Clustering Algorithms

Matthew Benvenuto, Mutasim Mim, Qinying Chen, Morgan Acevedo, Wayne Nguyen

May 20, 2022

1 Definitions, Dissimilarities, and Data Sets

1.1 Unsupervised learning

Unsupervised learning is the machine learning technique in which the goal is to directly infer the properties of a probability density without the help of a supervisor or teacher providing correct answers or degree-of-error for each observation. It allows the model to work on its own to discover patterns and information that was previously undetected. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition. An important concept when it comes to unsupervised learning is clustering.

1.2 Cluster analysis

Cluster analysis has many goals, which all relate to grouping or segmenting a collection of objects into subsets or "clusters," such that those within each cluster are more closely related to one another than objects assigned to different clusters. In addition, the goal is sometimes to arrange the clusters into a natural hierarchy. Central to all of the goals of cluster analysis is the notion of the degree of *similarity* (or *dissimilarity*) between the individual objects being clustered. A clustering method attempts to group the objects based on the definition of similarity supplied to it.

1.2.1 Proximity Matrices

Data is represented directly in terms of proximity between pairs of objects. These can be either similarities or dissimilarities. For example, in social science experiences, participants are asked to judge by how much certain objects differ from one another. Dissimilarities can then be computed by averaging over the collection of such judgements. This type of data can be represented by an $N \times N$ matrix $\mathbf{D}$, where N is the number of objects, and each element d_{ij}, records the proximity between ith and jth objects. These dissimilarity matrices are used as inputs in clustering algorithms.

1.2.2 Dissimilarities Based on Attributes

Often we have measurements x_{ij} for $i = 1, 2, \ldots, N$, on variables $j = 1, 2, \ldots, p$, which are also called *attributes*. Since the most popular clustering algorithms use a dissimilarity matrix as their input, we construct pairwise dissimilarities between the observations. We define $d_k(x_{ik}, x_{jk})$ as a dissimilarity between values of the kth attribute and define

$$D(x_i, x_j) = \sum_{j=1}^{p} d_k(x_{ik}, x_{jk})$$

as the dissimilarity between objects i and j.

Quantitative variables These variables are forms of numerical quantities, there is a numerical attachment. Examples include: height, weight, length.

Categorical variables A variable that has two or more categories, with no intrinsic ordering. An example is a binary response (Yes or No).

Ordinal variables These variables are categorical, but have an intrinsic ordering of such categories. An example is Low, Medium, High.

1.3 Combinatorial Algorithms

Combinatorial algorithms are computational procedures which are designed to help solve combinatorial problems. Combinatorial problems are problems involving arrangements of elements from a finite set and selections from a finite set. These problems can be divided into three basic types: (1) enumeration problems, (2) existence problems, and (3) optimization problems.

1.4 DNA Microarray Data Set

A data set that is implemented in this paper is the DNA microarray data set. It is an expression matrix of 6830 genes (rows) and 64 samples (columns), for the human tumor data.

2 Partition Based Clusterings

k-means clustering

Let C be the cluster assignment function, with $N_1, \dots, N_k$ be the sizes of the k-clusters for a given clustering. Let us denote by $\overline{x_i}$ the mean of the points in the i^{th} cluster (coordinate-wise mean). We define the loss function

$$L(C) = \sum_{i=1}^{k} \sum_{C(x)=i} \|x - \overline{x_i}\|_2^2.$$

We will show that

$$L(C) = \frac{1}{2} \sum_{i=1}^{k} \sum_{C(x)=i, C(x')=i} \frac{\|x - x'\|_2^2}{N_k},$$

which again, supports our intuitive notion of a 'good' clustering. We have, for a fixed i,

$$\begin{aligned} &N_k \sum_{C(x)=i} \|x_i - \overline{x_i}\|_2^2 \\ &= N_k \sum_{C(x)=i} \|x - \frac{1}{N_k} \sum_{j=1}^{N_k} x_j\|_2^2 \\ &= N_k \sum_{j=1}^{N_k} \sum_{l=1}^{p} (x_{jl} - (\frac{1}{N_k} \sum_{s=1}^{N_k} x_{sl}))^2 \\ &= N_k \sum_{l=1}^{p} \sum_{j=1}^{N_k} (x_{jl} - (\frac{1}{N_k} \sum_{s=1}^{N_k} x_{sl}))^2. \end{aligned}$$

We can thus fix a coordinate and consider the rest of the expression. Define $x_{jl} = y_l$ and note that

$$\begin{aligned} &N_k \sum_{j=1}^{N_k} (x_{jl} - (\frac{1}{N_k} \sum_{s=1}^{N_k} x_{sl}))^2 \\ &= N_k \sum_{j=1}^{N_k} (y_l - (\frac{1}{N_k} \sum_{s=1}^{N_k} y_l))^2 \\ &= N_k(\sum y_l^2 - \sum 2 \frac{y_l}{N_k} \sum y_{l'} + \frac{1}{N_k}(\sum y_l)^2) \\ &= N_k \sum y_l^2 - 2(\sum y_l)^2 + (\sum y_l)^2 \\ &= N_k \sum y_l^2 - (\sum y_l)^2 \\ &= \frac{1}{2} \sum (y_l - y_{l'})^2 \end{aligned}$$

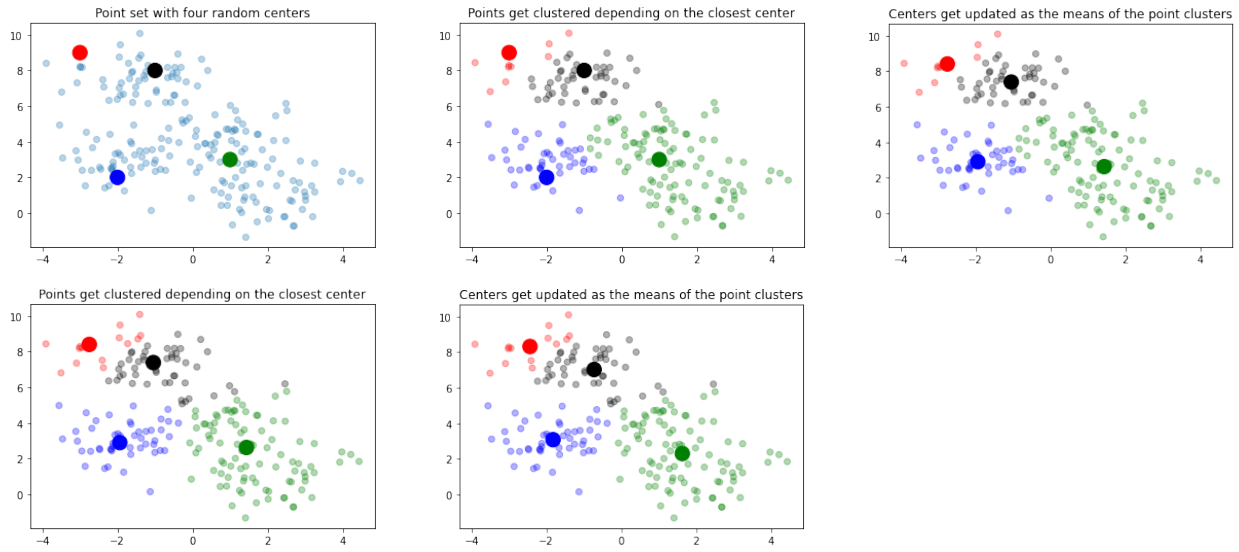
We provide below Llyod's algorithm for this optimization problem.

Llyod's Algorithm for finding a good clustering function.

- **Input** A sample collection in $\mathbb{R}^p$ and a positive integer k.
- **Output** A cluster assignment function C.

1. Start with a random collection of distinct points $y_1, \dots, y_k$ in $\mathbb{R}^p$ and arbitrary clustering function C.
2. Modify C so that each point x in the sample is assigned to the cluster associated with the y_i that is closed to x in the Euclidean sense.
3. Update the cluster centers as the mean of the points associated with that particular cluster.
4. Repeat steps 2 and 3 until convergence.

Table 1: Two iterations of Llyod's algorithm



Proposition 1.

To prove this, we will need the following lemma.

Lemma 1. *Let* $x_1, \dots, x_n \in \mathbb{R}^p$. *Then*

$$argmin_{x \in \mathbb{R}^p} \sum_n \|x - x_n\|_2^2 = \frac{\sum x_i}{n}.$$

Proof. For a fixed coordinate j,

$$\frac{\partial}{\partial x_j} \sum_{i=1}^{n} (x_j - x_{nj})^2 = 2nx_j - 2x_j \sum_n x_{nj}.$$

Setting the partial derivative to 0, we have $x_j = \frac{1}{n}\sum_n x_{nj}$. Note that since the objective function has not maximum, the mean, the unique critical point, must be the global minimum. This proves the proposition. $\square$

Proof. (of the proposition) Since the original point set is finite and we partition the set in k clusters, there are only a finite number of possible cluster functions. If we show that neither of the steps 2 and 3 increase the loss function, that will guarantee convergence. Assume that step 2 modifies point $\overline{x_1}, \ldots, \overline{x_1}$ to $y_1, \ldots, y_k$. Since cluster function changes value only when a sample point finds a closer cluster center, we have that

$$L(C) = \sum_{i=1}^{k} \sum_{C(x)=i} \|x - \overline{x_i}\|_2^2 \geq \sum_{i=1}^{k} \sum_{C(x)=i} \|x - y_i\|_2^2$$

But after this update, the centers y_i's gets modified to the corresponding means, say, y_i''s, and so, by the lemma above, we conclude that

$$L(C) = \sum_{i=1}^{k} \sum_{C(x)=i} \|x - \overline{x_i}\|_2^2 \geq \sum_{i=1}^{k} \sum_{C(x)=i} \|x - y_i\|_2^2 \geq \sum_{i=1}^{k} \sum_{C(x)=i} \|x - y_i'\|_2^2,$$

which proves the proposition. □

We should make a few remarks about Llyod's algorithm.

