CIS 6930: Introduction to Data Mining

Final Project Report

Submitted by

Unmey Mahaddalkar (3911 9642)
Tejas Sahasranaman (5521 8434)
Rushikesh Gawali (8330 4331)
Joel Menezes (9391 3491)

MULTISPECTRAL IMAGE IDENTIFICATION USING CLUSTERING ALGORITHMS

1) Introduction to Project Background and Purpose

The electromagnetic spectrum has been divided into diverse descriptive regions as infrared, ultraviolet and visible (red, green and blue) rays. Each region is classified based on its frequency.

1. Infrared (700nm to 1mm)
2. Ultraviolet (10nm to 380nm)
3. Visible light (380nm to 700nm)

These regions are further divided into spectral bands:

- Near Infrared (750-900 nm)
- Mid Infrared (1550-1750 nm)
- Far Infrared (2080-2350 nm)
- Thermal Infrared (10400-12500 nm)
- Radar
- Blue (450-515..520 nm)
- Green (515..520-590..600 nm)
- Red (600..630-680..690 nm)

Multispectral images allows capture of data images in all these regions and even beyond it. A multispectral image is a collection of various monochrome images captured using different sensors. The primary use of multispectral images lies in remote sensing applications, especially satellites that take several images of the surface being monitored in the visible and non-visible range to classify different regions on the surface.

The goal of our project is to use data containing a series of multispectral images of handwritten numbers between 0 and 9, written by 6 different individuals using two different ink pens-black and blue. The goal is to build various clustering algorithms that will help in differentiating each of the numbers from one another and the ink with which it was written.

2) Literature Survey

James P. Theiler and Gelen Gisler propose a variant of k-means clustering which uses both the spatial and the spectral properties from multispectral images to cluster the images received using remote sensing. These additional insights are received from segmented images which come at the cost of compactness of the clusters in the spectral domain.

S. K. Pal and P. Mitra have applied Expectation Maximization (EM) algorithm and Minimal Spanning Tree (MST) clustering to solve the segmentation of multispectral satellite images. They have realized that the performance of EM can be maximized by using rough-set theory which helps in faster convergence and also helps in avoiding the local minima problem.

H.G. WIlson, B. Boots and A. A. Millward have compared the results of various clustering algorithms on a small Landsat 7 ™ sub-image of the Hainan Province in China. The results of of the clustering procedures show that the results from hierarchical nearest neighbor linkage had the worst accuracy measure whereas the K-means partitional procedure provided the best classification result.

3) Algorithm Description and Implementation

1. DBSCAN Clustering

Algorithm:

DBSCAN algorithm is the most popular density based algorithm. The major steps involved in this algorithm are:

1. Initially the epsilon and minimum points within a cluster (minpoints) values are chosen.
2. The points having more than the minpoints in its neighborhood are identified. These points are called core points and are connected to form a cluster.
3. If any of the non-core points lie within the radius of the cluster, then it's assigned to the cluster, otherwise it is treated as noise.

Complexity:

1. Time complexity is mostly based on the computation of neighborhood points for each core point. The worst case time complexity is $O(n^2)$.
2. Space complexity is based on the type of method used to calculate the distance between two points. Both the methods have their own drawback:
    - If the distance is recomputed for each point, the space complexity decreases. However, the time complexity increases. This method is used in the DBSCAN package.
    - Another way to calculate distance is by generating a distance matrix for all the points. This will increase the space complexity to $O(n^2)$.

Advantages:

- This algorithm partitions the data into clusters but does not require specifying the number of clusters in advance like in k-means algorithm.
- In this technique, a dense region of data points forms a cluster. Density is measured based on the number of data points within a user specified radius.
- The clusters can be of any arbitrary shape unlike k means which forms spherical clusters. Also, since the clusters are dense regions of data points, the outliers (noise) can be easily identified and removed.

Disadvantages:

- It does not work well for high dimensional datasets
- Selecting the parameters-epsilon and minimum points in a cluster-is tricky.
- Identifying the high-density clusters is difficult when the data is much more spread out and random.

Implementation:

1. Initially, the dataset was converted into a list of lists data structure. The epsilon and minpoints values can be specified based on the dataset chosen.
2. For each point in the list, check if the point has been previously visited or not. If it has not been visited, change its status to 'visited' and find the neighbors for this point. The neighbors are found by computing the distance between the given point and the other points which lie within the radius of this point. Euclidean distance was used to calculate the distance between the points.
3. Once the neighbors are determined, check if the neighborhood points are above the minpoints threshold. If yes, then classify it as a cluster and check if this cluster can be extended further. If no, the points are classified as noise points.
4. Performance measures like accuracy, precision, recall, running time and f-score were computed.

