1 K-Nearest Neighbors Algorithm

In the learning algorithms we have seen, we select a hypothesis space and learn a set of parameters $(\theta_0, \theta_1, \dots, \theta_d)$ or a set of weights $(w_0, w_1, \dots, w_d)$ from the training data so that the target function $f(\bar{x})$ is approximated as $\hat{y} = h_\theta(\bar{x})\ or\ h_w(\bar{x})$, where $\bar{x}$ is the training sample and f is the classification or regression function. $\hat{y}$ is the learned version of the target function. Once the model parameters are computed, we do not need the training data. These methods are called parametric models.

A non-parametric model is one that cannot be characterized by a fixed set of parameters. A family of non-parametric models is known as Instance Based Learning methods. Instance based learning is based on the memorization of the dataset. The learning phase is absent, the cost is in the computation of the prediction. This kind learning is also known as lazy learning. However, the dataset should always be available.

1.1 K-NN Algorithm

K-NN classification falls in the supervised learning family of instance-based algorithms or lazy learning algorithms. K-nearest neighbors uses the notion of local neighborhood to obtain a prediction. The k-Nearest Neighbors (k-NN) algorithm is a simple, yet powerful, supervised learning algorithm used for both classification and regression tasks,

i. The value of 'k', which is the number of nearest neighbors to consider is decided first. This is a hyperparameter to be selected before running the algorithm.

ii. For a given test point (the data point to be classified or for which the target value in regression is to be predicted), the distance between this test point and all the points in the training dataset is calculated.

   The distance measures that are normally used are Euclidean distance and Manhatten distance measures. Let $X = < x_1, \dots, x_j, \dots x_d >$ be the training sample. Let $Z = < z_1, \dots, z_j, \dots z_d >$ is the test sample. The Euclidean distance $d(X, Z)$ is calculated as

$$d(X, Z) = \sqrt{\sum_j w_j (x_j - z_j)^2}$$

   The Manhatten distance is calculated as

$$d(X, Z) = \sum_j w_j |x_j - z_j|.$$

   $w_j$is the weight given to the sample j depending upon the nature of the dataset. Normally, it is equal to 1.

iii. Identify the 'k' points in the training data that are closest to the test point. These are the 'k' nearest neighbors.

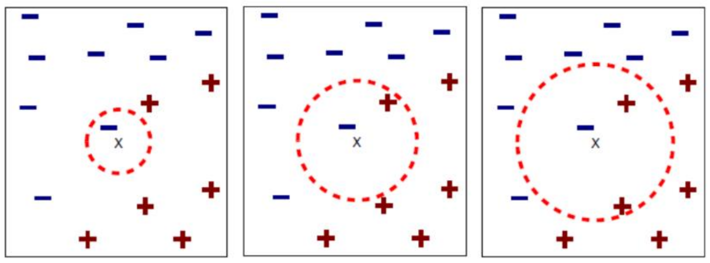
iv. For classification purposes, count the number of neighbors belonging to each class. The class with the highest count among the 'k' neighbors is assigned to the test point.

v. For regression tasks, calculate the average (or sometimes the weighted average) of the target values of the 'k' nearest neighbors. The result is the predicted value for the test point in regression.

The k-NN algorithm is intuitive and easy to implement, but it can be computationally expensive for large datasets, as it involves calculating distances between the test point and all training points. It's also sensitive to the choice of 'k' and the distance metric used.

```
Algorithm:The k-nearest neighbor classification algorithm
Input:    Let k be the input number of nearest neighbors and D
          be the set of training examples $\{\langle \overline{X}, Y \rangle\}$. Let $\overline{Z}$ be the
          test sample to be classified
for each training sample $\overline{X}$ do
    Compute distance$d = (\overline{X}, \overline{Z})$ and place $(d, \langle \overline{X}, Y \rangle)$ in L
Select $L_k$,the set of k closest Training examples/classes from L
$Y' =$ Class with the highest frequency in $L_k$
```

The following diagram shows the samples with two features of a dataset plotted on the $X - Y$ plane. The sample X represents the test point. In (a) the nearest neighbor is a negative class sample. In (b) the nearest neighbors belong to each of the classes. In (c) two nearest neighbors belong to the positive class and one of the neighbors belong to the negative class.



