CSE6250: Big Data Analytics in Healthcare
Homework 3
Rajesh Pothamsetty

2.3.b K-means Clustering

Percentage Cluster	Case	Control	Unknown
Cluster 0	0%	0%	5%
Cluster 1	91%	82%	73%
Cluster 2	9%	18%	22%
	100%	100%	100%

Table 1: Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 0	95%	0%	29%
Cluster 1	0%	100%	70%
Cluster 2	5%	0%	1%
	100%	100%	100%

Table 2: Clustering with 3 centers using filtered features

2.4.b Clustering with Gaussian Mixture Model (GMM)

Percentage Cluster	Case	Control	Unknown
Cluster 0	5%	6%	24%
Cluster 1	0%	0%	3%
Cluster 2	95%	94%	73%
	100%	100%	100%

Table 1: Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 0	3%	0%	38%
Cluster 1	16%	100%	35%
Cluster 2	80%	0%	27%
	100%	100%	100%

Table 2: Clustering with 3 centers using filtered features

2.5 Clustering with Streaming K-Means

2.5.a

- Streaming Kmeans is very much needed when there is huge amount of data or a continuous stream of data.
- When the source of the data is changing over time, kmeans algorithm may not be effective if the underlying properties of the data also changes with time. Forgetfulness an extended parameter helps to balance out the relative importance of new data versus the history.
- In our problem, we used a decay factor of 1, which is to treat the entire data from the beginning equally. If the properties of the data changes it is required to change/tune decay factor as well. This disadvantage can be overcome an extra parameter called half-life.
- **Update Rule**: Consider simple weighted average of mean of two sets of numbers, sum of product of mean and count of numbers. Only difference is adding relative importance of the two sets of numbers given by α $(decay\ factor)$.

$$C_{t+1} = \frac{C_t n_t \alpha + x_t m_t}{n_t \alpha + m_t}$$

$$n_{t+1} = n_t + m_t$$

$C_{t+1} - center\ of\ cluster\ including\ new\ batch$
$C_t - center\ of\ existing\ cluster$
$n_t - number\ of\ data\ points\ in\ existing\ cluster$
$m_t - number\ of\ data\ points\ in\ new\ cluster$

2.5.c For K = 3

Percentage Cluster	Case	Control	Unknown
Cluster 0	11%	9%	47%
Cluster 1	8%	63%	17%
Cluster 2	80%	28%	36%
	100%	100%	100%

Table 1: Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 0	10%	0%	2%
Cluster 1	4%	100%	70%
Cluster 2	86%	0%	28%
	100%	100%	100%

Table 2: Clustering with 3 centers using filtered features

2.5.b For different values of K (Streaming K means)

k	Streaming K-Means All Features	Streaming K-Means Filtered Features
2	0.57863	0.64402
5	0.62148	0.88651
10	0.59707	0.88341
15	0.68682	0.88306

Observation: For k = 3, streaming Kmeans outperformed Kmeans and GMM in the All feature case. Since StreamingKmeans processed the data in batches, it has a good initialization compared to Kmeans and GMM given the number of iterations. It has better class division/purity with high dimensional data. In the Filtered data case it is in the same boat as the kmeans and GMM given the number of iterations.

Self-Note: Spark SetRandomCenters initializes centers based on a N(0,1) distribution, this may not be very ideal depending on the data set.

2.6 Discussion on K-means and GMM

2.6.a

Observation: Case and Control classes similarly distributed in Kmeans and GMM**2.3.b**: Filtered data has clear division of the classes Case and Control which implies more defined features. As expected Unknown is slightly spread out than the other two classes but significantly aligned with Control class.

2.4.b : Due to the soft clustering property of GMM with Filtered data Unknown class is spread out across clusters, maybe with more iterations we could see a clear inclination.
Similar to Kmeans in the division of Case and Control classes.

2.6.b

k	K-Means All Features	K-Means Filtered Features	GMM All Features	GMM Filtered Features
2	0.47381	0.66092	0.47381	0.39324
5	0.64642	0.89134	0.53254	0.87133
10	0.59490	0.88375	0.66676	0.86375
15	0.69008	0.88962	0.67489	0.89169

GMM in general should outperform Kmeans due to its soft clustering property and no restrictions on the shape of the clusters. As K increases GMM could outperform Kmeans due to soft assignment property.
Since k means can handle higher dimensional data, it should be used as a method to initialize GMM.

Observation:

1. With the number of clusters purity increased for Kmeans and GMM as expected.
2. Filtered data has higher purity than All feature data case as expected.
3. GMM has the problem of center initialization with high dimensional data, this could be the reason we don't see a monotonic positive difference in the purities.

3. Advanced phenotyping with NMF

3.b. For different values of K

k	NMF All Features	NMF Filtered Features
2	0.49268	0.54467
3	0.49485	0.54329
4	0.49431	0.54329
5	0.49376	0.54536

Observation:
Stable purity values: NMF is using a matrix reduction method to cluster points, which is to say as long as V(input) isn't singular, purity values are deterministic of V. Hence, we observe purity values are stable across K values compared to other models we have seen. This needs to be further studied.

3.c

Percentage Cluster	Case	Control	Unknown
Cluster 0	94%	81%	93%
Cluster 1	1%	2%	1%
Cluster 2	5%	17%	6%
	100%	100%	100%

Table 1: Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 0	96%	0%	35%
Cluster 1	1%	100%	53%
Cluster 2	3%	0%	12%
	100%	100%	100%

Table 2: Clustering with 3 centers using filtered features

Observation:
With All feature case, all the data points are grouped to one cluster. Reasons could be lesser K value, the noise due to high dimensionality of the data and lower number of iterations for convergence. With the filtered data, the data is grouped in separate clusters for Case and Control classes as we seen in Kmeans and GMM, as expected.

Point to remember(self-note)
NMF has received much attention due to its straightforward interpretability for applications, i.e., we can explain each observation by additive linear combination of nonnegative basis In the Principal Component Analysis (PCA), to the contrary, interpretation after lower rank approximation may become difficult when the data matrix is nonnegative since it allows negative elements in the factors.
https://www.cc.gatech.edu/~hpark/papers/GT-CSE-08-01.pdf and GT-CSE-08-01.pdf

3.d. NMF Multiplicative Update Rule

Euclidean distance between two non-negative matrices can be considered as cost function.

Cost function (J) = $||V - WH||^2$

('W' as Centroid Matrix and 'H' as a matrix with 1 non-zero value per row(hard clustering), the above cost function is similar to Kmeans)

Proof to why we can use MU rule to minimize cost function J. We will attempt to understand it's viability from the point of known optimization method the gradient descent.

1. The multiplicative factor ($\frac{W^TV}{W^TWH}$) is unity when V = WH, so if we can design the descent method keeping this in mind.
2. Differentiate the cost function w.r.t H gives

   Gradient = $2W^2H - 2WV$

   In the matrix format

   $\cong -(W^TV - W^TWH)$
3. An additive update that can reduce cost function can be

$$H_{t+1} = H_t - \eta(W^TWH - W^TV)$$

$$H_{t+1} = H_t + \eta(W^TV - W^TWH)$$

4. For sufficiently small values of η, the above update can reduce $||V - WH||^2$, hence the above equation can be seen as a equivalent to gradient descent.

5. Rescale the variable diagonally and set $\eta = \frac{H_t}{(W^TWH)_t}$, we can get to the multiplicative update rule which is $H_{t+1} = H_t \frac{W^TV}{W^TWH}$

6. Similarly, it can be used to prove the use of MU rule for W update as well.