527-Final-Report

Chris Chen

2023-06-03

In this Project, clustering on a dataset of 12000 observations (each with 64 features) is done. Two machine learning clustering methods were used and the results were compared and evaluated. Since it's unsupervised learning, we do not have the "one true answer", hence the goal is set to be **having clusters that are as clear as possible**. Additionally, due to realistic constraints (I have two other ongoing projects, one on disease prediction based on microbiome data using ML, the other on optimization using self-developed packages, both require huge amount of R computing time), **fast computation time** was set to be the secondary goal.

1. Preprocessing

A bit of data exploration reveals that all features have approximately 0 mean but they have different ranges. The distributions are approximately normal except for the first two features. No obvious outliers for any of the features.

Dimension reduction was done–the 64-dimension data was reduced to 2-dimension. This was done because of two reasons:

- Theoretically, there are no neighbors in high dimensionality–Euclidean distance breaks
- Realistically, my device (MacBook Pro with macOS Big Sur Version 11.6.1) and software (R version 4.3.0) was unable to run the clustering algorithms on data with dimension higher than 3: "Vector memory exhausted"

In the end, 2 dimensions were selected because the computation of Level-set (one of the selected clustering methods) is only tractable for 1 or 2 dimensions. 2 dimensional data also makes the visualization easier and makes the results more interpretable.

Among all the dimension reduction methods, Principal Component Analysis (PCA), Independent Component Analysis (ICA), t-distributed Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP) were considered due to their wide application range and package avaliability. Recall that our objective is to obtain clusters that are as clear as possible, we select the dimension reduction method based on these following criteria:

- It makes logical sense to apply this method
- It gives clear clusters in 2D

PCA was done first. The result produced was not good. We don't know if it's logically sound to apply it, since PCA assumes that there's a linear relationship between features. The variance of the first two Principal Components are not too much different from the rest. Moreover, the plot of the data is crowded–no clear clusters can be observed (see Figure 1).

ICA was designed to do signal identification, hence assumes that none of the features are gaussian. However, we've seen that about 62 out of 64 features are somewhat Gaussian, hence it's not logically sound to apply it. Also, the plot produced is crowded–no clear clusters, just like that of PCA (see Figure 2).

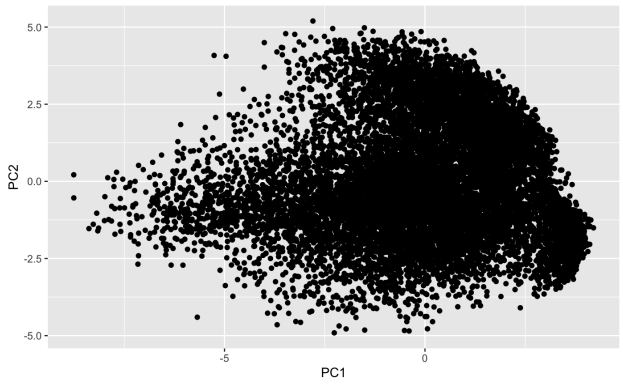

Figure 1: PCA Dimenstion Reduction Result

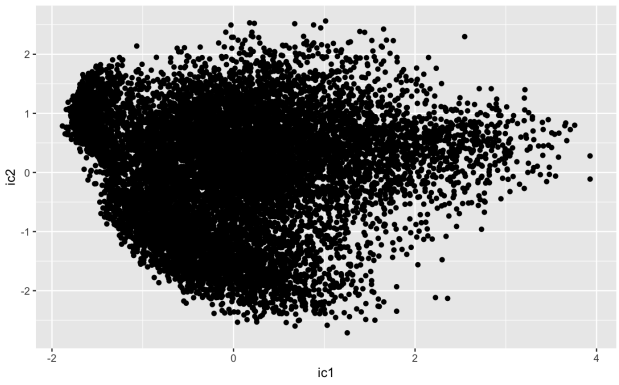

Figure 2: ICA Dimenstion Reduction Result

t-SNE and UMAP are very commonly used on RNA data due to their superior performance. However, there are some reasons that make UMAP better than t-SNE for our set of goals. First is from a theoretical reason: t-SNE only retains local structures, due to the assumption and construction of the distance between two points to be proportional to probability density under a Gaussian centered at one of them, while UMAP can retain global structures. Secondly, application-wise, UMAP allows usage of manhattan distance, which is better for high dimensional data. Lastly and realistically, t-SNE is slower (package Rtsne) than UMAP (package uwot). Arguably, t-SNE can preserve some global structure at very large perplexity values, but the computation time at such large values can be astronomical on my device, making UMAP a better choice for us.

