Clustering High-Dimensional Data via KNN Density Level-Sets and Self-Organizing Maps

Haijing Zong

ABSTRACT

I applied two unsupervised approaches—(i) KNN level-set density clustering and (ii) Self-Organizing Maps (SOM)—to a 10000×600 feature matrix. Both methods indicate that the data form a single, highly concentrated density peak, making compact, well-separated clusters elusive. According to the internal metrics (Silhouette≃3.86, DB≃-0.022, CH≃63.15), method(ii), which I output data as 'clust2_out.txt', produced marginally better separation than method(i), but overall cluster quality remains poor.

Keywords: KNN, PCA, SOM

1. PCA

Firstly, I standardized the data.

After that, I use PCA to reduce dimensionality, mainly because Colab said that I run out of RAM.

Graph a shows that The first few principal components explain most of the variance, while the following components explain less and less variance, with diminishing marginal benefits. Based on this, I can determine at which principal component number to "cut off" to retain key information while significantly reducing dimensionality.

The second graph shows the "Explained Variance Ratio" of each principal component. The horizontal axis is the principal component number, and the vertical axis is the percentage of the total variance of this component. Because your original data is 600-dimensional, the first principal component, although the "largest", only explains about $\frac{\lambda_1}{\sum_j \lambda_j} \approx \frac{30}{600} \approx 0.05$

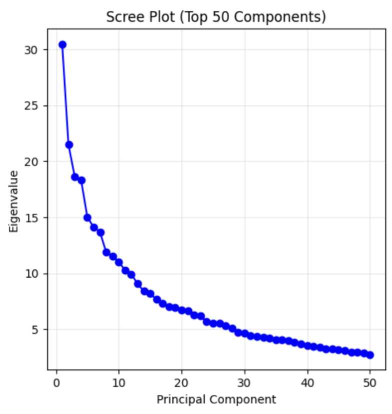
Figure c is the variance explanation rate: how much of the total variance can be explained by the first k PCs, and the number of PCs corresponding to each threshold (80%, 85%, 90%, 95%, 99%) is marked to help you select the dimension after dimensionality reduction.

Figure d shows the data projected onto the first two principal component spaces: a scatter plot is drawn using random sampling points to visualize the distribution structure or possible clusters of the samples after dimensionality reduction.

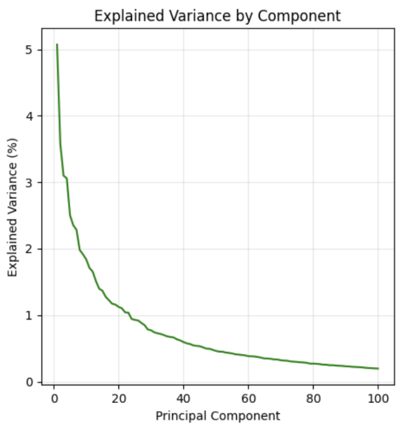
Figure e is a load heat map: it shows the coefficients (loadings) of the first 10 PCs on the first 50 original features, and it can be seen which features contribute the most to which principal components, as well as the positive and negative directions.

Figure f shows the "reconstruction error" curve after reconstructing the sample with the first k PCs as the number of principal components k increases, which helps to evaluate the degree of information loss after dimensionality reduction intuitively.

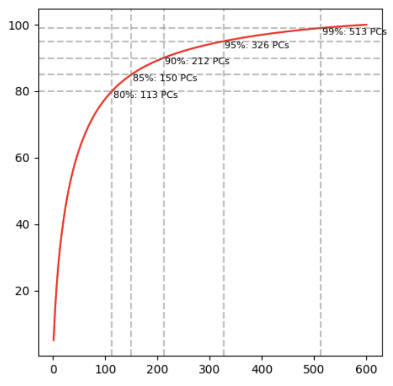
Thus, in the end, I successfully reduced the dimensionality to 326, and at the same time can explain the variance at 95.02%.



