Clustering

This chapter addresses the problem of finding structure in data, but unlike the case of classification, there is only one source of information to solve the problem: the data itself. We can foresee that a central concept in clustering (and unsupervised learning in general) is the quantification of dissimilarity: samples in a cluster are “more similar” among themselves than with other samples in the data set. There are basically two clustering methods:

- **Local criteria**: Hierarchical clustering, minimum variance (mostly neural networks as the ART and competitive networks) and valley seeking which are able to provide non-elliptical clusters.

- **Global criteria**: a global cost function is provided. The most widely used is MacQueen’s K means clustering.

K-Means
The basic idea is to minimize the distance of the samples to a predetermined number of centers, i.e.

\[ J = \sum_{i=1}^{K} \sum_{n \in C_i} |x_n - \gamma_i|^2 \quad \gamma_i = \frac{1}{N_i} \sum_{n \in C_i} x_n \]

Amazingly, this can be done incrementally by first finding the center closest to the sample and then update it by

\[ \gamma_i(n+1) = \gamma_i(n) + \eta(x(n) - \gamma_i(n)) \]
Information theory and its descriptors seem very appropriate to substitute the Euclidean distance in this algorithm. There have been several IT based clustering methods, starting from Watanabe, but the availability of the nonparametric estimators are particularly appealing.

**Differential Clustering using Renyi’s entropy.**

Let us assume that we cluster based on K-Means clustering and we try to improve the results by using entropy. An intuitive idea is to re-assign each sample to the cluster for which the incremental entropy is the smallest (because it means that it fits “naturally” in that cluster).

In an equation,

\[ H(C_i + x) - H(C_i) < H(C_k + x) - H(C_k) \quad \text{for} \quad k = 1, \ldots, K \quad k \neq i \]

Notice that this lends itself nicely to the IP estimator. Although entropy is not scale environment, reasonable results were obtained.
Clustering Evaluation Function

Let us assume that we have two data subgroupings \( p(x) \) and \( q(x) \). The cross information potential (CIP) measures dissimilarity because it evaluates the IPF created by \( p(x) \) in the locations created by \( q(x) \). Recall that it can be estimated by

\[
\hat{V}(p, q) = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} G_{\sigma^2}(x_i - x_j)
\]

where here the indices refer to different subgroupings. The clustering evaluation function is defined as.

\[
CEF(p, q) = -\log \hat{V}(p, q)
\]

From its relation to the CIP it is easy to show that CEF is always positive, it is symmetric and obeys the triangular inequality but it is not zero when \( p(x) = q(x) \) so it is a pseudo distance.

Let us understand CEF as a nonlinear weighted distance. When we use the Euclidean distance we are effectively weighting the difference between samples by their value,

\[
\overline{D}_2(X_1, X_2) = \frac{1}{N_1 N_2} \sum_{x_i \in X_1} \sum_{x_j \in X_2} (x_i - x_j)^2
\]

If we compare the two we see that CIP just weights the differences in samples with the Gaussian kernel (or any other kernel). So this gives a nonlinear weighting of the sample difference.
CEF and divergence measures

Notice that CEF is intimately related to the Bhattacharyya distance (square root of the product of PDFs). We can also link it to $D_{CS}(p,q)$

$$D_{CS}(p,q) = -\ln \left( \frac{\int p(x)q(x)dx}{\sqrt{\int p^2(x)dx \int q^2(x)dx}} \right) = -\ln \left( \frac{V(p,q)}{\sqrt{\int p^2(x)dx \int q^2(x)dx}} \right)$$

where we recognize the numerator of the log as the CIP. On the other hand we know that for optimization one can drop the log without affecting the solution. We can also think of the optimization of the ratio as the optimization of the numerator subject to a constraint given by the denominator. So we see that optimizing CIP provides the same solution as the CEF and this is also one of solutions of optimizing $D_{CS}(p,q)$.