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

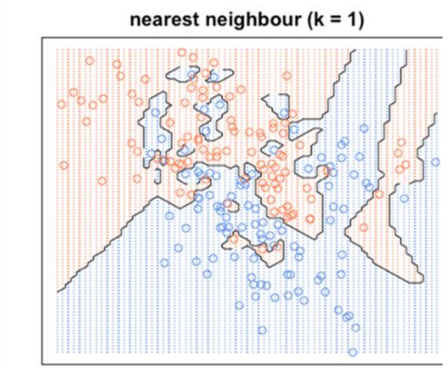
K-nearest neighbors of a record x are data points
that have the k smallest distance to x

The following is a worked-out example of applying K-NN algorithm for the finding the class of a test sample. The dataset and the distances of the test sample are shown. With $k = 3$, it can be observed that the target class of the test sample in the last row of the dataset is 1.

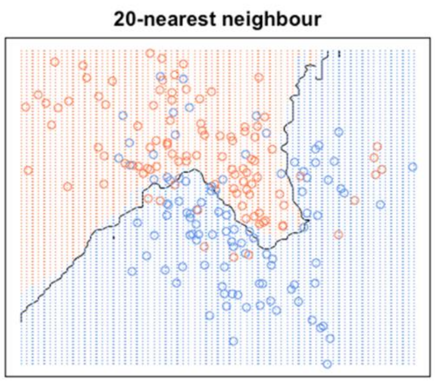
SL	SW	Class
7.2	3	0
7.4	2.8	0
7.9	3.8	0
6.4	2.8	0
6.3	2.8	0
6.3	2.5	1
6.1	2.8	1
6.4	2.9	1
6.6	3	1
6	2.6	?

SL	SW	Class	d
7.2	3	0	1.26
7.4	2.8	0	1.414
7.9	3.8	0	2.247
6.4	2.8	0	0.447
6.3	2.8	0	0.360
6.3	2.5	1	0.316
6.1	2.8	1	0.223
6.4	2.9	1	0.5
6.6	3	1	0.721
6	2.6	?	1

Nearest neighbor algorithm is easily misled by noisy/irrelevant features. $K = 1$ captures the finest structure of the input samples but is vulnerable to noise. The diagram shows the predicted classes for a two-class dataset with two features. It may be observed that the classification accuracy is very high.



The following diagram shows the predicted boundaries for the same dataset with $k = 20$. The boundary is more generalized. However, the training accuracy is low. It can be easily concluded by observing the spread of the data that the misclassified examples belong to noise.



2 Unsupervised learning-Clustering

Unsupervised learning is a machine learning technique used to find patterns and relationships in data without target labels. It involves exploring the structure of data to extract meaningful information. Common tasks include clustering, dimensionality reduction, and anomaly detection. Clustering is a fundamental task in unsupervised learning. It involves grouping similar data points together based on some similarity measure. The goal is to partition the data into distinct groups, or clusters, where data points within the same cluster are more similar to each other than to those in other clusters.

In hard clustering, each data point is assigned to exactly one cluster. The assignment is based on a distance metric or similarity measure, such as Euclidean distance or cosine similarity. K-means is a popular algorithm for hard clustering, where the goal is to minimize the within-cluster sum of squares.

Soft clustering, also known as fuzzy clustering, allows data points to belong to multiple clusters with varying degrees of membership. Instead of assigning data points to a single cluster, each point is assigned a probability of membership value for each cluster. Expectation-Maximization (EM) is an algorithm for soft clustering.

Clustering has numerous applications across various domains, including:

i. Customer segmentation in marketing.
ii. Document clustering in natural language processing.
iii. Image segmentation in computer vision.
iv. Anomaly detection in cybersecurity.

2.1 K-means Clustering

Given a training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}, x^{(i)} \in R^n$, the output is k (which is a hyper-parameter) cohesive clusters. No labels $y^{(i)}$ are given. Given a new x find out which cluster does it belong to?

k-Means clustering is a popular unsupervised learning algorithm used to partition a dataset into k distinct, non-overlapping subsets (clusters). The algorithm works as follows.

Choose the number of clusters, k, that the dataset is to be partitioned into.

Randomly select k points from the dataset as initial cluster centroids. These points can be actual data points or chosen randomly within the data space.

Assignment Step: Assign each data point in the dataset to the nearest cluster centroid. The nearest centroid is typically determined using the Euclidean distance, although other distance metrics can be used. This creates k clusters, each containing the points closest to one of the k centroids.