The result produced by t-SNE and UMAP can be seen in Figure 2. t-SNE (see Figure 3) gives relatively clear clusters, but the distance between clusters are quite small, probably due to the fact that the global structure is not retained. UMAP (see Figure 4) gives very clear clusters. It seems that there are approximately 6 clear clusters, maybe 8 (if we treat the upper middle cluster and the upper right cluster as four clusters) not-that-clear clusters.

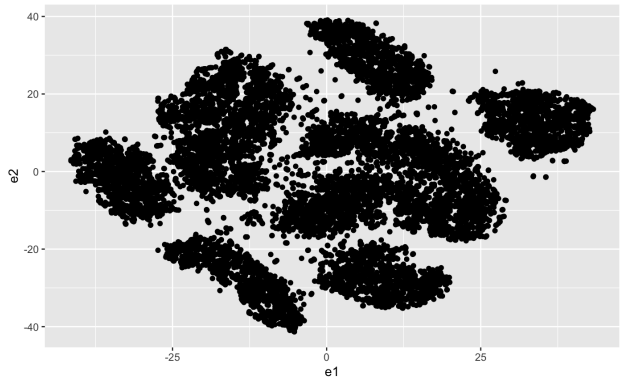

Figure 3: t-SNE Dimenstion Reduction Result

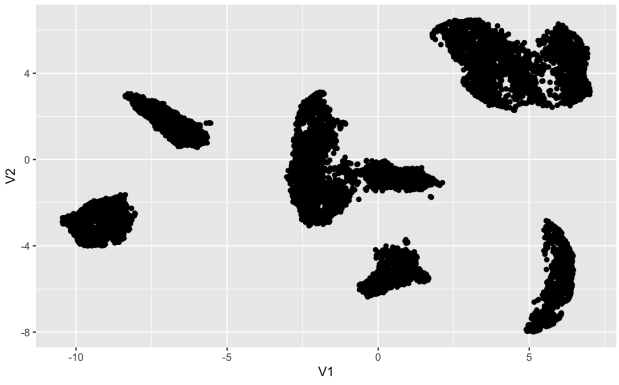

Figure 4: UMAP Dimension Reduction Result

2. Methods Description

2.1 Mean Shift

Clustering methods used in this project are Mean Shift and Level-set with Kernel Density Estimator. Intuitively, Mean Shift is essentially a mode-finding process. The empirical density of the data is estimated using a Kernel Density Estimator (KDE) first, requiring a dataset and specification of the kernel and the

smoothing parameter h–the bandwidth; then for each data point we shift it in a direction that has higher density (a region with more points) until convergence–it arrives at a local maximum; and lastly, all the points shifted to the same peak will be assigned to the same cluster. Each iteration in the shifting procedure is basically calculating a weighted average of all the points with the weights proportional to their densities and inversely proportional to their distances from this point.

2.2 Level-set with KDE

Level-set with Kernel Density estimator has the same first step as Mean Shift–they estimate the data density with KDE. Then a set of level curves, ranging from 0 to the supremum of the density, are selected; the selection criteria can differ–for example, some require the level curves to be evenly spaced while others may require the proportions of points under each level curve is evenly spaced. Third, find the level sets corresponding to each of the level curves, with the level set defined as the set of points with density higher than that level curve. Lastly, if the level set is disconnected, then each connected component is in one cluster.

2.3 Similarity and Difference

These two methods are very similar in their clustering procedure. Both algorithm first estimate the density of the data using a Gaussian kernel, then select certain peaks in the empirical data density curve as clusters, and lastly, points are assigned to clusters, giving us the final clustering.

They differ by some extent in the second and last stages. In Mean Shift, the number of clusters equals to the total number of isolated peaks in the empirical data density curve, and each point is assigned to the peak they are under. The isolated peaks are essentially the modes, the local maxima with gradient 0, and hence are found by steepest ascent. On the other hand, in Level-set, the number of clusters is more complicated due to the fact that points can be unclustered–if a point's density is lower than a threshold (level curve), it's unclustered. If each of the unclustered points are treated as a singleton cluster, then the number of clusters equals the number of isolated peaks with densities higher than the threshold plus the number of unclustered points; if each of the unclustered points are sent to its nearest cluster, then the number of clusters equals the number of isolated peaks with densities higher than the threshold.

