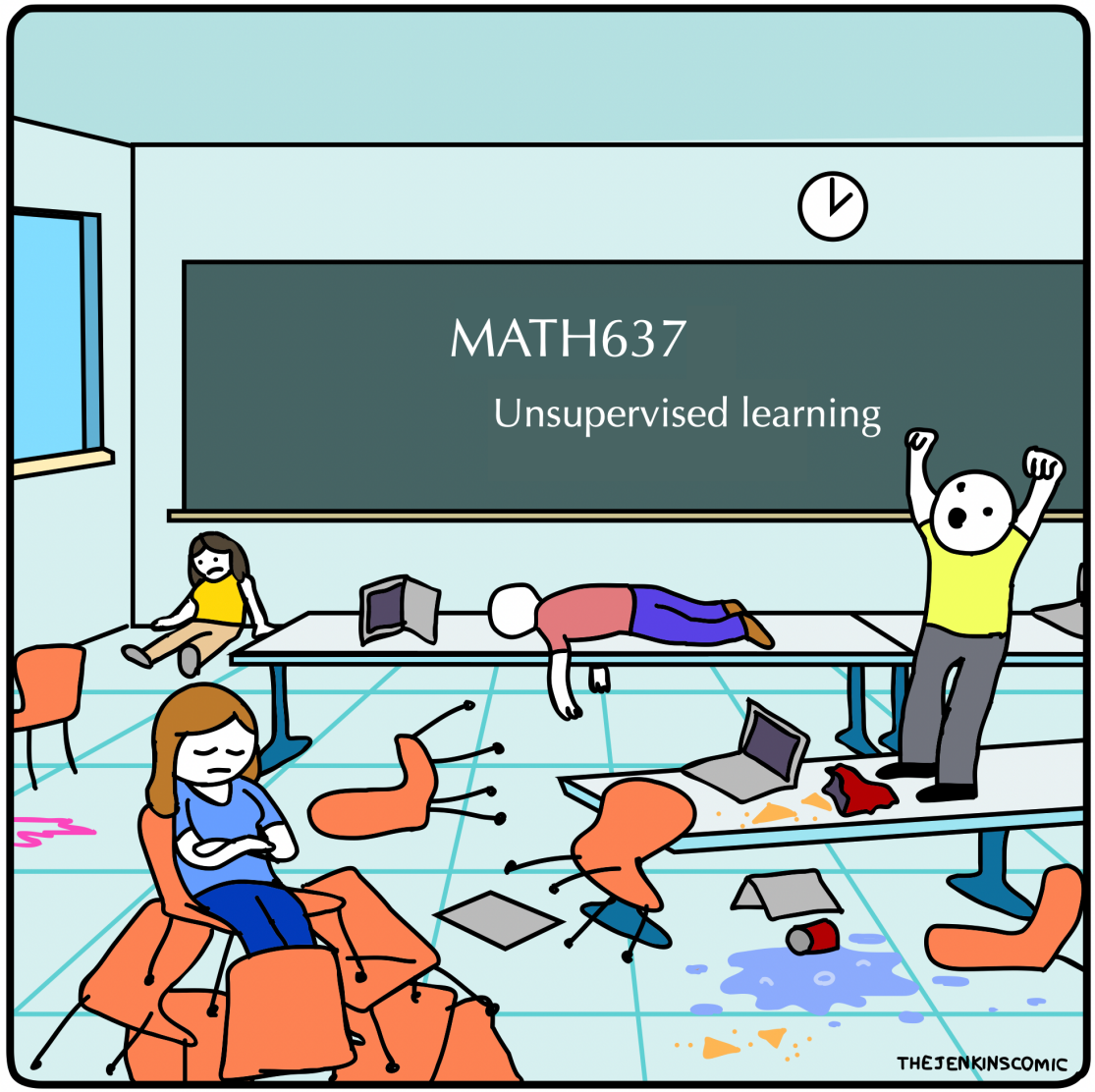
Mathematical techniques in data science

Lecture 12: Clustering using kernel k-means

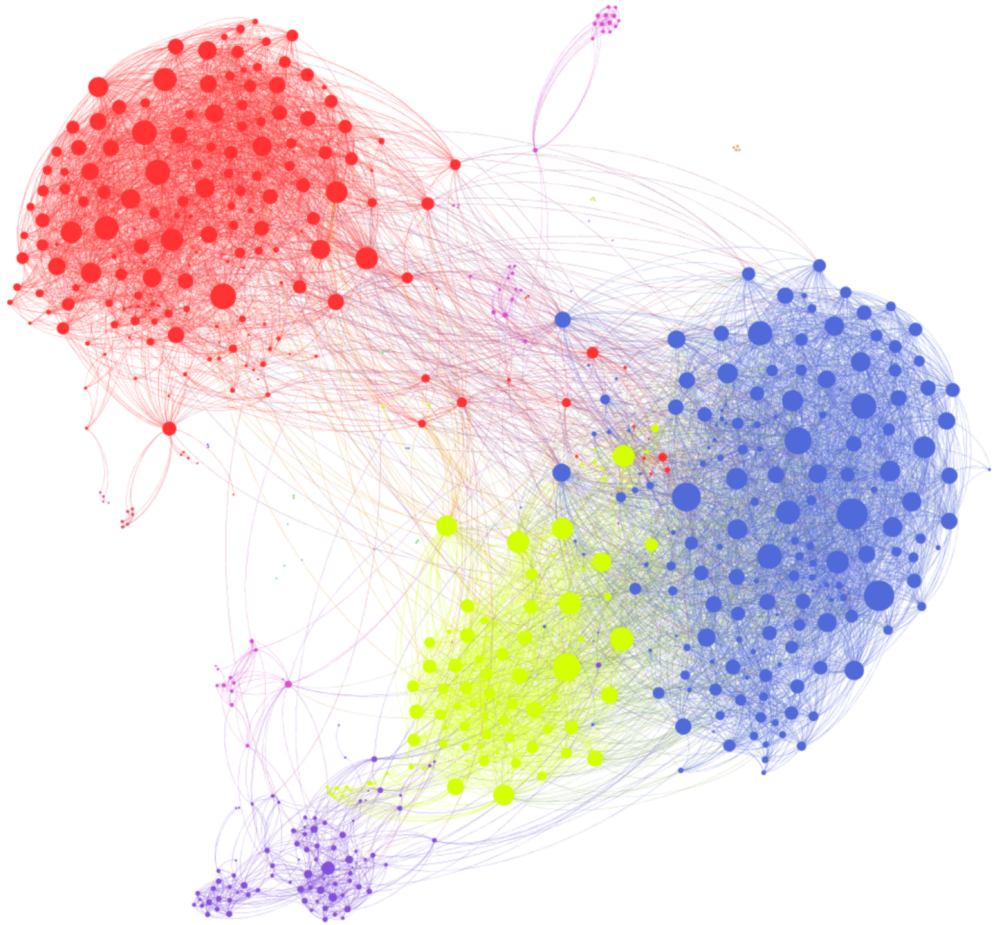
Unsupervised learning



Supervised and unsupervised learning

- Supervised learning problems
  - Labelled data (X, Y) with joint density $P(X, Y)$
  - We are mainly interested in the conditional density $P(Y|X)$.
- Unsupervised learning problems
  - Data X is not labelled and has density $P(X)$
  - We want to infer properties of $P(X)$