2. K-Means Clustering

Algorithm:

1. Select k points among n data points as the initial centroids
2. While the centroids from previous and next iterations are different:
    i. Form k clusters by assigning all points to the closest cluster.
    ii. Recompute the centroid of each cluster.

Complexity:

Time complexity of k-medoids clustering algorithm is O(nKId)

n: number of points

K: number of clusters
I: number of iterations
d: number of attributes

Implementation:

The algorithm was implemented in python and uses the following libraries and its components

Numpy - list data structure to store dataset and to find Euclidean distance.
Copy - deepcopy function to copy the distance matrix for modification
Pandas - Read the csv datafile

3. K-Medoids Clustering

Algorithm:

3. Select k points among n data points as the initial medoids
4. Calculate the closest distances of each data point to each of the chosen medoids and associate it to the closest one.
5. While the cost of the configuration decreases:
    1. For each medoid m, for each non-medoid data point o:
    i.Swap m and o, recompute the cost (sum of distances of points to their medoid)
    ii. If the total cost of the configuration increased in the previous step, undo the swap

Complexity:

Time complexity of k-medoids clustering algorithm is O(k(n-k)2)

Implementation:

The algorithm was implemented in python and uses the following libraries and its components
csv - read the dataset csv and read it into a numpy array
Numpy - list data structure to store dataset , medoids and distance matrix
Copy - deepcopy function to copy the distance matrix for modification
Scipy - calculate the spatial distance between any two points
matplotlib - draw a scatterplot of all clusters in different colors
Datetime - to record the start and end time of the execution

4. SNN Clustering

Algorithm:

1) Compute the similarity matrix.

This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points.

2) Sparsify the similarity matrix by keeping only the k most similar neighbors.

This corresponds to only keeping the k strongest links of the similarity graph.

3) Find the SNN density of each point

Using a user specified parameter,Eps, find the number points that have an SNN similarity of Eps or greater to each point. This is the SNN density of the point.

4) Find the core points

Using user specified parameter, MinPts, find the core points, i.e., all points that have an SNN density greater than MinPts.

5) Form clusters from the core points

If two core points are within a radius, Eps, of each other they are placed in the same cluster.

6) Discard all noise points

All non-core points that are not within a radius of Eps of a core point are discarded.

7) Assign all non-noise, non-core points to clusters

This can be done by assigning such points to the nearest core point

Implementation:

1) The algorithm was implemented in python and uses the following libraries and its components
2) csv - read the dataset csv and read it into a numpy array
3) Numpy - list data structure to store dataset and distance matrix
4) Numpy.copy– deepcopy function to copy the distance matrix for modification
5) Numpy.dot,Numpy.sqrt functions - calculate the spatial distance between any two points
6) Datetime - to record the start and end time of the execution

Complexity:

For a dataset with n records,

Time Complexity : O(n2)

Space Complexity : O(n2)

Advantages:

1) Can estimate the relative density, i.e., probability density, in a region
2) f we use SNN similarity then we can obtain a more robust definition of density
3) SNN Density can identify Core, Border and Noise points

Disadvantages:

1) Space Complexity is O(n2) which hurts especially when dataset is large.
2) Difficult to implement as compared to k-means,k-medoids, etc.

4) Data source and dataset description

Source:
The dataset of Multispectral images was taken from Kaggle which is a platform for predictive modelling and analytics competitions.The link to the dataset is given below.

https://www.kaggle.com/xiaozhouwang/multispectralimages

Description:
In this dataset , six different people were asked to draw numbers from 0 to 9 on similar sheets of paper using two pens of black and blue inks.

Different sensors were used to capture 10 grayscale images of size (350 x 350) of these sheets.To fully exploit the additional information which is contained in the multiple bands, the images are considered as one multi-spectral image rather than as a set of monochrome graylevel images. In this case , for an image with 10 bands, the brightness of each pixel as a point in a 10-dimensional space is represented by a vector of length 10.

The dataset has 713 csv files containing 122500 rows of multispectral image each of which represents one pixel of a 'colored digit' (for example blue 1 or black 4 etc)

In each file , the columns represent pixels for 10 grayscale images (350 x 350) that represent 10 channels for the multispectral image, where X, Y represent the location of the pixel, and channel0 - channel9 represent channels. There is a labels csv as well that contains labels for each pixel csv file.

5) Data cleaning and pre-processing

Cleaning:
Rows with empty column values were discarded.

The X and Y location parameters range from 0 to 349
The attributes channel0 - channel 9 range from 0 to 255.