3. Clustering Strategy and Parameter Choices

3.1 Initial Attempt

First, the two methods were run without any parameter tuning. For Mean Shift, 191 clusters were obtained (See Figure 5). The result is very chaotic, we have too many clusters, and there are no boundaries between clusters. For Level-set, 9 clusters were obtained (See Figure 6), and the result actually look pretty good already. Except that the two rightmost clusters have unexpected cuts in them, other points are clustered very well. But clearly, the result we currently have can be improved, especially for Mean Shift. We need to do parameter tuning.

3.2 Mean Shift Parameter Tuning

In the Section 2.1, we mentioned that Mean Shift requires a dataset, a kernel, and a smoothing parameter h. Kernel correspond to the parameter "kernelType" and h correspond to the parameter "bandwidth" in the function meanShift(). Additionally, this function has "alpha" to control if the mean shift algorithm will be approximated through Newton's Method, "iterations" to control number of times to perform the shifting procedure, "epsilon" to control when to terminate the procedure based on difference between consecutive iterations, and "epsilonCluster" to control the minimum distance between distinct clusters. "alpha" needs to be set to 0, because we don't want approximation using Newton's method. "iterations" and "epsilon"

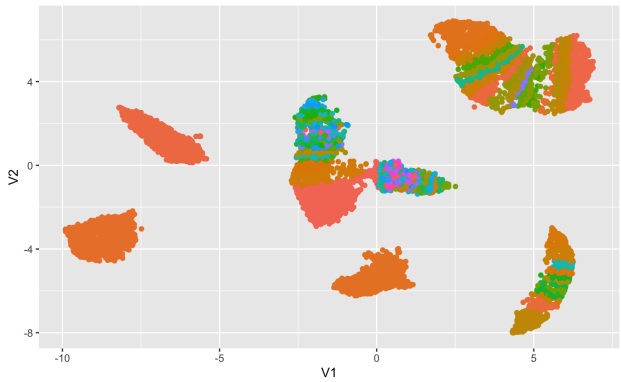

Figure 5: Mean Shift Result with Untuned Parameters

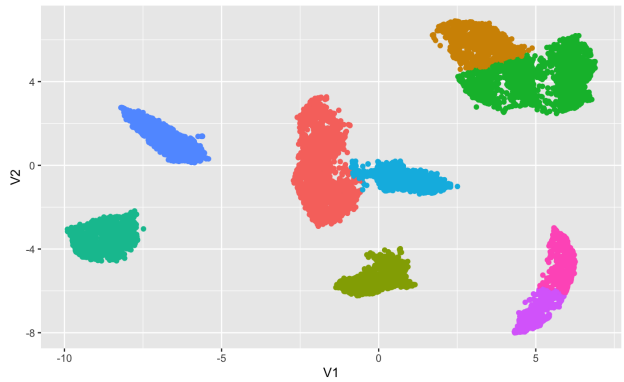

Figure 6: Level-set Result with Untuned Parameters

control the stopping condition, so modification of one would suffice. "iterations" was chosen because it's more straightforward to understand. "epsilonCluster" was not modified because the distance between distinct clusters is not meaningful in our case–UMAP does not retain any distance information.

- "kernelType" was set to be Gaussian
- iterations was set to 100, because after a few trials we found that the clustering is guaranteed to converge at 100 iterations, the total number of clusters converged to 6, and runtime converged to about 60 seconds. This is valid for h ranging from 0.5 to 5. Figure 7 illustrate one such example with Gaussian Kernel and bandwidth set to (1, 1).
- "bandwidth" was tuned to satisfy our goal: we want to have clusters that are as clear as possible. A metric called "silhouette score" was used reflect "clearness" of clustering

Silhouette is calculated as follows: For $i \in C_I$,

$$s(i) = \frac{b(i) - a(i)}{max\{b_i, a_i\}}$$

where

$$b(i) = min_{J \neq I}\{\frac{1}{|C_J|}\sum_{j \in C_J} d(i,j)\}$$

and

$$a(i) = \frac{1}{|C_I| - 1}\sum_{j \in C_I, i \neq j} d(i,j)$$

Note that when $|C_I| = 1$, $s(i) = 0$.

