George Mason, gtrivell, G00946692  
CS-484-001, Instructor: Prof. Jessica Lin  
HW3 Clustering Report  
Miner (Team) ID: gtrivell  
Iris Dataset Miner Rank: 2, Iris Dataset Miner Public V-Score: 1.00  
Image Dataset Miner Rank: 10, Image Dataset Miner Public V-Score: 0.60

**Preprocessing**For the iris dataset, I first attempted iterations without any preprocessing. Attempting to add a variance threshold resulted in worse scores. However, when adding a normalizer with variance thresholding, it resulted in 1.00.

For the image dataset, I went through the same process of attempting it without any adjustments, then adding variance threshold, and lastly trying with a normalizer. As a last resort, I used TSNE and got a better result than variance threshold.

**Clustering Approach**

Bisecting K-Means that chooses the next centroid as the point furthest from the current centroid to handle empty clusters.

Process inputted file into a single cluster (A nested array)

While there are less than K clusters:

Get the cluster with the highest SSE (In the first run, it would just be the whole set)

Start to bisect this cluster

While the SSE of the bisected clusters continues to improve (Additionally, iterates up to X times even when not improving to ensure that a lower SSE can’t be achieved):

Get two different random points in current cluster. If the first run, start with the points furthest from the center

Distribute the points in the cluster to these new temporary centers. If the overall SSE is lower than the record, save the iteration

Add the new clusters to the list of clusters

Readjust all points to see if any of the new centers are a better fit

For each line in the data, add the index of the cluster in which the line’s data ended up in

**Initialization**: After running either HW3.py or HW3Image.py, follow console instructions to input the test file name. HW3.py assumes K of 3 and HW3Image.py assumes K of 10.

**Distance Metric**: Scikit-learn’s mean\_squared\_error. However, beyond measuring the distance between points with this, calculations were done manually.

**Feature Selection/Dimension Reduction**

I noticed there were only 4 features in the Iris dataset and was curious if one could be irrelevant, so I removed it using variance thresholding.

Introducing VT to the image dataset, even with a value of 0, resulted in removing features. This told me that useless features are present that could be skewing K-means, so I gradually upped the strength until results were satisfactory. However, results were not satisfactory, so I tried TSNE.

**Evaluation Metrics**

The average SSE of the clusters resulting from the last bisection.

**Results & Conclusions**

Iterations is the X value mentioned previously.

**Iris Testing:**

Without Preprocessing

|  |  |
| --- | --- |
| Iterations | Accuracy |
| 20 | 0.73 |
| 100 | 0.76 |
| 1000 | 0.75 |

Even though iterations increase, accuracy stays inconsistent. This reflects my choice to randomly select centers to test.

Variance Threshold Added

|  |  |  |
| --- | --- | --- |
| Iterations | Feature Amount | Accuracy |
| 100 | 3 | 0.67 |
| 1000 | 2 | 0.69, 0.60 |

From here, variance threshold was removed, and I improved my algorithm to select better clusters. Cluster selection now only starts counting amount of iterations once the SSE stays constant. The counter resets if it encounters a lower SSE.

With New Algorithm

|  |  |
| --- | --- |
| Iterations | Accuracy |
| 200 | 0.65 |
| 500 | 0.75 |
| 1000 | 0.75 |

Adding a normalizer here raised the accuracy to 0.95.

Reintroducing the variance threshold got the accuracy to 1.00.

**Image Testing:**

Without Preprocessing

|  |  |
| --- | --- |
| Iterations | Accuracy |
| 20 | 0.22 |
| 150 | 0.23 |

20 iterations took 7 minutes while 150 iterations took 44 minutes

Variance Threshold Added

|  |  |  |
| --- | --- | --- |
| Iterations | Feature Amount | Accuracy |
| 20 | 649 | 0.27 |
| 20 | 667 | 0.21 |

With New Algorithm + Variance Threshold

|  |  |  |
| --- | --- | --- |
| Iterations | Feature Amount | Accuracy |
| 20 | 668 | 0.40 |
| 100 | 668 | 0.28 |
| 30 | 649 | 0.28 |

Results still wildly inconsistent due to random selection with a large dataset. It is hard to tell whether reducing features or adjusting iteration count is the correct choice.

Normalizer Added

|  |  |  |
| --- | --- | --- |
| Iterations | Feature Amount | Accuracy |
| 50 | 668 | 0.46 |
| 50 | 649 | 0.40 |
| 25 | 608 | 0.45 |
| 1000 | 668 | 0.44 |

TSNE Added

|  |  |  |
| --- | --- | --- |
| Iterations | Feature Amount | Accuracy |
| 15 | ALL | 0.56 |
| 40 | 668 | 0.60 |

Variance thresholding was removed for the first run and reintroduced for the last run. Additionally, the TSNE algorithm was adjusted to be exact in the last run, sacrificing performance for results. It took about an hour. (To save time when grading, the line [LineHolder = TSNE(method = 'exact').fit\_transform(LineHolder)] can be changed to [LineHolder = TSNE().fit\_transform(LineHolder)])

I didn’t initially have my algorithm choose a center by finding the points furthest from the current center because I was afraid that this would result in the outliers being chosen. However, at the time, I also did not have variance thresholding. To substantially speed up my program for large datasets, I would choose some of the furthest points from the current center to test rather than random points so that iterations could be very low. I also think that if the program ran long enough with high enough iterations and the correct number of features, it would eventually be accurate, but this is not realistic.