Kernel k-means

Johannes Oster

Faculty of Computer Science, University of Vienna

September 4, 2023

Table of Contents

1 Motivation

2 Foundations

3 Demo

4 Evaluation Results

5 Lessons Learned and Outlook

Baseline

- 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

Baseline

- 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

Kernel Methods Visualized

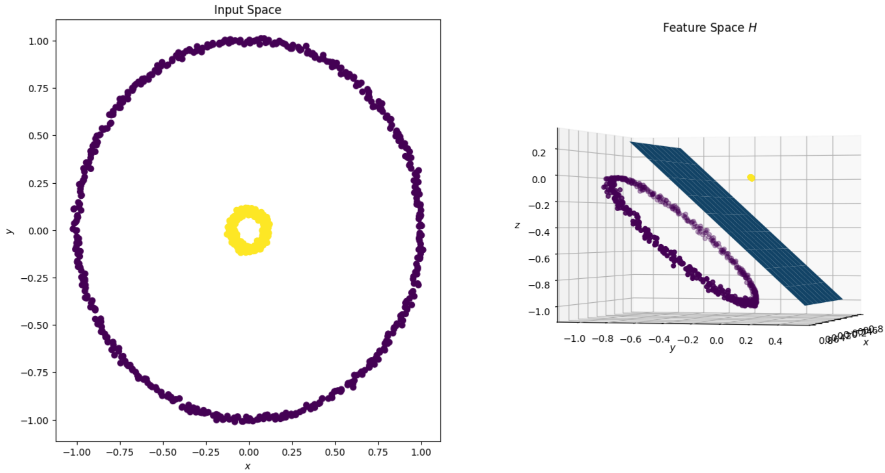


Figure: Visualizing how an explicit transformation $\phi(x_i)$ leads to linear separability.

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 [1]

[1] Agglomerative, Spectral, and Gaussian Mixture clustering

Objective of the algorithm

- 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 eucledian distance between each data point x_i and the center c_l of its respective cluster π_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 $c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$

Objective of the algorithm

- Data points: $x_i \in \mathbb{R}^N$
- Partitioning: $\{\pi_{l=1}^k\}$
- Non-linear transformation: $\phi : \mathbb{R}^N \to H$

- **Objective**: Minimize the squared eucledian distance between each data point x_i and the center c_l of its respective cluster π_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 $c_l = \frac{\sum_{x_j \in \pi_l} \Phi(x_j)}{|\pi_l|}$

Kernel Functions and Kernel Matrix

- Explicitely calculating transformation ϕ for each data point is computational intensive
- Kernel-Function: $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$
- Kernel-Matrix: $K_{ij} = k(x_i, x_j) \ \forall \ x_i, x_j \in X \subseteq \mathbb{R}^N$
- $k : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ 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. $\forall c \in \mathbb{R}^N, c^T K c \geqslant 0$
- Examplaratory Kernels:
  - $k_{Gaussian}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{Polynomial}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{Sigmoid}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

Kernel Functions and Kernel Matrix

- Explicitely calculating transformation ϕ 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. $\forall c \in \mathbb{R}^N, c^T K c \geqslant 0$
- Examplaratory Kernels:
  - $k_{Gaussian}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{Polynomial}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{Sigmoid}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

Kernel Functions and Kernel Matrix

- Explicitely calculating transformation ϕ for each data point is computational intensive
- Kernel-Function: $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$
- Kernel-Matrix: $K_{ij} = k(x_i, x_j) \; \forall \; x_i, x_j \in X \subseteq \mathbb{R}^N$
- $k : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ 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. $\forall c \in \mathbb{R}^N, c^T K c \geqslant 0$
- Examplaratory Kernels:
  - $k_{Gaussian}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{Polynomial}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{Sigmoid}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

Kernel Functions and Kernel Matrix

- Explicitely calculating transformation ϕ 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. $\forall c \in \mathbb{R}^N, c^T K c \geqslant 0$
- Examplaratory Kernels:
  - $k_{Gaussian}(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
  - $k_{Polynomial}(x_i, x_j) = (x_i \cdot x_j + \theta)^d$
  - $k_{Sigmoid}(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

Objective Function to minimize

- Current Objective Function:

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

- Substitute Scalar Product:

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

- For each data point, find a cluster where

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

Objective Function to minimize

- 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:

