

# Kernel k-means

Johannes Oster

Faculty of Computer Science, University of Vienna

September 3, 2023

- 1 Motivation
- 2 Foundations
- 3 Demo
- 4 Evaluation Results
- 5 Lessons Learned and Outlook

- Traditional Clustering Algorithms
  - use distance-based measures
  - are effective for linearly separable data
  - struggle with non-linear, complex real-world data structures
- Kernel Methods
  - implicitly map data points to higher dimensions
  - thereby make non-linear data potentially linearly separable

- Traditional Clustering Algorithms
  - use distance-based measures
  - are effective for linearly separable data
  - struggle with non-linear, complex real-world data structures
- Kernel Methods
  - implicitly map data points to higher dimensions
  - thereby make non-linear data potentially linearly separable

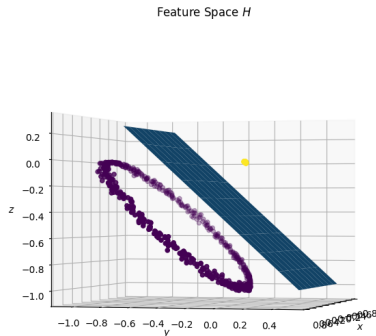
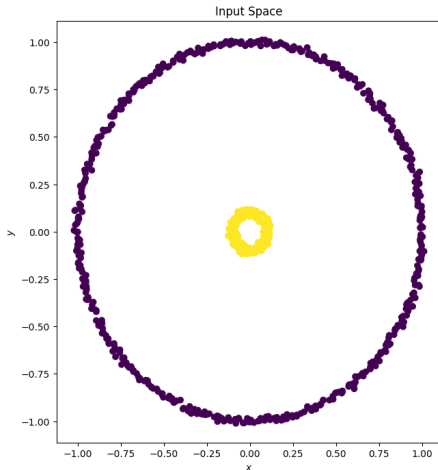


Figure: Visualizing how an explicit transformation  $\phi(x_i)$  leads to linear separability.

# The goal of this thesis was to

- implement the Kernel k-means algorithm in Python
- evaluate the implementation qualitatively on different data sets
- benchmark the implementation against other clustering algorithms<sup>1</sup>

---

<sup>1</sup>Agglomerative, Spectral, and Gaussian Mixture clustering

- Data points:  $x_i \in \mathbb{R}^N$
- Partitioning:  $\{\pi_{l=1}^k\}$
- Non-linear transformation:  $\phi : \mathbb{R}^N \rightarrow H$
- **Objective:** Minimize the squared euclidian distance between each data point  $x_i$  and the center  $c_l$  of its respective cluster  $\pi_l$  in  $H$ .
- $D(\{\pi_{l=1}^k\}) = \sum_{l=1}^k \sum_{x_i \in \pi_l} \|\Phi(x_i) - c_l\|^2$  with  $c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$

- Data points:  $x_i \in \mathbb{R}^N$
- Partitioning:  $\{\pi_{l=1}^k\}$
- Non-linear transformation:  $\phi : \mathbb{R}^N \rightarrow H$
- **Objective:** Minimize the squared euclidian distance between each data point  $x_i$  and the center  $c_l$  of its respective cluster  $\pi_l$  in  $H$ .
- $D(\{\pi_{l=1}^k\}) = \sum_{l=1}^k \sum_{x_i \in \pi_l} \|\Phi(x_i) - c_l\|^2$  with  $c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$



- Explicitly calculating transformation  $\phi$  for each data point is computational intensive
- Kernel-Function:  $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$
- Kernel-Matrix:  $K_{ij} = k(x_i, x_j) \forall x_i, x_j \in X \subseteq \mathbb{R}^N$
- $k : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is a valid kernel function if, and only if, it is symmetric and positive semidefinite, that is:
  - 1  $k(x_i, x_j) = k(x_j, x_i)$
  - 2  $\forall c \in \mathbb{R}^N, c^T K c \geq 0$
- Exemplary Kernels:
  - $k_{\text{Gaussian}}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{\text{Polynomial}}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{\text{Sigmoid}}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