No further normalization is required as all values have a fixed range and therefore one feature does not have a major influence over other column features on the similarity calculation and clustering process.

Pre-processing:
All csv files were combined into a single csv file containing all rows of all multispectral images with their ground truth label appended as the additional column.

The label column is converted into a numeric categorical variable which maps an integer to the ground truth label (for example , blue0 to 1 , blue1 to 2 .. black9 to 20)

A proportionate sample in the scale of 10,000 to 1 million was taken to suit our implementations of different clustering algorithms according to the amount of data they could process.

The performance and coverage of each of those algorithms are compared and contrasted in a later section.

6) Experimental Results and Performance Comparison

The individual performance metrics for each implemented algorithm is given below

DBSCAN Algorithm

Number of records	Running Time(in seconds)	Accuracy	Average Precision	Average Recall
2500	89.5821	10.5981	0.01774	0.09921
5000	423.8974	9.4668	0.01859	0.06894
9000	2058.133	8.8621	0.01754	0.06611

K-Means Algorithm

Number of records	Running Time(in seconds)	Accuracy	Average Precision	Average Recall
10000	4.57	15.078	0.3391	0.1011
50000	22.87	16.791	0.3412	0.09921

122500	167.135	17.231	0.3337	0.06894

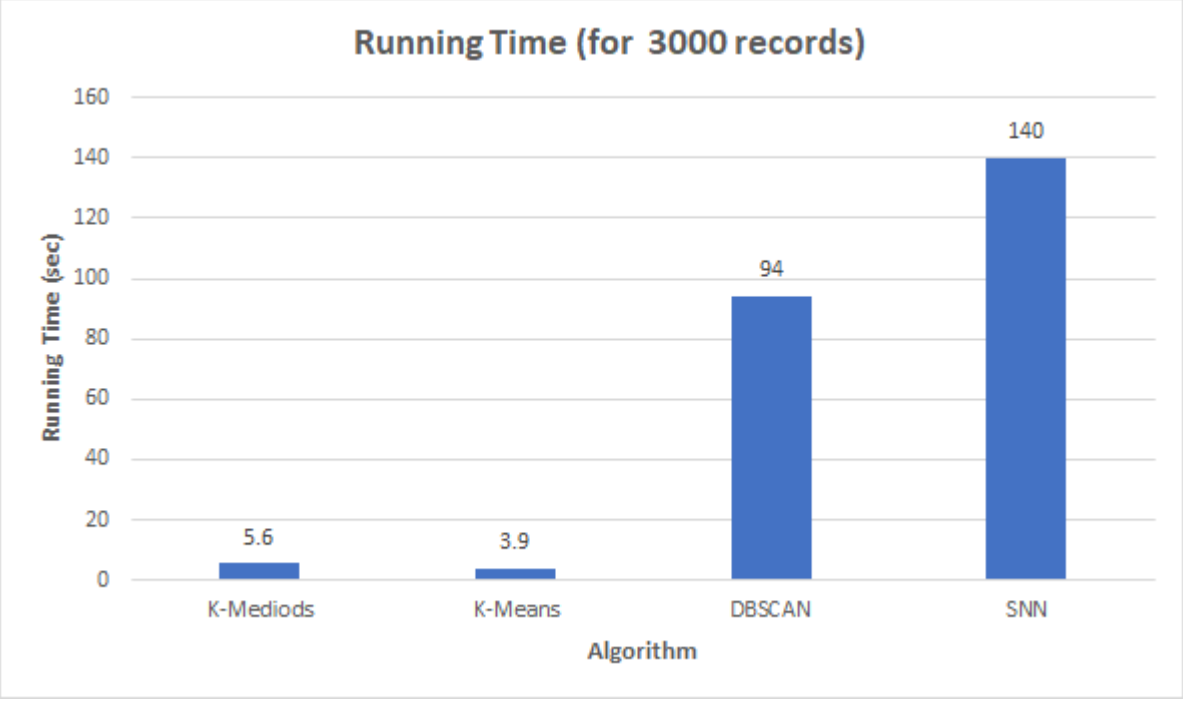
K-Medoids Algorithm

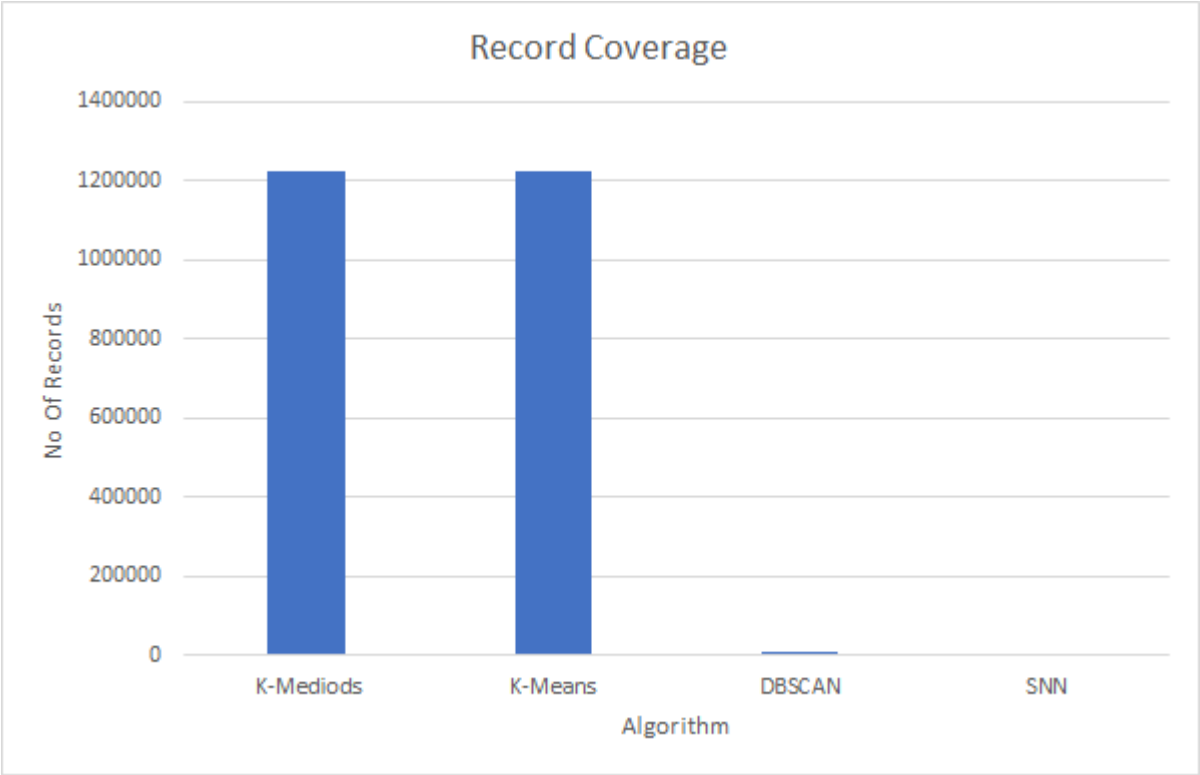
Number of records	Running Time(in seconds)	Accuracy	Average Precision	Average Recall
10000	5.634	23.356	0.03463	0.09453
50000	23.456	12.565	0.01298	0.10023
122500	46.201	5.451	0.04547	0.03575

4. SNN Algorithm

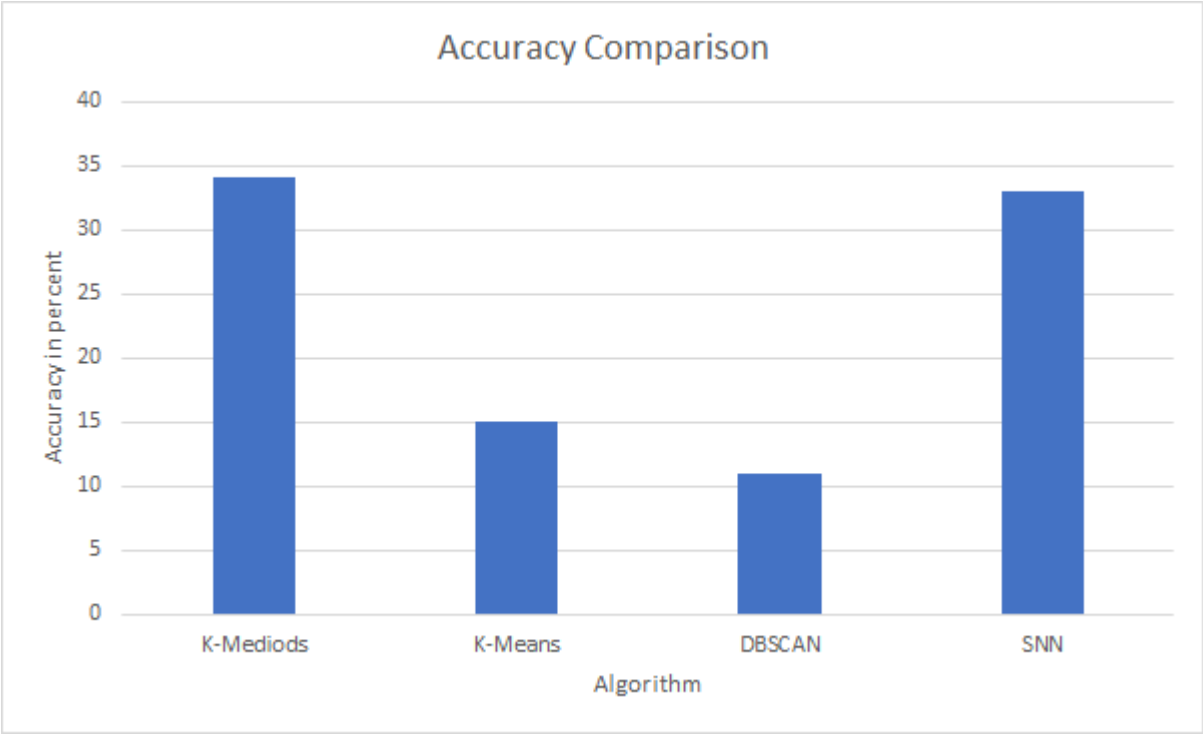
Number of records	Running Time(in seconds)	Accuracy	Average Precision	Average Recall
1000	14.382	35.832	0.362	0.115
2000	60.689	34.339	0.343	0.108
3000	140.823	33.377	0.333	0.098

