Secure Mean-shift Clustering using CrypTen

Qiang Fu, Jingtao Li

March 2021

0.1 Goal

To implement a MPC-based mean-shift clustering using CrypTen.

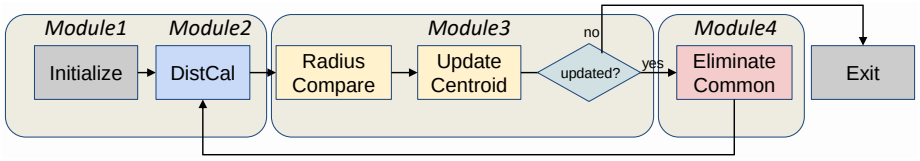

Figure 1: Mean-shift Clustering Process (Modular Version)

0.2 Background

The process of mean-shift clustering is shown in Fig. 1. Vanilla version of the mean-shift clustering takes input points P and a radius R (aka. bandwidth) as input.

- (Module 1) The algorithm starts from every data point by initialize centroids C at the exact location of input points.
- (Module 2) The distance of every centroids to every input points are calculated.
- (Module 3) Input points within the radius R hyper sphere centered at each centroid are assigned to that centroid. The mean of the associated data points is used to update the centroids for the next iteration. If no centroid changes, the program ends.
- (Module 4) Redundant centroids are removed if multiple centroids have the same value.

We repeat Module 2, 3, 4 until the mean value does not change anymore, which implies its convergence. With more and more centroids being eliminated at module 4, this whole process leaves only a few important cluster centers as the output.

0.3 Analysis

The algorithm is similar to the k-means clustering, but it does not need the user to give a input for the number of clusters. As a drawback, it needs to initialize a large number of centroids at the beginning (equal to the number of input points P). As a results, the computation complexity is polynomial. The "Distcal" module needs to calculate $|P|^2$ distance and later "Radius Compare" "Update Centroid" modules need to do $|P|^2$ comparison and addition. This polynomial complexity is a serious problem considering MPC-based secure implementation.

Thus, heuristics shall be considered for the implementation. [1] describes a dust heuristic for mean-shift clustering. Instead of initialization as vanilla version does, it sets a small number of random sampled points called dusts D as initial centroids, where $|D| \ll |P|$. The only problem with this heuristic is that it may result in inaccurate cluster compared to the vanilla version. But it turns out to be accurate enough according to [1].

0.4 Implementation Detail

The secure protocol starts with global agreement on the value of radius R and some random seed that is used in random points generation.

Assume we have multiple parties who have their own data points and want to securely compute clusters, we use CrypTen to implement the secure version of mean-shift clustering, where we use different threads to simulate different parties.

Input/Module 1. The input data we use is randomly generated. We firstly generates some center points randomly on an x-y plane, and points are generated using Gaussian distribution around each center points. Then, we use the global random seed to sample a small number of dusts from the input points (save as index). An example of this process is shown at Fig. 2.

Then, secret shares are created for all parties from the generated private input and secret shares of the dusts can be collected using the dusts' index.

Module 2. Distance between each centroid and input points pair are computed. CrypTen multiplication and addition functions are used to compute the euclidean distance between two points. The code is available in the github link (provided in Appendix).

Module 3. First the distance from Module 2 is compared with a selected radius to check if points are within the range of each centroid. A new centroid is then calculated. If any centroid changed, it will be updated and pass an identification of True or False.

Module 4. CrypTen equal functions are used to compare whether any two centroids are the same. A nested loop of comparison is used to eliminate the redundant centroids.

Output.

The output is the secret shares of the final derived centroids.

0.5 Results

0.5.1 Experimental Settings

We use a laptop with Intel Core i7-4510U CPU (2 core, 4 thread) to run the experiments. We select a base setting taking n_point = 100, n_center = 8 for the data, and taking radius = 0.1, n_party = 2 and n_dust = 8 for the algorithm setting. The generated centroids are very close to the one generated by mean-shift in sklearn. An example is shown in Fig. 3.

0.5.2 Time Breakdown

Iteration-wise. The first iteration is the longest in execution time, and the following iterations becomes shorter due to the reduced number of centroid. The execution time scales linearly with the number of centroid.

Phase-wise. Inside one iteration, most of the time is spending on Module2 and module3. By taking average across all of our experiments. We found they have a fixed ratio of around 7:4.

0.5.3 Sweeping Results (MeanShift Parameter).

We investigate taking different value of meanshift settings. We sweep the number of dust, the number of party, and radius values and we record the average time per iteration. The result is shown in Fig. 4. The time increases linearly with n_dust and the radius has little effect on the time, which are expected. And we also observe that the time increase dramatically with increasing number of parties.

However, the radius affects the accuracy of the clustering. As shown in Fig. 5, if the radius is small, the cluster is scattered too far apart and more sensitive to the initialization (where we use the dust heuristic). If the radius is too large, clusters tend to get close to each other and finally merge into one. Neither too small or too large can achieve a good clustering performance.

0.5.4 Sweeping Results (Data Parameter).

Then, we investigate the role of the dataset. We sweep the number of center (for generatign the clustered data) and the number of points in total. As shown in Fig. 6, the time scales almost linearly with the number of data which is expected. But it does not change much with the number of center.

0.5.5 Observation of Erroneous Results.

We found the generated centroids sometimes have negative value or very deviated from the ground-truth centroid when the data size is large. The arithmetic calculation seems to be not working correctly. Later we confirmed that this is related to a flaw in the Crypten division implementation.

References

[1] Jung Hee Cheon, Duhyeong Kim, and Jai Hyun Park. Towards a practical cluster analysis over encrypted data. In *International Conference on Selected Areas in Cryptography*, pages 227–249. Springer, 2019.