Silhouette score range from -1 to 1, high positive value means clean, appropriate clustering, 0 means crowded, or no clean clusters, and negative values means inappropriate clustering. "Appropriate" is defined as "all points belong to the clusters they are in right now and does not belong to other clusters", and inappropriate is defined as "some points does not belong to the cluster they are in right now, but belong to some other clusters".

However, a complication is involved: silhouette score computation requires distances, but our dimension reduction method does not preserve distance in the original data. If we choose to use distance in the original data, the problem is we are not sure if the clustering result would be the same using the original data; if we choose to use distance in dimension-reduced data, then the problem is the distances are not meaningful. If the technology allows, the **correct** way to do it would be using original data to calculate the distance AND do the clustering, then the silhouette score would be meaningful and can be used to tune h.

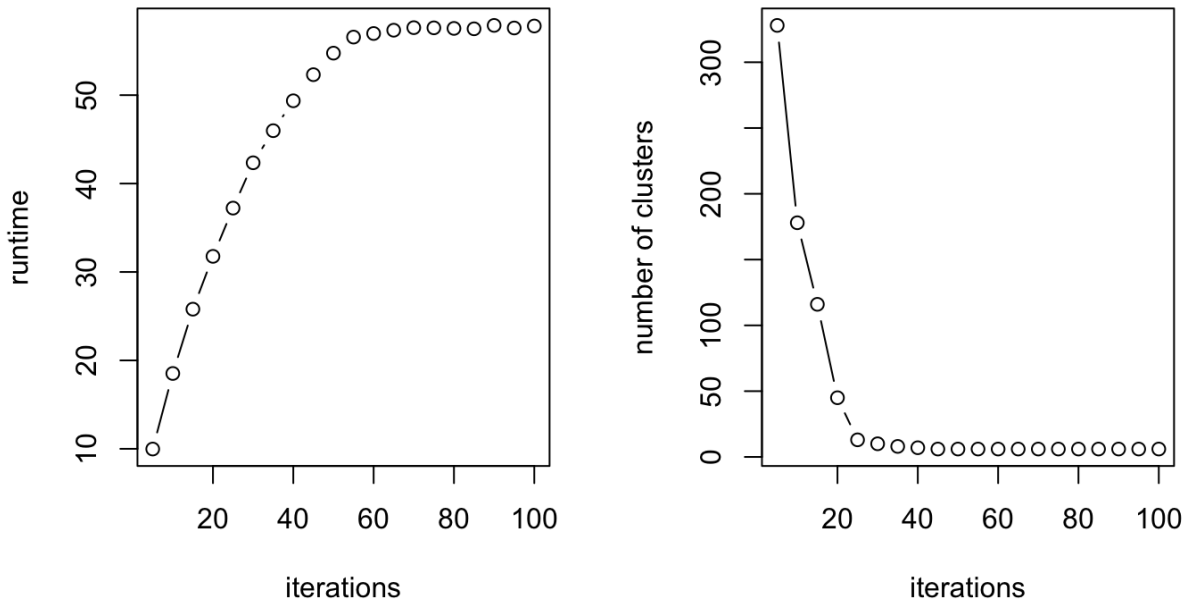

Figure 7: Convergence in Runtime (in seconds) and Number of Clusters of Mean Shift Algorithm over iterations with Gaussian Kernel and bandwidth (1, 1)

kernelType	iterations	bandwidth
Gaussian	100	(1.2, 1)

Table 1: Tuned Parameter of Mean Shift (meanShift)

All other metrics that reflect "clearness" of clustering involves distance calculation, making all of them suffer from this complication. Therefore, silhouette score was still used to tune h, but a "double criteria" was used to "fix" this problem to some extent: our objective in parameter tuning becomes:

- maximization of silhouette score in original dataset, using the clusters obtained from dimension-reduced data
- maximization of silhouette score in dimension-reduced data, since it would still be helpful to our 2-D visualization

In the end, h was tuned to be (1.2, 1). The summary of the tuned parameters is provided in the following table 1. The clustering result of meanShift using these parameters is presented in Figure 9.