- There is a precise upper bound of the number of iterations before convergence.
- Despite convergence, Llyod's algorithm may not converge to the global minimum.

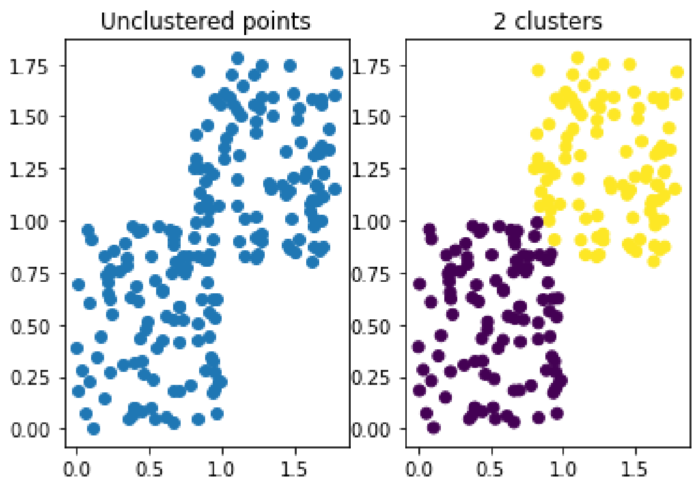
Below is an example of the usage of the Scikit-learn API for K-means clustering.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

points = np.concatenate((np.random.random((100,2)), np.random.random((100,2)) + (0.8,0.8))
, axis=0)

plt.subplot(121)
plt.title('Unclustered points')
plt.scatter(points[:,0], points[:,1])

plt.subplot(122)
plt.title('2 clusters')
kmeans = KMeans(n_clusters=2).fit(points)
labels = kmeans.labels_
plt.scatter(points[:,0], points[:,1],c=labels)
```



An application: Vector Quantization and Image Compression

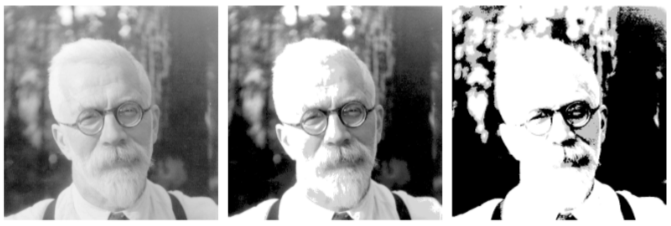


FIGURE *Sir Ronald A. Fisher (1890 – 1962) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a 1024×1024 grayscale image at 8 bits per pixel. The center image is the result of 2×2 block VQ, using 200 code vectors, with a compression rate of 1.9 bits/pixel. The right image uses only four code vectors, with a compression rate of 0.50 bits/pixel*

k- means clustering algorithm can be used for image compression. In the figure above, the left image is a 1024×1024 grayscale image, so each of the pixels can be represented by an integer between 0 and 255- requiring 8 bits- with the entire image requiring 1 megabytes of data. Let us divide the entire image into 2×2 blocks and consider each block a vector in $\mathbb{R}^4$. We then run Lloyd's algorithm with $K = 200$. We can then approximate the image by replacing each 2×2 block by its cluster centroid. In the figure above, the middle picture is the result. The approximated picture can be stored by recording the 200 centroids, requiring 200×8 bits, which is negligible, and 512×512 grid of information representing the corresponding centroids for each block, requiring $\log_2(200)$ bits per block. Overall, the storage requirement is $\frac{\log_2(k)}{4 \times 8}$, which is only 0.239 of the original amount of storage. We could do similar compression with only $K = 4$, producing the right image but using only 0.063 fraction of the original amount of storage.

k-mediods Clustering

The k-means clustering applies to points in Euclidean space, where we can compute the mean of a collection of points. For general dissimilarity matrix, it may not be possible. In this case, we modify the step 2 in the k-means clustering by replacing the mean of the corresponding cluster by the point in the cluster with minimal total distance from the remaining points in the cluster.

k-mediotd clustering for finding a good clustering function.

- **Input** A sample collection in $\mathbb{R}^p$ and a positive integer k.
- **Output** A cluster assignment function C.

1. Start with a random collection of distinct points $y_1, \dots, y_k$ in $\mathbb{R}^p$ and arbitrary clustering function C.
2. Modify C so that each point x in the sample is assigned to the cluster associated with the y_i that is closed to x in the Euclidean sense.
3. Update the cluster centers as the point in the corresponding cluster that has the minimum total distance from the remaining cluster points.
4. Repeat steps 2 and 3 until convergence.

Remark Each iteration of step 3 in Llyod's algorithm requires computation proportional N_k, while for k-mediods, it is proportional to N_k^2 - making the k-mediods algorithm computationally intensive than its k-means algorithm.

How to choose k?

Usually determining a good value of k is defined as a part of the problem- for example, if we have k employees, we may want to partition all tasks in k groups. Choosing k can also be data based. For this, we examine total within cluster dissimilarity $W_1, \ldots, W_l$ for $K = 1, 2, \ldots, l$. Usually, as l increases, W_l decreases. The intuition underlying this approach is that if there are actually K^* distinct groupings of the observations (as defined by the dissimilarity measure), then for $K < K^*$ the clusters returned by the algorithm will each contain a subset of the true underlying groups. Thus, for $K > K^*$, we newer partition will in fact, break up a natural group of points into two parts, so the decrease in W will be much less than the case when $K < K^*$. That is, we should expect a sharp decrease in $W_l - W_{l+1}$ for $l = K^*$, i.e., $\{W_K - W_{K+1} \mid K < K^*\} \gg \{W_K - W_{K+1} \mid K > K^*\}$. Thus, a good choice of K would be where the graph of W_K has a sharp decrease. We reproduce a representive figure from Hastie, et. al.

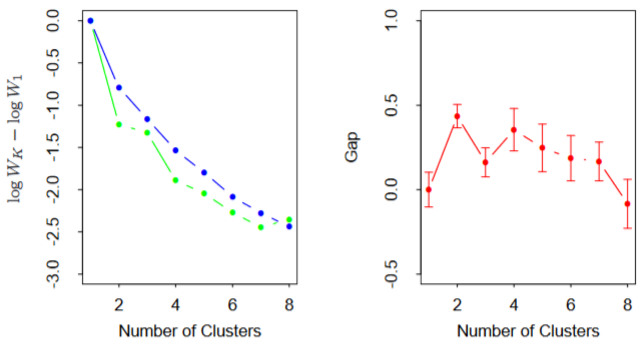


FIGURE 14.11. *(Left panel): observed (green) and expected (blue) values of* $\log W_K$ *for the simulated data of Figure 14.4. Both curves have been translated to equal zero at one cluster. (Right panel): Gap curve, equal to the difference between the observed and expected values of* $\log W_K$*. The Gap estimate* K^* *is the smallest* K *producing a gap within one standard deviation of the gap at* $K + 1$*; here* $K^* = 2$.

3 Hierarchical Clustering

Hierarchical clustering is an unsupervised clustering that seeks to build a hierarchy of clusters. Unlike k-means clustering, hierarchical clustering does not need to pre-define the number of clusters. The hierarchical representation can be visualized by the dendrogram tree. At the highest level we have the whole data, while at the lowest level we have all the single-point clusters. There are two main types for hierarchical clustering: Agglomerative clustering and divisive clustering.

Dendrogram

What is a dendrogram? A dengrogram is a type of tree diagram that shows hierarchical relationships between different sets of data. In the following example, we generate 10 random points and plot the dendrogram.

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage

X = np.random.random((10,2))
Z = linkage(X, 'single')
fig = plt.figure(figsize=(20, 10))
dn = dendrogram(Z)
plt.axhline(y=0.33, color='r', linestyle='--')
plt.show()

labels = range(0, 10)
plt.figure(figsize=(10, 7))
plt.subplots_adjust(bottom=0.1)
plt.scatter(X[:,0],X[:,1], label='True Position')
for label, x, y in zip(labels, X[:, 0], X[:, 1]):
plt.annotate(
   label,
    xy=(x, y), xytext=(5, 5),
   textcoords='offset points', ha='right', va='bottom')
plt.show()
```

