Critical Review

Topic - **Fast and Provably Good Seeding for *k*-Means**

- Olivier Bachem

Link for Paper : https://las.inf.ethz.ch/files/bachem16fast.pdf (NIPS 2016)

Link for our Implementation of review code :
https://github.com/skygup7/CSN-382-Machine-Learning/tree/master/CourseProject

15114004 Akash Gupta

15114033 Harsh Kumar Bansal

INTRODUCTION

This paper extends the work of Bachem in reducing the running time of the k-means++ procedure. Bachem suggested sampling points using a Markov process (instead of computing the complete distribution after every iteration and then sampling from this distribution). Since computing the distribution is a costly operation, this technique reduces the running time significantly for very large datasets. However, in order to show that the sampling probabilities is similar to that of the k-means++ procedure, we need to show bounds on the mixing time of the Markov process. This in turn imposes constraints on the datasets on which this technique may be applied. The current paper approaches the problem in a slightly different manner. They try to argue that the Markov process based algorithm works for any dataset if one allows some additive factors in the approximation guarantee. They also argue that in some cases this additive approximation factor does not cause any serious problems. Furthermore, they show that on many real datasets, the additive term is small for reasonable values of parameters. As in Bachem work, the process is significantly faster than the k-means++ seeding procedure.

K-means

The *k*-means is an algorithm to find cluster centers that minimizes the intra-class variance, i.e. the sum of squared distances from each data point being clustered to its cluster center (the center that is closest to it). Although finding an exact solution to the *k*-means problem for arbitrary input is NP-hard, the standard approach to finding an approximate solution (often called Lloyd's algorithm or the *k*-means algorithm) is used widely and frequently finds reasonable solutions quickly.

However, the *k*-means algorithm has at least two major theoretical shortcomings:

- First, it has been shown that the worst case running time of the algorithm is super-polynomial in the input size.
- Second, the approximation found can be arbitrarily bad with respect to the objective function compared to the optimal clustering.

The *k*-means++ algorithm addresses the second of these obstacles by specifying a procedure to initialize the cluster centers before proceeding with the standard *k*-means optimization iterations is called **seeding.** With the *k*-means++ initialization, the algorithm is guaranteed to find a solution that is *O(log k)* competitive to the optimal *k*-means solution.

K-means++

K-means++ is an algorithm for choosing the initial values (or "seeds") for the *k*-means clustering algorithm.

K-means Algorithm

1. Take one center C_1, chosen uniformly at random from X .
2. Take a new center C_i , choosing x ∈ X with probability = $D(x)^2/\sum_{x\in\mathcal{X}} D(x)^2$.
3. Repeat Step 2. until we have taken k centers altogether.
4. Proceed as with the standard k-means algorithm.

   We call the weighting used in Step 2 simply “D weighting”.

This seeding method yields considerable improvement in the final error of k-means. Although the initial selection in the algorithm takes extra time, the k-means part itself converges very quickly after this seeding and thus the algorithm actually lowers the computation time.The k-means++ algorithm guarantees an approximation ratio *O(log k)* in expectation (over the randomness of the algorithm), where *k* is the number of clusters used.

For n data samples of d-dimensions having k number of clusters have

$$Runtime = O(nkd)$$

The drawback of k-Means++ is that it does not scale easily to massive data sets since both its seeding step and every iteration of Lloyd's algorithm require the computation of all pairwise distances between cluster centers and data points.

K-MC^2

Above K-means++ highlights the need for a fast and scalable seeding algorithm. Such an approach was presented by Bachem et al. (2016) who propose to approximate k-means++ using a Markov chain Monte Carlo (MCMC) approach and provide a fast seeding algorithm that produces probably good clustering with some assumptions on the data.

For n data samples of d-dimensions having k number of clusters have

Runtime = $O(mk^2d)$ (m is markov chain length)

The method consists in an approximation of k-Means++ using Markov chain Monte Carlo (MCMC) approach,where the data points are states. The first chain state is a random sampled data point. A randomized process determines if the chain transitions to other random data points. The decision of state transition is dependent of the initial distances of all data points, that are computed once, as part of a pre-processing step. Unlike k-Means++ it only requires a full pass through all the dataset.

The stationary distribution of K-MC^2 is the same as k-Means++ given enough steps in the chain are simulated. Under natural assumptions on the data generating distribution, Bachem showed that the computational complexity of k-means++ can be greatly decreased while retaining the same *$O(\log k)$* guarantee on the solution quality.

The drawback of this approach is that these assumptions(mainly I.I.D) may not hold and that checking their validity is expensive.

