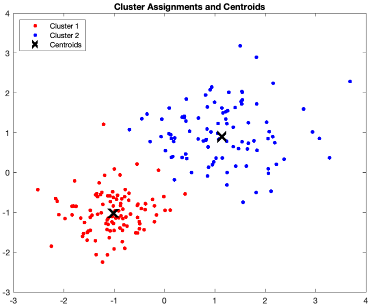
Machine Learning

Deniz Gündüz and Krystian Mikolajczyk

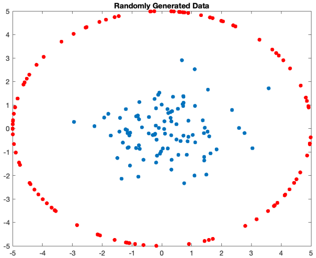
Department of Electrical and Electronic Engineering
Imperial College London

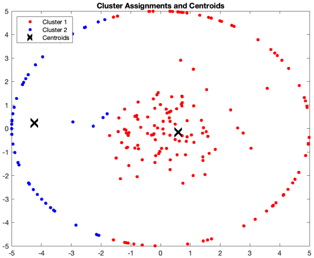
k-Means Clustering



- Goal is to group points into clusters based on similarity
- In k-means, separating boundary is linear

Non-linear Boundaries





- *k*-means fails to cluster across nonlinear boundaries
- Solution is to cluster in high-dimensional kernel space
- Can we retain the complexity through the kernel trick?

k-Means Algorithm: Formal Description

- Let $\mathcal{X}$ be a space with some distance metric d.
  (e.g., $\mathcal{X} = \mathcal{R}^d$ and $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$)
- Dataset: $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\},\ \boldsymbol{x}_i \in \mathcal{X}$
- We want to create k clusters $\mathcal{C}_1, \ldots, \mathcal{C}_k$
- The cluster center of $\mathcal{C}_i$ is given by

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}(\mathcal{C}_i) = \operatorname*{argmin}_{\boldsymbol{\mu} \in \mathcal{X}} \sum_{\boldsymbol{x} \in \mathcal{C}_i} d(\boldsymbol{x}, \boldsymbol{\mu})$$

- For Euclidean distance, $\boldsymbol{\mu}_i$ is simply the average of the samples in $\mathcal{C}_i$

k-Means Algorithm: Formal Description

Consider $\mathcal{X} = \mathbb{R}^d$ and $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|^2$

1. Initialize cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathbb{R}^d$ randomly
2. Repeat until convergence
   - For $i = 1, \ldots, n$, set
   $$c_i = \operatorname*{argmin}_j d(\boldsymbol{x}_i, \boldsymbol{\mu}_j)$$
   - For $j = 1, \ldots, k$, set
   $$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^n \mathbb{1}\{c_i = j\}\boldsymbol{x}_i}{\sum_{i=1}^n \mathbb{1}\{c_i = j\}}$$

Kernel k-means

- Assume all points are mapped to their images $\Phi(\boldsymbol{x}_i)$
- Let $\boldsymbol{K} = \{K(\boldsymbol{x}_i, \boldsymbol{x}_j)\}_{i,j=1,\ldots,n}$ denote the kernel matrix, where $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i)^T \Phi(\boldsymbol{x}_j)$
- Let $\mathcal{C}_1, \ldots, \mathcal{C}_k$ denote the k clusters with the corresponding cluster centers $\boldsymbol{\mu}_1^\Phi, \ldots, \boldsymbol{\mu}_k^\Phi$
- Let $n_i = |\mathcal{C}_i|$ denote the number of points in cluster $\mathcal{C}_i$
- We have

$$\boldsymbol{\mu}_i^\Phi = \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in \mathcal{C}_i} \Phi(\boldsymbol{x}_j)$$

Kernel *k*-means

- k-means objective can be written as

$$\min_{\mathcal{C}_1,\ldots,\mathcal{C}_k} \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in \mathcal{C}_i} \|\Phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i^{\Phi}\|^2$$

- Can be expressed in terms of the kernel function.
- Let's use the same greedy algorithm as before.

Distance in Kernel Space

- Challenge: We cannot calculate the cluster center in the feature space
- We don't need to.
- In cluster assignment, we need to find the minimum distance from each point to each cluster center.

$$
\begin{aligned}
\|\Phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i^{\Phi}\|^2 &= \|\Phi(\boldsymbol{x}_j)\|^2 - 2\Phi(\boldsymbol{x}_j)^T \boldsymbol{\mu}_i^{\Phi} + \|\boldsymbol{\mu}_i^{\Phi}\|^2 \\
&= \Phi(\boldsymbol{x}_j)^T \Phi(\boldsymbol{x}_j) - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} \Phi(\boldsymbol{x}_j)^T \Phi(\boldsymbol{x}_a) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} \sum_{\boldsymbol{x}_b \in \mathcal{C}_i} \Phi(\boldsymbol{x}_a)^T \Phi(\boldsymbol{x}_b) \\
&= K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} \sum_{\boldsymbol{x}_b \in \mathcal{C}_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b)
\end{aligned}
$$

Distance in kernel space can be computed using the kernel function!

Cluster Assignment

- Assign point $\boldsymbol{x}_j$ to cluster c_j:

$$
\begin{aligned}
c_j &= \operatorname*{argmin}_i \left\{ \|\Phi(\boldsymbol{x}_j) - \boldsymbol{\mu}_i^{\Phi}\|^2 \right\} \\
&= \operatorname*{argmin}_i \left\{ K(\boldsymbol{x}_j, \boldsymbol{x}_j) - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) + \frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} \sum_{\boldsymbol{x}_b \in \mathcal{C}_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b) \right\} \\
&= \operatorname*{argmin}_i \left\{ \underbrace{\frac{1}{n_i^2} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} \sum_{\boldsymbol{x}_b \in \mathcal{C}_i} K(\boldsymbol{x}_a, \boldsymbol{x}_b)}_{\text{independent of point } \boldsymbol{x}_j} - \frac{2}{n_i} \sum_{\boldsymbol{x}_a \in \mathcal{C}_i} K(\boldsymbol{x}_a, \boldsymbol{x}_j) \right\}
\end{aligned}
$$

Kernel k-means Algorithm

1. Start with random partitioning of points to k clusters
2. Find closest cluster for each point
3. Reassign clusters
4. Repeat until cluster assignment remains the same over iterations

RBF Kernel k-means