We compared experimentally the assessment of dissimilarity given by CEF with many others for unimodal clusters.
Multiple Clusters

If we redefine CIP as

\[ \hat{V}(p,q) = \frac{1}{2N_1N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} M(x_i,x_j)G_{\sigma\sqrt{2}}(x_i - x_j) \]

where \( M(x_i,x_j) = M(x_i)M(x_j) \) is the clustering membership function defined as \( M_i(x_i) = 1 \) iff \( x_i \in p(x) \).

We can generalize the CIP for arbitrary number of classes \( C \) as

\[ CIP(x,s) = \frac{1}{2N_1...N_C} \sum_{i} \sum_{j} M(x_{ij},s)G_{\sigma\sqrt{2}}(x_i - x_j) \]

where \( M(x_{ij},s) = s(x_i) \cup s(x_j) \). Since \( s^k(x_i) = 1 \) \( x_i \in C_k \) is a bit string, this generalizes to an arbitrary number of pairs.

CEF for Reclustering.

As differential entropy, CEF can also be used to recluster K-Means clusters to improve their performance. The tests used a greedy approach.
Note that this works even when the number of samples in each cluster are different

Tests with Mixture of Gaussians and Classifiers
### Comparison of the MLP trained with backpropagation and CEF

<table>
<thead>
<tr>
<th>DATA SET2</th>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>342</td>
<td>8</td>
</tr>
<tr>
<td>Class 2</td>
<td>9</td>
<td>341</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SET2</th>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>340</td>
<td>10</td>
</tr>
<tr>
<td>Class 2</td>
<td>12</td>
<td>338</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SET3</th>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>145</td>
<td>5</td>
</tr>
<tr>
<td>Class 2</td>
<td>9</td>
<td>341</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATA SET3</th>
<th>Class 1</th>
<th>Class 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>144</td>
<td>6</td>
</tr>
<tr>
<td>Class 2</td>
<td>9</td>
<td>341</td>
</tr>
</tbody>
</table>
Gradient descent Clustering based on $D_{CS}$

One of the difficulties of gradient descent applied to a clustering cost function is the fact that the partnership function is discrete (0/1). One way of overcoming this difficulty is to create a fuzzy boundary. Our implementation has very interesting properties. First we will be working with the argument of the log in $D_{CS}$, i.e.

$$J_{CS}(p,q) = \frac{\int p(x)q(x)dx}{\sqrt{\int p^2(x)dx \int q^2(x)dx}}$$

and use the IP estimator and the membership function to obtain

$$\hat{J}_{CS}(p,q) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (1 - m_i^T m_j) G_{\sigma^2} (x_i - x_j)$$

For $k$ clusters this generalizes to

$$\hat{J}_{CS}(p,q) = \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{N} \sum_{j=1}^{N} (1 - m_i^T m_j) G_{\sigma^2} (x_i - x_j)$$

This algorithm assumes that the number of clusters is known.
Adaptation with Lagrange multipliers

The goal is to minimize $J_{CS}(m_1, \ldots, m_K)$ because this corresponds to maximizing $D_{CS}$. But we have to modify the crisp membership vectors to employ gradient descent. Let us define the equivalent constrained optimization problem

$$\min_{m_1, \ldots, m_N} \hat{J}_{CS}(m_1, \ldots, m_N) \quad subject \quad m_j^T 1 - 1 = 0, \quad j = 1, \ldots, N$$

Let us make the change of variables $m_i(k) = v_i^2(k)$, which yields

$$\min_{v_1, \ldots, v_N} \hat{J}_{CS}(v_1, \ldots, v_N) \quad subject \quad v_j^T v - 1 = 0, \quad j = 1, \ldots, N$$

This is equivalent to making the membership vectors $v_i$

$$\frac{\partial \hat{J}_{CS}}{\partial v_i} = \left( \frac{\partial \hat{J}_{CS}^T}{\partial m_i} \frac{\partial m_i}{\partial v_i} \right)^T = \Gamma \frac{\partial \hat{J}_{CS}}{\partial m_i} \rightarrow 0 \quad \Gamma = diag(2\sqrt{m_{i1}}, \ldots, 2\sqrt{m_{iK}})$$

Notice that if all the elements are positive, then this corresponds to a variable stepsize because the direction of the original gradient never changes. So we enforce this by adding a small value $\alpha$ to all the components. This “noise” also helps the adaptation.
The solution of the optimization can be constructed with the Lagrangian

\[ L = \hat{J}_{CS}(v_1, ..., v_N) + \sum_{j=1}^{N} \lambda_j (v_j^T v_j - 1) \]

where the \( \lambda_i \) are the Lagrange multipliers.