AFK-MC^2 (Assumption Free K-MC^2)

The proposed AFK-MC^2 method is presented as a simple, yet fast, alternative for k-Means seeding that produces probably good clustering **without** assumptions on the data. Ideally, it should also retain the theoretical guarantees of k-means++ and provide equally competitive clusterings in practice.While K-MC^2 requires assumptions about the data generating distribution (in our implementation uniformity), this algorithm works without such assumptions. It the true D^2-sampling distribution with regards to the first center C_1 as a proposal distribution that can approximate non-uniform distributions.

This means an added runtime cost of *O(nd)* to calculate the initial distribution, but performance improvements for non-uniform samples n data samples of d-dimensions having k number of clusters have

Runtime = $O(nd+mk^2d)$ (m is markov chain length)

In contrast to the previously introduced algorithm K-MC2 , it produces provably good clusterings even without assumptions on the data. As a key advantage, ASSUMPTION-FREE K-MC^2 allows to provably trade off solution quality for a decreased computational effort. Extensive experiments illustrate the practical significance of the proposed algorithm: It obtains competitive clusterings at a fraction of the cost of k-means++ seeding and it outperforms or matches its main competitor K-MC^2 on all considered data sets.

OBJECTIVE

The objective of this paper is to provide fast and competitive seedings for k-Means clustering without prior assumptions on the data. As key contributions, Research Paper

(1) propose a simple yet fast seeding algorithm for k-Means,

(2) show that it produces provably good clusterings without assumptions on the data,

(3) provide stronger theoretical guarantees under assumptions on the data generating distribution,

(4) extend the algorithm to arbitrary distance metrics and various divergence measures,

(5) compare the algorithm to previous results, both theoretically and empirically, and,

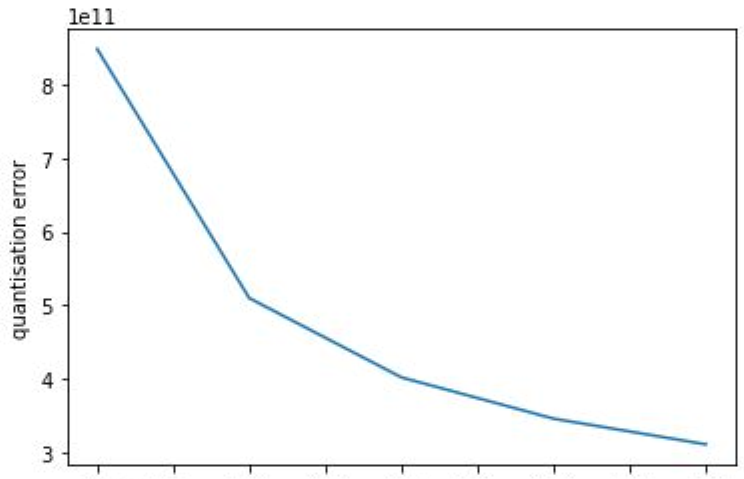
(6) demonstrate its effectiveness on several real-world data sets.

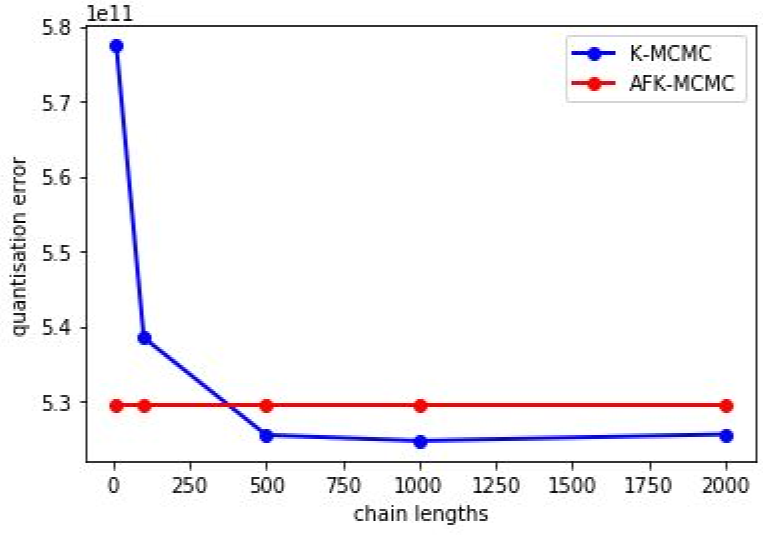
Experimental Analysis

Implementation 1

Protein Homology Dataset (http://osmot.cs.cornell.edu/kddcup/datasets.html)