3.3 Level-set with KDE Parameter Tuning

As discussed in Section 2.2, Level-set takes in a kernel, a smoothing parameter h, and a set of level curves. Kernel correspond to "kernel", h correspond to "h", and level curves correspond to "n.grid" parameter in the function pdfCluster. Additionally, this function takes in an "hmult" parameter, which is a shrink factor that is multiplied to h used in the KDE.

A noteworthy point is how pdfCluster deal with level-sets. "n.grid" is actually the total number of level-sets.

By default, it's calculated as

$$n.grid = \lfloor (5 + \sqrt{n})4 + 0.5 \rfloor$$

, an empirical rule of thumb. Then, the level sets are determined by requiring the proportions of points under each level curve to be evenly spaced using quantiles. For example, for our dataset, $n.grid = \lfloor (5 + \sqrt{12000})4 + 0.5 \rfloor = 458$, so a total of 458 level curves are used. The first (lowest) level curve is set to be the height such that the proportion of points in the data under it to be $\frac{1}{458} = 0.00218$, the second (second lowest) level curve at a height such that the proportion under it is $\frac{2}{458} = 0.00437$, etc. etc., and the 458-th (highest) will have all the data below it. Modifying this parameter mostly changes the running time–because the level-sets are calculated for EACH level curve–but barely changes the clustering result, since as long as there are "enough" level curves–there are curves to cover each of the valleys and peaks, the result does not change. 458 is more than enough. Empirically speaking, we tested that when n.grid is larger than 200 (up to 10000), the clustering result does not change. So, we set this parameter to 200 for faster computation. Additionally, in this function, unclustered points are sent to the nearest cluster, so no singleton clusters.

"hmult" was set to 1–the default is 0.75 for data has less than or equal to 6 dimension. There's no point in changing it and changing h at the same time. However, it is useful in a way that it allows us to observe if we constantly increase h, what would happen. As shown in Figure 8, number of clusters will converge to 1 if we keep increasing h. Note that although the x-label is "h", it is in fact "hmult", because it's better for visualization and more straightforward to understand since h is 2 dimensional (here h is set to (1, 1) for correctness). But keep in mind that h can have different values, here is just a heuristic illustration of the effect of increasing h.