The solution is given by

\[
\begin{align*}
\frac{\partial L}{\partial v_i} &= \frac{\partial \hat{J}_{CS}}{\partial v_i} + \sum_{k=1}^{N} \lambda_k \frac{\partial (v_k^T v_k - 1)}{\partial v_i} = 0 \\
\frac{\partial L}{\partial \lambda_j} &= v_j^T v_j - 1 \quad j = 1, ..., N
\end{align*}
\]

The vectors \( v_i \) are derived as

\[
\frac{\partial \hat{J}_{CS}}{\partial v_i} + 2\lambda_i v_i = 0 \quad \Rightarrow v_i^+ = -\frac{1}{2\lambda_i} \frac{\partial \hat{J}_{CS}}{\partial v_i} \quad i = 1, ..., N
\]

where the “+” means updated vector. From this we can get the \( \lambda_i \) as

\[
v_i^+ v_i^+ - 1 = 0 \quad \Rightarrow \lambda_i = \frac{1}{2} \sqrt{\frac{\partial ^2 \hat{J}_{CS}^T}{\partial v_i^2} \frac{\partial \hat{J}_{CS}}{\partial v_i}}
\]

At convergence or after a prescribed number of iterations, the largest value of \( v \) is set to 1 and the others to zero. Initial conditions are random. The complexity of this algorithm is \( O(N^2) \).

Our experience shows that the kernel size should be annealed slowly from a large to a small value to avoid local minima.
Data set

(a) $\sigma$ versus no. iterations.

(b) Misclassifications (% mean over 20 trials) versus no. iterations.
In the Iris data set the DCS clustering yields 9% error which is not good. The reason is the difference in dynamic range of features. One can either use an asymmetric kernel or normalize the features. Although different they provide similar results.
Mean Shift Algorithms and Renyi’s entropy

Another interesting clustering methodology is the mean shift algorithm originally proposed by Fukunaga. The big advantage is that they do not require the selection of the number of clusters (the kernel size effectively controls the number of clusters).

The stationary point of the gradient of the PDF estimated by Gaussian kernels can be written as a fixed point update

\[ p_{X,\sigma}(x) = \frac{1}{N} \sum_{i=1}^{N} G_{\sigma}(x-x_i), \quad \nabla p_{X,\sigma}(x) = 0 \quad x(n+1) = m(x(n)) = \frac{\sum_{i=1}^{N} G_{\sigma}(x-x_i)x_i}{\sum_{i=1}^{N} G_{\sigma}(x-x_i)} \]

This algorithm developed by Fukunaga is called the blurring mean shift GBMS. In order to avoid the blurring, we need to keep a version of the original data X0, and slightly modify the solution to read

\[ x(n+1) = m(x(n)) = \frac{\sum_{i=1}^{N_0} G_{\sigma}(x-x_{0_i})x_{0_i}}{\sum_{i=1}^{N_0} G_{\sigma}(x-x_{0_i})} \]

which has been called the Gaussian mean shift (GMS). They have been shown very useful because they are very fast and do not need a preset number of clusters not a stepsize. GMS was shown to be stable and yielding the modes of the data.
There is an interesting interpretation of these algorithms in terms of Renyi’s entropy. Let us define a cost function that minimizes Renyi’s quadratic entropy or maximizes the IP.

\[ J(X) = \max_X nH_2(X) = \max_X \log V(X) \]