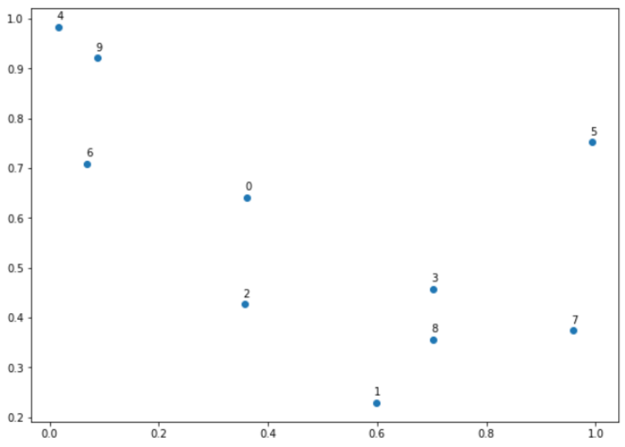


Figure 1: Random generate 10 points

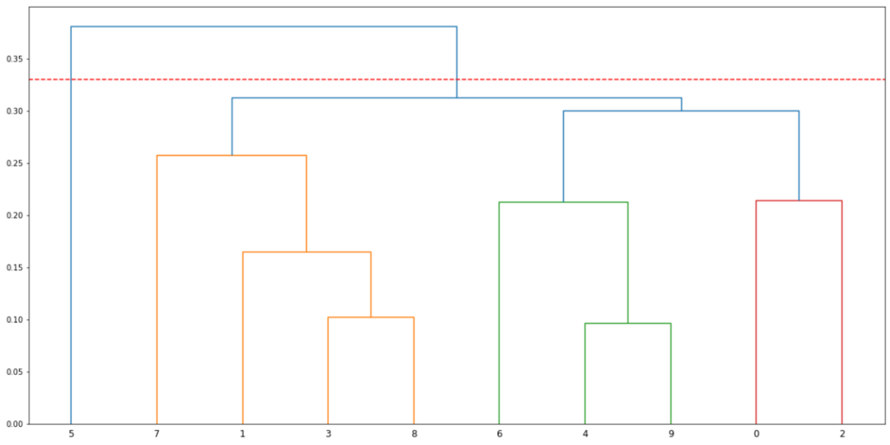


Figure 2: Dendrogram for 10 random generated points

In Figure 1, the 10 randomly generated points are labeled $0-9$. Figure 2 shows the dendrogram for these 10 data points, and we choose the single linkage in this case. We can cut the dendrogram tree with a horizontal line where it an traverse the maximum distance up and down without intersecting the merging points. For example, in this figure, we have the horizontal red dashed line which is a cutting line that tells us how many clusters we have at this point. Clearly, we have a single-point 5 as one cluster and another cluster that contains all other points. Moreover, by first looking at bottom of the dengrogram, we can tell that points $3,8$, $4,9$, $0,2$ form three clusters respectively according to the dissimilarity and linkage we choose. Then, as we go up the dendrogram, we notice that $7,1,3,8$ merge to form a cluster, $6,4,9$ merge and form another new cluster. After that, $7,1,3,8,6,4,9,0,2$ merge to form a new cluster, and eventually merge the single point 5 to form one whole cluster.

Agglomerative hierarchical clustering

The figure we showed above is an example of agglomerative hierarchical clustering. First we have some data points, and we treat them as single-point clusters. Then we merge clusters based on the metric and linkage we choose, and eventually we would obtain one entire clusters that contains all the data points.Here is the algorithm for agglomerative clustering.

1. Given N data points$\{x_i, i=1,2,...,N\}$. We consider each point as a single-point cluster. Initially, we have N clusters.

2. Compute the dissimilarity matrix $D=(d_{ij})$ where $d_{ij}=d(x_i,x_j)$ in some given metric, $i,j=1,2,...,n$

3. Find the smallest dissimilarity between two clusters(two points in this case), say D_{IJ} and merge clusters I and J to be the new cluster IJ. Now we have $N-1$ clusters.
4. Find the smallest dissimilarity between two clusters and merge the clusters. Now we have $N-2$ clusters.
5. Repeat step 4 until we are left with only one cluster.

For step 4, to compute the dissimilarity between two clusters that have more than 1 data point respectively, we introduce three popular linkage methods:

Single linkage:

$$d_{A,B} = min\{d(a,b) : a \in A, b \in B\}$$

Complete linkage:

$$d_{A,B} = max\{d(a,b) : a \in A, b \in B\}$$

Average linkage:

$$d_{A,B} = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b)$$

Now we want to compare these three linkage methods, and try to analyze some features by visualizing the dendrograms and scatter plots. First, we do the single linkage method.

```
    X = np.random.random((200,2))
    Z = linkage(X, 'single')
    fig = plt.figure(figsize=(20, 10))
    dn = dendrogram(Z)
    cluster1 = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='single')
    cluster1.fit_predict(X)
    plt.subplot(131)
    plt.title('2 clusters')
    plt.scatter(X[:,0],X[:,1], c=cluster1.labels_, cmap='rainbow')
```

We certainly can generate more than 200 points to see a general pattern, but the dendrogram will be too messy to look at, so we stick with 200 points which is quite a large amount of data points. Figure 3 shows the dendrogram using single linkage method, and we can clearly see that a huge cluster is developing along with some small clusters. Now we look at Figure 4 which shows the scatter plots for the clusters formed eventually. As we can see, we have a big chain of points that forms one cluster, and towards the end of the clustering process, we are left with several single-point clusters with that big chain of cluster.

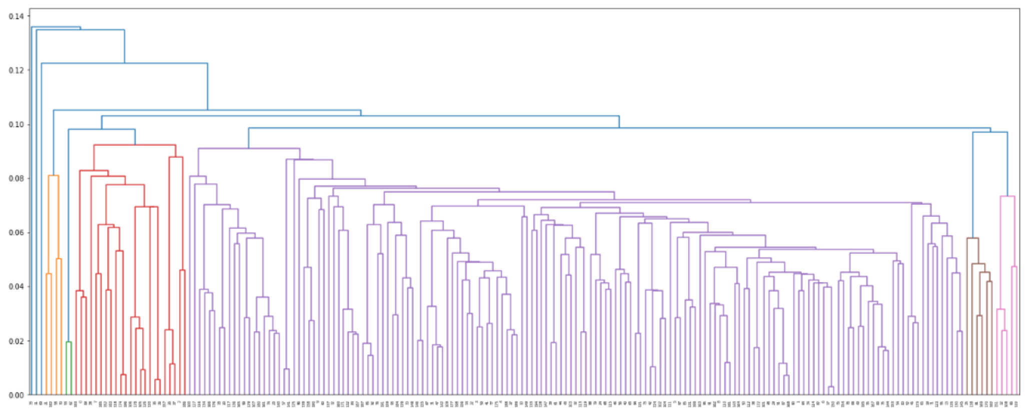


Figure 3: Dendrogram for 200 randomly generated points using single linkage method

Figure 5 below shows the dendrogram using complete linkage method. We have four big clusters, and we do not have any single point cluster left as we go up the hierarchy. Now we look at the scatter plots shown in Figure 6, we have 4 big clusters, and then merge to 3 clusters, and eventually merge to 2 clusters that are almost the

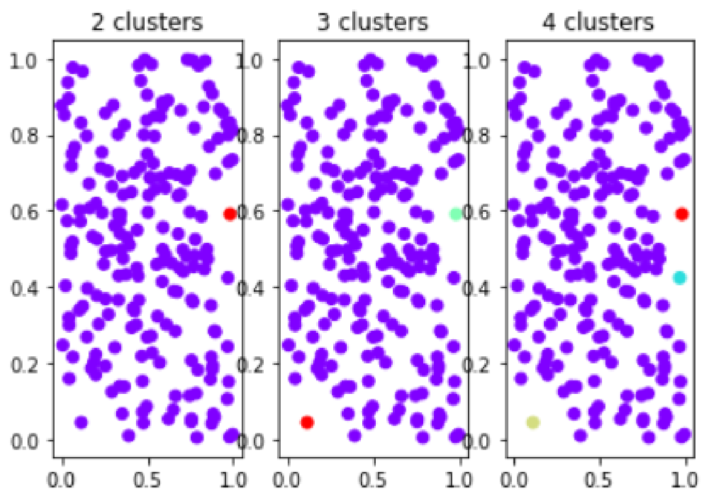


Figure 4: Left: Single linkage with 2 clusters;Middle: Single linkage with 3 clusters;Right: Single linkage with 4 clusters

same size.