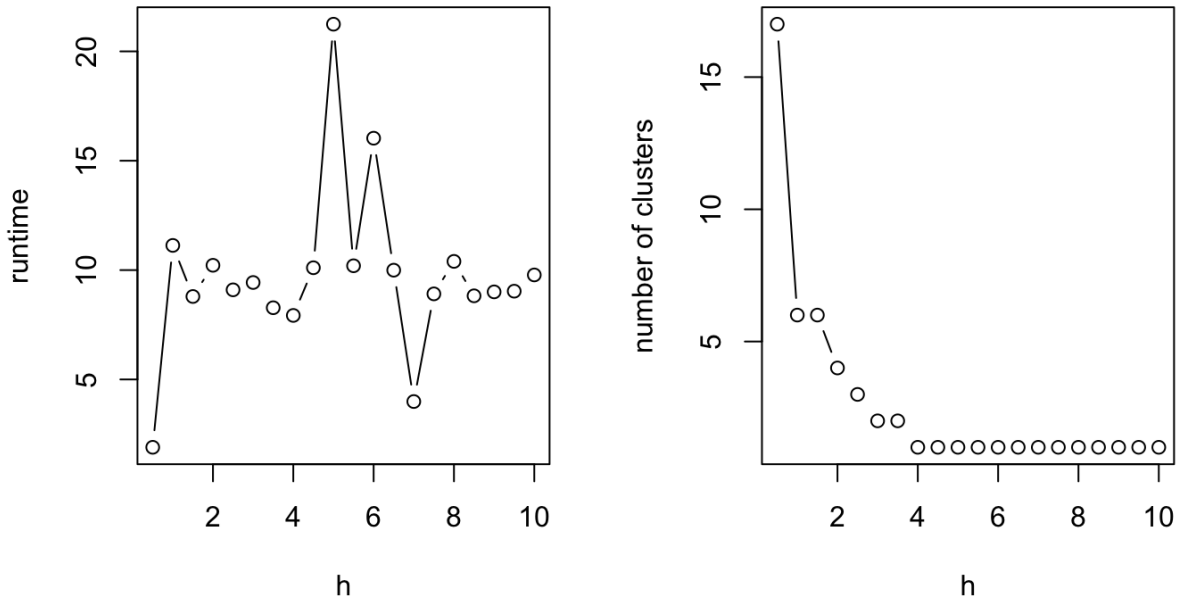

Figure 8: Runtime (in minutes) and Number of Clusters as hmult Increases

"kernel" was set to be Gaussian, h has a default of (1, 0.8)–this was calculated by assuming the underlying distribution is Normal. This set of value is asymptotically optimal. However, using the same h tuning procedure as described in Section 3.2, we need at h to be at least (1, 1) to achieve the "double criteria"

kernel	hmult	h	n.grid
Gaussian	1	(1.2, 1)	200

Table 2: Tuned Parameter of Level-set with KDE (pdfCluster)

objective. In the end, h was still set to be (1.2, 1) because it's better for clustering method comparison and it can also achieve the "double criteria" objective. The summary of the tuned parameters is provided in the following table 2. The clustering result of Level-set with KDE using these parameters is presented in Figure 10.

4. Evaluation

4.1 Visual Evaluation

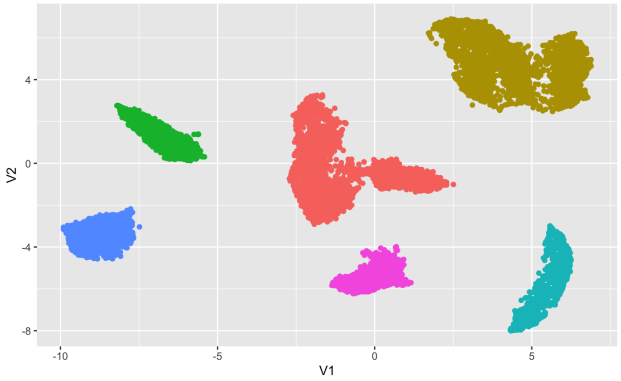

Figure 9: Mean Shift Result with Tuned Parameters

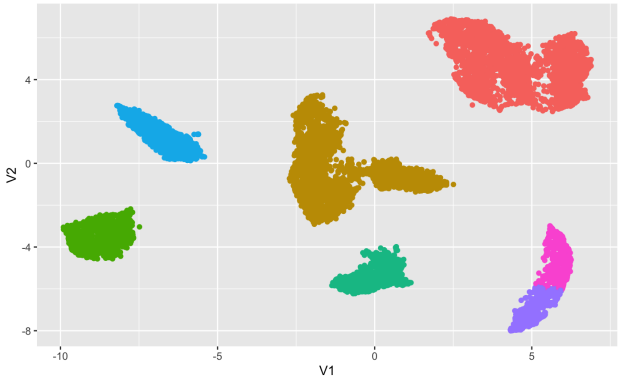

Figure 10: Level-set Result with Tuned Parameters

As shown in Figure 9 and Figure 10, our clustering result is quite good. We obtained clear clusters, and the margins are all very clean. 6 clusters were obtained by Mean Shift, and 7 clusters were obtained by Level-set. The only difference between these two clustering is that–in terms of the visualization–Level-set cut the lower right large cluster into two smaller clusters.

4.2 External Validation

Confusion Matrix of the two clusterings are presented in Table 3. Mis-classification Error $d_{ME} = 0.0499$, Jaccard Index $Jac = 0.995$. This means two clusterings are very similar. The result is quite unsurprising due to the similarity between these two methods: both algorithm first estimate the density of the data using a Gaussian kernel, then select certain peaks in the empirical data density curve as clusters, and lastly assign the points to their corresponding clusters.

In the first stage–estimation of density of data, we used the same kernel and the same bandwidth. So, it's safe to say that the estimated data density are exactly the same. The second stage is very similar, since Mean Shift look for "peaks", i.e. highest density regions and Level-set look for regions with higher-than-threshold density regions. The peaks that were selected by these two methods should be very similar. The last stage, again, is the same–points are assigned to their closest "high density regions", i.e., clusters. These explain why out clusterings are very similar to each other. Mean Shift picked 6 peaks, Level-set picked 7 peaks, with 6 agreeing with those picked by Mean Shift. The one remaining looks like a sub-peak of one of the 6 peaks.

