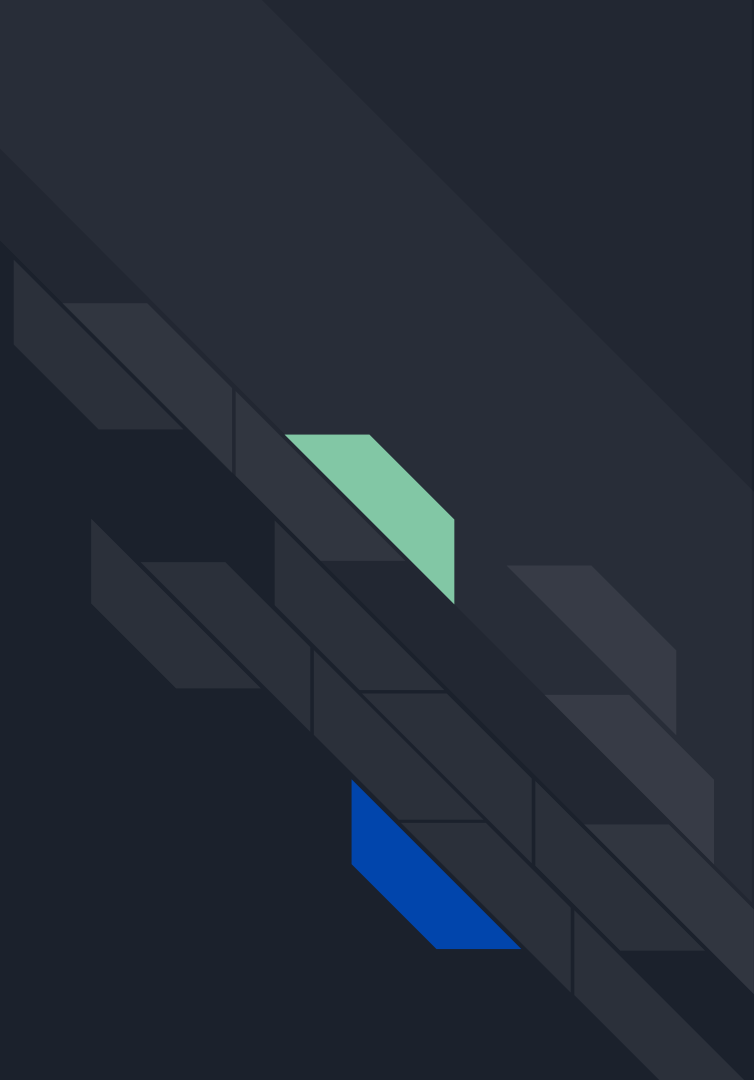


A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is a light green color. They are positioned diagonally, with the blue one partially covering the green one.

Web-Scale k-Means clustering

Cheikh Tidiane Diop

Goal of the project





Goal of the project

Present two modifications to the popular k-means clustering algorithm to address the extreme requirements for latency, scalability, and sparsity encountered in user-facing web applications by orders of magnitude compared to the classic batch algorithm.

For this presentation , we are focusing on the first modification which relies on the mini-batch optimization for k-means clustering




Overview

- Data used
- Implementation of the k-means clustering using the minibatch optimization
- Performing k-Means using the MiniBatch optimization, evaluate the time for our model to be trained and visualize the cluster
- Performing k-Means using the standard Batch optimization , evaluate the time for our model to be trained and visualize the clusters
- Observations based on the results
- Going further



About our data set

[Enron Dataset](#) which comprises 500,000+ emails from 150 employees of the Enron Corporation. After cleaning the dataset and preprocessing it, we will perform the k-means clustering on 18000 email subjects.

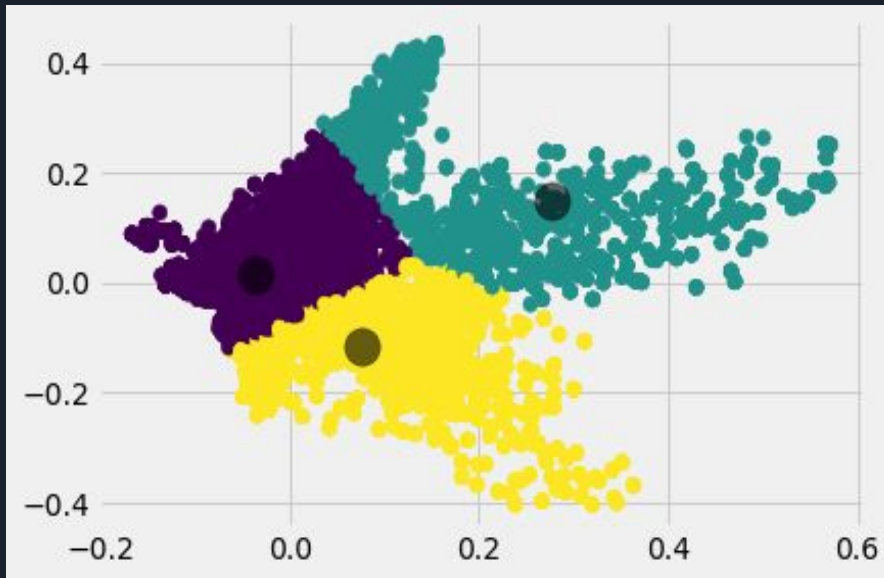


Implementation of the k-means clustering using the minibatch optimization

1. Initialize the clusters centers
2. Assign samples to their closest clusters using the euclidean metric
3. Update the clusters centers (where the minibatch optimization intervenes)
4. Repeat step 2 and 3 till the maximum number of iterations is reached

