The Global Kernel $k$-Means Clustering Algorithm

Grigorios Tzortzis and Aristidis Likas

Department of Computer Science
University of Ioannina, Greece
Clustering

- Partition of a dataset into homogeneous groups

Given a dataset \( X = \{x_1, x_2, \ldots, x_N\} \) of objects, we aim to partition this dataset into \( M \) disjoint clusters \( C_1, C_2, \ldots, C_M \).

- When the objects are data vectors, the most well-known algorithm for the above task is \( k\)-Means
**k-Means**

- Each cluster \( C_k \) is represented by its center \( m_k \) (mean of the cluster elements)
- Finds **local minima** w.r.t. the **clustering error**

\[
E(m_1, \ldots, m_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(x_i \in C_k) \|x_i - m_k\|^2
\]

- (sum of cluster variances)

- **Drawbacks**
  - **Highly dependent on the initial positions of the centers**
  - **Identifies only linearly separable clusters**

- **Improvements**
  - Multiple restarts, Global \( k \)-Means (Likas et al. [2003])
  - Kernel \( k \)-Means
Global $k$-Means

- An incremental, deterministic clustering algorithm that runs $k$-Means several times
- Finds near-optimal solutions wrt clustering error

Idea: a near-optimal solution for $k$ clusters can be obtained by running $k$-means from an initial state $(m_1, m_2, ..., m_{k-1}, x_n)$

- the $k$-1 centers are initialized from a near-optimal solution of the (k-1)-clustering problem $(m_1, m_2, ..., m_{k-1})$
- the $k$-th center is initialized at some data point $x_n$ (which?)

Consider all possible initializations (one for each $x_n$)
Orange circles: optimal initial position of the cluster center to be added
Global $k$-Means - Algorithm

In order to solve the $M$-clustering problem:

1. Solve the $1$-clustering problem (trivial)
2. Solve the $k$-clustering problem using the solution of the $(k-1)$-clustering problem $(m_1, m_2, ..., m_{k-1})$
   a) Execute $k$-Means $N$ times, initialized as $(m_1, m_2, ..., m_{k-1}, x_n)$ at the $n$-th run ($n=1, ..., N$).
   b) Keep the solution corresponding to the run with the lowest clustering error as the solution with $k$ clusters $(m_1, m_2, ..., m_k)$
3. $k:=k+1$, Repeat step 2 until $k=M$.

✓ Avoids the initialization problem of $k$-Means
✓ Locates near optimal partitions w.r.t. clustering error
✓ All intermediate solutions for $k=1, ..., M-1$ are also found: useful when searching for the number of clusters
   ▪ Requires $MN$ runs of $k$-Means to find $M$ clusters
Kernel \( k \)-Means

- Data points are mapped from input space to a higher dimensional feature space through a transformation \( \phi \)

\[
E(m_1, \ldots, m_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(x_i \in C_k) \| \phi(x_i) - m_k \|^2 \quad \text{where} \quad m_k = \frac{\sum_{i=1}^{N} I(x_i \in C_k) \phi(x_i)}{\sum_{i=1}^{N} I(x_i \in C_k)}
\]

\( \checkmark \) Identifies non-linearly separable clusters in input space

- Minimizes the clustering error in feature space
Kernel \( k \)-Means

- **Kernel trick**
  - A kernel function corresponds to the inner products in feature space i.e.
    \[
    K_{ij} = \phi^T(x_i)\phi(x_j),\ |\phi(x_i) - \phi(x_j)|^2 = K_{ii} + K_{jj} - 2K_{ij}
    \]
  - Computation of distances from centers in feature space:
    \[
    \|\phi(x_i) - m_k\|^2 = K_{ii} - \frac{2\sum_{j=1}^{N} I(x_j \in C_k)K_{ij}}{\sum_{j=1}^{N} I(x_j \in C_k)} + \frac{\sum_{i=1}^{N} \sum_{l=1}^{N} I(x_j \in C_k)I(x_l \in C_k)K_{ij}}{\sum_{i=1}^{N} \sum_{l=1}^{N} I(x_j \in C_k)I(x_l \in C_k)}
    \]
  - No need to explicitly define transformation \( \phi \)

- **Difference from k-means**
  - The cluster centers are not explicitly defined
  - Each cluster \( C_k \) is described by its training data

- **Finds local minima** - Strong dependence on initialization
Global Kernel \( k \)-Means

Based on the ideas of the Global \( k \)-Means and Kernel \( k \)-Means algorithms we propose the Global Kernel \( k \)-Means algorithm