- Explicitly calculating transformation  $\phi$  for each data point is computational intensive
- Kernel-Function:  $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$
- Kernel-Matrix:  $K_{ij} = k(x_i, x_j) \forall x_i, x_j \in X \subseteq \mathbb{R}^N$
- $k : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is a valid kernel function if, and only if, it is symmetric and positive semidefinite, that is:
  - 1  $k(x_i, x_j) = k(x_j, x_i)$
  - 2  $\forall c \in \mathbb{R}^N, c^T K c \geq 0$
- Exemplary Kernels:
  - $k_{\text{Gaussian}}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{\text{Polynomial}}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{\text{Sigmoid}}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

- Explicitly calculating transformation  $\phi$  for each data point is computational intensive
- Kernel-Function:  $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$
- Kernel-Matrix:  $K_{ij} = k(x_i, x_j) \forall x_i, x_j \in X \subseteq \mathbb{R}^N$
- $k : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is a valid kernel function if, and only if, it is symmetric and positive semidefinite, that is:
  - 1  $k(x_i, x_j) = k(x_j, x_i)$
  - 2  $\forall c \in \mathbb{R}^N, c^T K c \geq 0$
- Exemplary Kernels:
  - $k_{\text{Gaussian}}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{\text{Polynomial}}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{\text{Sigmoid}}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

- Explicitly calculating transformation  $\phi$  for each data point is computational intensive
- Kernel-Function:  $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$
- Kernel-Matrix:  $K_{ij} = k(x_i, x_j) \forall x_i, x_j \in X \subseteq \mathbb{R}^N$
- $k : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$  is a valid kernel function if, and only if, it is symmetric and positive semidefinite, that is:
  - 1  $k(x_i, x_j) = k(x_j, x_i)$
  - 2  $\forall c \in \mathbb{R}^N, c^T K c \geq 0$
- Exemplary Kernels:
  - $k_{\text{Gaussian}}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{\text{Polynomial}}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{\text{Sigmoid}}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

## ■ Current Objective Function:

$$D(\{\pi_{l=1}^k\}) = \sum_{l=1}^k \sum_{x_i \in \pi_l} \|\Phi(x_i) - c_l\|^2 \text{ with } c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$$

## ■ Substitute Scalar Product:

$$\begin{aligned} D(\{\pi_{l=1}^k\}) &= \sum_{l=1}^k \sum_{x_i \in \pi_l} \left\| \Phi(x_i) - \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|} \right\|^2 \\ &= \sum_{l=1}^k \sum_{x_i \in \pi_l} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right) \end{aligned}$$

## ■ For each data point, find a cluster where

$$\arg \min_{l \in \{1, 2, \dots, k\}} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right)$$

- Current Objective Function:

$$D(\{\pi_{l=1}^k\}) = \sum_{l=1}^k \sum_{x_i \in \pi_l} \|\Phi(x_i) - c_l\|^2 \text{ with } c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$$

- Substitute Scalar Product:

$$\begin{aligned} D(\{\pi_{l=1}^k\}) &= \sum_{l=1}^k \sum_{x_i \in \pi_l} \left\| \Phi(x_i) - \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|} \right\|^2 \\ &= \sum_{l=1}^k \sum_{x_i \in \pi_l} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right) \end{aligned}$$

- For each data point, find a cluster where

$$\arg \min_{l \in \{1, 2, \dots, k\}} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right)$$

- Current Objective Function:

$$D(\{\pi_{l=1}^k\}) = \sum_{l=1}^k \sum_{x_i \in \pi_l} \|\Phi(x_i) - c_l\|^2 \text{ with } c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$$