Update Step: Recalculate the centroids of each cluster thus formed. The new centroid is the mean (average) of all the points assigned to that cluster. Specifically, for each cluster, sum the coordinates of all points in the cluster and then divide by the number of points in that cluster. This gives the new centroid.

Repeat Step: Repeat the Assignment and Update steps until the centroids no longer change significantly, or until a maximum number of iterations is reached. This means the algorithm has converged, and the clusters are stable. Once the algorithm has converged, it outputs the final positions of the centroids and the clusters of data points.

Prediction Step: Given a test data point, calculate the distance of the test sample from the cluster centroids. The test sample is attributed with the cluster to which it is found to be the nearest.

Consider the following example. The dataset comprises of 2D samples. The test sample is at the end of the table.

x_1	x_2
1	2
5	1
8	1
2	1
6	1
8	2
2	2
7	1
9	1
2	3
6	2
9	2
3	2
6	3
10	1
3	4
6	4
10	2
4	5

Let $k = 3$. Assume the three Centroids as,

$\mu_1 = (3,4), \mu_2 = (5,1), \mu_3 = (8,2)$

The assignment step results as follows.

x_1	x_2	c
1	2	1
5	1	2
8	1	3
2	1	2
6	1	2
8	2	3
2	2	1
7	1	3
9	1	3
2	3	1
6	2	2
9	2	3
3	2	1
6	3	2
10	1	3
3	4	1
6	4	3
10	2	3
10	9	

$$d(x^1, \mu_1) = \sqrt{4+4} = \sqrt{8}$$
$$d(x^1, \mu_2) = \sqrt{16+1} = \sqrt{17}$$
$$d(x^1, \mu_3) = \sqrt{49+0} = \sqrt{49}$$

The update step results as follows.

$\mu_1 = (2.2,2.6), \mu_2 = (5,1.6), \mu_3 = (8.3,1.7)$

The process is repeated till the cluster allocations of samples do not change or for the maximum number of iterations is reached. The following is the algorithm.

```
Initialize k centroids $\mu_1, \mu_2, \dots, \mu_k \in R^n$
Repeat until convergence (no new cluster assignments)
    for each $x^{(i)}$
        find $c^{(i)} = j \ni arg\min_j d(x^{(i)}, \mu_j)$
    for each $j$
        $\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} == j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} == j\}}$
```

2.2 Hierarchical Clustering

In contrast to K-means clustering, this method builds a hierarchy of clusters either in a bottom-up (agglomerative) or top-down (divisive) manner. **Agglomerative Clustering** starts with each data point as a separate cluster and merges the closest pairs of clusters step by step until all points are in one cluster. **Divisive Clustering** starts with all data points in a single cluster and recursively splits the cluster into smaller clusters.

Agglomerative Clustering:

Initially, each cluster contains a single point and the distance can be calculated as Euclidian distance between the clusters/data points. However, as we merge the datapoints into clusters, the question arises regarding the way distance is to be calculated between two clusters, each comprising of several data points. The following explains the same.

Cluster distance measures.

Single link: smallest distance between the points in the clusters.

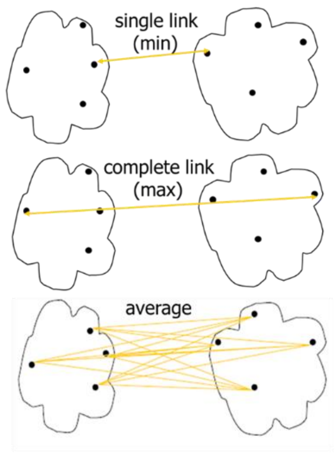
$$d(C_i, C_j) = \min\{d(x_{ip}, x_{jq}\}$$

Complete link: smallest distance between the points in the clusters.

$$d(C_i, C_j) = \max\{d(x_{ip}, x_{jq}\}$$

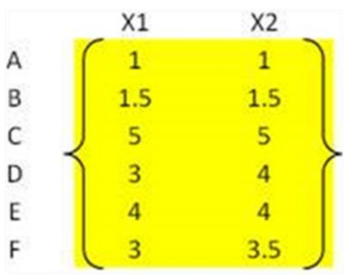
Average link: smallest distance between the points in the clusters.