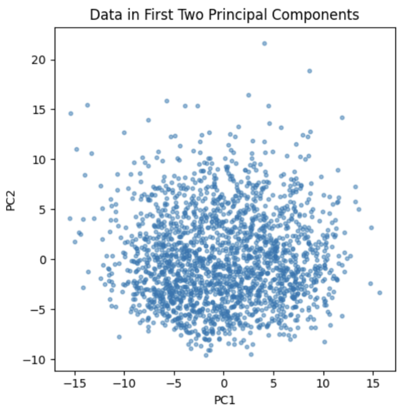
(a) Scree Plot (Top 50 PCs)



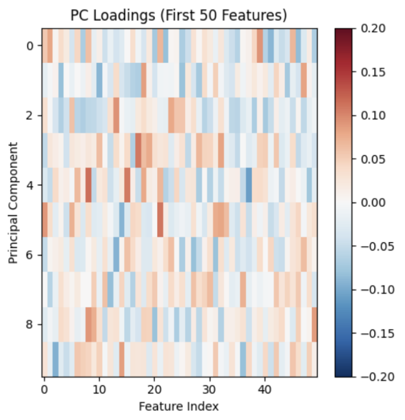
(b) Explained Variance by Component



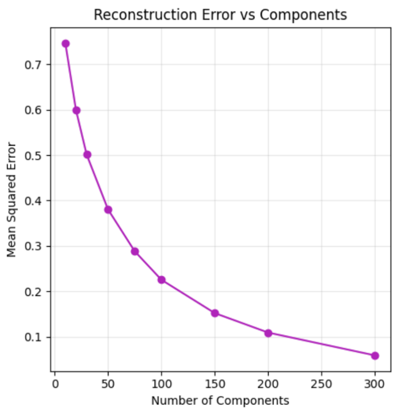
(c) Cumulative Explained Variance



(d) Projection on PC1 vs PC2



(e) PC Loadings Heatmap



(f) Reconstruction Error vs. #PCs

Figure 1. Summary of PCA diagnostics. (a) Scree plot. (b) Explained variance ratio. (c) Cumulative explained variance. (d) Scatter of first two PCs. (e) Heatmap of PC loadings. (f) Reconstruction error analysis.

2. LEVEL SET + KNN

2.1. Intuition

Firstly, I use KDE to estimate the density distribution of the data, then I define a density threshold (level), and define the area with density higher than this threshold as a "level set".

These high-density areas are considered potential clusters. I then use KNN to connect points in the same level set. Specifically: For each point in a high-density area, if there are other points in its k nearest neighbors that are also in the same high-density area, connect them as the same cluster.

2.2. Steps

Mathematically, level set is choosing a lambda, such that $f(\hat{x}) > \lambda$. How to choose lambda is a problem. I do not think there's an optimized way to do that, so I manually tested different lambdas. I use KDE to estimate $f(\hat{x})$, and thus the bandwidth selection will affect the result. Initially, I tried to think of some optimized way (such as leave one out data driven method) to find optimal h that gives minimum MSE, but it turns out either because of the dimension is too large (326), the calculation is extremely slow, or because the result I calculated did not give me any good clustering result. **This suggests that the true cluster boundaries in the data are thin and sharp, and that the previous smoothing was obscuring the structure.** During all this experiments, I find that smaller h<1, seems better. Thus, in the end, I set h as 0.7.

Then I define an indication function, $\mathbf{1}_C(i) = \begin{cases} 1, & \hat{f}(X_i) \geq \lambda, \\ 0, & \text{otherwise.} \end{cases}$ to find all the index of $X_{reduced}$ which satisfies $C = \{i : \hat{f}(X_i) \geq \lambda\}$. Same as before, I manually tested different lambdas. I chose the top 12% percentile density to fall in to set C.

I count the points that are in the set $C = \{i : \hat{f}(X_i) \geq \lambda\}$, it has 1234, which counted for 12.3% of all the observations.

I then construct mutual KNN. For each point $i \in C$, I find their k's nearest neighbor sets.

I then construct $A \in \{0,1\}^{n_{\text{core}} \times n_{\text{core}}}$, among which $A_{i,j} = 1$.

Extract connected components on an undirected graph (C, E): 1. Get n_{comp} components $C_1, \ldots, C_{n_{\text{comp}}}$. 2. $labels_core[i] = c$means that the $i - th$ point in the core point index list belongs to the $c - th$ connected component.

Then, I try to find low-density set $\mathcal{L} = \{i : \hat{f}(X_i) < \lambda\}$, connect these points to their closest cluster. Initially, I assigned all the low-density points to the clusters. However, I find that in some clusters, there are only 10 or 15 or even lower number of points. So, I adopted another strategy: Set a maximum acceptable distance threshold max_dist, and only assign points with distance < max_dist to the cluster, and keep the others as -1 (noise).