- Substitute Scalar Product:

$$\begin{aligned} D(\{\pi_{l=1}^k\}) &= \sum_{l=1}^k \sum_{x_i \in \pi_l} \left\| \Phi(x_i) - \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|} \right\|^2 \\ &= \sum_{l=1}^k \sum_{x_i \in \pi_l} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right) \end{aligned}$$

- For each data point, find a cluster where

$$\arg \min_{l \in \{1, 2, \dots, k\}} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right)$$

**input** : Data set  $X$ , Number of clusters  $k$ , Kernel function  $k\_func$ ,  
Convergence tolerance  $tol$

**output:** Cluster assignments  $A$

Compute kernel matrix  $K = k\_func(X)$

Initialize  $k$  clusters

**while** *change in assignments  $A$  or change in objective function  $> tol$*   
**do**

**for** *each data point  $x_i \in X$*  **do**

        Assign it to the cluster  $\pi_l$  that minimizes:

$$\arg \min_{l \in \{1, 2, \dots, k\}} \left( K_{ii} - 2 \frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right)$$

**end**

**end**

**return** cluster assignments  $A$



- Qualitative evaluation of performance on different data sets
- Quantitative benchmarking:
  - *Alternative Algorithms*: Agglomerative, Spectral, and Gaussian Mixture clustering
  - *Metrics*: Silhouette Score, Calinski-Harabasz Index, Davies-Bouldin Index, Sum of Variances
  - *Scenarios*:
    - fixed dimensionality with increasing data set size
    - increasing dimensionality with fixed data set size

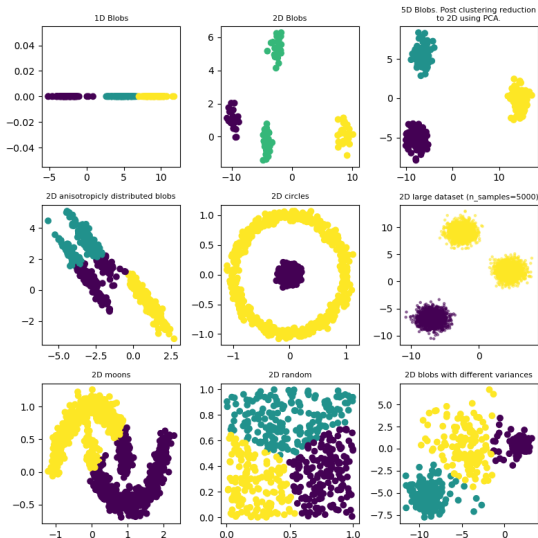
- Measures how similar a data point is to its own cluster compared to others
- Range:  $[-1;1]$ 
  - 1 ...good clustering
  - 0 ... overlapping clustering
  - -1 ... poor clustering
- $a(x_i) \in \pi_l \dots$  average dissimilarity of  $x_i$  to all other data points of  $\pi_l$
- $b(x_i) \dots$  average dissimilarity of  $x_i$  to all data points of its nearest neighbouring cluster
- $s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))}$
- $S(\{\pi_{l=1}^k\}) = \frac{1}{N} \sum_{l=1}^k \sum_{x_i \in \pi_l} s(x_i)$

- Measures between-cluster dispersion mean  $B_k$  over the within-cluster dispersion  $W_k$
- Range:  $[0; \text{inf}]$  where higher values indicate better clustering
- $B_k = \sum_{l=1}^k |\pi_l| \|c_l - c\|^2$
- $W_k = \sum_{l=1}^k \sum_{x_i \in \pi_l} \|x_i - c_l\|^2$
- $CHI(\{\pi_{l=1}^k\}) = \frac{B_k/(k-1)}{W_k/(N-k)}$