$$d(C_i, C_j) = \text{avg}\{d(x_{ip}, x_{jq}\}$$

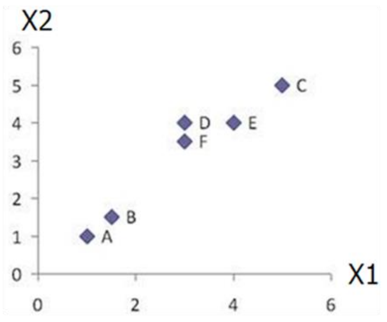


The dataset comprises of six 2D samples. The objective is to perform Agglomerative clustering. Initially, each data point is in its own cluster. The process of agglomerative clustering is implemented with the help of a distance matrix. Initially, the distance matrix comprises of rows and columns corresponding to each data point. The cells contain the distance between the points representing the row and column of each cell. Of course, it would be enough to fill either the lower or higher diagonal matrix.

Consider the following example dataset.



$$\begin{array}{c|cc} & X1 & X2 \\ A & 1 & 1 \\ B & 1.5 & 1.5 \\ C & 5 & 5 \\ D & 3 & 4 \\ E & 4 & 4 \\ F & 3 & 3.5 \end{array}$$

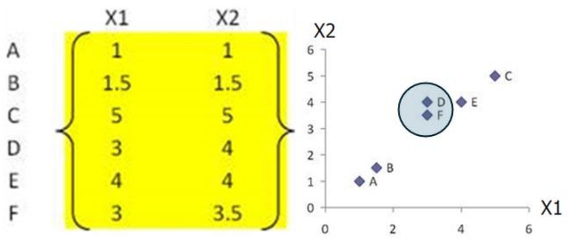


The distance matrix is calculated as follows. We find that the (single point) clusters $D \text{ and } F$ are nearest and hence are merged. The new distance matrix will have the rows/columns corresponding to $D \text{ and } F$ as merged. Distance measure is made in accordance with **Single Link.**

d	A	B	C	D	E	F
A	0					
B	0.77	0				
C	5.66	4.95	0			
D	3.61	2.92	2.24	0		
E	4.24	3.54	1.41	1.00	0	
F	3.20	2.50	2.50	0.50	1.12	0

$$d_{AB} = \sqrt{(x_1^A - x_1^B)^2 + (x_2^A - x_2^B)^2}$$

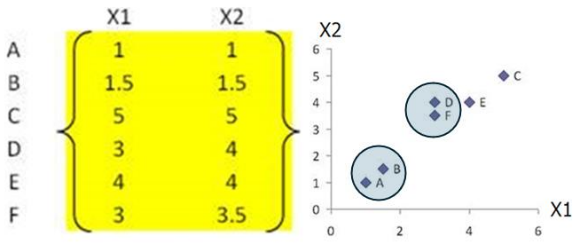
Merge Closest Clusters



$$\begin{array}{c|cc} & X1 & X2 \\ A & 1 & 1 \\ B & 1.5 & 1.5 \\ C & 5 & 5 \\ D & 3 & 4 \\ E & 4 & 4 \\ F & 3 & 3.5 \end{array}$$

If $D \text{ and } F$ are merged, then we have to recompute the distances of the clusters $A, B, C, \text{and } E$ from the cluster $\{D, F\}$. For example, to calculate the distance between the clusters $A \text{ and } \{D, F\}$, we compute the minimum of distance between $(A, D) \text{ and } (A, F)$. The resulting matrix appears below. In the new matrix, we find that the distance between $A \text{ and } B$ clusters is the minimum and hence will be merged in the next step.

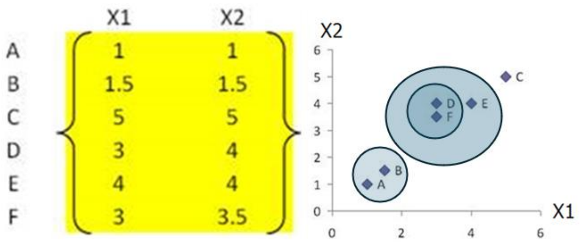
d	A	B	C	D, F	E
A	0				
B	0.77	0			
C	5.66	4.95	0		
D, F	3.20	2.50	2.24	0	
E	4.24	3.54	1.41	1.00	0



$$\begin{array}{c|cc} & X1 & X2 \\ A & 1 & 1 \\ B & 1.5 & 1.5 \\ C & 5 & 5 \\ D & 3 & 4 \\ E & 4 & 4 \\ F & 3 & 3.5 \end{array}$$

The new distance matrix appears below. The clusters $\{D, F\}$ and E are merged in the next step.