Clustering

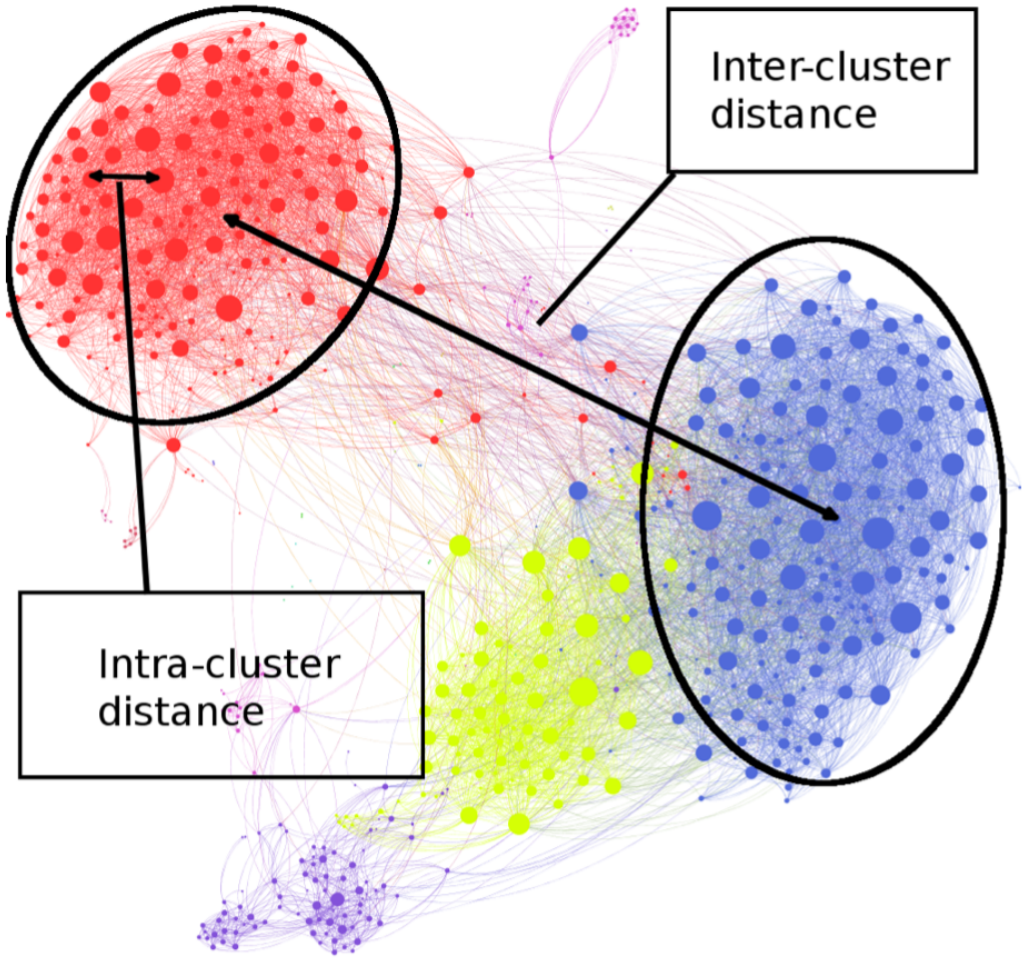


- Unsupervised problem
- Want to label points according to a measure of their similarity

Clustering

We try to partition observations into "clusters" such that:

- Intra-cluster distance is minimized.
- Inter-cluster distance is maximized.



K-means clustering

The K-means algorithm is a popular algorithm to cluster a set of points in $\mathbb{R}^p$.

- We are given n observations $x_1, x_2, \ldots, x_n \in \mathbb{R}^p$.
- We are given a number of clusters K.
- We want a partition $\hat{S} = \{S_1, \ldots, S_K\}$ of $\{x_1, \ldots, x_n\}$ such that

$$\hat{S} = \operatorname*{argmin}_S \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

where $\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ is the mean of the points in S_i (the "center" of S_i).

K-means clustering

- We want a partition $\hat{S} = \{S_1, \ldots, S_K\}$ of $\{x_1, \ldots, x_n\}$ such that

$$\hat{S} = \underset{S}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2,$$

  where $\mu_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$ is the mean of the points in S_i (the "center" of S_i).
- The above problem is NP hard.
- Efficient approximation algorithms exist (converge to a local minimum though).

Lloyd's algorithm

Lloyds's algorithm for K-means clustering

- Denote by $C(i)$ the cluster assigned to x_i.
- Lloyds's algorithm provides a heuristic method for optimizing the K-means objective function.