	1	2	3	4	5	6	7
1	0	3500	0	0	0	0	0
2	3617	0	0	0	0	0	0
3	0	0	0	0	1220	0	0
4	0	0	0	0	0	599	659
5	0	0	1229	0	0	0	0
6	0	0	0	1176	0	0	0
7	0	0	0	0	0	0	0

Table 3: Confusion Matrix of the Two Clustering

4.3 Internal Validation

Essentially what we are trying to evaluate here are two things: how stable are these methods, and how well we achieved our goal–getting clusters that are as clear as possible. Therefore, two metrics/procedures were used to help us evaluate: bootstrap + Rand Index and silhouette score.

The boostrapping with Rand Index evaluation procedure is the following:

- Perturb the data by resampling with replacement
- For each perturbed dataset, we do dimension reduction with UMAP
- Then we apply the clustering function and obtain a vector of assignments for the perturbed dataset
- Repeat the above procedure for ten times to get 10 vectors of assignments
- Calculate the Rand Index for each pair (45 pairs in total) and take the average

This was done for both methods. In the end, we obtained a 0.739 average Rand Index for Mean Shift and a 1 average Rand Index for Level-set. It seems that Level-set is more stable than Mean Shift, probably because "identifying the peak" can be more fickle than "identifying regions that has high density", especially when the number of level curves is large.

Average silhouette score is calculated for both methods. Mean Shift has an average silhouette score of 0.658 and Level-set has an average silhouette score of 0.645. It seems that Mean Shift gives more "clean cut" clusters, probably due to the fact that Mean Shift is about peak-finding, so sub-peaks are less likely to be included comparing to Level-set, since level set look for peaks that are above a certain threshold in density. However, we have to keep in mind that silhouette score is not very meaningful since the distance between points are not preserved. Indeed, we can compute the silhouette score using the original data and the current clustering, but that approach would also be problematic since the clustering result could be very different using the original data.

5. Discussion

5.1 Strength and Limitations

We achieved our goal in obtaining clusters that are as clear as possible. The clustering result we get is also very stable. The Result in general is within expectation. However, there are a few things that could be improved, but all the limitations were imposed by myself–my poor method picking strategy. I originally picked these two methods because I think they are very similar–only differ in high-density-region-picking method–thus any difference in the clustering result could be attributed to this difference. This is true indeed, but kernel density estimation just takes too much time. I tried other methods–kmeans, hierarchical, and DBSCAN, all of them are extremely fast, and can be performed on the original data. This leads to lower bootstrapping times, lower efficiency in parameter tuning, and a lot of information loss since we could only use 2-dimension data (64 originally!). This dimension reduction made all the distance related metrics somewhat meaningless–silhouette

score, within-cluster sum of squares, compactness, etc. etc, hence affected our parameter tuning and method evaluation severely.

5.2 Future Directions

I'm very interested in parametric clustering methods, say k-means and hierarchical clustering. I dabbled the theory behind and attempted to cluster the data with these two methods. See Figure 11 and Figure 12, the metrics used to determine the optimal number of clusters are silhouette score and gap statistic. I chose to use 9 in the end because I wanted to see where are the rest "hidden" clusters. Additionally, all the data were used for these two methods–no dimension reduction used during clustering, UMAP was only used for visualization.