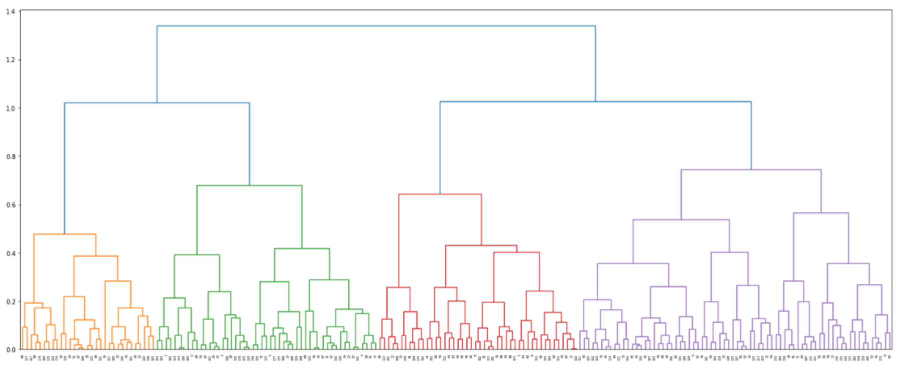


Figure 5: Dendrogram for 200 randomly generated points using complete linkage method

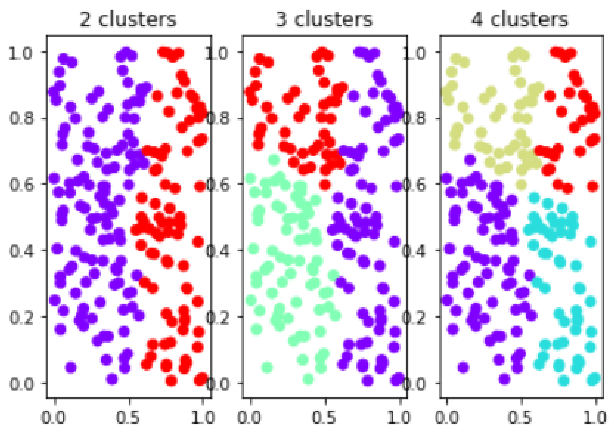


Figure 6: Left: Complete linkage with 2 clusters;Middle: Complete linkage with 3 clusters;Right: Complete linkage with 4 clusters

Figure 7 shows the dendrogram using average linkage method. We see that there are four big clusters formed without single-point clusters. This is similar to what we observe from complete linkage method. But the scatter plots shows that we have 2 clusters in the end with one of them much larger than the other.

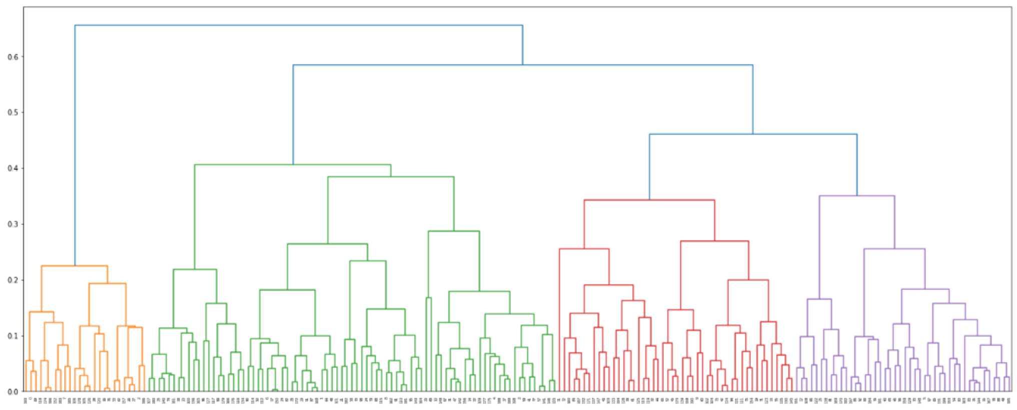


Figure 7: Dendrogram for 200 randomly generated points using average linkage method

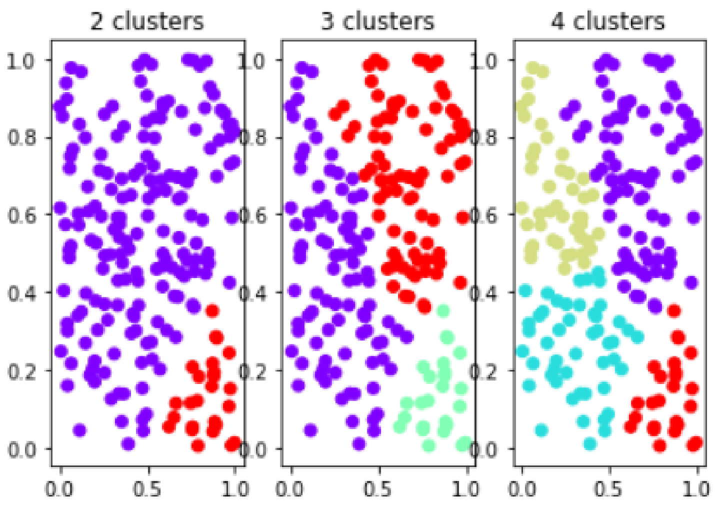


Figure 8: Left: Average linkage with 2 clusters;Middle: Average linkage with 3 clusters;Right: Average linkage with 4 clusters

In conclusion, we observe that single linkage method produces a huge cluster with several single-point cluster in the end; complete linkage method has some spatially compact clusters, and merge to 2 clusters that have similar size in the end; average linkage method is similar to complete linkage method that it produces several clusters, but they do not necessarily have the same size.

Divisive Hierarchical Clustering

It is exactly the opposite of the Agglomerative Hierarchical Clustering. Here we assign all data points as one single cluster and then separate the cluster to two clusters with the following steps:

1. Initially, we have all the data points. Treat them as one single cluster$B$, called the ”remainder” group.
2. For each point in cluster B, compute the average dissimilarity from all others in the cluster. We denote this quantity (*) Pick the one with the largest average dissimilarity. Put it into a new cluster A, called the ”splinter” group.
3. For each point in cluster B, compute the average dissimilarity between that point and all the points in cluster A. We call this quantity(**). If the difference (*-**) are all negative then we stop the process. If

there are some positive values, then take the point with the largest difference and put it into cluster A and repeat step 3. If we want more than 2 clusters in the end, we can keep going with this algorithm until we have all singleton clusters.

This is the most used Divisive Hierarchical Clustering algorithm called "Diana", proposed by MacNaughton-Smith, Williams, Dale, and Mockett. Figure 9 shows the dendrogram for 200 generated points using divisive

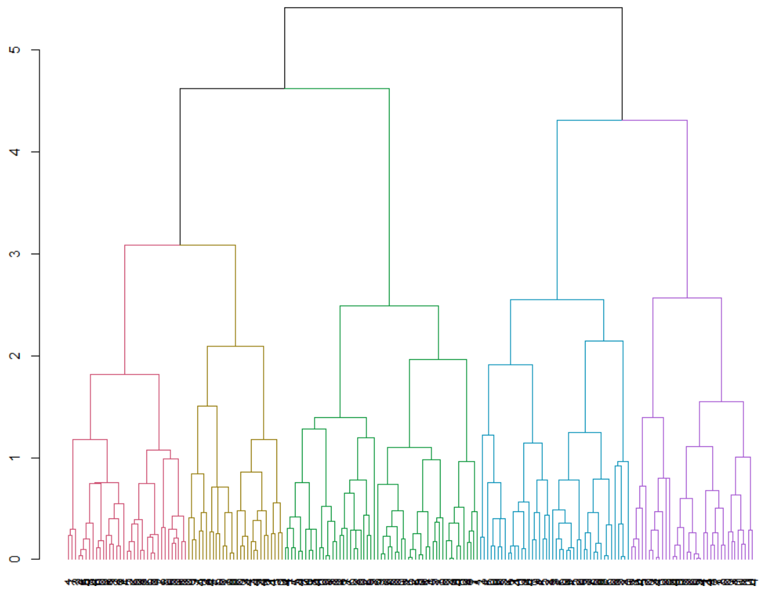


Figure 9: Dendrogram for 200 randomly generated points using divisive hierarchical clustering

hierarchical clustering. To view this dendrogram, we start at the top, and then traces down to the number of clusters that we desire. At the bottom, we have all the singleton clusters. Compared to the hierarchical clustering dendrograms above, we notice that this is similar to the one using complete linkage method because we have several big clusters in the middle of the dendrogram.
Now we look at some real data presented in Izenamn's book. The primate scapulae data consist of measurements on the scapulae of five genera of adult primates representing Hominoidea; that is, gibbons (Hylobates), orangutans (Pongo), chimpanzees (Pan), gorillas (Gorilla), and man (Homo).The measurements consist of indices and angles that are related to scapular shape, but not to functional meaning. Researches believe that these scapular shape measurements could be useful in classifying living primates.Figure 10 shows the dendrograms from the single-linkage, average-linkage, and complete-linkage agglomerative hierarchical methods and the dendrogram from the divisive hierarchical method. As we can see from the graph, the dendrogram for single linkage method has several singleton clusters even towards the end of the clustering process. But for complete linkage hierarchical clustering and divisive hierarchical method, they do not have isolated points. This corresponds to what we have observed previously that single linkage method tends to have more isolated points in the end, while complete linkage method and divisive hierarchical clustering tend to have big clusters that have similar sizes.

Conclusion

Divisive clustering is more complex, but more efficient if we do not generate a complete hierarchy all the way down to singleton clusters. Also, divisive algorithm is more accurate in global distribution of data when making top-level partitioning decision while Agglomerative clustering makes decisions by considering the local patterns without taking into account the global distribution data. As we have explored before, there are different linkage methods for agglomerative clustering which indicates that there are more patterns of data for users to choose from.

