

HW3 - Theory questions

Valeriia Kravchik - 342390093

January 13, 2021

Q1 - Clustering

- (a) Note that the K-means algorithm is quite sensitive to noise and unnecessary data (outliers), since it is usually quite strongly influenced by extreme values. At the same time, a rather similar K-medoids algorithm, which is essentially a variant of K-means, is much more reliable in relation to noise and outliers.[1]

K-means clustering assigns each point to the cluster whose center is nearest. The center of the cluster is defined as the average of all points in the cluster. The algorithm attempts to minimize the intra-cluster variance defined by[2]:

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

where there are k clusters $S_i, i = 1, 2, \dots, k$ and μ_i is the center of all points $x_j \in S_i$.

K-medoids clustering is computed using the PAM-algorithm (PAM is short for Partitioning Around Medoids). It chooses datapoints as centers in contrast to the K-means algorithm. The PAM-algorithm is based on the search for k representatives (called medoids) among all elements of the dataset. When having found k representatives k clusters are now generated by assigning each element to its nearest medoid. Then it finds a local minimum for the objective function[3]:

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - c_i)^2$$

where there are k clusters $S_i, i = 1, 2, \dots, k$ and c_i is the medoid of S_i . So we know that the K-means uses a sum of squared Euclidean distances for minimization. But we know that K-medoids method tries to minimize a sum of pairwise dissimilarities. Little reminding about Euclidean distance between u and v . It is:

$$|u - v| = \sqrt{\sum_{i=1}^n (u_i - v_i)^2}$$

Let me visualize a robustness of K-medoid, when working with noise: Consider a dataset of 5 points in 2D (Figure 1):
(1,1),(1,2),(2,1),(2,2)(3,3):

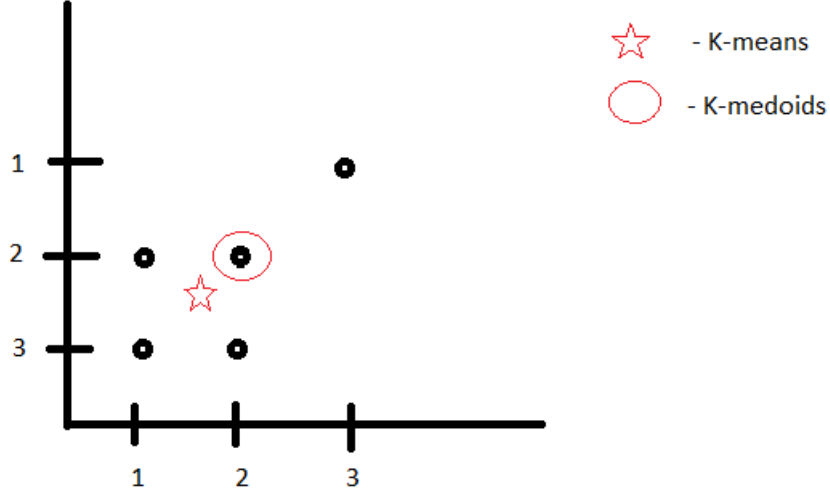


Figure 1: K-means vs K-medoids

Center of data using K-means:

$$X_{mean} = \frac{1 + 1 + 2 + 2 + 3}{5} = 1.8$$

$$Y_{mean} = \frac{1 + 1 + 2 + 2 + 3}{5} = 1.8$$

For K-medoids center of data. We need to arrange X_i in ascending order = 1,1,2,2,3, and from here the median is the middle for, that is 2. The same for Y_i , the medoid = 2 (Figure 1).

Let us to produce the outlier in point (100,100) (Figure 2):

Center of data using K-means:

$$X_{mean} = \frac{1 + 1 + 2 + 2 + 3 + 100}{6} = 18.16$$

$$Y_{mean} = \frac{1 + 1 + 2 + 2 + 3 + 100}{6} = 18.16$$

For K-medoids center of data. We need to arrange X_i in ascending order = 1,1,2,2,3,100.

$$X_{median} = \text{average of } (\frac{n}{2} \text{ term} + \frac{n+1}{2} \text{ term}) = \frac{2+2}{2} = 2$$

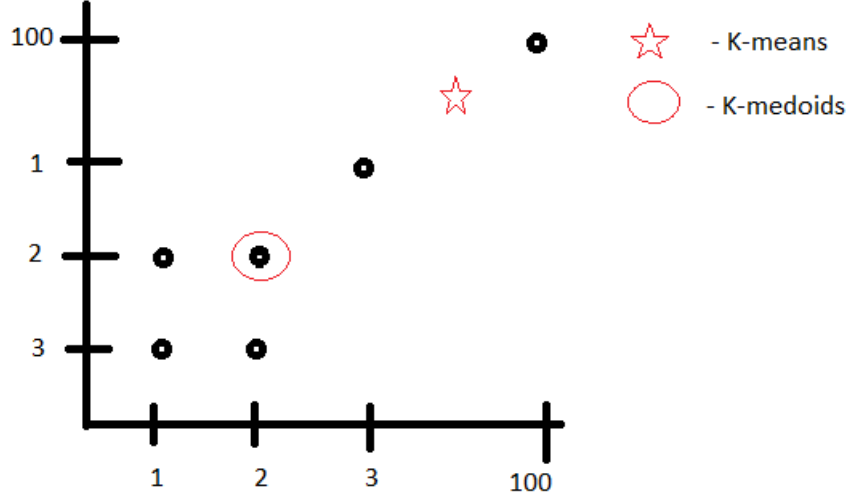


Figure 2: K-means vs K-medoids with outlier

We need to arrange Y_i in ascending order = 1,1,2,2,3,100.

$$Y_{median} = \text{average of } \left(\frac{n}{2} \text{ term} + \frac{n+1}{2} \text{ term}\right) = \frac{2+2}{2} = 2$$

We can pay attention that after adding the outlier the center for K-means moved from (1.8,1.8) to (18.16,18.16). But the K-medoids is still the same! Thus we can say that K-medoids algorithm is more robust to outliers and noise than K-means algorithm.