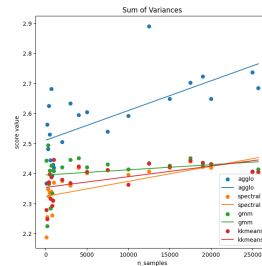
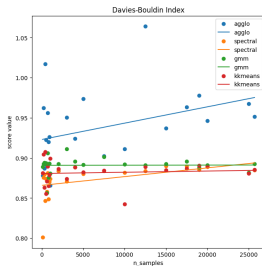
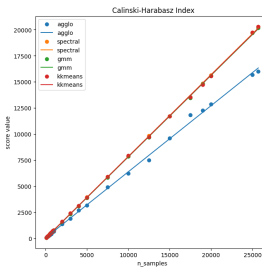
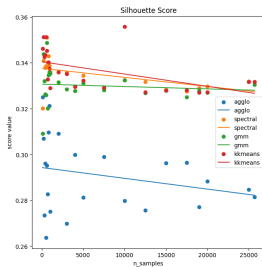
- Measures the average similarity of each cluster with its most similar one
- Range:  $[0; \infty]$  where lower values indicate better clustering
- $S(\pi_l) = \frac{1}{|\pi_l|} \sum_{x_i \in \pi_l} \|x_i - c_l\|^2$
- $R(\pi_l, \pi_m) = \frac{S(\pi_l) + S(\pi_m)}{\|c_l - c_m\|}$
- $DBI(\{\pi_{l=1}^k\}) = \frac{1}{k} \sum_{l=1}^k \max_{m \neq l} R_{lm}$

- Sums variance  $V$  for each point across all clusters
- Range:  $[0; \infty]$  where lower values indicate better clustering
- $V(x_i \in \pi_l) = ||x_i - c_l||^2$
- $E(\{\pi_{l=1}^k\}) = \sum_{l=1}^k \sum_{x_i \in \pi_l} V_{x_i}$

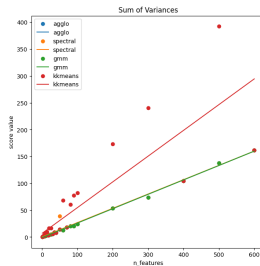
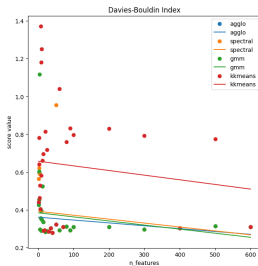
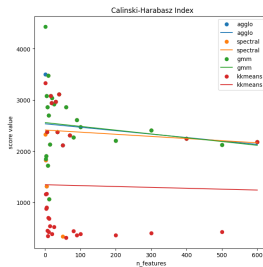
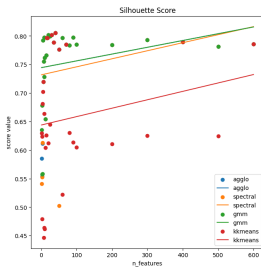
See Jupyter Notebook



# Results - Increasing Data Set Size







The custom implementation demonstrated

- similarly stable performance with growing data set sizes
- comparably large standard deviations in all scores and significantly higher Sum of Variances with growing dimensionality.
- detailed results can be found in the thesis under "5.2 Results"

## ■ Unexpected Findings

- Unexpected to find Silhouette score increasing with increasing dimensionality going against the typical 'curse of dimensionality'
- Instability as opposed to constant decline in quality of custom implementation for increased dimensionality

## ■ Future Work

- Optimization for High-dimensional Data (e.g. PCA)
- Kernel function performance comparisons
- Additional Performance metrics like computation time and resource intensity
- Increased diversity of data sets in benchmarking

## ■ Unexpected Findings

- Unexpected to find Silhouette score increasing with increasing dimensionality going against the typical 'curse of dimensionality'
- Instability as opposed to constant decline in quality of custom implementation for increased dimensionality

## ■ Future Work

- Optimization for High-dimensional Data (e.g. PCA)
- Kernel function performance comparisons
- Additional Performance metrics like computation time and resource intensity
- Increased diversity of data sets in benchmarking