Performing k-Means using the MiniBatch optimization, evaluate the time for our model to be trained and visualize the cluster

Clustering



Running Model Iteration 0

Running Model Iteration 100

Running Model Iteration 200

Running Model Iteration 300

Running Model Iteration 400

Running Model Iteration 500

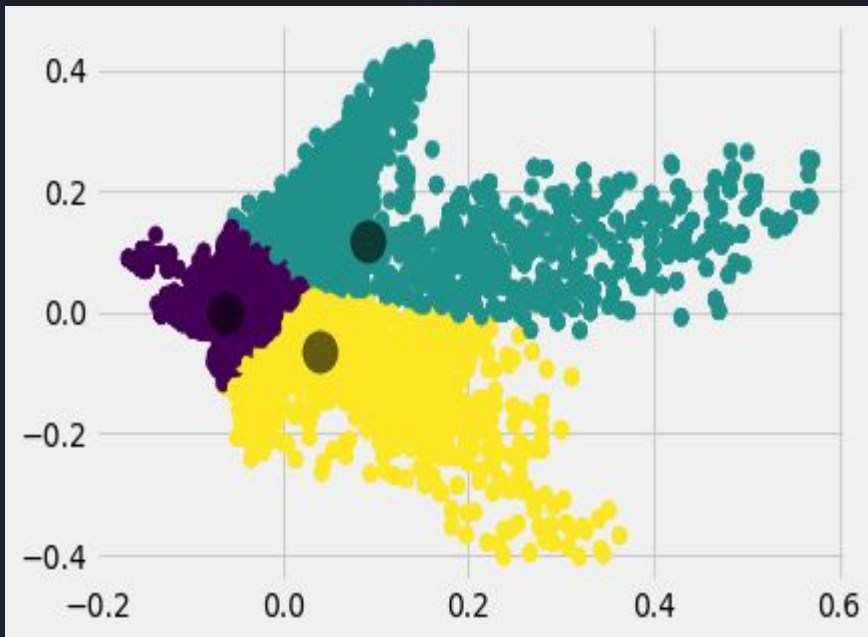
Model finished running

CPU times: user 12.1 s, sys: 17.4 ms,
total: 12.1 s

Wall time: 12.1 s

Performing k-Means using the Standard Batch optimization, evaluate the time for our model to be trained and visualize the cluster

Clusters



Running Model Iteration 0

Running Model Iteration 100

Running Model Iteration 200

Running Model Iteration 300

Running Model Iteration 400

Running Model Iteration 500

Model finished running

CPU times: user 3min 22s, sys: 15.6 s,
total: 3min 37s

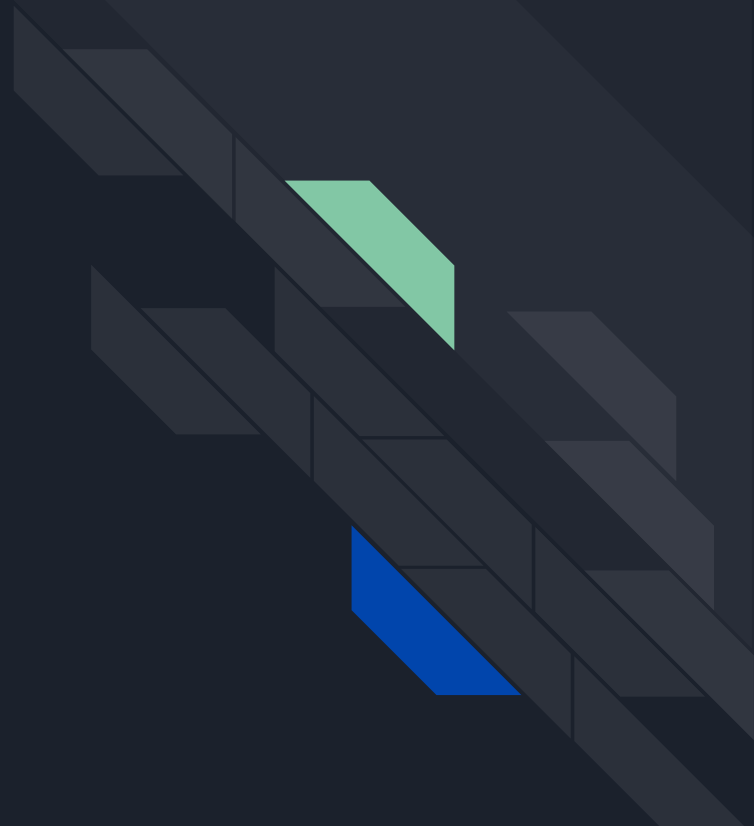
Wall time: 3min 20s



Observations based on the results

From the two previous images displaying clustering using Batch on one hand and Minibatch on the other hand, we don't observe so much differences in the clusters. While in terms of computational cost, the K-Means algorithm implemented using Minibatch took 12 seconds to be trained while the k-Means algorithm implemented using the Batch took 3 mins to be trained

Going further





How can we improve the k-Means clustering?

1. Use a different method to initialize the centroids: k-means++
2. Use the elbow method to get the optimal number of clusters
3. Use the L1 regularizer to promote sparsity so that our model learns the informative features
4. As we are using web-scale data, large dataset, using k-means with sklearn will lead to better results