Since SNN clustering algorithm could only scale a maximum of 3000 records . The performance comparison has also been done for the same amount of data .





For Max 3000 records



7) Limitations

1) During our implementation we found out that the algorithms or our implementation of these wasn't very accurate. We would have liked higher accuracy.
2) Since we were implementing these algorithms on our local machines, the space limitations hurt us tremendously, especially for algorithms like SNN and DBSCAN which require O(n2) space where n is the number of records in the dataset.
3) Also because of the limited processing capability of our machines, the execution time for algorithms was quite high.

8) Potential Future Improvement

1) We can work on improving the accuracy of our implementation of these algorithms.
2) We can also improve upon the time and space requirements.
3) We can implement other clustering approaches like CURE and BIRCH to tackle the data size and improve accuracy.

9) References:

[1] CLAUSI, D.

K-means Iterative Fisher (KIF) unsupervised clustering algorithm applied to image texture segmentation

In-text: (Clausi, 2002)

Your Bibliography: Clausi, D. (2002). K-means Iterative Fisher (KIF) unsupervised clustering algorithm applied to image texture segmentation. *Pattern Recognition*, 35(9), pp.1959-1972.

[2] LIANG, B. AND ZHANG, J.

KmsGC: An Unsupervised Color Image Segmentation Algorithm Based onK-Means Clustering and Graph Cut

In-text: (Liang and Zhang, 2014)

Your Bibliography: Liang, B. and Zhang, J. (2014). KmsGC: An Unsupervised Color Image Segmentation Algorithm Based onK-Means Clustering and Graph Cut. *Mathematical Problems in Engineering*, 2014, pp.1-13.

[3] ., K. V., ., P. A. P., ., R. M. AND ., S. M.

Multispectral Image Clustering Using Enhanced Genetic k-Means Algorithm

In-text: (. et al., 2007)

Your Bibliography: ., K., ., P., ., R. and ., S. (2007). Multispectral Image Clustering Using Enhanced Genetic k-Means Algorithm. *Information Technology Journal*, 6(4), pp.554-560.

[4] MOREIRA, G., SANTOS, M. Y., PIRES, J. M. AND GALVÃO, J.

Understanding the SNN Input Parameters and How They Affect the Clustering Results

In-text: (Moreira et al., 2015)

Your Bibliography: Moreira, G., Santos, M., Pires, J. and Galvão, J. (2015). Understanding the SNN Input Parameters and How They Affect the Clustering Results. *International Journal of Data Warehousing and Mining*, 11(3), pp.26-48.

[5] MULTISPECTRAL IMAGE CLASSIFICATION | KAGGLE

In-text: (Kaggle.com, 2017)

Your Bibliography: Kaggle.com. (2017). *Multispectral Image Classification | Kaggle*. [online] Available at: https://www.kaggle.com/xiaozhouwang/multispectralimages [Accessed 12 Dec. 2017].

[6] VANDERPLAS, J.

In Depth: k-Means Clustering | Python Data Science Handbook

In-text: (VanderPlas, 2017)

Your Bibliography: VanderPlas, J. (2017). *In Depth: k-Means Clustering | Python Data Science Handbook*. [online] Jakevdp.github.io. Available at: https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html [Accessed 12 Dec. 2017].