- An incremental deterministic algorithm that employs Kernel \( k \)-Means as a local search procedure
  - At each stage of the algorithm a new cluster is added as in Global \( k \)-Means
- Main idea
  - Given a near-optimal solution \((C_1, C_2, \ldots, C_{k-1})\) with \( k-1 \) clusters:
    - A near-optimal solution with \( k \) clusters can be obtained by running kernel \( k \)-means from an initial state
      \[
      (C_1, \ldots, C_l := C_l - \{x_n\}, \ldots, C_{k-1}, C_k = \{x_n\}) \quad x_n \in C_l
      \]
  - Which \( x_n \)? Check all possible initializations (one for each \( x_n \))
Orange circles: optimal initialization of the cluster to be added
Global Kernel $k$-Means - Algorithm

In order to solve the $M$-clustering problem:

1. Solve the 1-clustering problem with Kernel $k$-Means (trivial solution)
2. Solve the $k$-clustering problem using the solution to the (k-1)-clustering problem
   a) Let $(C_1, C_2, ..., C_{k-1})$ denote the solution to the (k-1)-clustering problem
   b) Execute Kernel $k$-Means $N$ times, initialized during the $n$-th run as
      $$(C_1, ..., C_l := C_l - \{x_n\}, ..., C_{k-1}, C_k = \{x_n\}) \quad x_n \in C_l$$
   c) Keep the run with the lowest clustering error as the solution with $k$ clusters $(C_1, C_2, ..., C_k)$
   d) $k := k+1$
3. Repeat step 2 until $k=M$. 
Global Kernel $k$-Means

- **Advantages**
  - ✓ Initialization independent
  - ✓ Finds near optimal solutions w.r.t the clustering error in feature space.
  - ✓ Identifies non-linear separable clusters in input space

- When solving the $M$-clustering problem the solutions with $1, ..., M$ clusters are also found

- **Increased Complexity**
  - To solve for $M$ clusters we must run Kernel $k$-Means $MN$ times
Fast Global Kernel $k$-Means

Global Kernel $k$-Means complexity is high for large datasets we propose a speeding up scheme: Fast Global Kernel $k$-Means

- How is the complexity reduced?
  - To add a new cluster $k$ given the solution for the $(k-1)$-clustering problem, instead of executing Kernel $k$-Means $N$ times, it is **executed only once** from state
    \[
    (C_1, ..., C_l := C_l - \{x_n^*\}, ..., C_{k-1}, C_k = \{x_n^*\}) \quad x_n^* \in C_l
    \]
  - $x_n^*$ provides the **greatest reduction in clustering error** in the first iteration of kernel $k$-means
Fast Global Kernel $k$-Means - Details

- $C_k = \{x_n\}, \quad m_k = \phi(x_n)$
- $C_k$ allocates all points $x_i$ that are closer (in feature space) to $x_n$ than to their cluster center (in the solution with $(k-1)$ clusters):
  $$|| \phi(x_i) - \phi(x_n) ||^2 < d_i$$
- $d_i$ is the distance in feature space between $x_i$ and its cluster center in the $(k-1)$-clustering solution
- The reduction in clustering error due to the reallocation is
  $$b_n = \sum_{i=1}^{N} \max(d_i - ||\phi(x_n) - \phi(x_i)||^2, 0)$$
  $$n^* = \arg \max b_n$$
- Run Kernel $k$-Means once from initial partition
  $$(C_1, \ldots, C_l := C_l - \{x_{n^*}\}, \ldots, C_{k-1}, C_k = \{x_{n^*}\}) \quad x_{n^*} \in C_l$$
Experimental Evaluation

- We compared **Global Kernel $k$-Means**, **Fast Global Kernel $k$-Means** and **Kernel $k$-Means with multiple restarts**
  - On artificial data
  - On MRI segmentation

- Global Kernel $k$-Means and the fast version were run **once**
- Kernel $k$-Means was **restarted 100 times**
- We compared the algorithms in terms of clustering error
Artificial Datasets

- We created three datasets
  - i) Two rings dataset (2 clusters),
  - ii) five copies of two rings (10 clusters),
  - iii) ‘IJCNN 2008’ logo (9 clusters)

- In all the experiments we used a Gaussian kernel

\[ K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2 / (2\sigma^2)\right) \]
Artificial Datasets - Results