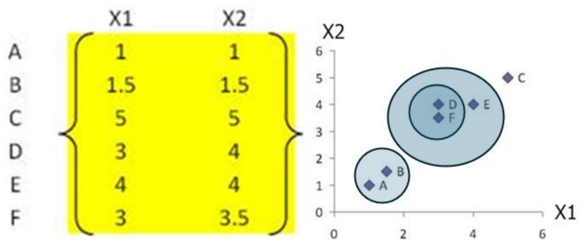
d	A, B	C	D, F	E
A, B	0			
C	4.95	0		
D, F	2.50	2.24	0	
E	3.54	1.41	1.00	0



$$\begin{array}{c|cc} & X1 & X2 \\ A & 1 & 1 \\ B & 1.5 & 1.5 \\ C & 5 & 5 \\ D & 3 & 4 \\ E & 4 & 4 \\ F & 3 & 3.5 \end{array}$$

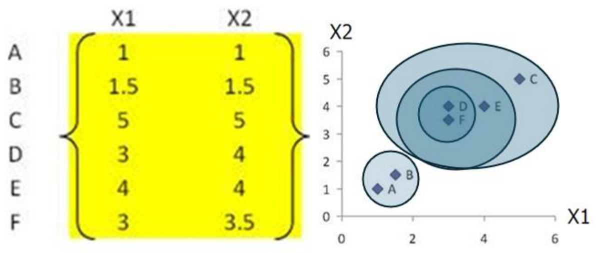
The subsequent steps are

d	A, B	C	D, F, E
A, B	0		
C	4.95	0	
D, F, E	2.50	1.41	0



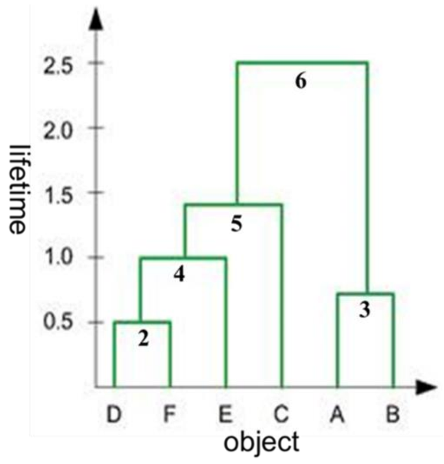
$$\begin{array}{c|cc} & X1 & X2 \\ \hline A & 1 & 1 \\ B & 1.5 & 1.5 \\ C & 5 & 5 \\ D & 3 & 4 \\ E & 4 & 4 \\ F & 3 & 3.5 \end{array}$$

d	A, B	D, F, E, C
A, B	0	
D, F, E, C	2.50	0



$$\begin{array}{c|cc} & X1 & X2 \\ \hline A & 1 & 1 \\ B & 1.5 & 1.5 \\ C & 5 & 5 \\ D & 3 & 4 \\ E & 4 & 4 \\ F & 3 & 3.5 \end{array}$$

The above steps can be shown in the form of a dendogram. A dendrogram is a diagrammatic representation of hierarchical merging of clusters (vertical lines/bars). The vertical axis of the dendrogram typically represents the distance between clusters. The height of each horizontal line (branch) indicates the distance between the clusters (vertical lines) being merged.



Lifetime of a Cluster: The distance between a cluster is created and that it disappears/merges with another cluster during clustering. Lifetime of A, B, C, D, E and F is 0.71, 0.71, 1.41, 0.5, 1.0 and 0.5 respectively.

k-cluster Lifetime : The distance from that k clusters between that emerge to that k clusters vanish. The distance from that k clusters become k−1 clusters.