Start with a "cluster centers" assignment $m_1^{(0)}, \ldots, m_K^{(0)}$. Set $t := 0$. Repeat:

1. Assign each point x_j to the cluster whose mean is closest to x_j:

$$S_i^{(t)} := \{x_j : \|x_j - m_i^{(t)}\|^2 \leq \|x_j - m_k^{(t)}\|^2 \ \forall k = 1, \ldots, K\}.$$

2. Compute the average $m_i^{(t+1)}$ of the observations in cluster i:

$$m_i^{(t+1)} := \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j.$$

Example, Dense

Convergence of Lloyd's algorithm

Note that Lloyds's algorithm uses a greedy approach to sequentially minimize:

$$\sum_{i=1}^{K} \sum_{x_j \in S_i} \|x_j - m_i\|^2.$$

- Both steps of the algorithm decrease the objective.
- Thus, Lloyds's algorithm converges a local minimum of the objective function.

There is no guarantee that Lloyds' algorithm will find the **global** optimum.

Local mean

Lloyd's algorithm: initiation step

- There is no guarantee that Lloyds' algorithm will find the global optimum
- As a result, we use different starting points
- Common initiation schemes:
  - The Forgy method: Pick K observations at random and use these as the initial means
  - Random partition: Randomly assign a cluster to each observation and compute the mean of each cluster
  - kmeans++ (default in sklearn)

kmeans++ initiation

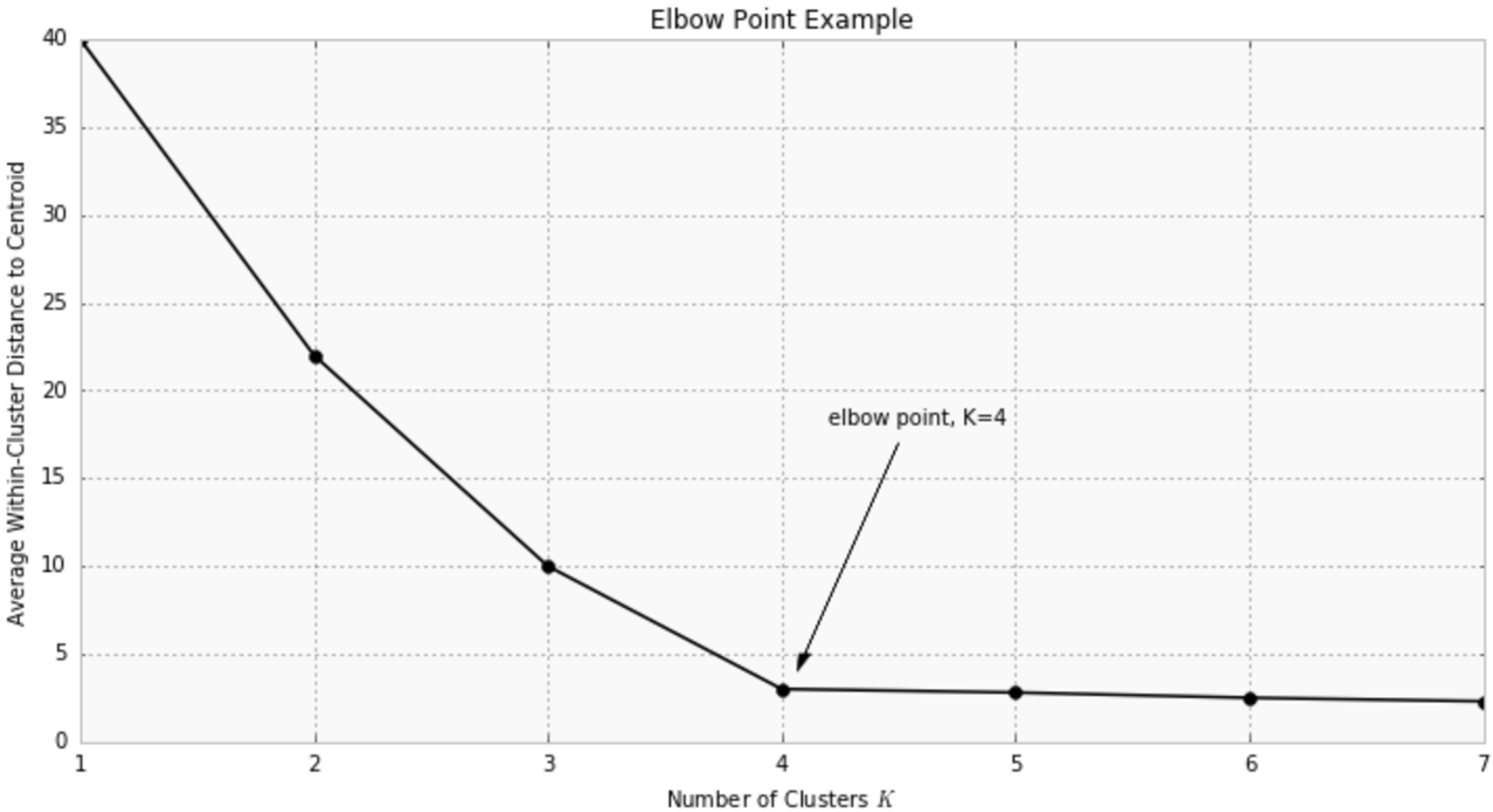
Intuition: spreading out the k initial cluster centers is a good thing