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

- For each data point, find a cluster where

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

Objective Function to minimize

- Current Objective Function:

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

- Substitute Scalar Product:

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

- For each data point, find a cluster where

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

Algorithm Outline

```
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:
         $\arg\min_{l \in \{1,2,\ldots,k\}} \left( K_{ii} - 2\frac{\sum_{x_j \in \pi_l} K_{ij}}{|\pi_l|} + \frac{\sum_{x_j, x_h \in \pi_l} K_{jh}}{|\pi_l|^2} \right)$
    end
end
return cluster assignments $A$
```

Evaluation Strategy

- 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

Metric: Silhouette Score

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

Metric: Calinski-Harabasz Index

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

Davies-Bouldin Index

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

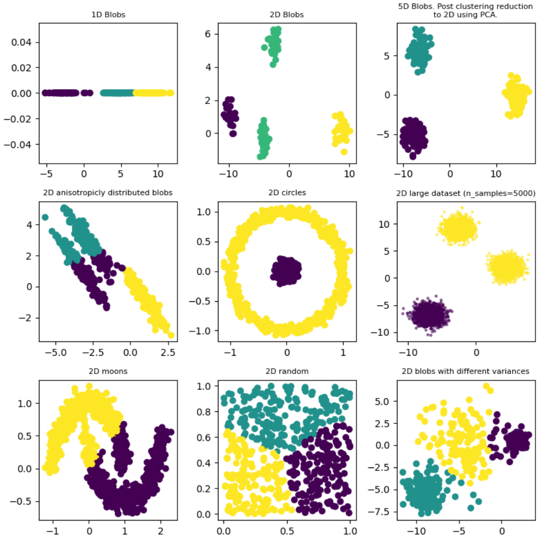
Sum of Variances

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

Demo

▸ See Jupyter Notebook

Results - Qualitative Evluation



1D Blobs

2D Blobs

5D Blobs. Post clustering reduction to 2D using PCA.

2D anisotropicly distributed blobs

2D circles

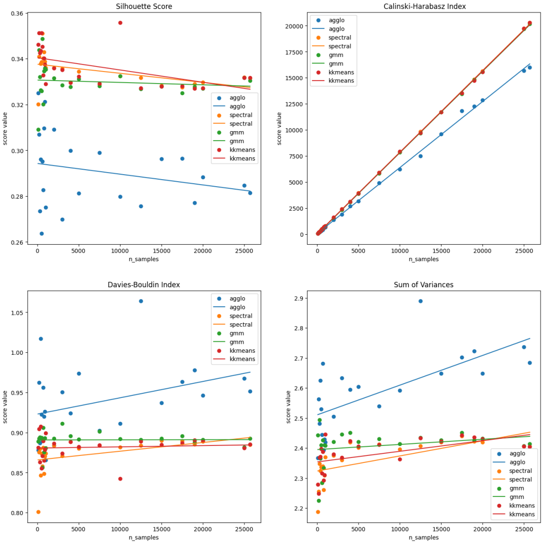
2D large dataset (n_samples=5000)

2D moons

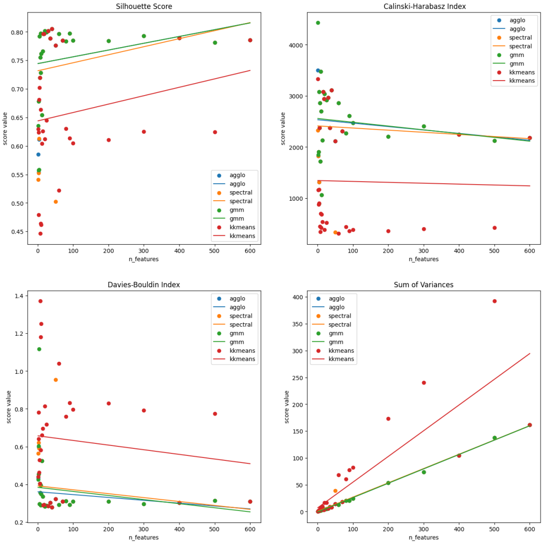
2D random

2D blobs with different variances

Results - Increasing Data Set Size



Results - Increasing Dimensionality



Results - Summary

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”

Lessons Learned and Outlook

- 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

Lessons Learned and Outlook

- 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