist of 10235 images splitted across 10 classes and the images are mainly of size 64X64 with 3 color channels. Also the folders structure of saved images is already splitted for train and test.

2.1 Data loading and processing

Images were scaled, normalized and fed into a batched Tensor with the dimensions (6399, 64, 64, 3) for train and (1221, 64, 64, 3) for test. The decrease in total data count was caused by having to discard some images from specific classes. This was done because certain classes had significantly more data than others.

Labels were also loaded based on the folder name and converted to numerical value. They will be useful when comparing to a supervised model.

2.2 Feature extraction

Features were extracted using a pretrained "vgg16" model without the classification layers. Images we're resized to a (B, C, H, W) form to comply to the needed input shape of the model and the inference outputted a batched Tensor with 4096 features per data point. The used weights we're trained on the imagenetV1 dataset.

Chapter 3

Kmeans tuning

The first tune strategy was to find out the appropriate number of clusters, because the number might differ from the number of classes giving that the labeled classes might be splitted in more clusters with simmilar features.The tested number of clusters we're "10,15,20,25,50,80,160" and their performance was based on test accuracy and some metrics for unsupervised learning. The accuracy was computed on data not given as input and labeles we're decided using a cluster tho label map constructed using train inputs.

clusters	inertia	homo	silhouette	train_acc	test_acc
10.0	271880032.000	0.398	0.062	0.545	0.516
15.0	254061008.000	0.415	0.048	0.538	0.521
20.0	242366704.000	0.461	0.048	0.606	0.572
25.0	233491136.000	0.495	0.048	0.615	0.574
50.0	209296208.000	0.550	0.037	0.663	0.609
80.0	194929952.000	0.580	0.031	0.680	0.631
160.0	174136160.000	0.624	0.030	0.708	0.649

Figure 3.1: Cluster tuning

Giving this data we picked 25 clusters as the right number of clusters giving that at this point inertia was lower than previous values, accuracy on train and test we're high and giving that they are pretty close the model is not overfitting and at least for the last 2 epochs silhouette score stayed the same.

Furthermore we tuned the initialization of the centroids between "k-means++", "random" and PCA generated components using 25 clusters.

init	time	inertia	homo	silhouette
k-means++	22.698s	20177540.000	0.486	0.046
random	7.690s	20164680.000	0.487	0.028
PCA-based	2.560s	20156484.000	0.492	0.032

Figure 3.2: Centroid initialization tuning

For the final tune we will tune the algorithm used by Kmeans for distance.

algorithm	time	inertia	homo	silhouette	acc_train	acc_test
lloyd	24.657s	233758128.000	0.481	0.034	0.587	0.545
elkan	21.489s	233758144.000	0.481	0.026	0.587	0.545

Figure 3.3: Algorithm used for kmeans.

Giving that all the metrics except silhouette are the same for both runs we will base our decision on silhouette value and we will keep the "lloyd" algorithm.

Chapter 4

DB-SCAN tuning

Db-scan has two important parameters to tune respectively "eps" that represents the maximum distance between samples and "min_samples" that represents minimum data points to form a cluster. Will first try to tune "eps" using a "min_samples" that allows Db-scan to work without clustering all data as noise.

eps_val	time	silhouette	homo	complet	nr_clusters
0.1	7.689	-0.223	0.001	0.190	4
0.5	7.282	-0.223	0.001	0.190	4
2	7.924	-0.223	0.001	0.190	4
10	7.584	-0.304	0.005	0.233	24
30	7.499	-0.363	0.076	0.183	220
60	7.896	0.149	0.014	0.125	19
80	13.604	0.368	0.002	0.110	5

Figure 4.1: Eps tuning.

We can observe we achieve a good silhouette when we have an eps with

a value of 40, 60 or 80. We are now gonna tune the "min_samples" for 40,60 respectively 80 eps.

min_samples	time	silhouette	homo	complet	nr_clusters
2	8.929	0.368	0.002	0.110	5
3	8.719	0.395	0.001	0.097	3
4	8.641	0.409	0.001	0.087	2
5	8.845	0.409	0.001	0.087	2
6	9.110	0.409	0.001	0.087	2
7	9.075	0.409	0.001	0.087	2

Figure 4.2: Min_samples tuning.

From this table we concede that giving the rising of the silhouette score and the fall of other parameters, this dataset is not appropriate for the task.

Chapter 5

Results

Final results can be seen in the following table. From this table we deduce that compared to the random and the supervized method Kmean accuracy is good, anyway we can also deduce that this dataset was not good for clustering with the DBSCAN.

algorithm	time	silhouette	acc_train	acc_test
random	0.002003908157348633	none	0.12012012012012012	0.12285012285012285
SVM	35.261393785476685	none	0.9997597597597597	0.8247338247338247
kmeans	26.77710747718811	0.042468223720788956	0.6072072072072072	0.579033579033579
dbscan	23.04601550102234	0.40913838148117065	none	none

Figure 5.1: FINAL RESULTS