<table>
<thead>
<tr>
<th>Method/Dataset</th>
<th>Two Rings $\sigma = 1$</th>
<th>Ten Rings $\sigma = 1.8$</th>
<th>‘IJCNN 2008’ $\sigma = 0.7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global kernel $k$-means</td>
<td>320.17</td>
<td>966.87</td>
<td>27.97</td>
</tr>
<tr>
<td>Fast global kernel $k$-means</td>
<td>320.17</td>
<td>1073.18</td>
<td>27.97</td>
</tr>
<tr>
<td>Kernel $k$-means (100 runs)</td>
<td>Mean: 334.4</td>
<td>Std: 6.4</td>
<td>1107.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>177.24</td>
</tr>
<tr>
<td></td>
<td>Min: 320.17</td>
<td></td>
<td>981.53</td>
</tr>
<tr>
<td></td>
<td>Max: 351.05</td>
<td></td>
<td>1765.29</td>
</tr>
</tbody>
</table>

Global Kernel $k$-Means
Fast Global Kernel $k$-Means
Kernel $k$-Means (12/100 runs)

Global Kernel $k$-Means
Fast Global Kernel $k$-Means
Kernel $k$-Means (5/100 runs)
Artificial Datasets - Conclusions

- Global Kernel $k$-Means in all cases finds the solution with the lowest clustering error and identifies the structures present in the dataset
  - This algorithm identifies near optimal solutions

- Performance of fast Global Kernel $k$-Means is very close to the original algorithm except for the second dataset

- Kernel $k$-Means is very sensitive to initializations
  - For the second dataset it never solves the 10 rings
  - During the restarts near optimal but also very bad solutions are found
  - Number of restarts? We are never sure if they suffice
MRI Segmentation

- We used MRI images downloaded from the BrainWeb site (www.bic.mni.mcgill.ca/brainweb/)
  - We segmented slices of a 3-d brain image
  - In those slices seven classes prevail: background, CSF, grey matter, white matter, muscle/skin, skin and skull
  - We performed clustering into 7 clusters
  - The ground truth is also available (class for each pixel)
  - Large datasets: 181 x 217 = 39277 pixels
MRI Segmentation - Kernel Definition

- Typical approaches cluster each pixel based on its intensity
- The use of kernel k-means enables the use of additional information: pixel intensity + local histogram

- We used a composite kernel for MRI segmentation:

\[ K_{ij} = \exp \left( \frac{-\|I(i) - I(j)\|^2}{2\sigma^2} \right) \left[ \sum_{z=1}^{Bins} \sqrt{P_z(i)P_z(j)} \right] \]

- Global Kernel k-Means is slow (large dataset)

- We compare Fast Global Kernel k-Means to Kernel k-Means (100 random restarts)
## MRI Segmentation - Results

<table>
<thead>
<tr>
<th>Method/Slice</th>
<th>Slice 60</th>
<th></th>
<th>Slice 80</th>
<th></th>
<th>Slice 100</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CE</td>
<td>ME</td>
<td>CE</td>
<td>ME</td>
<td>CE</td>
<td>ME</td>
</tr>
<tr>
<td>Fast global kernel k-means</td>
<td>5208.32</td>
<td>19.89%</td>
<td>5064.99</td>
<td>14.1%</td>
<td>5010.15</td>
<td>15.82%</td>
</tr>
<tr>
<td>Kernel k-means (100 runs)</td>
<td>Mean</td>
<td>5286.95</td>
<td>5244.39</td>
<td>5094.85</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Std</td>
<td>66.29</td>
<td>127.63</td>
<td>141.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Min</td>
<td>5207.65</td>
<td>5064.27</td>
<td>5009.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Max</td>
<td>5364.68</td>
<td>5477.84</td>
<td>5808.77</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Kernel k-means best**: 3, 12, 28 out of 100 runs

- **Fast Global Kernel k-Means equals the best of Kernel k-Means**
  - This solution is much better than the average clustering error achieved by Kernel k-Means
  - The **100 restarts are 20 times slower** than Fast Global Kernel k-Means (16 hours vs. 45 minutes)
MRI Segmentation - Examples

<table>
<thead>
<tr>
<th>Original MRI</th>
<th>Ground Truth</th>
<th>Fast Global kernel k-Means</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slice 80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slice 100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

• We have proposed the Global Kernel k-means:
  ▫ an incremental deterministic approach for clustering in feature space
  ▫ effectively solves the initialization problem of kernel k-means
  ▫ provides near-optimal solutions in terms of the clustering error in feature space
  ▫ Solves all intermediate k-clustering problems for k=1,...,M

• Several techniques can be used to improve computational time
  ▫ fast global kernel k-means
  ▫ Use only a subset with L<<N training data as candidates for the initialization of the new cluster center
  ▫ This subset can be selected through preprocessing using exemplar-based clustering methods (e.g. L-medoids, affinity propagation, convex mixture models).