In the end, I use the same evaluation matrix as self-organizing map.

— Scores for KNN — Davies-Bouldin : 2.383466120623888

Silhouette : -0.06789695589215106

Calinski-Harabasz : 5.263166446154424

All three indices point the same way:

1. Clusters are not well separated.

2. Internal cohesion is weak—points within a cluster are almost as far from each other as from other clusters.

3. A negative Silhouette means a significant fraction of samples would sit closer to some other cluster's center.

In other words, the algorithm is failing to discover meaningful structure in this feature space. I have to say that no matter what parameter combinations I tried, (h, λ, k), I do not think this method can produce good result.

2.3. Possible explanation on the bad result

KNN density + Level-set clustering relies on the "valleys between density peaks" to draw boundaries; when the data is highly concentrated, the density field is almost flat - no matter how the threshold is adjusted, no meaningful clusters can be distinguished, only a large cluster or noise fragments can be obtained. This is not because the algorithm is "bad", but because the data itself lacks a separable density structure.

3. SELF ORGANIZING MAP

3.1. algorithm

The following is summarized from Wikipedia:

1. Goal

Project high-dimensional data onto a fixed low-dimensional grid (usually 2D) while trying to maintain the topological relationship of "near is still near, far is still far".

2. Structure

• Grid: preset rectangular or hexagonal node array.

• Weight vector: Each node has a weight vector with the same dimension as the input space, representing the prototype position of the node.

3. Basic loop

1. Sample selection: Extract an input **x** from the data set.

2. Determine the best matching unit (BMU): Find the node r^* with the closest Euclidean distance to **x** among all nodes.

3. Neighborhood update: Execute

$$\mathbf{w}_r(t+1) = \mathbf{w}_r(t) + \eta(t)\, h_{r^*,r}(t)\big[\,\mathbf{x}(t) - \mathbf{w}_r(t)\big]$$

for each node r, where $\eta(t)$ is the learning rate (decreasing over time), and

$$h_{r^*,r}(t) = \exp\!\left(-\frac{\|r - r^*\|^2}{2\,\sigma^2(t)}\right)$$

is the neighborhood kernel (the width $\sigma(t)$ also decreases).

4. Parameter annealing

During the training process, $\eta(t)$ and $\sigma(t)$ gradually decay from large to small, first globally sorting and then locally fine-tuning.

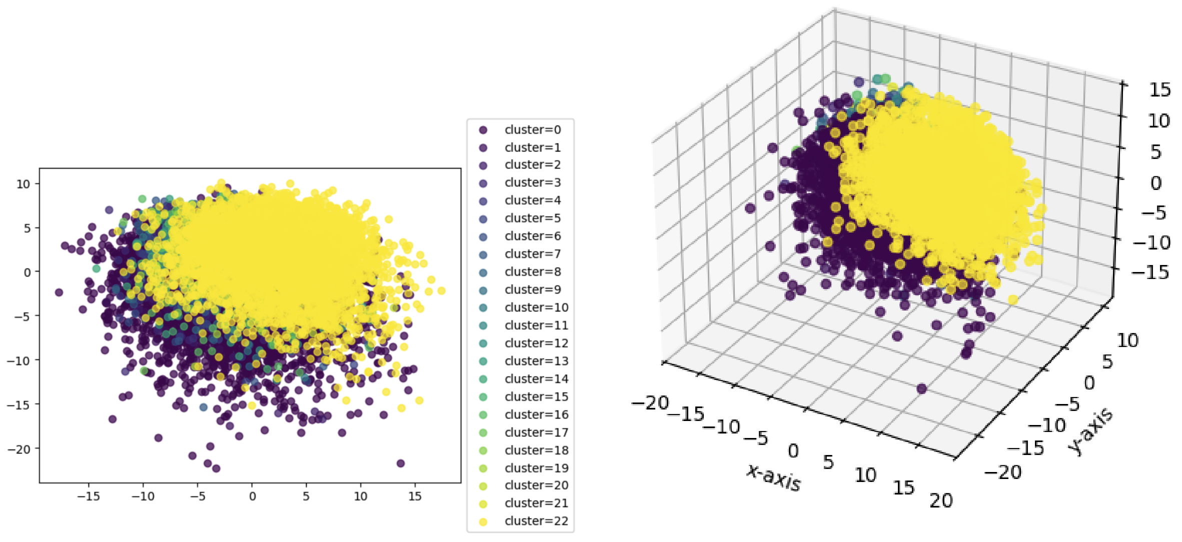
5. Convergence

If the Robbins–Monro condition is satisfied (the square of the learning rate is summable, but the sum itself is not summable), and the neighborhood width eventually converges to 0, the weight vector converges to a local optimal vector quantization result and maintains a high topological consistency.