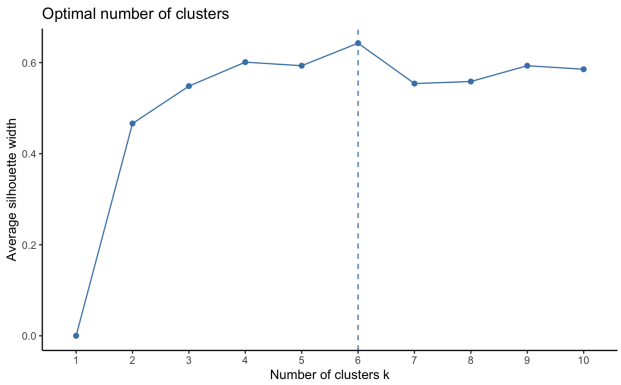

Figure 11: Optimal Number of Clusters by Silhouette

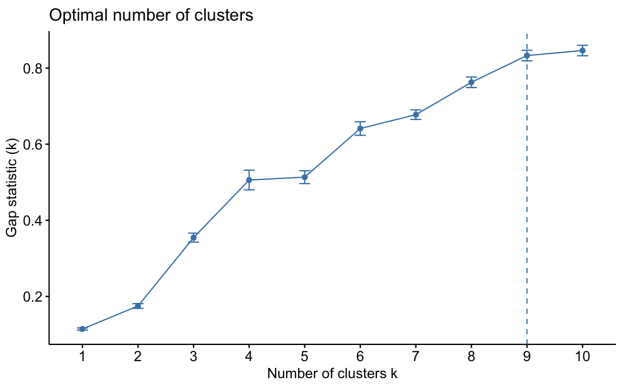

Figure 12: Optimal Number of Clusters by Gap

The result is provided in Figure 13 and Figure 14. It's quite magical that even using all the data the clustering result can be well-reflected on the 2-dimension visualization. It seems that there are two smaller clusters in the upper middle cluster, three in the upper right cluster, and two in the lower right cluster. Level-set captured the two in the upper middle cluster and the two in the lower right cluster cleanly, and two in the upper right somewhat vaguely. This was without parameter tuning. After tuning, we only have seven left–we lost one in the upper right and one in the upper middle. This has a lot to do with our parameter tuning objective–to obtain clusters that are as clear as possible.

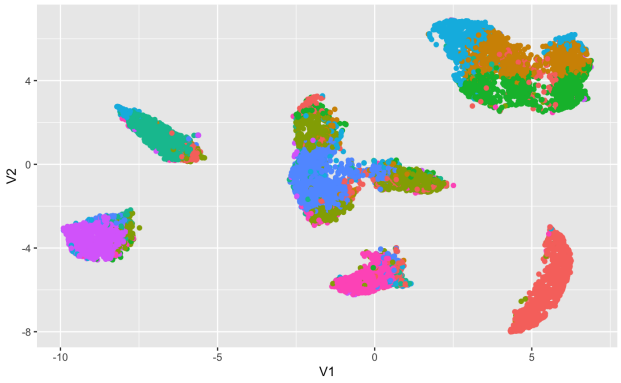

Figure 13: K-means Result

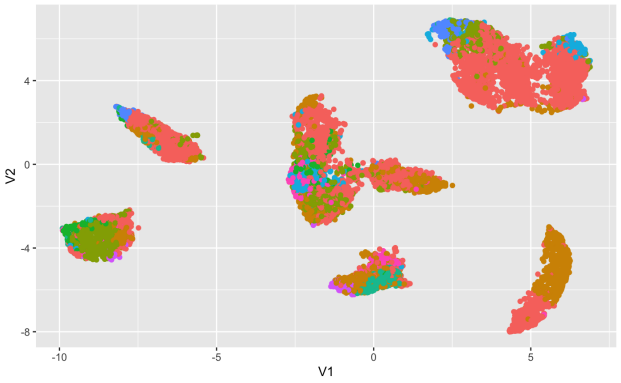

Figure 14: Hierarchical Clustering Result

However, we can see that the clustering result with k-means and hierarchical doesn't have the clean cut margins. Also, all clusters have some points that are "far away from home", which reason are to be discovered. In the future, more time can be devoted to do comparison between parametric clustering and nonparametric clustering, and maybe between density based methods, centroid based methods, and graph based methods.

References

1. Azzalini, A., Menardi, G. (2014). Clustering via nonparametric density estimation: the R package pdfCluster. Journal of Statistical Software, 57(11), 1-26, URL http://www.jstatsoft.org/v57/i11/.
2. Azzalini A., Torelli N. (2007). Clustering via nonparametric density estimation. Statistics and Computing. 17, 71-80.
3. Menardi, G., Azzalini, A. (2014). An advancement in clustering via nonparametric density estimation. Statistics and Computing. DOI: 10.1007/s11222-013-9400-x.
4. Cheng, Y. (1995). Mean shift, mode seeking, and clustering. IEEE transactions on pattern analysis and machine intelligence, 17(8), 790-799.
5. Fukunaga, K., & Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. IEEE transactions on information theory, 21(1), 32-40.
6. Lisic, J. (2015). Parcel Level Agricultural Land Cover Prediction (Doctoral dissertation, George Mason University).