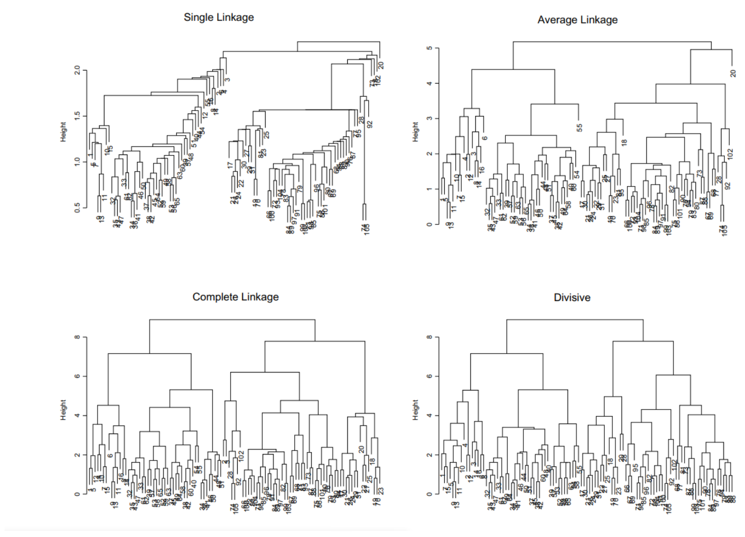
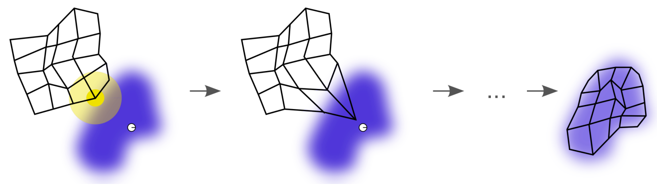


Figure 10: Dendrograms from hierarchical clustering of the primate scapulae data. Upper-left panel: single linkage. Upper-right panel: average linkage. Lower-left panel: complete linkage. Lower-right panel: divisive.

4 Self Organizing Maps

Definition: An unsupervised machine learning technique used to produce low-dimensional representation of a higher dimension. This could be thought of as a projection from a higher dimensional space onto a lower dimensional subspace, while preserving the topological structure.



This iterative adjustment that pulls the weight vectors of neighboring neurons towards each other is what makes the network self-organizing. This can be thought of as a combination of dimensionality reduction and clustering, where each cluster in n-dimensional feature space corresponds to a neuron in the 1 or 2 dimensional array. The self-organization allows us to see structure in the clustering.

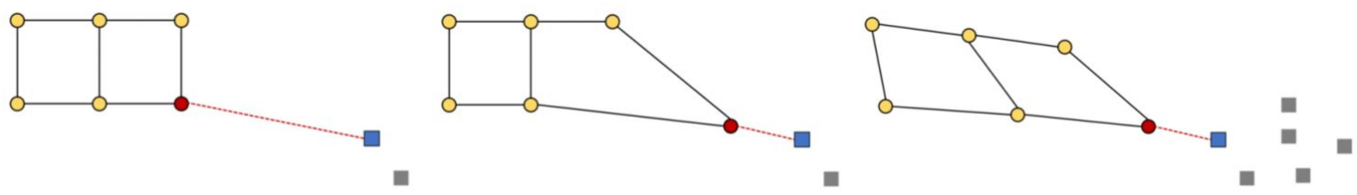
4.1 Overview

Start with a collection of observations. The process of self organizing maps first initializes a collection of neurons each with an associated weight vector in the n-dimensional feature space. The aim of the self organizing map algorithm is to recursively adjust the weight vectors, so that the weight vector for neighbor vectors are close in the feature space. Consequently, nearby observations in the data set are assigned to same neuron or nearby neurons. This iterative adjustment that pulls the weight vectors of neighboring neurons towards each other is what makes the network self-organizing. This can be thought of as a combination of dimensionality reduction and clustering, where each cluster in n-dimensional feature space corresponds to an observation. The self-organization allows us to see structure in the clustering.

4.2 Training Process

1. Initialize Neural Network Weights.
2. Randomly Select an input. [Data Point Coordinate]
3. Compute the Eucliden Distance between the selected point and all of the neurons.
4. Select the Winning Neuron. [Closest Distance]
5. Update the Neuron Weights.
6. Decrease the learning rate and the neighberhood size.
7. Repeat 2-6 until all the data points are selected.

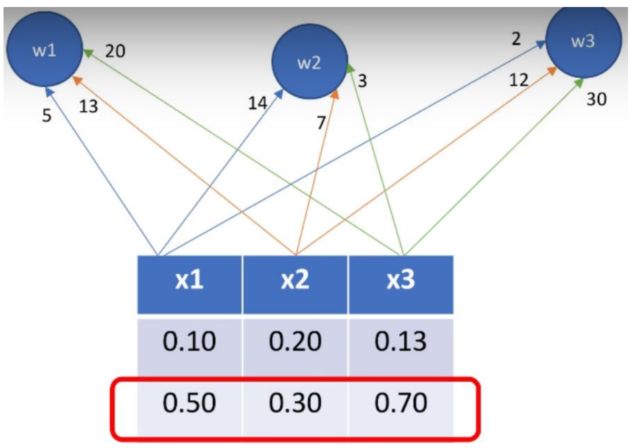
Table 2: One iteration of Training Process



Note: After each iteration of updating the weight vectors within the neighborhood the self organizing map algorithm decreases its radius. This happens after each full iteration, until the size of the neighborhood will shrink to the size of just one node. If a node is found to be within the neighborhood then its weight vector is adjusted.

4.3 Mathematical Example:

Lets say there are two points (0.10,0.20,0.13) and (0.50, 0.30, 0.70) in $\mathbb{R}^3$, and three neurons with corresponding weights between point and each neuron. We randomly select an input and compute the Eucliden Distances.



x1	x2	x3
0.10	0.20	0.13
0.50	0.30	0.70

Selected Input:

$$x_1 = .5, x_2 = .30, x_3 = .70$$

Weights:

Weight 1:

$$w_{1,1} = 5, w_{1,2} = 13, w_{1,3} = 20$$

Weight 2:

$$w_{2,1} = 14, w_{2,2} = 7, w_{2,3} = 3$$

Weight 3:

$$w_{3,1} = 2, w_{3,2} = 12, w_{3,3} = 30$$

Distances:

General Formula:

$$d_k = \sqrt{\sum_{i=1}^{n}(x_i - w_{ki})^2}$$

$$d_1 = \sqrt{\sum_{i=1}^{n}(x_i - w_{1i})^2} = \sqrt{(.5-5)^2 + (.3-13)^2 + (.7-20)^2} = 23.5$$

$$d_2 = \sqrt{\sum_{i=1}^{n}(x_i - w_{2i})^2} = \sqrt{(.5-14)^2 + (.3-7)^2 + (.7-3)^2} = \mathbf{15.2}$$

$$d_3 = \sqrt{\sum_{i=1}^{n}(x_i - w_{3i})^2} = \sqrt{(.5-2)^2 + (.3-12)^2 + (.7-30)^2} = 31.6$$

Results

d_2 is the winning distance, which corresponds to the winning neuron W_2 [Best Matching Unit]

What Next?

Apply the weight update formula to all of the neighborhood neurons and update the weights to attract in the direction of the Best Matching Unit, and recompute the Results until all of the observations have been processed.

4.3.1 Weight Update Formula

$$\Delta_{ij} = \eta(t) * T_{j,I(x)} * (x_i - w_{i,j})$$

where the **Learning Rate**:

$$\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_\eta}\right)$$

where the **Topological Neighberhood**:

$$T_{j,I(x)} = \exp\left(-\frac{D^2_{j,I(x)}}{2\sigma(t)^2)}\right)$$

where the **Neighberhood size**:

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_0}\right)$$

where the **Lateral Distance Between Neurons**:

$$D_{j,i} = ||w_j - w_i||$$

The parameters are t,i,j, I(x) where t: epoch, i:neuron, j:neuron, I(x):winning neuron
The hyper-parameters are : $\eta_0, T_\eta, T_0, \sigma$