- Choose one center uniformly at random from among the data points.
- For each data point x, compute $D(x)$, the distance between x and the nearest center that has already been chosen.
- Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$
- Repeat Steps 2 and 3 until k centers have been chose

Choosing k

- Elbow method
- Cross-validation
- Average silhouette method

Elbow method



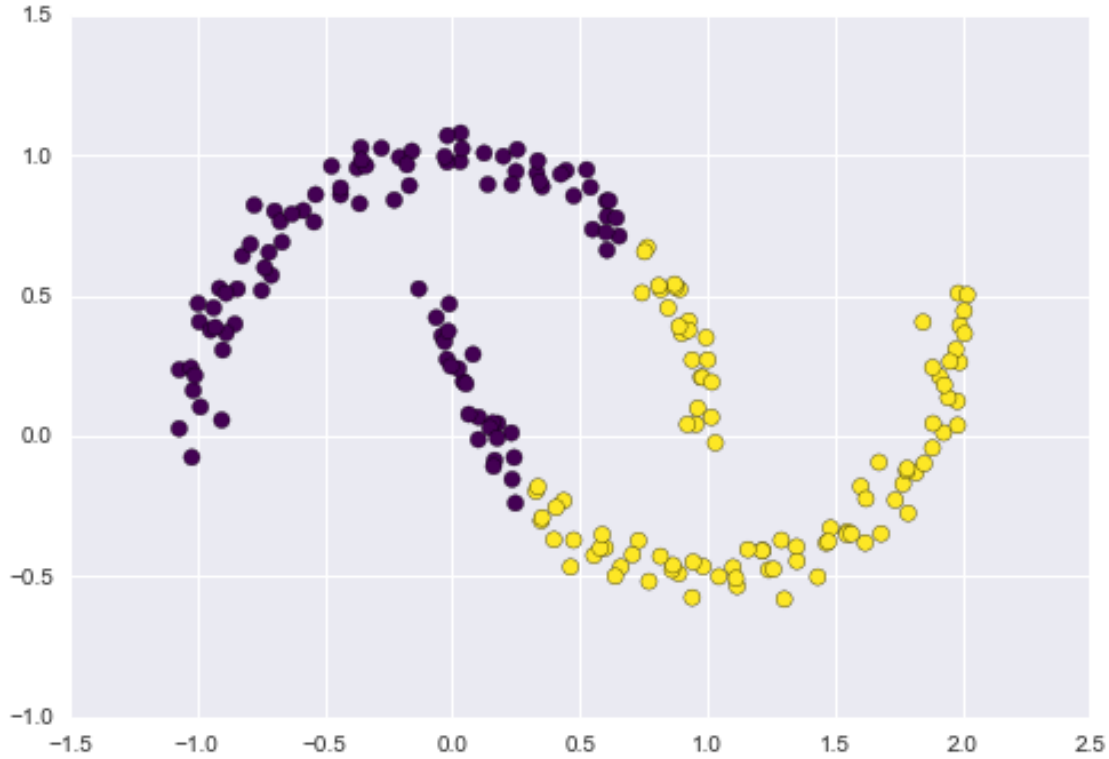
Silhouette method

- a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation)
- ranges from $[-1, 1]$
- The Silhouette coefficient is defined for each sample and is composed of two scores:
  - a: The mean distance between a sample and all other points in the same class.
  - b: The mean distance between a sample and all other points in the next nearest cluster
- The Silhouette coefficient (sklearn.metrics.silhouette_score) for a single sample is then given as:

