# Fast Noise Removal for k-Means Clustering

Venkata Sasank Mudigonda (vmudigon@masonlive.gmu.edu)

## I. Source Paper

The paper in [1], titled "**Fast Noise Removal for k-Means Clustering**" was published in April 2020 by *Sungjin Im* and *Mahshid Montazer Qaem* from University of California at Merced, *Xiaorui Sun* from University of Illinois at Chicago, and *Benjamin Moseley* and *Rudy Zhou* from Carnegie Mellon University. This paper aims to introduce a quick noise pre-processing step to speed up the $k$-means clustering algorithms.

## II. Source Code

The paper is implemented in the Python programming language. Below are the list of libraries used.

- **NumPy** - for efficient matrix computations
- **SciPy** - for perform efficient computation of pairwise distances among two sets of points
- **pandas** - to read the data sets stored in different formats
- **matplotlib** - for plotting the graphs for analyses

All the algorithms ($k$-means, $k$-means–, $k$-means++, NK-means, and sample coreset) are organised in separate *.py* files. The $Project.ipynb$ is used to import all the algorithms, run the experiments and plot the graphs stating the results.

## III. Methodology

### A. Notation

I have implemented most important algorithms used for final evaulation. They are $k$-MEANS, $k$-MEANS–, $k$-MEANS++, NK-MEANS, and SAMPLE_CORESET using predominantly NumPy. I have dedicated each algorithm a section to state the algorithm and to give sufficient explanation.

Below is the notation used across all the algorithms in the paper.

- $n$ - number of elements
- $x, y \in \mathbb{R}^d$ are elements in sets $X, Y \in \mathbb{R}^d$. $Y$ is a coreset of $X$.
- $d(x, y)$ is the distance metric. We define $d(x, Y) = min_{y \in Y}(d(x, y))$
- for a point $x$, a ball centered with radius $r$ is given by $B(x, r) = \{y \in X | d(x, y) \leq r\}$
- $X_z(C) \subset X$ is a subset of points of $X$ excluding $z$ points with highest assignment costs
- for a set of $k$ centers $C \subset R^d$ and $z \in N$ outliers, we define $z - cost$ of C by

$$f_z^X = \sum_{x \in X_z(C)} d^2(x, C)$$

- Thus the $z$-cost of $C$ is the cost of clustering $X$ with $C$ while excluding the $z$ points with highest assignment costs
- $L$ is the set of $l$ farthest points from the centers $C$ in each iteration in the $k$-MEANS– algorithm

### B. Lloyd's algorithm for the k-means problem

Lloyd's algorithm in [2], also called the $k$-means algorithm is one of the first attempts to solve the $k$-means problem. It randomly initializes $k$ centers and tried to optimize the centers in an iterative fashion and ensuring improvement in the objective function. The sets $S_j$ are the sets of points to which $\mu_j$ is the closest center. After recording the points in each cluster, an average of these points in each cluster is computed to give rise to new cluster centers. The formal algorithm is listed in **Algorithm 1**.

**Algorithm 1** Lloyd's Algorithm

**procedure** $k$-MEANS($X, k$)
  $\mu_1, \ldots, \mu_k \leftarrow$ randomly chosen centroids
  **while** Objective function still improves **do**
    $S_1, \ldots, S_k \longleftarrow \phi$
    **for** $i \in 1, \ldots, n$ **do**
      $j \leftarrow arg\min_j \parallel x_i - \mu_j \parallel^2$
      Add $i$ to $S_j$
    **end for**
    **for** $j \in 1, \ldots, k$ **do**
      $\mu_j \longleftarrow \frac{1}{|S_j|} \sum_{i \in S_j} x_i$
    **end for**
  **end while**
  **return** $\{\mu_1, \ldots, \mu_k\}$
**end procedure**

### C. k-means++

To address the sensitivity of Lloyd's algorithm to random initialization of cluster center David Arthur, & Sergei Vassilvitskii in [3] proposed a new initialization strategy to the Lloyd's algorithm. Their algorithm is given by the following steps.

1) Choose an initial center $c_1$ uniformly at random from $X$.
2) Choose the next center $c_i$, selecting $c_i = x^{'} \in X$ with probability $\frac{D(x^{'})^2}{\sum_{x \in X} D(x)^2}$
3) Repeat step 2 until we have chosen a total of $k$ centers
4) Proceed with Lloyd's algorithm for solving the $k$-means problem

### D. k-means−−

The $k$-means– is an algorithm proposed by Chawla & Aristides in [4] to solve the $k$-means with outliers problem. It is solving the $k$-means problem by identifying $k$ centers by excluding $z$ outliers. The algorithm removes $l$ outliers in each iteration and recomputes the cluster centers using Lloyd's approach. See **Algorithm 2** for a formal algorithm.