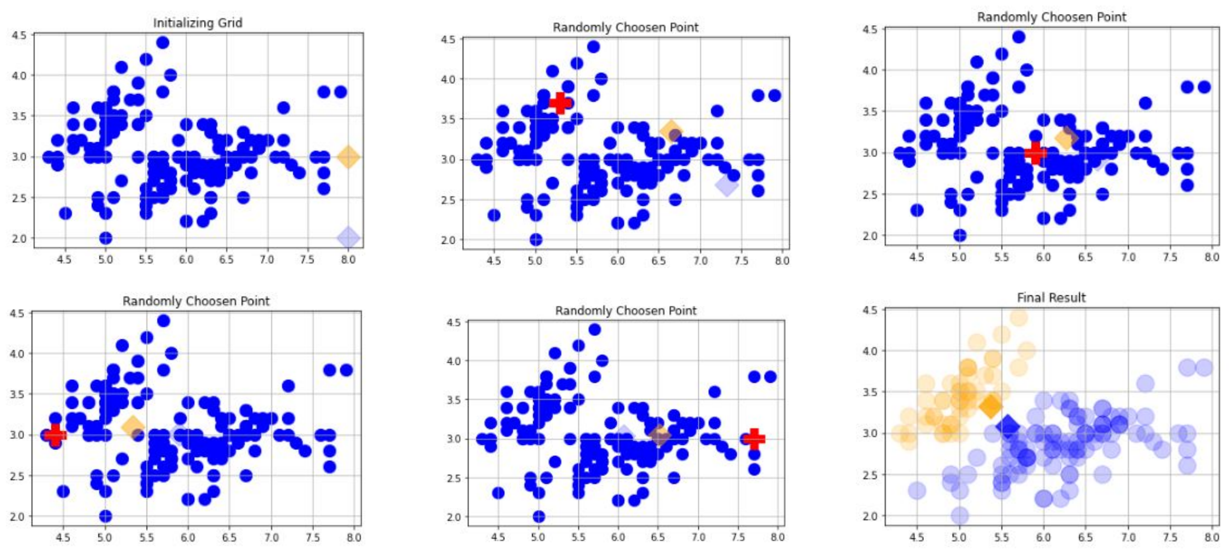
4.3.2 Basic Updating Weights Formula:

$$New_{Weights} = Old_{Weights} + Learning_{Rate}(Vector_{Input} - Old_{Weights})$$
$$w_{2,1} = 14 + 0.5(0.5 - 14) = 7.5$$
$$w_{2,2} = 7 + 0.5(0.30 - 7) = 3.65$$
$$w_{2,3} = 3 + 0.5(0.70 - 3) = 1.85$$

4.4 Self Organizing Maps Demonstration

4.4.1 Two neuron Demo

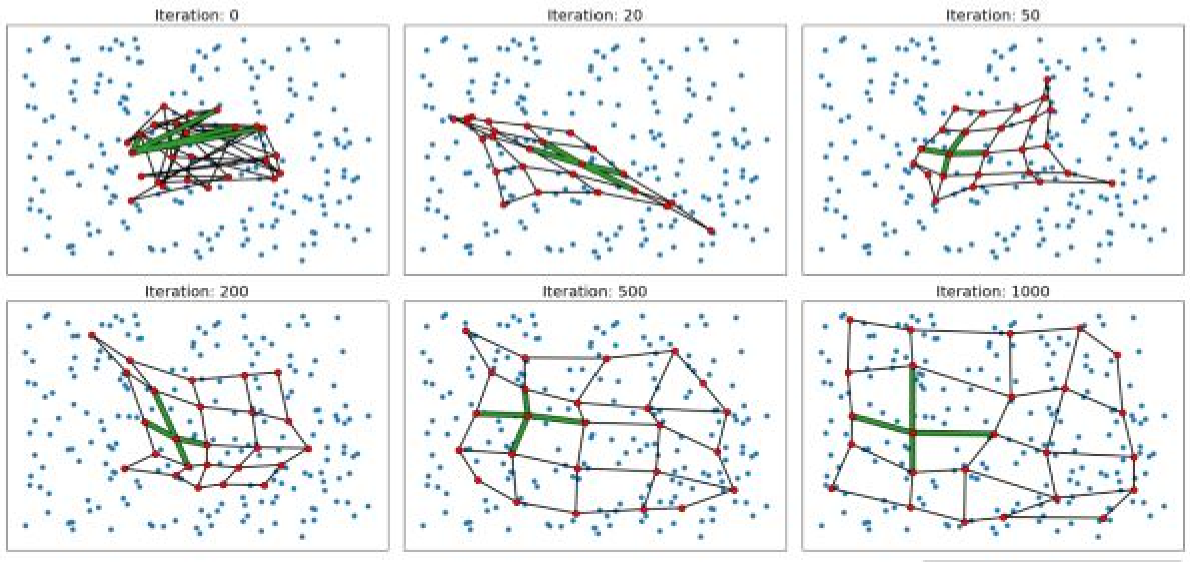
Table 3: Four iterations of Self Organizing Map Algorithm with two neurons



Note: This shows the competitive learning process for each winning neuron attracting towards a randomly chosen data point, until all the data points have been exhausted. The points nearest to the yellow neuron were classified orange, the other points closest to the blue neuron were classified as blue.

4.4.2 Twenty-Five Neuron Demo

Table 4: 1000 iterations of Self Organizing Map Algorithm with 25 Neurons



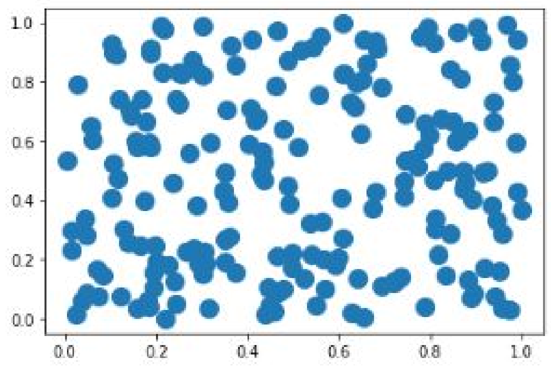
Note: Initially the learning rate is high, with the lattice adjusting rapidly, but then each iteration the weight update formula's neighborhood and learning rate decay. The grid then eventually converges on a topological structure and the training process is complete.

Coding Example Frequency Plot:

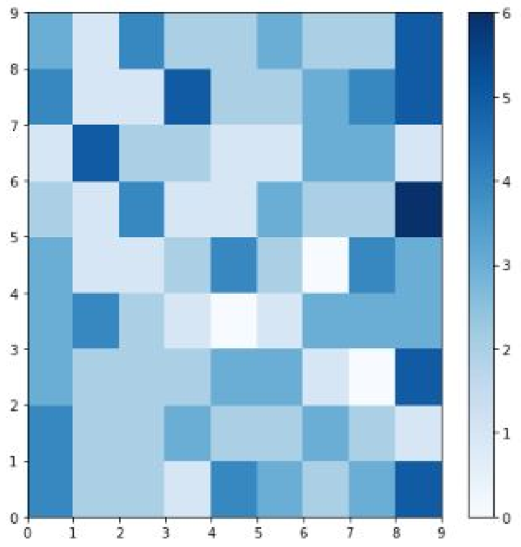
```
!pip install minisom
from minisom import MiniSom
import random
import matplotlib.pyplot as plt
import numpy as np
data = np.concatenate((np.random.random((100,2)), (np.random.random((100,2)))), axis=0)
n_neurons =9
m_neurons =9
som = MiniSom(7,7,len(data[1]),sigma=1.0,learning_rate=0.5)
som.random_weights_init(data)
som.train_random(data,100) # training with 100 iterations
som = MiniSom(n_neurons, m_neurons, len(data[1]), sigma=1.5, learning_rate=.5,
neighborhood_function='gaussian', random_seed=0)

som.pca_weights_init(data)
som.train(data, 10000, verbose=True)  # random training
plt.figure(figsize=(7, 7))
frequencies = som.activation_response(data)
plt.pcolor(frequencies.T, cmap='Blues')
plt.colorbar()
plt.show()
plt.scatter(data [: ,0] ,data[:,1], s=150)
```

Table 5: Randomly generated data points



Scatter Plot



Frequency Plot

Iris Data Set U-matrix:

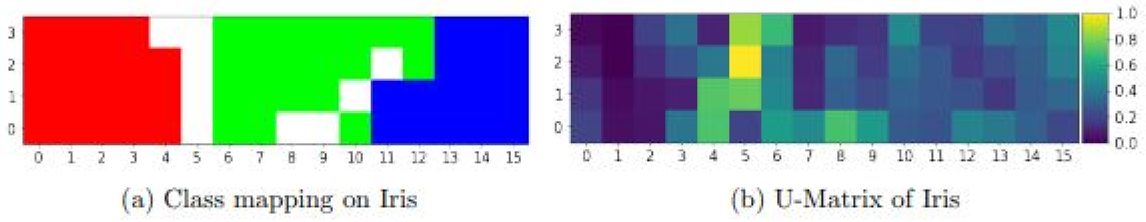
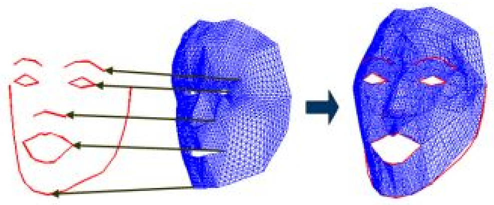


Figure 11: The darker colored regions in the U-matrix emphasize a high density, while the lighter colored regions show a low density. In the U-matrix plot between the red and green there is a low density region splitting the classes nicely, but between the green and the blue region there is a mixture of classes so there is no concrete split of the two classes.