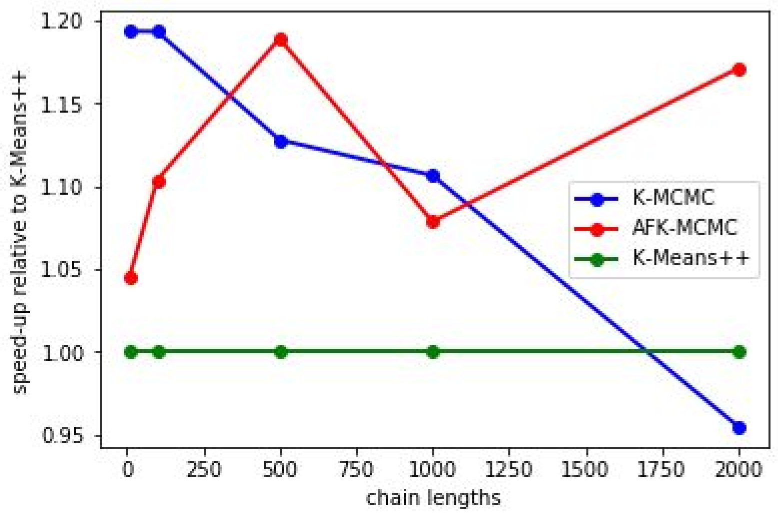
We train our model with Protein Homology Dataset.This Dataset contains “bio_train.dat” file.The dataset, “bio_train.dat”, contains 145,751 instances of data, each with 74 features. First, the optimum number of clusters are determined by “Elbow Method” as given below.





As can be seen from the graph, the elbow can be pointed at n=200. Hence, we take k=200 (number of clusters) for the K-Means algorithm. After running K-Means++, K-MC^2 , AFK-MC^2 for different chain lengths, we get the following result.

From the figure, we can see that as the chain length increases, the quantisation error decreases and approaches towards the K-Means++ base line (quantisation error for K-Means++ = 3.1x10^{11}). Also, the calculated speed-up relative to K-Means++ is given by the following graph.



The figure clearly shows significant speed-up for various chain lengths relative to K-Means++

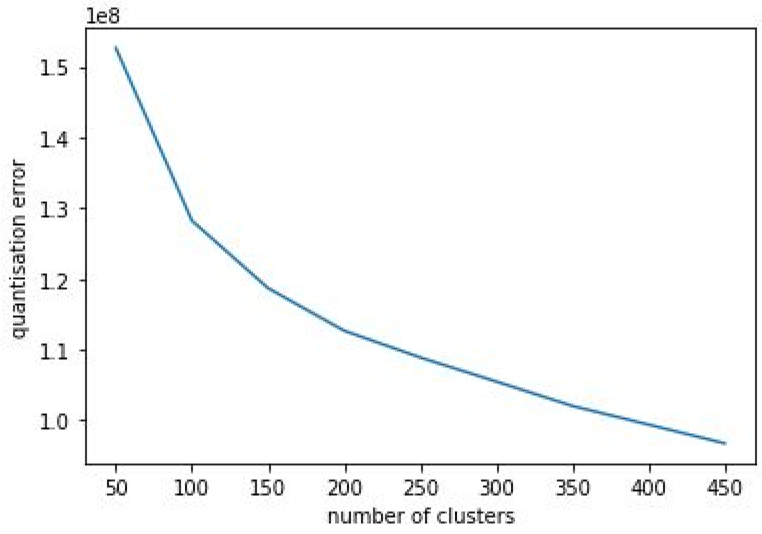
Hence, the algorithm is verified on the Protein Homology Dataset (KDD)

Implementation 2

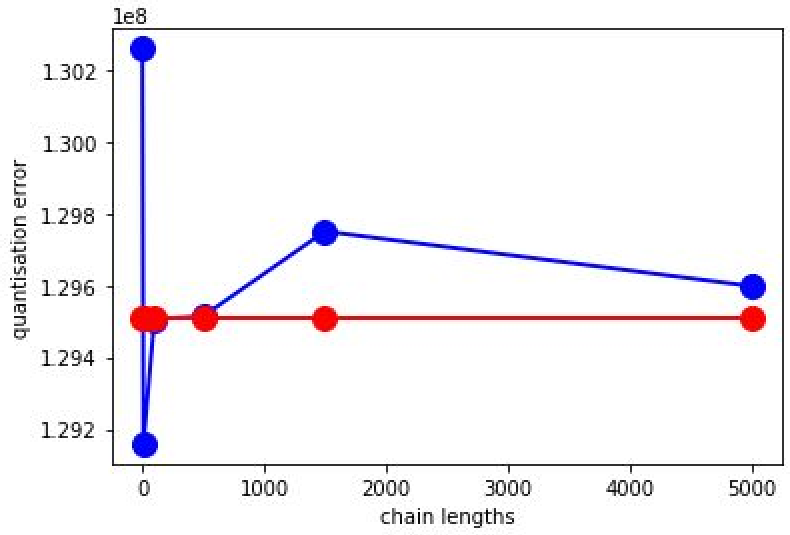
Oxford Flower Dataset (http://www.robots.ox.ac.uk/~vgg/data/flowers/)