$$s = \frac{b - a}{\max(a, b)}$$

Issues with k-means

- k-means is limited to linear cluster boundaries



- Solution: adding non-linearities to the model

Kernel k-means

Kernel k-means = kernel trick + k-means

- Ideas:
  - maps the data to a high-dimensional space (called feature space) by a non-linear function ϕ to separate the clusters linearly
  - Using this high-dimensional representation to run k-means
  - Project the data back to the original space to identify the clusters
- Note: the kernel trick works best if we don't have to construct $\phi(x)$ explicitly, but can compute

$$K(x, y) = \langle \phi(x), \phi(y) \rangle$$

- For k-means, we need to compute

$$\|\phi(x_i) - m_j\|^2$$

Kernels

Polynomial Kernel	$\kappa(\mathbf{a}, \mathbf{b}) = (\mathbf{a} \cdot \mathbf{b} + c)^d$
Gaussian Kernel	$\kappa(\mathbf{a}, \mathbf{b}) = \exp(-\|\|\mathbf{a} - \mathbf{b}\|\|^2 / 2\sigma^2)$
Sigmoid Kernel	$\kappa(\mathbf{a}, \mathbf{b}) = \tanh(c(\mathbf{a} \cdot \mathbf{b}) + \theta)$

Kernel k-means = kernel trick + k-means

Note that

$$\begin{aligned}\|\phi(x_i) - m_j\|^2 &= \langle \phi(x_i) - m_j, \phi(x_i) - m_j \rangle \\ &= \langle \phi(x_i), \phi(x_i) \rangle - 2\langle \phi(x_i), m_j \rangle + \langle m_j, m_j \rangle\end{aligned}$$

Given a cluster C_j, its center (on feature space) is

$$m_j = \frac{1}{|C_j|} \sum_{b \in C_j} \phi(b)$$

Thus

$$\langle \phi(x_i), m_j \rangle = \frac{1}{|C_j|} \sum_{b \in C_j} \langle \phi(x_i), \phi(b) \rangle = \frac{1}{|C_j|} \sum_{b \in C_j} K(x_i, b)$$

Kernel k-means = kernel trick + k-means

Note that

$$\begin{aligned}\|\phi(x_i) - m_j\|^2 &= \langle \phi(x_i) - m_j, \phi(x_i) - m_j \rangle \\ &= \langle \phi(x_i), \phi(x_i) \rangle - 2\langle \phi(x_i), m_j \rangle + \langle m_j, m_j \rangle\end{aligned}$$

Given a cluster C_j, its center (on feature space) is

$$m_j = \frac{1}{|C_j|} \sum_{b \in C_j} \phi(b)$$

Thus

$$\langle m_j, m_j \rangle = \frac{1}{|C_j|^2} \sum_{b,c \in C_j} K(b, c)$$

Kernel k-means

Input: K: kernel k: number of clusters

Output: $C_1, ..., C_k$: partitioning of the points

1. Initialize the k clusters: $C_1^{(0)}, ..., C_k^{(0)}$.
2. Set $t = 0$.
3. For each point $\mathbf{a}$, find its new cluster index as

$$j^*(\mathbf{a}) = \text{argmin}_j \|\phi(\mathbf{a}) - \mathbf{m}_j\|^2, \text{ using (2).}$$

4. Compute the updated clusters as

$$C_j^{t+1} = \{\mathbf{a} \ : \ j^*(\mathbf{a}) = j\}.$$

5. If not converged, set $t = t + 1$ and go to Step 3; Otherwise, stop.