A Appendix

A.1 Code and Validation

Code and Validation can be found at Here

A.2 Result Figures

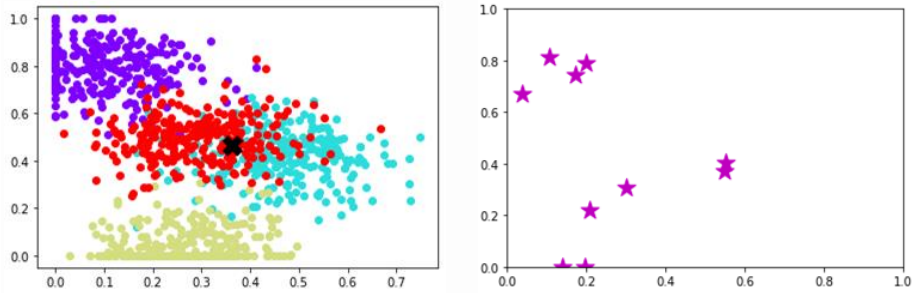

Figure 2: An example of randomly generated input points P, and randomly sampled dusts D using the dust heuristics in [1].

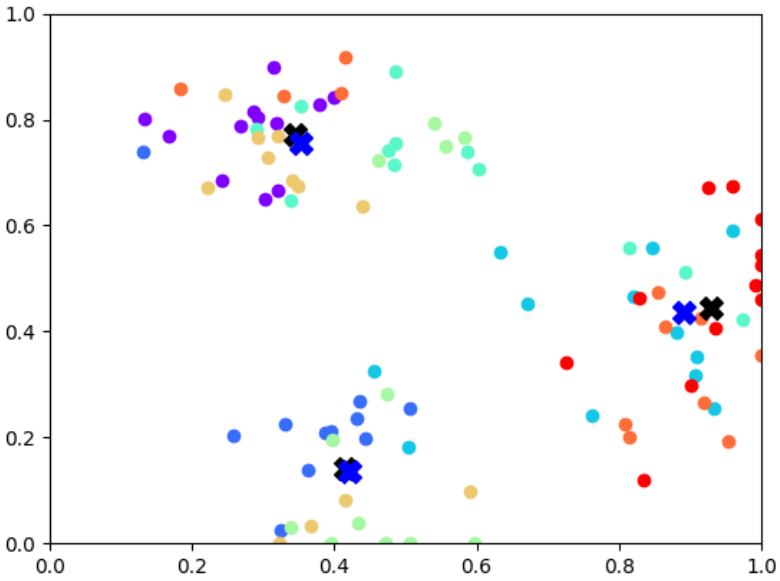

Figure 3: A clustering example. The blue marks are centroids generated by our secure mean-shift, and the black marks are centroids generated by mean-shift in sklearn.

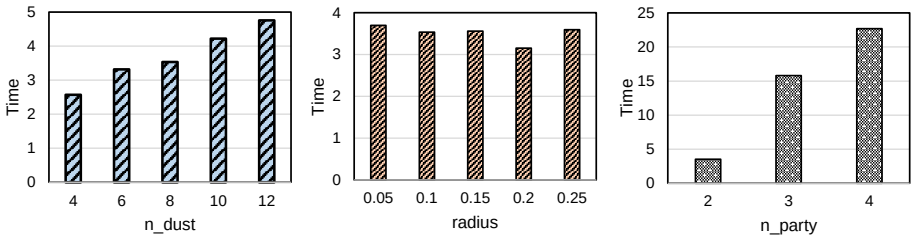

Figure 4: Results of sweeping the algorithm parameters.

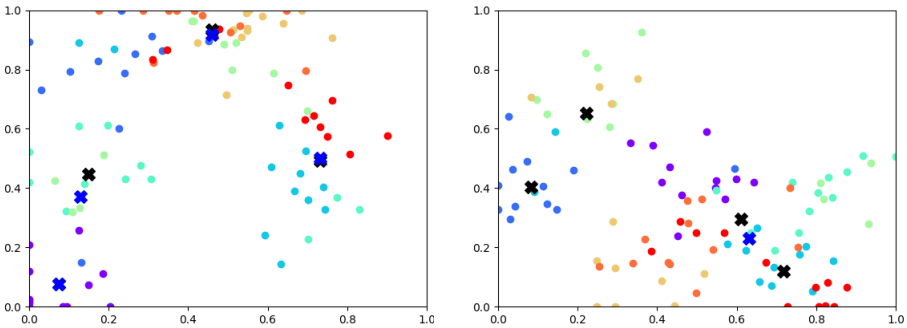

Figure 5: Clustering results of different radius. Left: radius = 0.05. Right: radius = 0.20.

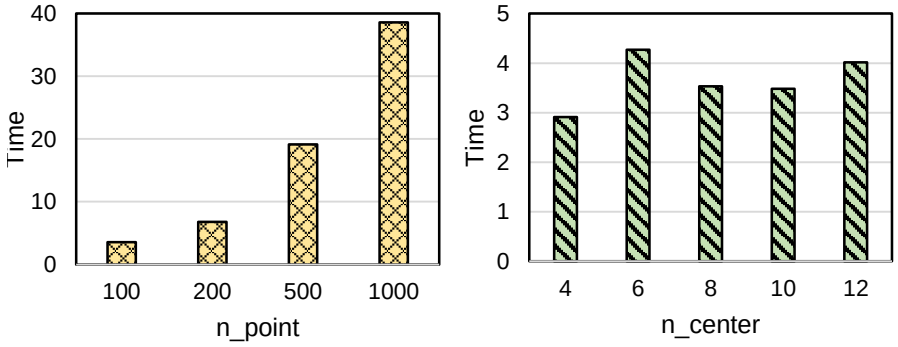

Figure 6: Results of sweeping the data parameters.