**Algorithm 2** Algorithm for $k$-means−−

**procedure** $k$-MEANS−−$(X, k, l, d)$
  $C_0 \leftarrow \{k$ random points of $X\}$
  $i \leftarrow 1$
  **while** no convergence achieved **do**
    Compute $d(x, C_{i-1}) \forall x \in X$
    Re-order the points in $X$ such that $d(x_1, C_{i-1}) \geq \ldots \geq d(x_n, C_{i-1})$
    $L_i \leftarrow \{x_1, \ldots, x_l\}$
    $X_i \leftarrow X \setminus L_i = x_{l+1}, \ldots, x_n$
    **for** $j \in \{1, \ldots, k\}$ **do**
      $P_j \leftarrow \{x \in X_i | c(x|C_{i-1}) = c_{i-1,j}\}$
      $c_{i,j} \leftarrow mean(P_j)$
    **end for**
    $C_i \leftarrow \{c_{i,1}, \ldots, c_{i,k}\}$
    $i \leftarrow i + 1$
  **end while**
  **return** $C, L$
**end procedure**

### E. NK-means

It is the algorithm proposed in [1] by the authors that reduces the k-means with outliers to the stan-dard k-means problem. This enables the usage of otherimplementations of k-means along with NK-means. See **Algorithm 3** for a formal algorithm.

**Algorithm 3** Algorithm for NK-means

**procedure** NK-MEANS$(X, k, z, A)$
  Suppose we know the optimal objective value $Opt := Opt(X, k, z)$
  Initialize $r \leftarrow 2(Opt/z)^{1/2}, Y \leftarrow \phi$
  **for** each $x \in X$ **do**
    Compute $B(x, r)$
    **if** $|B(x, r)| \geq 2z$ **then**
      Mark $x$ as $heavy$
    **end if**
  **end for**
  **for** each $x \in X$ **do**
    **if** $|B(x, r)|$ contains no heavy points **then**
      Update $Y \leftarrow Y \cup x$
      Mark $x$ as $heavy$
    **end if**
  **end for**
  **return** C $\leftarrow A(X \backslash Y, k)$
**end procedure**

### F. Coreset Construction

To scale down large datasets, the authors also proposed the **Algorithm 4** which creates a coreset upon which theclustering algorithms are applied. A Coreset is a weighted setof points $Y$ of a larger set of points $X$ when it holds true thata good clustering of $Y$is also a good clustering of $X$. This way, at a smaller scale, the clustering algorithms run faster,especially if the time complexity is huge.

**Algorithm 4** Algorithm for Coreset Construction

**procedure** SAMPLECORESET$(X, k, z)$
  Let $p = max(\frac{36}{z}\log(\frac{4nk^2}{z}), 36\frac{k}{z}log(2k^3))$
  **if** $p > 1$ **then**
    Output $Y \leftarrow k$-MEANS++$(X, 32(k + z))$
  **else**
    Let $S$ be a sample drawn from $X$, where each $x \in X$ is included in $S$ independently with probability $p$.
    Output C $\leftarrow k$-MEANS++$(X\ Y, k)$
  **end if**
  **return** C $\leftarrow A(X \backslash Y, k)$
**end procedure**

## IV. EXPERIMENTS

### A. Datasets

The following datasets have been used to examine the above defined algorithms.

- **KDD** [5]: $n = 4898431, d = 34, k = 3, z = 45747$. The 7 non-numeric features were dropped.
- **SKIN-$\Delta$** [7]: $n = 245057, d = 3, k = 10, z = 0.01n$. Only the first 3 features were used.
- **SUSY-$\Delta$** [8]: $n = 5M, d = 18, k = 10, z = 0.01n$
- **POWER-$\Delta$** [6]: $n = 2049280, d = 7, k = 10, z = 0.01n$. The first 2 features, date and time were dropped

The outliers ($z$) are added to each normalized data set from the uniform distribution of $[-\Delta, \Delta]$. The data set SKIN-5 implies that the original data has been normalised and $z$ outliers drawn from the uniform distribution defined by the bounds [-5, 5] are added to the normalized data set. I have generated the datasets of SKIN, SUSY and POWER with $\Delta = \{5, 10\}$, while keeping KDD as is.