Interesting Application:



Levente Saj´o, Mikl´os Hoffmann, and Attila Fazekas decided to train a self organizing map on a set of stereo images to accurately produce 3D models that clustered emotions. This was quite an interesting research paper showing mappings from 2d to 3d points for the purpose of emotion recognition. This is of particular interest for marketing and different business applications.

5 Clustering Variables

We are able to use the same clustering methods for variables as we did for clustering observations with the exception being the measure of distance between variables. We generally use a distance matrix based upon the correlation matrix for the r variables. A pair of variables with relatively large correlation are thought to be "close" to each other. If we standardize each of the r variables across the n observations to have zero mean and unit variable, then it is not difficult to show that

$$\frac{1}{2(n-1)}\sum_{i=1}^{n}(x_{ji}-x_{ki})^2 = 1-\rho_{jk} \in [0,2],$$

where ρ_{jk} is the sample correlation between variables X_j and X_k. This shows us that using squared Euclidean distance, $\sum_i (x_{ji}-x_{ki})^2$, is equivalent to using $1-\rho_{jk}$ as a dissimilarity measure.

5.1 Gene Clustering

Clustering the thousands of genes measured using a microarray experiment is a popular use of variable clustering. The idea of gene clustering is grouping together genes that share similar expression patterns. The $(r \times n)$ data matrix $\chi = (x_{ij})$ contains the gene-expression data derived from a microarray experiment, where i indexes the row (gene), j indexes the column (tissue sample). Thus, x_{ij} is a measurement of how strongly the ith is expressed in the jth sample.

5.2 Principal-Component Gene Shaving

Suppose the goal is to discover a gene cluster that has high variability across samples. Let $\mathcal{S}_k$ denote the set of indices of a cluster of k genes. Consider the jth tissue sample and compute the average gene-expression over the k genes for that sample,//

$$\overline{x}_{j,\mathcal{S}_k} = \frac{1}{k}\sum_{i\in\mathcal{S}_k} x_{ij}, j = 1,2,\ldots,n.$$

Compute the average over all n tissue samples,

$$\overline{x}_{\mathcal{S}_k} = \frac{1}{n}\sum_{j=1}^{n}\overline{x}_{j,\mathcal{S}_k} = \frac{1}{kn}\sum_{j=1}^{n}\sum_{i\in\mathcal{S}_k} x_{ij}.$$

The empirical variance of $\overline{x}_{\mathcal{S}_k}$ is defined by

$$v\hat{a}r\{\overline{x}_{\mathcal{S}_k}\} = \frac{1}{n}\sum_{j=1}^{n}(\overline{x}_{ij},\mathcal{S}_k - \overline{x},\mathcal{S}_k)^2.$$

Given all possible clusters of size k, we can search for that cluster $\mathcal{S}_k$ with the highest $v\hat{a}r\{\overline{x}_{\mathcal{S}_k}\}$. Unfortunately, such a search procedure is computationally infeasible because it entails evaluating $\binom{r}{k}$ different subsets, which gets big very quickly for r large, as would be common in gene clustering.

Gene shaving is a method for clustering genes, in which the goal is to identify small subsets of highly correlated genes that vary as much as possible between samples. This method differs from those previously described because the genes are allowed to be included as members of more than one cluster.
Consider a linear combination of the jth column gene expressions,

$$z_j = \mathbf{a}^\tau \mathbf{x}_j = \sum_{i=1}^{r} a_i x_{ij},$$

where $\mathbf{x}_j = (x_{1j}, ..., x_{rj})^\tau$, $\mathbf{a} = (a_1, ..., a_r)^\tau$, the $\{a_i\}$ are positive, negative, or zero weights, and $\sum_{i=1}^{r} a_i^2 = 1$. The goal is to find the coefficients $\{a_i\}$ such that the variance of Z_j is maximized.
The solution is given by the first principal component (PC1) of the r rows of χ are referred to as eigengenes. We can eliminate unimportant genes by computing the inner product (or correlation) of each gene with PC1 and "shave off" those genes (rows of χ) with the $100\alpha\%$ smallest absolute inner products. This shaving process decreases the size of the set of available genes, say to k_1 genes. We then perform the shaving process on that subset to get a subset k_2. Thus, we eventually get a finite sequence of nested gene clusters, $\mathcal{S}_r \supset \mathcal{S}_{k_1} \supset \mathcal{S}_{k_2} \supset \ldots \supset \mathcal{S}_1$.
Next, we need to decide on k and $\mathcal{S}_k$. For a given k, we define the ANOVA-type decomposition of the total variance,

$$V_T = \frac{1}{kn} \sum_{i \in \mathcal{S}_k} \sum_{j=1}^{n} (x_{ij} - \overline{x}_{\mathcal{S}_k})^2 = V_B + V_W,$$

where

$$V_B = \frac{1}{n} \sum_{j=1}^{n} (\overline{x}_{j,\mathcal{S}_k} - \overline{x}_{\mathcal{S}_k})^2,$$

$$V_W = \frac{1}{n} \sum_{j=1}^{n} [\frac{1}{k} \sum_{i \in \mathcal{S}_k} (x_{ij} - \overline{x}_{j,\mathcal{S}_k})^2]$$

are the between-variance and within-variance, respectively. The percentage of the total variance explained by the gene cluster $\mathcal{S}_k$ can be represented as the natural statistic

$$R^2(\mathcal{S}_k) = \frac{V_B}{V_T} \times 100$$

(the larger the value of R^2, the more coherent the gene cluster). A permutation argument is applied to the R^2-value in the above function to determine the cluster size k. The "significance" of the R^2-value is judged by comparing it with its expectation computed under a suitable reference null distribution, which assumes the rows and columns of χ are independent. When we randomly permute the elements of each row of χ to get χ^*. If we do this B times to get χ^{*b}, b = 1,2, ..., B. We apply the shaving process to get $\mathcal{S}_k^{*b}$ and then compute $R^2(\mathcal{S}_k^{*b})$, b = 1,2, ..., B.
The *gap statistic* is defined by

$$Gap(k) = R^2(\mathcal{S}_k) - \overline{R^2(\mathcal{S}_k^*)}$$

where $\overline{R^2(\mathcal{S}_k^*)}$ is the average of the all $\{R^2(\mathcal{S}_k^{*b})$, b = 1,2, ..., $B\}$. The value of k that maximizes the gap is $\hat{k}$. An important graphical technique is to plot the *gap curve*, which is the plot of *Gap*(k) against cluster size k. Set $\hat{k} = \hat{k}(1)$. After determining the number, $\hat{k}(1)$, of genes and their identities, we look for a second gene cluster. First, we remove the effects of the first cluster gene. The first *supergene* is computed, $\overline{\mathbf{x}}^{(1)} = (\overline{x}_1^{(1)}, ..., \overline{x}_n^{(1)})^\tau$, a row vector of average genes corresponding to the first cluster $\mathcal{S}_{\hat{k}^{(1)}}$, where $\overline{x}_j^{(1)} = \sum_{i\ in \mathcal{S}_{\hat{k}^{(1)}}} x_{ij}/\hat{k}^{(1)}, j = 1, 2,$ ..., r. We then orthogonalize χ by regressing each row of χ on the supergene $\overline{\mathbf{x}}^{(1)}$ and replacing the rows of χ by the residuals from each such regression, which gives the matrix χ_1. We run the shaving process on χ_1 and use the gap statistic to obtain $\hat{k}^{(2)}$, the second gene cluster $\mathcal{S}_{\hat{k}^{(2)}}$, and the second supergene $\overline{\mathbf{x}}^{(2)}$. We repeat this a total of t times, where t is prespecified, by modifying χ and $\overline{\mathbf{x}}$ each step.

5.3 Example: Colon Cancer Data

The colon cancer data is a set of 92 genes (columns) and 62 tissue samples (rows). The gene expression heatmap for the color cancer data is displayed in Figure 12.

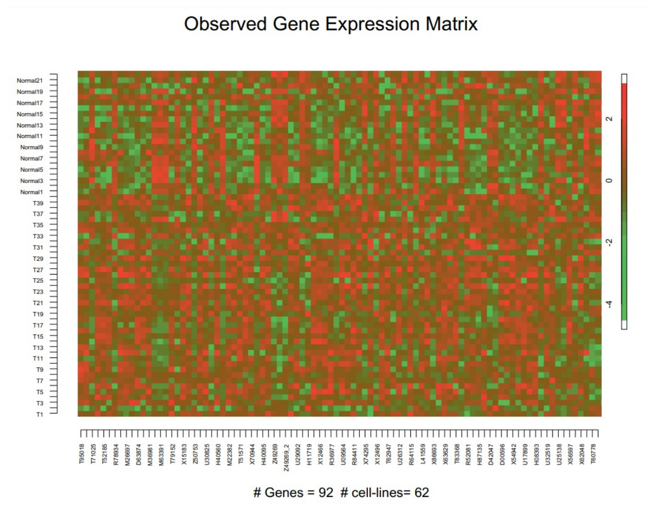


Figure 12: Gene expression heatmap of 92 genes (columns) and 62 tissue samples (rows) for the colon cancer data. The tissue samples are divided into 40 colon cancer samples (T1–T40) and 22 normal samples (Normal1–Normal22).