Since the log is a monotonic function it will not change the location of the extremum, and let us compute the information force for the data set that is evolving over the forces (using a Gaussian kernel)

\[ F(x_k) = \frac{1}{2\sigma^2} \sum_{j=1}^{N} G_\sigma(x_k - x_j)(x_j - x_k) = 0 \]

rearranging this equation we get exactly the GBMS. If we apply the update on the original data set \( X_0 \), we have to compute the cross information potential \( V(X,X_0) \) and maximize

\[ J(X) = \max_X V(X;X_o) \]

\[ = \max_X \frac{1}{NN_o} \sum_{i=1}^{N} \sum_{j=1}^{N_o} G_\sigma(x_i - x_o_j) \]

If we compute the forces we end up with the GMS

\[ x_k(n + 1) = m(x_k(n)) = \frac{\sum_{j=1}^{N} G_\sigma(x_k - x_0_j)x_0_j}{\sum_{j=1}^{N} G_\sigma(x_k - x_0_j)} \]

Stopping the GMS uses a tolerance

\[ \frac{1}{N} \sum_{i=1}^{N} d_n(x_i) < tol, \quad d_n(x_i) = \| x_i(n) - x_i(n-1) \| \]
Example: 16 Mixture of Gaussians

(a) R16Ga Dataset

(b) apriori probabilities

(a) Good Mode finding ability of GMS algorithm

(b) Poor mode finding ability of GBMS algorithm
Graph Theoretical Clustering

The advantage of the graph theoretical approach is that it does not require parametric assumptions about the data nor the number of clusters. It uses a proximity graph where each node is a sample, and between pairs of nodes an edge is formed with a weight that measures the distance between the data samples. So clustering corresponds to partitioning the graph.

More recently the problem has been reformulated as to find the cost of cutting the graph into two (or more) sub graphs

$$Cut(\Gamma_1, \Gamma_2) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k_{ij}$$

where $k_{ij}$ are the edge weights that need to be removed to produce the cut. Shi and Malik proposed the normalized cut as

$$NC(\Gamma_1, \Gamma_2) = \frac{Cut(\Gamma_1, \Gamma_2)}{Assoc(\Gamma_1, \Gamma)} + \frac{Cut(\Gamma_1, \Gamma_2)}{Assoc(\Gamma_2, \Gamma)}$$

where $Assoc(\Gamma_m, \Gamma) = \sum_{i=1}^{N_m} \sum_{j=1}^{N} k_{ij}$ is the total weight from connections of the subgraph to the full graph. These authors optimized the normalized cut based on the eigenvalues of the Laplacian matrix $L = D - K$ where $D$ is a diagonal matrix of elements $d_m = \sum_{i=1}^{N} k_{mi}$ and $K = [k_{lj}]$ is the affinity matrix.
These methods are called *spectral clustering methods* and they outperform the conventional clustering methods. However, the connection weighting is heuristic, they need to store the affinity matrix, computation is heavy and the choice of the kernel is heuristic. ITL answers some of these shortcomings.

**Graph Creation with Information Forces**

Let us construct a directed tree from the samples (node). Each node $i$ initiates a branch pointing to another node $j$, which is called the predecessor of $i$. The root does not have a predecessor.

If we have more than one cluster the issue is how to divide the nodes by the region where the density of samples is the lowest. In chapter 2 we saw that this is exactly what the information forces do! They push the samples to the center of the clusters. So in principle we can use ITL to create directed graphs. Notice that the kernel size is going to be very important to define the boundaries of each graph.
Creating Directed Trees
The procedure to create a directed tree with IF is very straightforward. Node j is defined as the predecessor of node i if it satisfies the condition that node j lies closest to the direction of the force $F_i$ acting on i under the following constraints:

1- The distance $\|x_i - x_j\| \leq 3\sigma$

2- $F_i(x_i - x_j) \geq 0$

3- Node j can not be one of i's children

The first rule makes sure that we do not include samples that are in different clusters (just look in the immediate neighborhood of the sample). Condition 2 uses the directions of the IF (that are global) to create the directed trees. The third condition ensures that there is a single root. When applied to the previous data set only 3 errors are made and annealing is NOT necessary!
When we applied this method to real datasets the results are also pretty good. In the WINE database (13 dimensions), the clustering achieved with the IF directed trees is ($\sigma=0.32$)

