Report: Indian Pines

Noor Ahmed
noorahmedds@gmail.com

I. Method overview

A. Introduction

The overall pipeline involves reading of corrected Indian Pines Dataset. Preprocessing attempted was standardization and PCA reduction. The libraries in primary use were Sklearn and matplotlib for visualisation. Note: The main metric used for validation was Homogenity score between the ground truth labels and retrieved clusters.

II. Parameter Values

In this section I will justify all parameter values as well as the chosen methods

A. Clustering Method

```
Kmeans++ | 17 classes, Homogenity Score:  0.36617811906988856
Kmeans++ | 17 classes, Completeness Score:  0.36617811906988856
Random initial centroids | 17 classes, Homogenity Score:  0.3653839152662964
Random initial centroids | 17 classes, Completenees Score:  0.3653839152662964
Initialised with PCA Components | 17 componenets | 17 classes, Homogenity Score:  0.3574293018654008
Initialised with PCA Components | 17 componenets | 17 classes, Completeness Score:  0.3574293018654008
Initialised Kmeans++ | Projected Feature Vectors | 17 componenets | 17 classes, Homogenity Score:  0.363176458770896
Initialised Kmeans++ | Projected Feature Vectors | 17 componenets | 17 classes, Completeness Score:  0.363176458770896
6
```

Fig. 1. Methods used and the H-Scores retrieved

Findings, None of the initially chosen clustering methods gave a great boost against simple kmeans. Other experiments were also conducted with dbscan but because its a density based clustering method a lot of the pixels were considered outliers. The '0' class was a major problem for DBScan as that class was not highly separable.

B. TSNE

I also attach TSNE visualisation in Fig. 2 which implements a non linear minimisation to learn separation for given distribution. I experimented with this to see how separable the classes were without any feature extraction.

It is to be noted that if the number of componenets used was to be increased for TSNE the separation would be much more apparent but. TSNE was not used for experimentation as embedding generation took up to 20 minutes on my i5 machine without any acceleration.

C. Number of clusters

In Fig. 3 it can be seen that the homogenity score increases as we increase the number of classes/clusters in kmeans. But around 20 clusters we start to hit diminishing returns. In all further experiments I used 20 as number of clusters as that gave the best compromise between speed and accuracy.

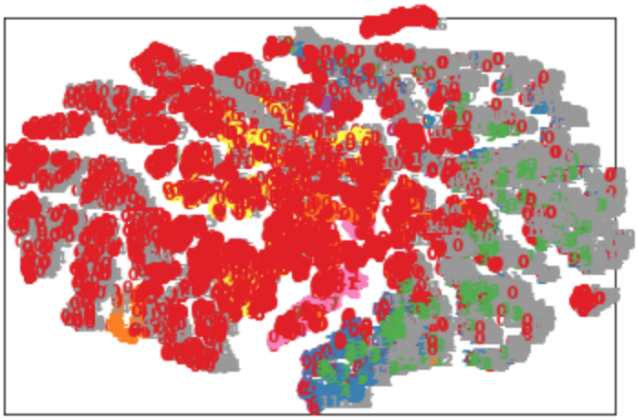

Fig. 2. TSNE generated embeddings with classes written on the plot

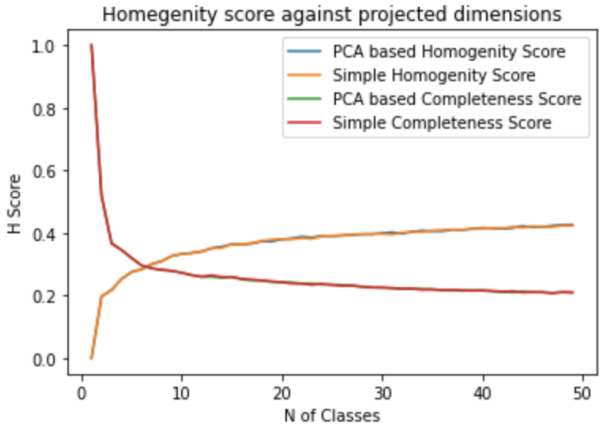

Fig. 3. Homogenity score increases as classes increase but with diminishing returns

D. Number of projected dimensions to reduce to: PCA

Fig4. contains experiment visualisation. Number of projected dimensions was another parameter that I needed to search over. I did a simple linear search over several values. And found that I got highest H-score at 30 dimensions. So i chose that as the parameter value.

E. Effect of Preprocessing

Fig. 5 shows the effect of standardizing input data.

F. Reference Papers and Implementations

(https://github.com/nshaud/DeepHyperX) here i found standardization technique i used

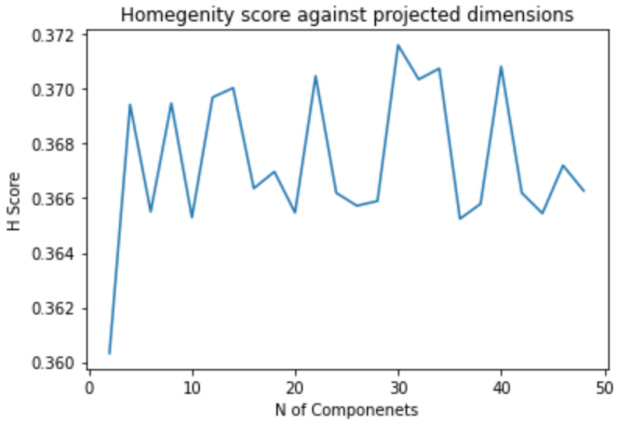

Fig. 4. Number of PCA Components to project the data onto for preprocessing

```
No Preprocessing Score:  0.36034101456045853
Standardization:  0.3818534828658725
```

Fig. 5. Effect of standardization of H-Score

(https://github.com/gokriznastic/HybridSN) here i confirmed the optimality of the number of reduced dimensions. This repository also uses 30 dimensions to reduce the dimensions of their input data

G. Final Results

Fig 6 shows final results. Note: The colors are assigned randomly to the clusters/classes. When compared to the ground truth the structural integrity is maintained and both images look very texturally similar.

IMAGE CREATED BY MY METHOD

```
In [21]: 1 show_cluster_img(estimator.labels_)
```

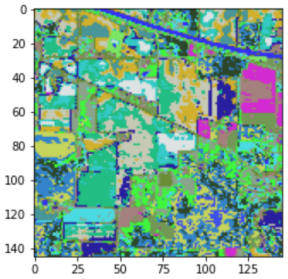

IMAGE CREATED BY FROM THE ORIGINAL GROUND TRUTH LABLES

```
In [19]: 1 show_cluster_img(labels)

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
(145, 145, 3)
```

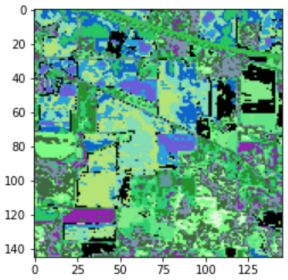

Fig. 6. Final Result