- (b) Let $z = \sum_{i=1}^n (x_i - \mu)^2$ for any μ

From the maxima minima principle we know, that z has it's minimum when:

$$\frac{\partial z}{\partial \mu} = 0 \text{ and } \frac{\partial^2 z}{\partial^2 \mu} > 0$$

$$\begin{aligned} \frac{\partial z}{\partial \mu} &= \frac{\partial}{\partial \mu} \left[\sum_i x_i^2 - 2\mu \sum_i x_i + \sum_i \mu^2 \right] = \\ &= \frac{\partial}{\partial \mu} \left[\sum_i x_i^2 - 2\mu \sum_i x_i + n\mu^2 \right] = 0 - 2 \sum_i x_i + 2n\mu \end{aligned}$$

from $\frac{\partial z}{\partial \mu} = 0$

$$\implies -2 \sum_i x_i + 2n\mu = 0$$

$$\implies -\sum_i x_i + n\mu = 0$$

$$\implies \mu = \frac{1}{n} \sum_i x_i$$

For $\mu = \frac{1}{n} \sum_i x_i$, $\frac{\partial^2 z}{\partial^2 \mu} = 2n > 0$

z is minimized at $\mu = \frac{1}{n} \sum_i x_i$

Hence, $\mu = \operatorname{argmin}_{\mu} \sum_i (x_i - \mu)^2$

Bonus Let $z = \sum_{i=1}^n |x_i - \mu|$

set arrange x_i^s such that $x_1 < x_2 < \dots < x_n$

for $\mu \in \operatorname{set}[x_1, x_n]$ we get:

$$\begin{aligned} \sum_{i=1}^n |x_i - \mu| &= \sum_{i=2}^{n-1} |x_i - \mu| + |x_1 - \mu| + |x_n - \mu| = \\ &= \sum_{i=2}^{n-1} |x_i - \mu| + (\mu - x_1) + (x_n - \mu) = \sum_{i=2}^{n-1} |x_i - \mu| + (x_n - x_1) \end{aligned}$$

with the same argument applied to all the series we got:

$$z = \sum_{i=1}^n |x_i - \mu| = |x_{\frac{n+1}{2}} - \mu| + |x_n - x_1| + \dots + |x_{\frac{n+3}{2}} - x_{\frac{n-1}{2}}| = |x_{\frac{n+1}{2}} - \mu| + \operatorname{const}$$

z gets to the minimum, where $z' = 0$ and $z'' > 0$. z is minimized when $\mu = x_{\frac{n+1}{2}}$ (median), hence $\operatorname{median} = \operatorname{argmin}_{\mu} \sum_i |x_i - \mu|$

Q2 - SVM

Linear kernel means that in the graph we will get the line. Parameter C tells to the SVM optimizer how much you want to avoid missclassifying in the training set. For large values C, the optimization will choose a smaller-margin hyperplane. Accordingly the small values of C the optimization will choose the bigger margin. For tiny C you will get missclassified examples, even if you data linear separable.

Polynomial kernel in general defined as: $K(X_1, X_2) = (a + X_1^T X_2)^b$. b - is the degree of the kernel, a is a constant term [4]. In the polynomial kernel we simply calculate the dot product by increasing the power of the kernel. With large b the separable line will look very complex.

RBF kernel is function whose value depends on the distance from the origin or from some point. The γ parameter influence on the optimization as fitting optimizer. That means that if value of γ increases, the model gets overfits, as the value of γ decreases the model underfits, and the same is true about the C parameter for RBF kernel.

From this explanations I can conclude, that:

for picture A the answer is 2

for picture B the answer is 6

for picture C the answer is 3

for picture D the answer is 1

for picture E the answer is 5

for picture F the answer is 4

References

- [1] S. Mannor, X. Jin, J. Han, X. Jin, J. Han, X. Jin, J. Han, and X. Zhang, “K-Medoids Clustering,” in *Encyclopedia of Machine Learning*. Springer US, 2011, pp. 564–565. [Online]. Available: https://link.springer.com/referenceworkentry/10.1007/978-0-387-30164-8_426
- [2] S. P. Lloyd, “Least Squares Quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [3] L. Kaufman and P. J. Rousseeuw, Eds., *Finding Groups in Data*, ser. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc., mar 1990. [Online]. Available: <http://doi.wiley.com/10.1002/9780470316801>
- [4] “SUPPORT VECTOR MACHINES(SVM). Introduction: All you need to know... | by Ajay Yadav | Towards Data Science.” [Online]. Available: <https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>