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>59</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_1$</td>
<td>(59)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_2$</td>
<td>(71)</td>
<td>50</td>
<td>6</td>
</tr>
<tr>
<td>$C_3$</td>
<td>(48)</td>
<td>0</td>
<td>46</td>
</tr>
</tbody>
</table>

and for the IRIS database the results are slightly better than the optimization of $D_{CS}$ ($\sigma=0.095$)

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>49</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$C_1$</td>
<td>(50)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_2$</td>
<td>(50)</td>
<td>42</td>
<td>3</td>
</tr>
<tr>
<td>$C_3$</td>
<td>(50)</td>
<td>5</td>
<td>44</td>
</tr>
</tbody>
</table>
Graph Cutting with ITL- The Information Cut
It is possible to frame the gradient based clustering based on the DCS as a novel Information Cut algorithm and an alternative to spectral clustering. In fact, the elements of the affinity matrix can be related to the information potential between pairs of samples,\[
\int G_\sigma(x,x_l)G_\sigma(x,x_r)dx = G_{\sigma,2}(x_l,x_r) = k_{ll}
\]
Moreover, if we estimate the cross information potential we obtain
\[
\int \hat{p}_1(x)\hat{p}_2(x)dx = \frac{1}{N_1N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k_{ij}
\]
which can be interpreted as an ITL derived graph-cut. Hence, the samples corresponding to \(p_1(x)\) are related to the nodes of graph \(\Gamma_1\), and the samples corresponding to \(p_2(x)\) with the nodes of a graph \(\Gamma_2\). Same thing for the denominator of the DCS, so we have what we called the Information Cut (IC) as
\[
IC(\Gamma_1,\Gamma_2) = \frac{\sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k_{ij}}{\sqrt{\sum_{i=1}^{N_1} \sum_{i=1}^{N_2} k_{ii'} \sum_{j=1}^{N_2} \sum_{j=1}^{N_2} k_{jj'}}}
\]
The denominators are indeed what is called the volume of a graph, so we can rewrite as
\[
IC(\Gamma_1,\Gamma_2) = \frac{Cut(\Gamma_1,\Gamma_2)}{\sqrt{Vol(\Gamma_1)Vol(\Gamma_2)}}
\]
So instead of solving the eigendecomposition in spectral clustering we can use the gradient based clustering using the $D_{CS}$. The advantage is that it is a sample by sample procedure, but the disadvantage is that it is sensitive to local minima. We can use the kernel annealing to avoid them, but the annealing rates have to be determined for each problem. To decrease computation we can use random sampling of the data, or use the incomplete Choleski decomposition.

When compared with the normalized cut, the information cut provided better results.
But remember that the procedure is now sensitive to local minima, and kernel annealing needs to be used.

The kernel size does not influence performance that much here.
The information cut outperforms the normalized cut in the results in the WINE data set (96.6% versus 97.2%) and also in the PENDIGIT (73.4% versus 84.4%). Moreover we showed that stochastic sampling the data down to 20% does not affect the accuracy of the results that much.

<table>
<thead>
<tr>
<th>M</th>
<th>0.1N</th>
<th>0.2N</th>
<th>0.3N</th>
<th>0.4N</th>
<th>0.5N</th>
</tr>
</thead>
<tbody>
<tr>
<td>%</td>
<td>83.5</td>
<td>84.4</td>
<td>84.7</td>
<td>85.3</td>
<td>84.1</td>
</tr>
<tr>
<td>M</td>
<td>0.6N</td>
<td>0.7N</td>
<td>0.8N</td>
<td>0.9N</td>
<td>N</td>
</tr>
<tr>
<td>%</td>
<td>84.5</td>
<td>83.8</td>
<td>84.7</td>
<td>85.3</td>
<td>85</td>
</tr>
</tbody>
</table>