5−$cluster\ lifetime$=0.71−0.50=0.21

4−$cluster\ lifetime$=1.00−0.71=0.29

3−$cluster\ lifetime$=1.41−1.00=0.41

2−$cluster\ lifetime$=2.5−1.41=1.09

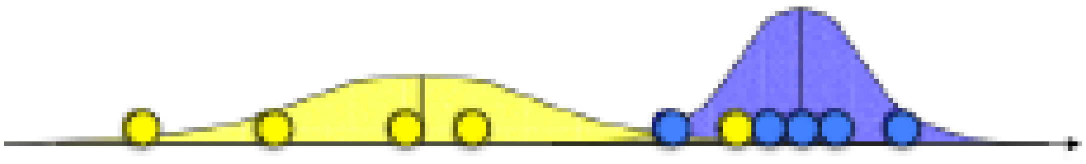
3 Expectation-Maximization Algorithm

Clustering can be classified as hard clustering and soft clustering. In hard clustering any training sample belongs finally to a fixed cluster. Clusters are mutually exclusive in hard clustering. Soft clustering advocates the probability with which a sample belongs to a cluster.

3.1 EM Algorithm

Assume that the samples come from k sources, where k is the number of clusters. We consider Gaussian distributions and hence the name Gaussian Mixture Model (GMM). Expectation-Maximization algorithm targets to identify the distributions (clusters) and their parameters. With parameters of GD known, we can compute $P\left(c_j \,|x\right)$, the probability with which the sample x belongs to the cluster c_j.

If we know the clusters from which points come, we can easily formulate the distributions.

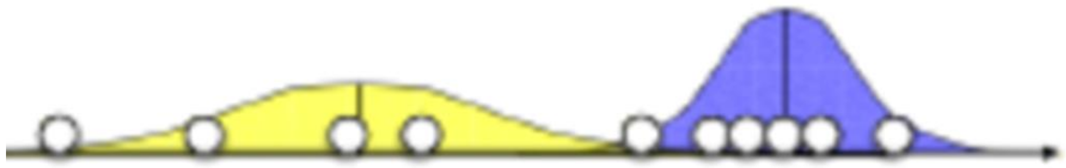


$$\mu_j = \frac{\sum x_i}{N}$$

$$\sigma_j = \sqrt{\frac{\sum(x_i - \bar{x})^2}{N-1}}$$

$$P\left(c_j \middle| x_i\right) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i-\mu_j}{\sigma_j}\right)^2}$$

However, we are given only the mixed samples but not their distributions. In fact, we have to separate out the samples into their respective clusters, compute the parameters of the distribution and be able to find out the distribution to which a new test sample belongs to. The following diagram shows the task to be performed. All the 1D samples are shown in white colour meaning that the whole set is given from which their distributions are to be extracted.



Initialization Step:

The approach is to fix the value of K (the number of clusters/distributions). Assume $\mu_j, \sigma_j, \pi_j \; for \; j = 1..k$. π_j is called latent/hidden variable and specifies the prior probability that

x_i is generated by jth distribution or cluster. It may be noted that $\sum_j \pi_j = 1$. The EM algorithm works as follows.

Expectation Step:

for each sample x_i:

compute responsibility(membership weight) r_{ij}, $probability\ of$ sample i is from distribution j as follows.

$\boldsymbol{r_{ij}} = \frac{\pi_j N(\boldsymbol{x_i}|\boldsymbol{\mu_j}, \boldsymbol{\sigma_j})}{\sum_j \pi_j N(\boldsymbol{x_i}|\boldsymbol{\mu_j}, \boldsymbol{\sigma_j})}$, which represents the Bayes theorem. π_j is the probability of the distribution j and $\boldsymbol{N(x_i|\mu_j, \sigma_j)} = \frac{1}{\sigma_j\sqrt{2\pi}}\boldsymbol{e}^{-\frac{1}{2}\left(\frac{x_i-\mu_j}{\sigma_j}\right)^2}$ is the probability that the sample x_i is from the given GD with the parameters μ_j, σ_j.

Maximization Step: In the maximization step, the parameters of the distributions are updated as follows.

$$\boldsymbol{N_j} = \sum_i \boldsymbol{r_{ij}}$$

$$\boldsymbol{\mu_j} = \frac{\boldsymbol{1}}{\boldsymbol{N_j}}\sum_i \boldsymbol{r_{ij}x_i}$$

$$\boldsymbol{\sigma_j} = \frac{\boldsymbol{1}}{\boldsymbol{N_j}}\sum_i \boldsymbol{r_{ij}}\left(\boldsymbol{x_i} - \boldsymbol{\mu_j}\right)^2$$

$$\boldsymbol{\pi_j} = \frac{\boldsymbol{N_j}}{\boldsymbol{m}}$$