### B. Experiments on the data

I ran Lloyd's, $k$-means++, $k$-means–, NK-means, sample coreset algorithms on the datasets and their variants mentioned above. The algorithms are evalutated using objective value ($l_2$-norm) recall, and the run time (specified in order in the tables 1 and 2).

## V. CONCLUSIONS

I could reproduce the results for $k$-means– algorithm computing the $l_2$-norm values, recall, and run time (in seconds). The entries hightlighted in bold font are the ones where I could attain a better result than the paper. For $k$-means–, it's performance on SKIN-10 and SUSY-10 datasets

| | SKIN-5 | SUSY-5 | POWER-5 | KDD |
|---|---|---|---|---|
| NK-MEANS | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| $k$-means– | 2.6456 | 56.3837 | 44.4747 | 155.8024 |
| | 0.751 | 0.7993 | 0.6298 | 0.0035 |
| | 2.15 | 110 | 27.3 | 68 |
| $k$-means++ (c) | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 |
| $k$-means++ (o) | 2.3034 | 52.9273 | 57.9480 | 146.0895 |
| | - | - | 0 | 0 |
| | 0.671 | 249 | 30 | 10.7 |

TABLE I

RESULTS WITH $\Delta = 5$

| | SKIN-10 | SUSY-10 | POWER-10 |
|---|---|---|---|
| NK-MEANS | 0 | 0 | 0 |
| | 0 | 0 | 0 |
| | 0 | 0 | 0 |
| $k$-means– | 2.1811 | 51.6976 | 57.0119 |
| | **0.9379** | **0.9815** | 0.9647 |
| | 2.02 | 110 | 27 |
| $k$-means++ (c) | 0 | 0 | 0 |
| | 0 | 0 | 0 |
| | 0 | 0 | 0 |
| $k$-means++ (o) | 7.1105 | 60.3155 | 196.0934 |
| | - | - | - |
| | 0.560 | 241 | 50.7 |

TABLE II

RESULTS WITH $\Delta = 10$

exceeded the paper. Fig. 1 and 2 show the performances measured in $l2$-norm values for $k$-means– and $k$-means++ (original dataset) across the datasets. For other algorithms, my computer is overwhelmed and the Python kernel gets restarted. I plan to run these algorithms in AWS Elastic Cloud Compute.

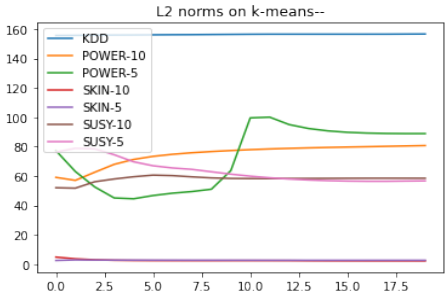


Fig. 1. $l_2$-norm values vs number of iterations of $k$-means–

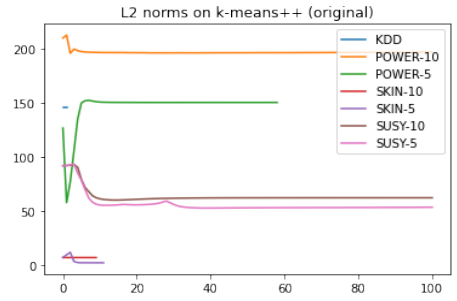


Fig. 2. $l_2$-norm values vs number of iterations of $k$-means++ (original dataset)

## REFERENCES


[1] Sungjin Im, Mahshid Montazer Qaem, Benjamin Moseley, Xiaorui Sun, & Rudy Zhou. (2020). Fast Noise Removal for $k$-Means Clustering.

[2] S. Lloyd, "Least squares quantization in PCM," in IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, March 1982, doi: 10.1109/TIT.1982.1056489.

[3] David Arthur, & Sergei Vassilvitskii (2007). K-means++: the advantages of careful seeding. In In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms.

[4] Chawla, Sanjay & Gionis, Aristides. (2013). K-means-: A unified approach to clustering and outlier detection. 189-197. 10.1137/1.9781611972832.21.

[5] Hettich, S. and Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

[6] Individual household electric power consumption data set. URL https:// archive.ics.uci.edu/ml/datasets/ Individual+household+electric+power+consumption

[7] Skin segmentation data set. URL https://archive.ics.uci.edu/ml/datasets/Skin+Segmentation

[8] SUSY data set. URL https://archive.ics.uci. edu/ml/datasets/SUSY.