We train our model on The Oxford flowers dataset with 102 classes. The dataset contains 8188 images for 102 different classes of flowers. Then we extracted FC7(Fully Connected Convolutional layer) features from the images using CNN model by TensorFlow (Link : https://github.com/skygup7/SummerProject2017/blob/master/feature-extraction-from-flower.ipynb)

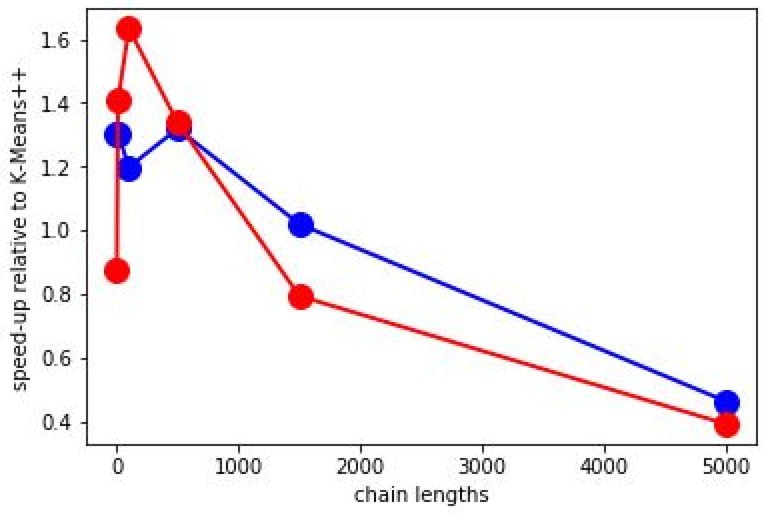
Then we find the optimum numbers of clusters for the features extracted by elbow method.



Here, we find that optimum number of clusters is 100. Hence, we take k=100 (number of clusters) for the K-Means algorithm. After running K-Means++, K-MC^2 , AFK-MC^2 for different chain lengths, we get the following result.



From the figure, we can see that as the chain length increases, the quantisation error decreases and approaches towards the K-Means++ base line (quantisation error for K-Means++ = 1.278x10^8). Also, the calculated speed-up relative to K-Means++ is given by the following graph.



The figure shows speed-up for various feasible chain lengths relative to K-Means++

Hence, the algorithm is verified on the Oxford Flowers Dataset (VGG)

Detailed steps are described in the IPython Notebook in GitHub link.

EXPERIMENTAL RESULTS'

For **Oxford Flower Dataset,** we get following Table Result :

Chain length	**KM++** (time in seconds)	**K-MC^2** (time in seconds)	**AFK-MC^2** (time in seconds)
1	1.618425599720831	1.2446888773138198	1.8473356289809089
10	1.618425599720831	1.240580574750129	1.1506127428969193
100	1.618425599720831	1.3510007535265687	0.9907419286690129
500	1.618425599720831	1.2267621276309	1.2095934804978241
1500	1.618425599720831	1.5875733533817766	2.0386110989942607
5000	1.618425599720831	3.49064142993866	4.124420849338276

For **Protein Homology Dataset,** we get following Table Result :

Chain length	**KM++** (time in seconds)	**K-MC^2** (time in seconds)	**AFK-MC^2** (time in seconds)
10	46.28615170261955	38.77869020646303	44.2850810276359
100	46.28615170261955	38.78491732161892	41.94123745155703
500	46.28615170261955	41.03727249867207	38.92301087353481
1000	46.28615170261955	41.83517716429397	42.91637492099144
2000	46.28615170261955	48.500249108554996	39.51891522953447

The both tables show the time taken by the algorithms for various chain lengths in 2 different datasets. It can be seen that K-MC^2 shows significant improvement in time with some assumptions and AFK-MC^2 shows significant improvement over K-Means++ without any assumptions on data. This result can also be visualised by the speedup graph for both datasets.

Conclusion

In this paper, bachem propose ASSUMPTION-FREE K-MC^2 is a simple and fast seeding algorithm for k-Means. In contrast to the previously introduced algorithm K-MC^2 , it produces provably good clusterings even without assumptions on the data. As a key advantage, ASSUMPTION-FREE K-MC^2 allows to provably trade off solution quality for a decreased computational effort. Extensive experiments illustrate the practical significance of the proposed algorithm: It obtains competitive clusterings at a fraction of the cost of k-means++ seeding and it outperforms or matches its main competitor K-MC^2 on all considered data sets.