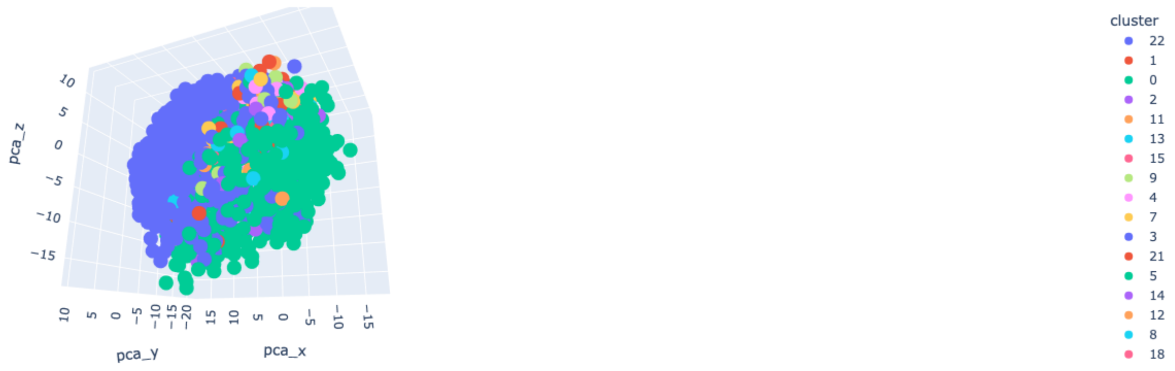
3.2. Steps

I set the neurons with 23*23. I used PCA to project the data into 2d and 3d, to visualize the result of SOM here.

In both of the graph, it looks like most of the points are centered together.



I also draw a dynamic 3-d graph (see below) that allows people to rotate it to see how the points are clustered in 3 main components. It shows similar results that most of the points are centered in two colors, while others are scattered around.



Possible reasons of this graph: The data itself is unimodal and has no obvious discrete clusters; or The neighborhood is too wide/the number of steps is insufficient during training, and the nodes have not yet differentiated. I tried to reduce sigma, and increase the number of iterations, however the graph is similar. And thus I didn't include the graph with different parameters here.

3.3. Evaluation

I calculate silhouette [1], Calinski-Harabasz, Davies-Bouldin, and Contingency Matrix.

— Scores for SOM —

Davies-Bouldin Index: 11.40123558546461

Silhouette Score: -0.003696408702285198

Calinski-Harabasz Index: 13.437482689997976

silhouette score: Not ideal for very high-dimensional data, but still interpretable. It measures, for each

[1] https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

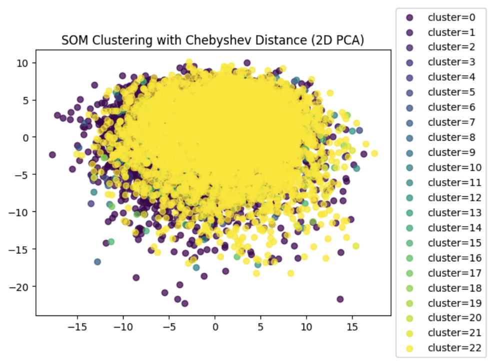
point, the difference between its mean distance to points in the same cluster and its mean distance to points in the nearest other cluster, normalized to the interval [-1, 1]. Typical “good” values are close to +1. Values near 0 indicate overlapping clusters, while negative values show many points are closer to another cluster than to their own.