We can apply the PC gene-shaving to the color cancer microarray described above. Figure 13 shows the gap curves for the first four clusters derived using the gene-shaving algorithm.

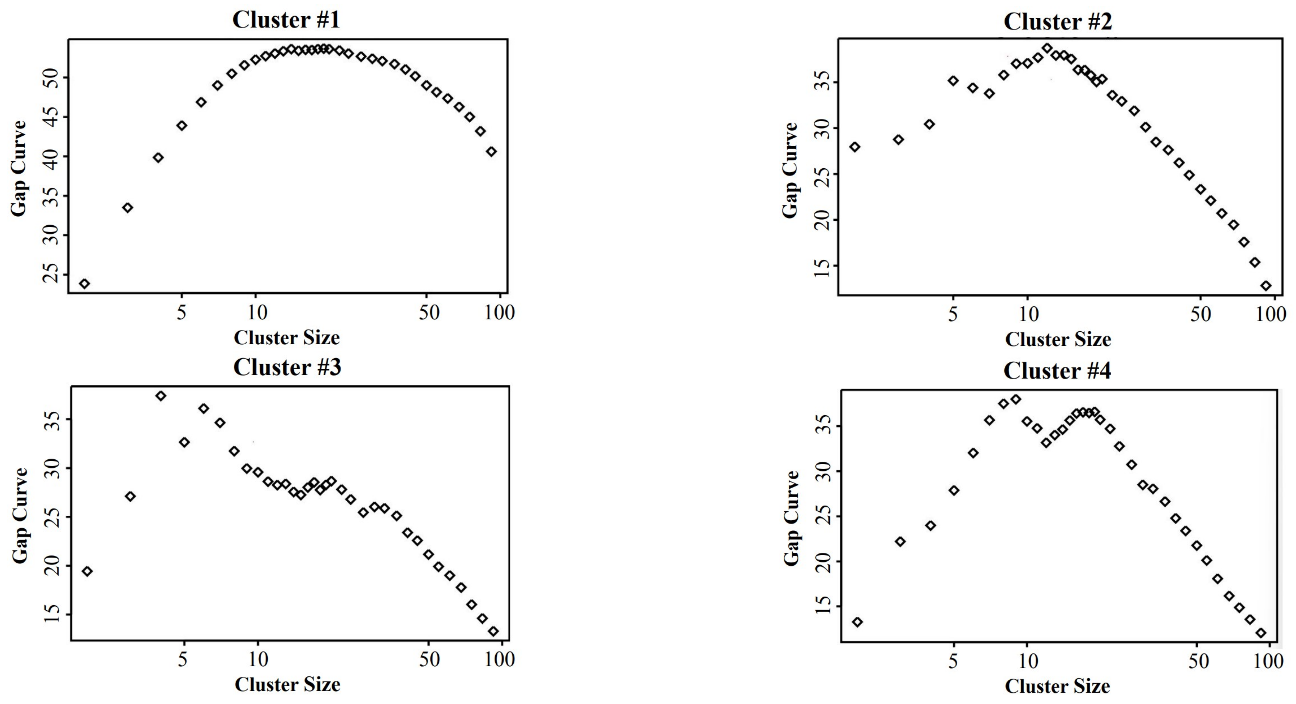


Figure 13: Gap curves for the first four clusters of colon cancer data. The gap estimate of cluster size is that value of k for which the gap curve is a maximum. The estimated cluster sizes are first cluster (top-left panel), 41; second cluster (top-right panel), 15; third cluster (bottom-left panel), 6; and fourth cluster (bottom-right panel), 19.

For each cluster, the value of k at which the gap curve attains its maximum is chosen to be the estimated size of the cluster.

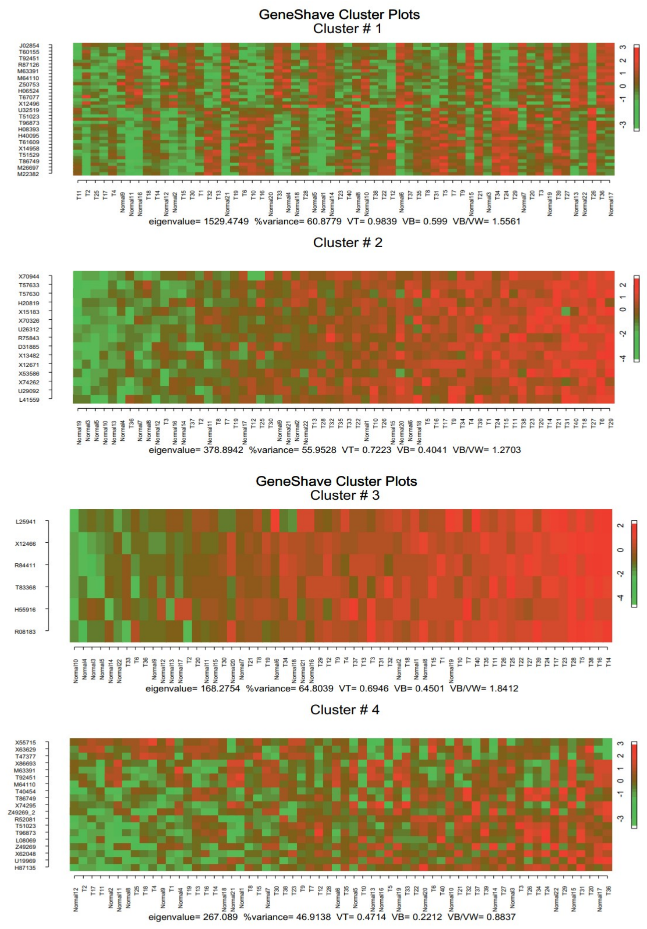


Figure 14: Heatmaps for the first four gene clusters for the colon cancer data, where each cluster size is determined by the maximum of that gap curve. The genes are the rows and the samples are the columns. The samples are ordered by the values of the column averages.

The estimated cluster sizes are 41, 15, 6, and 19. The four heatmaps for those gene clusters are demonstrated in Figure 14, where the samples are ordered by the values of the column averages. Each panal gives the values of the total variance V_T, the between-variance V_B, the ratio V_B/V_T, and $R^2 = V_B/V_T \times 100\%$, the percentage of total variance explained by that cluster. The largest R^2 value was that of the third cluster at 64.8%. The clusters in Figure 13 display different patterns of gene expression. The first cluster has an interesting feature in that the genes split into two equal-size subgroups. In other words, for a given tissue sample, when the “upper” subgroup of genes are strongly upregulated (red color), the “lower” subgroup are strongly downregulated (green color), and vice versa. Furthermore, whether the sample is a tumor sample or a normal sample impacts the red/green split. Next, the second and third clusters of genes have the same overall appearance. Specifically, the tumor samples tend to be upregulated, whereas normal samples tend to be downregulated. Finally, in the fourth cluster, the reds and greens are somewhat more randomly sprinkled around the heatmap, although there are pockets of adjacent cells that seem to share similar expression patterns.

6 Individual Statements

6.1 Mutasim Mim

6.2 Qinying Chen

I worked on hierarchical clustering. First I introduce what a dendrogram is, and then I explain the algorithms for agglomerative hierarchical clustering and divisive hierarchical clustering. For the linkage method, I use python built-in package for agglomerative clustering, and RStudio built-in package diana for the divisive clustering, just to figure out what the pattern is for each linkage method by generating some random data points. Then I compare with the dendrogram shown in Izemann's book that uses primate scapulae data.

6.3 Thang Xuan Nguyen

Wayne had personal problems so he was unfortunately not able to work on the project with us; however, he participated in the brainstorming sessions and inputted ideas on what we could do.

6.4 Matthew Benvenuto

I had worked on some basic definitions on the initial page, finishing touches for the introduction, and the self organizing maps section. In the self organizing maps section, I provided an introduction, an overview, a labeled algorithmic description of the training process, along with a mathematical example, and a detailed description of the weight update formula. I then coded a demonstration of two neurons on the iris data set and showed 100 iterations of the updating neurons to make a decision on how to classify the data points. I also wrote code to show a frequency plot of a U-matrix.

6.5 Morgan Acevedo

I worked on the clustering variables sections and the introduction. In the introduction, I described different definitions and data sets that were important in multiple sections so there was no overlapping of information. In the clustering variables section, I described what clustering variables are and how to cluster variables. I focused on gene clustering and the gene shaving method similar to the textbook. Finally, I was unable to locate the exact data set the textbook used to create my own images, so I used what the textbook demonstrated.

References

[1] Izenman, A. J. (2013). Modern multivariate statistical techniques: Regression, classification, and Manifold Learning. Springer.

[2] Hastie, T., Friedman, J., &; Tisbshirani, R. (2017). The elements of Statistical Learning: Data Mining, Inference, and prediction. Springer.

[3] Hastie, T., Tibshirani, R., Eisen, M. B., Alizadeh, A., Levy, R., Staudt, L., Chan, W. C., Botstein, D., &; Brown, P. (2000, August 4). 'gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. Genome biology. Retrieved May 23, 2022, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC15015/