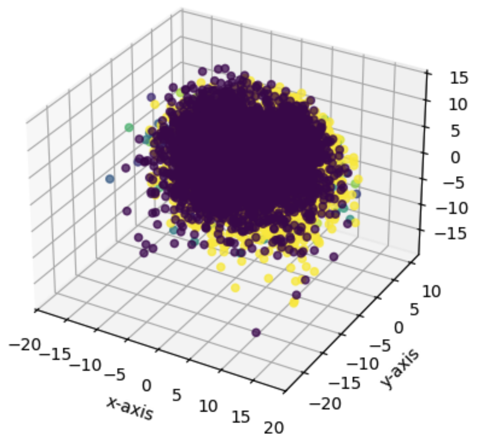
davies-bouldin score: A lower Davies–Bouldin index indicates better separation: it averages, over all clusters, the ratio of within-cluster scatter to between-cluster centroid distance. Values <1 are considered as good separation. (The index is always non-negative—there are no valid “negative” DB scores.) A score around 11 means clusters are wide and sit very close together, pointing to very poor structure.

calinski-harabasz score: This index is the ratio of between-cluster dispersion to within-cluster dispersion (both defined as sums of squared distances). Higher is better. For datasets with thousands of points, “good” partitions often yield scores in 100+. A value as low as 13 implies that inter-cluster variance is only marginally larger than intra-cluster variance—again indicating almost no meaningful clustering.

3.4. SOM with chebyshev distance

I tried chebyshev distance, but seems the result did not improve.



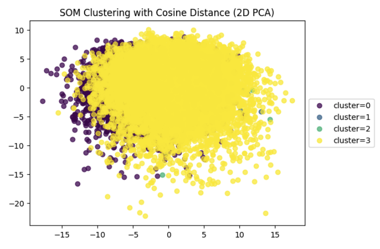


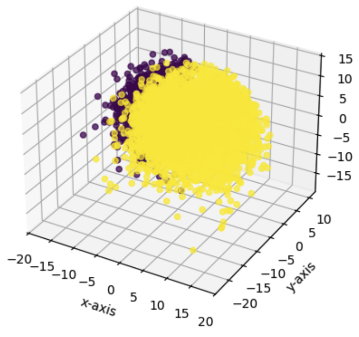
cluster
22
506
66
44
132
286
110
154
242
88
264
462
308
374
220
176
418
pca_x
pca_y
pca_z

Not only the graph, but also the scores are getting worse: — Scores for SOM Chebyshev—
Davies-Bouldin Index: 7.400318526378243
Silhouette Score: -0.03334639486779961
Calinski-Harabasz Index: 4.407773146343785

3.5. SOM with euclidean distance

I think the Euclidean distance also didn’t improve the result.

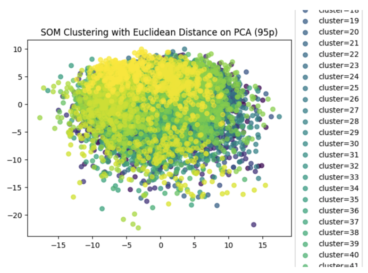


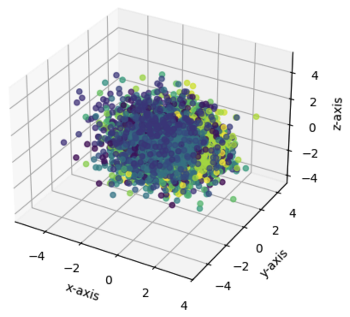


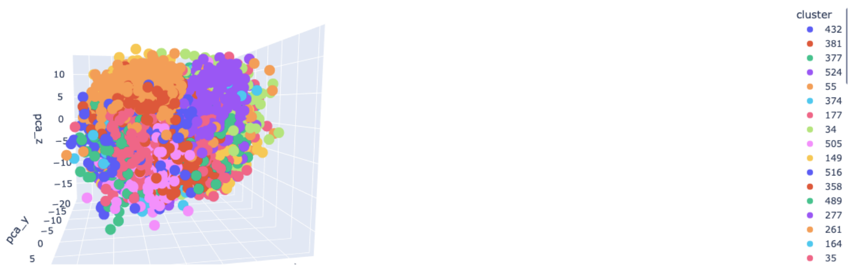
pca_z
pca_y
pca_x
cluster
15
0
10
5

3.6. Keep 95% variance

I then use PCA to reduce the dimension to 95% of the variance.







The scores are the best:

— Scores for SOM PCA95 on PCA —

Davies-Bouldin Index: 3.864776131679225

Silhouette Score: -0.022210333113241 09

Calinski-Harabasz Index: 63.15407558485201

Thus, I decide to use this set of data as my final result for clustering.